A* Search and Bound-Sensitive Heuristics for Oversubscription Planning

Michael Katz[1] **Emil Keyder**[2]

[1]IBM Research, Yorktown Heights, NY, USA
[2]Invitae Corporation, San Francisco, CA, USA
michael.katz1@ibm.com, emilkeyder@gmail.com

Abstract

Oversubscription planning (OSP) is the problem of finding plans that maximize the utility value of their end state while staying within a specified cost bound. Recently, it has been shown that OSP problems can be reformulated as classical planning problems with multiple cost functions but no utilities. Here we take advantage of this reformulation to show that OSP problems can be solved optimally using the A^* search algorithm, in contrast to previous approaches that have used variations on branch-and-bound search. This allows many powerful techniques developed for classical planning to be applied to OSP problems. We also introduce novel bound-sensitive heuristics, which are able to reason about the primary cost of a solution while taking into account secondary cost functions and bounds, to provide superior guidance compared to heuristics that do not take these bounds into account. We propose two such bound-sensitive variants of existing classical planning heuristics, and show experimentally that the resulting search is significantly more informed than with comparable heuristics that do not consider bounds.

Introduction

Oversubscription planning (OSP) problems are a family of deterministic planning problems. In contrast to classical planning, where a set of hard goals is specified and the planner searches for a minimal (or low) cost plan that reaches a state in which all of the goals are made true, oversubscription planning specifies a *utility function* that describes the benefit associated with achieving different possible states, and asks for a plan that achieves as high a utility as possible, but whose cost does not exceed a set bound (Smith 2004).

While domain-independent classical planning approaches have increasingly standardized around variations on A^* search and heuristics that are automatically extracted from the problem description (Bonet and Geffner 2001; Keyder and Geffner 2008; Haslum and Geffner 2000; Edelkamp 2001; Helmert et al. 2014; Helmert and Domshlak 2009), OSP has generally been solved with branch-and-bound algorithms and heuristics that compute an admissible estimate of the utility achievable from a state. In the context of OSP, an admissible estimate is one that does not *underestimate* the utility achievable from a state. In order to obtain these estimates, recent approaches often adapt classical planning techniques such as landmarks (Mirkis and Domshlak 2014; Muller and Karpas 2018) or abstractions (Mirkis and Domshlak 2013), and enhance them with reasoning that is specific to the context of OSP, such as the knowledge that there always exists an optimal plan that ends with a utility-increasing action, or that the cost bound for the problem can be reduced under specific conditions to aid the search algorithm in detecting that improving over the currently achieved utility is impossible. Though solving OSP tasks with A^* search would enable the integration of various search pruning techniques used in classical planning (Alkhazraji et al. 2012; Pochter, Zohar, and Rosenschein 2011), no previous approach was able to do so.

In contrast to these approaches, we show that general methods from classical planning, including A^* search, can be used in the OSP setting. This previously turned out to be the case for the related *net-benefit* planning problem, where classical planners solving a compiled version of the problem with action costs but no utilities were shown to outperform planners designed specifically for that task (Keyder and Geffner 2009). Here, we use a similar, recently proposed compilation that converts OSP problems into classical planning problems with multiple cost functions but no utilities (Katz et al. 2019a). In addition, we demonstrate that existing classical planning heuristics can be used to guide the search for optimal plans. While these heuristics are typically uninformative when applied to the compilation out-of-the-box (ignoring the secondary cost function), we show how they can be modified (without any specific reasoning about utilities) to render them sensitive to the secondary cost functions and bounds that are introduced by the compilation. Our experiments with A^* and the newly introduced estimators that we refer to as *bound-sensitive heuristics* show that they lead to informed searches that are competitive with, and often outperform, the state of the art in heuristic search for optimal OSP. We now briefly review the various flavors of planning that we consider in this work, and introduce the formalisms by which we describe them.

Background

We describe planning problems in terms of extensions to the SAS$^+$ formalism (Bäckström and Nebel 1995). A *classical planning task* $\Pi = \langle V, O; s_I, G, \mathcal{C} \rangle$ is given by a set of variables V, with each variable $v \in V$ having a finite do-

Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

main $dom(v)$, a set of actions O, with each action $o \in O$ described by a pair $\langle \mathsf{pre}(o), \mathsf{eff}(o) \rangle$ of partial assignments to V, called the *precondition* and *effect* of o, respectively, initial state s_I and goal condition G, which are full and partial assignments to V, respectively, and the cost function $\mathcal{C} : O \to \mathbb{R}^{0+}$. A state s is given by a full assignment to V. An action is said to be *applicable* in a state s if $\mathsf{pre}(o) \subseteq s$, and $s[\![o]\!]$ denotes the result of applying o in s, where the value of each $v \in V$ is given by $\mathsf{eff}(o)[v]$ if defined and $s[v]$ otherwise. An operator sequence $\pi = \langle o_1, \ldots, o_k \rangle$ is applicable in s if there exist states $s_0, \cdots, s_k$ such that (i) $s_0 = s$, and (ii) for each $1 \leq i \leq k$, o_i is applicable in s_{i-1} and $s_i = s_{i-1}[\![o_i]\!]$. We refer to the state s_k by $s[\![\pi]\!]$ and call it the end state of π. An operator sequence π is a plan for a classical planning problem if it is applicable in s_I and $G \subseteq s_I[\![\pi]\!]$. The cost of a plan π is given by $\mathcal{C}(\pi) = \sum_{o \in \pi} \mathcal{C}(o)$; the goal of optimal classical planning is to find a plan with minimal cost. We refer to a pair $v, \vartheta \in dom(v)$ as a *fact* and denote it by $\langle v, \vartheta \rangle$. We sometimes abuse notation and treat partial assignments as sets of facts.

An oversubscription planning (OSP) problem is given by $\Pi_{\text{OSP}} = \langle V, O, s_I, \mathcal{C}, \mathsf{u}, \mathsf{B} \rangle$, where V, O, s_I, and $\mathcal{C}$ are as in classical planning, $\mathsf{u}{:}\langle\langle v, \vartheta \rangle\rangle \to \mathbb{R}^{0+}$ is a non-negative valued utility function over variable assignments (facts), and B is a cost bound for the plan, imposing the additional requirement that only plans π such that $\mathcal{C}(\pi) \leq \mathsf{B}$ are valid. The *utility* of a plan π is given by $\sum_{\langle v, \vartheta \rangle \in s_I[\![\pi]\!]} \mathsf{u}(\langle v, \vartheta \rangle)$; the objective of OSP problems is to find valid plans with maximal utility.

A multiple cost function (MCF) problem is given by $\Pi_{\text{MCF}} = \langle V, O, s_I, G, \mathcal{C}_0, \mathscr{C} \rangle$, where V, O, s_I, and $\mathcal{C}_0$ are as in classical planning, $\mathcal{C}_0$ is the *primary cost function*, and $\mathscr{C} = \{\langle \mathcal{C}_i, \mathsf{B}_i \rangle \mid 1 \leq i \leq n\}$ is a set of n *secondary cost functions* $\mathcal{C}_i : O \to \mathbb{R}^{0+}$, and *bounds*, both non-negative. Valid plans for MCF planning problems fulfill the condition $\mathcal{C}_i(\pi) \leq \mathsf{B}_i$ for all secondary cost functions, and optimal plans for MCF planning have minimal primary cost $\mathcal{C}_0(\pi)$. In this paper we limit our discussion to MCF problems with a single secondary cost function, i.e. $n = 1$, as this is sufficient to represent the OSP problems that we consider.

Reformulating OSP Problems

It has recently been shown that an OSP problem can be compiled into an MCF planning problem with a single secondary cost function that corresponds to the cost function $\mathcal{C}$ of the original problem, and is constrained to not exceed the specified bound B (Katz et al. 2019a). The primary cost function for the problem, or the cost function to be optimized, results from compiling the utilities from the original problem into costs. Two different compilations have been proposed for this task. The first of these is the *soft goals compilation*, which adds a new hard goal to the problem for each variable v that has some value $\vartheta \in dom(v)$ with a non-zero utility, along with actions that are able to achieve this hard goal at different costs. The other is the *state-delta compilation* which does not introduce any new variables or actions, but instead encodes in the cost of each action the change in state utility that results from applying it. Here we consider only the soft goals compilation, as the state-delta compilation introduces negative action costs that A^* and existing classical planning heuristics are not designed to handle. Note, however, that our methods do not depend on the specific choice of compilation, as long as they remove utilities from the problem and do not introduce negative action costs.

The *soft goals compilation* was originally introduced in the context of net-benefit planning. Net-benefit planning is similar to oversubscription planning in that it introduces a utility function over the set of states, but differs in that it does not specify a bound on plan cost, having instead as its objective the maximization of the difference between utility and cost (Keyder and Geffner 2009). This compilation can be applied in the OSP setting to result in an MCF planning problem (Katz et al. 2019a, Definition 2):

Definition 1 *Let* $\Pi_{\text{OSP}} = \langle V, O, s_I, \mathcal{C}, \mathsf{u}, \mathsf{B} \rangle$ *be an oversubscription planning task. The* soft goals reformulation $\Pi^{sg}_{\text{MCF}} = \langle V', O', s_I, G', \mathcal{C}_0, \{\langle \mathcal{C}', \mathsf{B} \rangle\} \rangle$ *of* Π_{OSP} *is an* MCF planning task*, where*

- $V' = \{v' | v \in V\}$*, with* $dom(v') = \begin{cases} dom(v) \cup \{g_v\} & \mathsf{u}_{max}(v) > 0 \\ dom(v) & \textit{otherwise}, \end{cases}$
- $O' = O \cup \{o^{v,\vartheta} = \langle \{\langle v, \vartheta \rangle\}, \{\langle v, g_v \rangle\} \rangle \mid \vartheta \in dom(v), v \in V, \mathsf{u}_{max}(v) > 0\}$
- $G' = \{\langle v, g_v \rangle | v \in V, \mathsf{u}_{max}(v) > 0\}$,
- $\mathcal{C}_0(o) = \begin{cases} 0 & o \in O \\ \mathsf{u}_{max}(v) - \mathsf{u}(\langle v, \vartheta \rangle) & o = o^{v,\vartheta}, \end{cases}$
- $\mathcal{C}'(o) = \begin{cases} \mathcal{C}(o) & o \in O \\ 0 & \textit{otherwise}, \end{cases}$

with $\mathsf{u}_{max}(v) := \max_{\vartheta \in dom(v)} \mathsf{u}(\langle v, \vartheta \rangle)$ *denoting the maximum utility over the values of the variable* v*.*

In the reformulated problem, only the $o^{v,\vartheta}$ actions for which ϑ is not the maximum utility value of v have positive primary costs. These actions make explicit that a particular utility will not be achieved, and that the plan has instead chosen to achieve the associated hard goal g_v by accepting a cost penalty equal to the difference between the maximum achievable utility for that variable and the current utility. The *primary cost* of a plan π for the reformulated problem is then given by $\sum_{v \in V} \mathsf{u}_{max}(v) - \sum_{f \in s_I[\![\pi]\!]} \mathsf{u}(f)$.

While this compilation is sound as stated, two further optimizations can be made to reduce the state space of the resulting compiled problems. First, an arbitrary ordering can be introduced over V to ensure that the g_v values are achieved in a fixed sequence, to avoid searching over different orderings. Second, a new precondition fact that is deleted by the $o^{v,\vartheta}$ actions can be added to the original domain actions to ensure that $o^{v,\vartheta}$ actions happen only at the end of the plan and are not interleaved with the original domain actions. We make use of both of these optimizations in the experimental results presented here.

A^* for MCF Planning Problems

The A^* algorithm extends blind search techniques such as Dijkstra's algorithm by allowing the incorporation of admissible (non-overestimating) heuristics (Hart, Nilsson, and

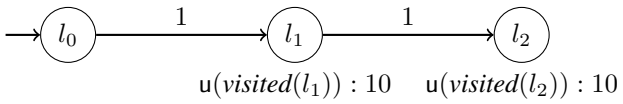

Figure 1: An OSP problem based on the VISIT-ALL domain.

Raphael 1968). In each iteration of its main loop, A^* picks a node n to expand with minimal $f(n) = g(n) + h(n)$ value, where $g(n)$ is the cost of the path to n, and $h(n)$ is an admissible estimate of the remaining cost to the goal. An optimal solution to the problem is found when a node n with minimal $f(n)$ value is a goal node.

To adapt A^* to the MCF planning setting, we store at each node n a set of accumulated path costs $g_i(n)$ resulting from each of the secondary cost functions $\mathcal{C}_1, \ldots, \mathcal{C}_n$, in addition to the accumulated primary cost $g_0(n)$. When a node is taken from the priority queue and expanded, generated successor nodes for which any $g_i(n) > \mathsf{B}_i$ can be immediately pruned, as all $\mathcal{C}_i$ are assumed to be non-negative, and they cannot constitute valid prefixes for solution paths.

One key optimization used in modern A^* implementations in the classical setting is duplicate detection, which allows states that are rediscovered during search to be discarded, if the new g value exceeds the cost of the path to the state that was previously found, or to be updated with a new parent, if the cost of the new path is less. In the MCF setting, care must be taken to ensure that newly discovered nodes are discarded (or replace existing nodes), only when they are dominated by (or dominate), the existing node in *all* cost dimensions. While the only necessary property of the open list from a correctness perspective is that it order nodes by increasing primary $f(n)$ value, the choice of a secondary ordering heuristic plays a role here: an ordering that causes a dominating node to be generated first and enables subsequently generated nodes to be immediately discarded as dominated results in superior performance. In our implementation of the algorithm, we therefore use an open list that orders nodes by increasing $g_i(n)$ value when their primary $f(n)$ values are the same.

Bound-Sensitive Heuristics

While any admissible heuristic can be used to guide search in MCF planning, classical planning heuristics that ignore bounds entirely are typically extremely uninformative. Consider the problem shown in Figure 1: the agent is initially at l_0, and can obtain a utility of 10 by visiting each of the locations l_1 and l_2. The costs of the actions *move*(l_0, l_1) and *move*(l_1, l_2) are both 1. In the compiled MCF version of this problem, each of the hard goals associated with the soft goals *visited*(l_*) can be achieved in two ways: with a 0-cost action that requires the location to have actually been visited, or with an action with cost 10 that has no preconditions. A naive heuristic that ignores the bound but is otherwise optimal will therefore give an estimate of 0, as both *visited*(l_1) and *visited*(l_2) can be made true if there are no limits on the agent's movement. If, however, $\mathsf{B} = 1$, the optimal $\mathcal{C}_0$ cost at l_0 is 10, as l_2 cannot be reached with cost $\leq \mathsf{B}_1$ and the agent must use the $o^{\textit{not-visited}(l_2)}$ action to achieve the associated hard goal with a cost of 10. Similarly, if $\mathsf{B} = 0$, the $\mathcal{C}_0$ cost of the optimal plan is 20, since $\mathcal{C}_1$ for all available actions exceeds the bound B. In practice, it turns out that the OSP versions of many classical planning problems have similar behavior: their state spaces are strongly connected, so any variable assignment can be achieved from any state, and classical planning heuristics that ignore bounds are no more informed than blind search.

In order to obtain estimates that take secondary cost bounds into account and can guide heuristic search towards feasible solutions, we therefore introduce *bound-sensitive heuristics*. In the following, we use $\mathbf{b} = \langle \mathbf{b}_1, \ldots, \mathbf{b}_n \rangle$ to denote a vector consisting of the available secondary cost bounds.

Definition 2 (Perfect bound-sensitive heuristic) *Given an MCF planning problem* $\Pi_{\text{MCF}} = \langle V, O, s_I, G, \mathcal{C}_0, \mathscr{C} \rangle$*, the* perfect bound-sensitive heuristic $h^*(s, \boldsymbol{b})$ *for a state* s *and cost bounds* $\boldsymbol{b}$ *is given by the minimal primary cost* $\mathcal{C}_0(\pi)$ *of a plan* π *for* s *such that* $\mathcal{C}_i(\pi) \leq \boldsymbol{b}_i$ *for* $i = 1, \ldots, n$*.*

By analogy with standard admissible heuristics, an *admissible bound-sensitive heuristic* is a non-overestimating bound-sensitive heuristic:

Definition 3 (Admissible bound-sensitive heuristic) *Given an MCF planning problem* $\Pi_{\text{MCF}} = \langle V, O, s_I, G, \mathcal{C}_0, \mathscr{C} \rangle$*, an* admissible bound-sensitive heuristic $h(s, \boldsymbol{b})$ *is a heuristic* h *such that* $h(s, \boldsymbol{b}) \leq h^*(s, \boldsymbol{b})$ *for all states* s *and cost bounds* $\boldsymbol{b}$*.*

Any classical planning heuristic that completely ignores $\mathcal{C}_i$ and B_i can be thought of as an admissible bound-sensitive heuristic that assumes $\mathbf{b} = \infty$. As the value of $\mathbf{b}$ decreases, the value of $h^*(s, \mathbf{b})$ can only increase. In general, it is useful to keep in mind the following property:

Theorem 1 *For all states* s *and cost bounds* $\boldsymbol{b}, \boldsymbol{b}'$ *such that* $\boldsymbol{b} \leq \boldsymbol{b}'$ *(where* $\leq$ *is interpreted as a pairwise comparison),* $h^*(s, \boldsymbol{b}) \geq h^*(s, \boldsymbol{b}')$*.*

Proof sketch: This follows from any plan π for s such that $\mathcal{C}_i(\pi) \leq \mathbf{b}_i$ also having the property that $\mathcal{C}_i(\pi) \leq \mathbf{b}'_i$ for $i = 1, \ldots, n$ since $\mathbf{b} \leq \mathbf{b}'$, while the opposite is not the case. ∎

Theorem 1 applied to MCF planning problems obtained from the soft goals compilations of OSP problems states that for any s, decreasing $\mathbf{b}$ *increases* $h^*(s, \mathbf{b})$, and decreases the achievable utility, since the primary cost here indicates the utility that the plan must accept as unachievable through $o^{v,\vartheta}$ actions with $\mathcal{C}_0(o^{v,\vartheta}) \geq 0$.

Bound-Sensitive h^{max}

The admissible classical heuristic h^{max} estimates the cost of a set of facts F as the cost of the most expensive fact $f \in F$,

and applies this approximation recursively to action preconditions to obtain the cost of the goal (Bonet and Geffner 2001):

$$h^{max}_{\mathcal{C}}(F,s) = \max_{f \in F} h^{max}_{\mathcal{C}}(f,s)$$

$$h^{max}_{\mathcal{C}}(f,s) = \begin{cases} 0 & f \in s \\ \min\limits_{o \in achievers(f,s)} h^{max}_{\mathcal{C}}(o,s) & otherwise \end{cases}$$

$$h^{max}_{\mathcal{C}}(o,s) = \mathcal{C}(o) + h^{max}_{\mathcal{C}}(\mathsf{pre}(o),s)$$

where $h^{max}_{\mathcal{C}}$ denotes the value of h^{max} computed with a cost function $\mathcal{C}$, and $achievers(f,s)$ denotes the set $\{o \mid f \in \mathsf{eff}(o)\}$. The h^{max} cost of a fact f that is not present in s is computed by choosing an action o from this set that achieves it with minimum possible cost. A bound-sensitive $h^{max}_{\mathbf{b}}$ can easily be obtained by replacing the set of achievers used to compute $h^{max}_{\mathcal{C}_0}$ with

$$achievers_{\mathcal{C}_0}(f,s,\mathbf{b}) = \{o \mid f \in \mathsf{eff}(o) \wedge \bigwedge_{i=1,\ldots,n} h^{max}_{\mathcal{C}_i}(o,s) \leq \mathbf{b}_i\}$$

where actions o for which any estimate $h^{max}_{\mathcal{C}_i}(o,s)$ exceeds $\mathbf{b}_i$ are not considered. Note that due to the admissibility of h^{max}, this restriction of the set of achievers is sound but not complete: it is guaranteed that any action removed from the set of achievers cannot be used in a valid plan, but there may be additional actions that cannot be achievers but are not pruned by the heuristic. In general, any admissible estimate $h^{max}_{\mathcal{C}_i}(o,s)$ could be used to compute $achievers_{\mathcal{C}_0}(f,s)$, but we have chosen h^{max} here for simplicity.

Theorem 2 *Bound-sensitive $h^{max}_{\mathcal{C}_0}$ using* any *admissible heuristic h' to compute the* achievers *set is an admissible bound-sensitive heuristic.*

Proof sketch: This follows from the fact that an admissible heuristic h' will only remove from the set of achievers those whose preconditions are provably unreachable within the bound, and the admissibility of h^{max} itself. ∎

Bound-Sensitive Merge-and-shrink

Merge-and-shrink heuristics h^{MS} are a family of abstraction heuristics that incrementally build a representation of the full state space of a problem (Helmert et al. 2014). The construction process begins with the set of transition systems induced over each state variable; at each step, two transition systems are selected to be *merged* and replaced with their synchronized product. Since the transition systems need to be represented explicitly in memory, before each merge step a *shrink* step is performed on the two selected transition systems to enforce a user-specified threshold on the size of the synchronized product. This is done by abstracting multiple states in the current representation into a single state (and thereby losing accuracy). The final output of the algorithm consists of a single abstract transition system in which multiple states and actions from the original task are mapped to a single state or transition, respectively. $h^{\text{MS}}(s)$ is then given by the cost of a shortest path from the abstract state representing s to the closest abstract goal state in the final transition system. This estimate is admissible, as the cost of the path in the final transition system cannot be larger than the cost of the path in the original problem.

To formulate a bound-sensitive variant $h^{\text{MS}}_{\mathbf{b}}$ of h^{MS}, we maintain for each transition in the abstract state space the minimum $\mathcal{C}_i$ cost for $i = 1,\ldots,n$ among all of the transitions from the original task represented by the transition. The distance $\mathcal{C}_i$ between any two abstract states s, s' then represents a non-overestimate of the secondary cost of reaching s' from s. A bound-sensitive heuristic value for a state s and bound $\mathbf{b}$ can be computed as the minimum $\mathcal{C}_0$ cost of a path π from s to an abstract goal state s_g whose $\mathcal{C}_i$ cost in the abstract state space does not exceed $\mathbf{b}_i$, for any i. The $\mathcal{C}_0$ cost of such a path can be computed with a modified version of Dijkstra's algorithm that stores secondary cost information for each node and discards nodes for which $\mathcal{C}_i > \mathbf{b}_i$ for any i.

Theorem 3 *Bound-sensitive h^{MS} is an admissible bound-sensitive heuristic.*

Proof sketch: This follows from the fact that the secondary costs used in the abstract state space are the minimums of the secondary costs $\mathcal{C}_i$ of the represented transitions in the original problem, and the proof of admissibility of standard h^{MS}. ∎

While the $h^{\text{MS}}_{\mathbf{b}}$ heuristic can be implemented by running Dijkstra's algorithm in the abstract state space for each heuristic computation, an important optimization when a single secondary cost function is present (which is the case in the compiled OSP problems that we consider) is to run Dijkstra only once during preprocessing, and compute the primary cost in the presence of different bounds on the secondary cost. This information can then be stored as a sequence of pairs $\langle\langle b_0, c_0\rangle, \ldots, \langle b_n, c_n\rangle\rangle$, where $b_0,\ldots,b_n$ is strictly increasing bounds on the secondary cost function and $c_0,\ldots,c_n$ are strictly decreasing heuristic estimates (recall Theorem 1). $h^{\text{MS}}(s,b)$ is then given by the first c_i such that $b_i \leq b$.

An important limitation that applies to both $h^{\text{MS}}_{\mathbf{b}}$ and $h^{max}_{\mathbf{b}}$ is that although they are admissible, they do not uphold certain guarantees made by their classical counterparts. While classical h^{MS} computes the optimal cost of a shortest path in the abstract state space, the bound-sensitive version is not able to compute the optimal shortest path cost in the presence of the bound, as this problem is NP-hard:

Theorem 4 *The shortest paths problem for graphs in the presence of bounds on even a* single *additional cost function is NP-hard.*

Proof sketch: By reduction from the knapsack problem (Karp 1972), modeled as a sequence of nodes with two edges between each pair of consecutive nodes. The edges in this construction represent either including or omitting an item

	25		50		75		100	
Coverage	BnB	A^*	BnB	A^*	BnB	A^*	BnB	A^*
elevators08	30	30	25	25	23	23	17	**18**
elevators11	20	20	19	19	18	18	14	**15**
miconic	96	96	65	65	55	55	50	**55**
mprime	35	35	**28**	27	24	24	19	19
mystery	29	29	**27**	26	21	21	18	18
openstacks14	**20**	19	**15**	13	7	7	3	3
parcprinter08	**17**	15	13	13	11	11	**11**	10
parcprinter11	**13**	12	9	9	7	7	6	6
parking11	**11**	10	1	1	0	0	0	0
parking14	**14**	12	4	4	0	0	0	0
pegsol08	30	30	30	30	**29**	28	27	27
pegsol11	20	20	20	20	**19**	17	17	17
pipes-tank	**35**	33	20	20	**16**	15	11	11
rovers	15	15	8	8	6	6	5	**6**
scanalyzer08	13	**14**	12	12	12	12	12	12
tetris14	17	17	14	14	**11**	10	9	9
tidybot11	20	20	20	20	**18**	17	13	13
tidybot14	20	20	18	18	**14**	13	6	6
transport08	17	17	15	15	12	**13**	11	11
transport11	15	15	11	11	8	**9**	6	6
transport14	13	**14**	9	9	9	9	7	7
trucks	**13**	12	8	8	6	6	5	5
visitall11	16	16	**12**	11	9	9	9	9
Sum other	661	661	494	494	413	413	375	375
Sum all	**1190**	1182	**897**	892	**748**	743	651	**658**

Table 1: Coverage results, BnB vs. A^* with blind heuristic.

from the final set, respectively, and their secondary and primary costs are determined by the *weight* and *value* functions for the knapsack objects. ■

In order to be polynomial, the h_b^{MS} heuristic prunes from the abstract state graph only the nodes for which the $\mathcal{C}_1$ costs of *all* possible shortest paths exceed $\mathbf{b}_1$, while allowing paths for which $\mathcal{C}_1 > \mathbf{b}_1$, as long as *some* path with $\mathcal{C}_1 \leq \mathbf{b}_1$ to the same node exists. A similar condition applies in the case of the cost-sensitive version of h^{max}, and the heuristic is not perfect in this setting even for problems with a single goal and at most a single precondition per action.

Returning to the example shown in Figure 1, we note that the cost-bounded versions of both h^{max} and h^{MS} are able to compute optimal costs for this problem. In the case of both h_b^{max} and h_b^{MS} (assuming a reasonable upper bound on the number of abstract states), this is due to the fact that there is a single path to each state in the compiled problem, and the considerations discussed above do not apply.

Experiments

We implemented our approach[1] in the Fast Downward planner (Helmert 2006), and evaluated it on a set of publicly available OSP benchmarks (Katz et al. 2019b). The set of benchmarks is taken from the International Planning Competitions of recent years, in which goal facts are replaced with utilities, and the bound set at 25%, 50%, 75%, or 100% of the cost of the optimal or best known solution to each

[1]Code available at https://github.com/emilkeyder/fd-2018-osp.

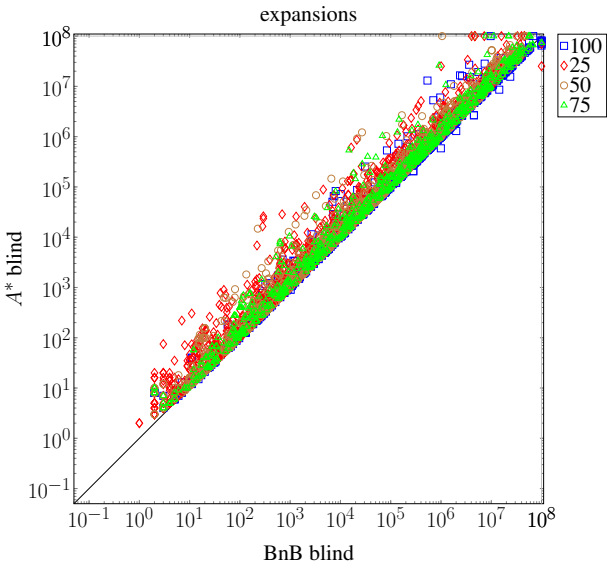

Figure 2: Expansions, BnB vs. A^* with blind heuristic.

problem. The baseline for our comparison is a blind branch-and-bound search, currently the best available configuration of heuristic search for oversubscription planning that we know of (Katz et al. 2019a). The experiments were performed on Intel(R) Xeon(R) CPU E7-8837 @2.67GHz machines, with time and memory limits of 30min and 3.5GB.

We first compare the baseline branch and bound search to our proposed approach of A^* search on the MCF compilation of the OSP task, using the blind heuristic for both algorithms. Since the compilation introduces intermediate states at which some but not all of the $o^{v,\vartheta}$ have been applied, we use a further optimization that avoids generating these nodes and applies all of the $o^{v,\vartheta}$ actions in a single step, reducing the state space to that of the original OSP task. The per-domain and overall coverage, as well as per-task node expansions comparing the two blind search approaches are shown in Table 1 and Figure 2, respectively. Overall, branch-and-bound search performs better, but only by a small margin. Thus, A^* search equipped with informative heuristics has the potential to improve over the state-of-the-art method.

Our next experiment tests this hypothesis. We compare blind A^* search to A^* using classical h^{max} and h^{MS}, as well as the two heuristics' bound-sensitive variants introduced here. For h^{MS}, we use exact bisimulation with an abstract state space threshold of $50k$ states and exact generalized label reduction (Sievers, Wehrle, and Helmert 2014), limiting the time for abstraction creation to 600 seconds. Domain-level and overall coverage comparison, as well as per-task node expansions for the various configurations and problem suites, are shown in Table 2 and Figure 3, respectively. We now report some observations from these experiments.

- Blind branch-and-bound search usually slightly outperforms blind A^* in terms of coverage, except for the 100% suite. The difference between the two may come down to the fact that A^* must do extra work in ordering the priority queue, while branch and bound search has no ordering

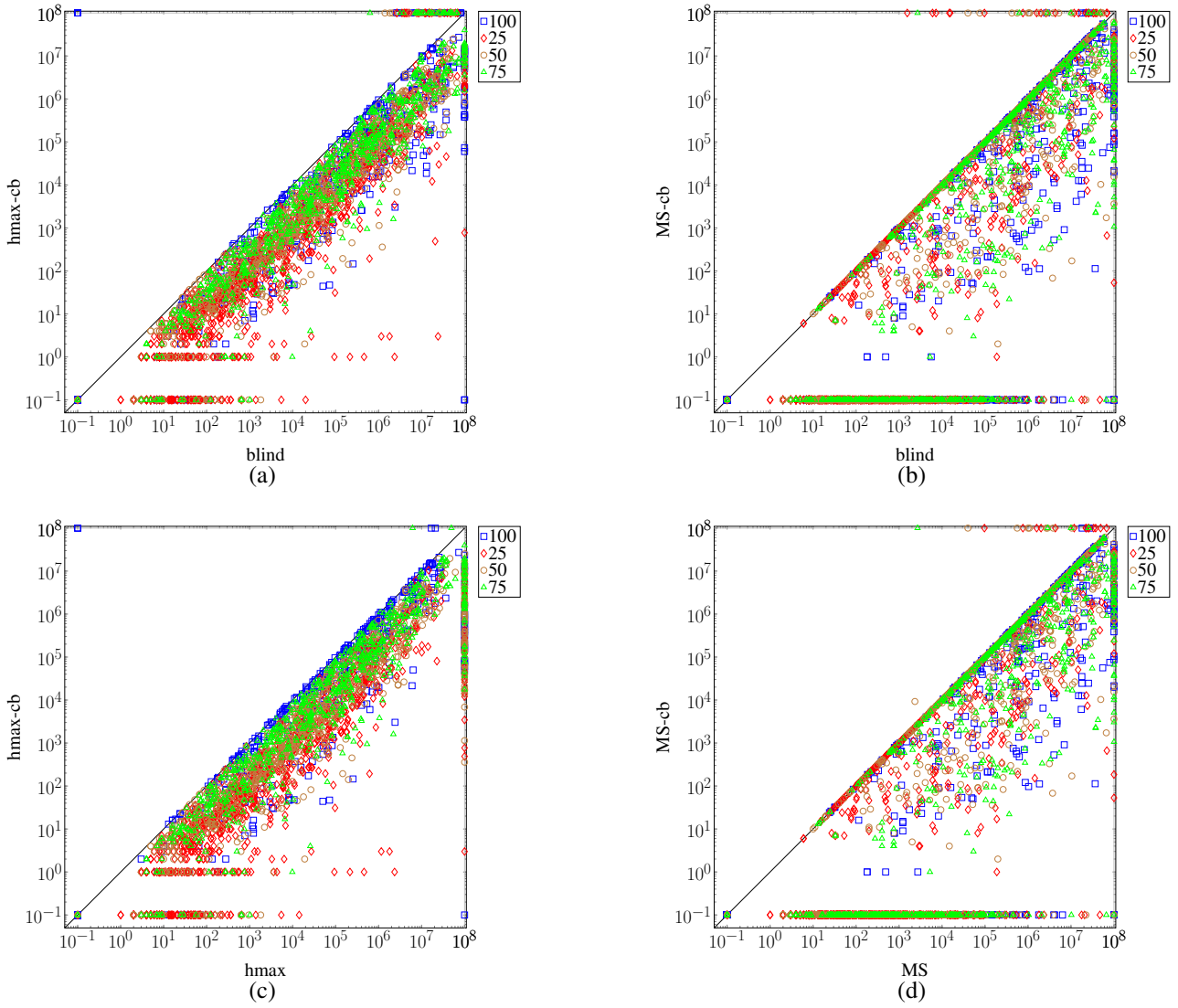

Figure 3: Expansions up to the last layer, A^* with blind heuristic vs. (a) bound-sensitive h_b^{max} and (b) bound-sensitive h_b^{MS}; A^* with (c) h_b^{max} vs h^{max} and (d) h_b^{MS} vs h^{MS}.

heuristic and can use a simple stack as its node queue.

- Blind search performs better than informed search in terms of overall coverage when bounds are low, but the effect diminishes as the bound increases and it becomes intractable to explore the full state space under the bound. Looking on the number of domains in which each configuration excels, for the 25% suite of problems, bound-sensitive h^{max} is on par with the blind branch-and-bound, both dominating all other approaches. Notably, bound-sensitive h_b^{MS} has the best overall coverage in the 50%, 75%, and 100% suites, where it solves 6, 35, and 43 more problems than blind A^*, respectively. The other 3 heuristics are generally outperformed by, or at best match the performance of blind search. Further, h_b^{MS} dominates all other approaches in the number of domains it excels on, for the 75% and 100% suites.
- Bound-sensitive heuristics are much more informative than their classical variants, sometimes decreasing expansions by orders of magnitude. Compared to non-bound-sensitive heuristics, they also almost always result in better coverage, sometimes by quite a large margin, for instance in the BLOCKSWORLD, MICONIC, and WOODWORKING domains for h_b^{MS} and the MPRIME, SOKOBAN, and PARC-PRINTER domains for h_b^{max}. In general, using bound-sensitive heuristics rather than classical ones does not impose any penalty, with the former almost always outperforming the latter.
- Limiting the resources available to h^{MS} at construction time proves to be essential to good performance in this setting. Without the 600 second time limit, there are

	25								50								75								100							
	BnB	bl	h_b^{max}	h^{max}	h_b^{MS}	h^{MS}	Total		BnB	bl	h_b^{max}	h^{max}	h_b^{MS}	h^{MS}	Total		BnB	bl	h_b^{max}	h^{max}	h_b^{MS}	h^{MS}	Total		BnB	bl	h_b^{max}	h^{max}	h_b^{MS}	h^{MS}	Total
BnB	-	**7**	**11**	**26**	**20**	**11**	**1190**		-	**4**	**13**	**27**	**12**	**8**	897		-	**6**	**16**	**23**	9	**5**	748		-	1	**15**	**19**	3	3	651
bl	2	-	11	**24**	**18**	**8**	1182		0	-	**12**	**26**	**10**	**5**	892		2	-	**16**	**22**	6	**2**	743		**4**	-	**18**	**20**	3	**3**	658
h_b^{max}	**11**	**12**	-	**28**	**15**	**14**	1170		8	10	-	**31**	10	**11**	881		13	13	-	**23**	8	11	726		11	10	-	**13**	7	10	631
h^{max}	0	0	0	-	4	2	1098		0	0	0	-	3	1	812		2	2	2	-	4	2	682		9	8	5	-	5	7	612
h_b^{MS}	7	7	9	**22**	-	7	1178		9	**10**	**12**	**30**	-	**9**	**898**		**19**	**20**	**23**	**33**	-	**18**	**778**		**18**	**17**	**28**	**30**	-	**17**	**701**
h^{MS}	0	0	8	**20**	**11**	-	1170		0	1	9	**26**	5	-	884		2	**2**	**15**	**23**	5	-	743		**7**	**3**	**18**	**21**	2	-	659

Table 2: Domain comparison of coverage for four domain suites defined by the 25%, 50%, 75%, and 100% of best known solution cost for the classical planning task as an OSP task cost bound. Branch-and-bound with blind heuristic (BnB) is compared to A^* with blind heuristic (bl) and with other considered heuristics. The values represent the number of domains in which the row configuration performs better than the column one. "Total" shows total coverage for the row configuration.

a number of problems on which the planner times out while trying to build the abstraction, and coverage drops signicantly. However, our implementation is currently unable to handle out of memory errors during the construction of the heuristic. This occurs in 225 tasks for the bound-sensitive and on 197 tasks for the non-bound sensitive versions of the heuristic. In all of these cases, the memory limit is exceeded in a matter of just a few seconds. Interestingly, blind A^* search is able to solve 35 and 15 out of these tasks, respectively. Further improvements in overall performance could therefore be obtained with an implementation that deals with memory exceeded errors more gracefully.

Finally, see Speck and Katz (2021) for a comparison to a recently proposed symbolic search based planner.

Related Work

One related area of research in the classical setting is that of *bounded-cost planning*, where the planner looks for *any* plan with (primary) cost below a given bound, similar to the treatment of the secondary cost in the OSP setting. Approaches proposed for this setting include dedicated search algorithms (Stern, Puzis, and Felner 2011) and heuristics that take into account accumulated cost and plan length at the current search node (Thayer and Ruml 2011; Haslum 2013; Dobson and Haslum 2017). These approaches work by preferentially expanding nodes in areas of the search space that are likely to have a solution under the cost bound. Optimal OSP, however, requires expanding all nodes that potentially lie on a path to state with maximal utility. Furthermore, it cannot be assumed that solutions necessarily achieve all soft goals.

Heuristics that are able to take into account bounds on secondary cost functions have also been investigated in the stochastic shortest path setting, where they were used as additional constraints in an LP-based heuristic to consider limitations on fuel or time resources (Trevizan, Thiébaux, and Haslum 2017).

The problem of Resource-Constrained Planning (RCP) is also related to the MCF setting (Nakhost, Hoffmann, and Müller 2012). RCP is of equal expressivity to MCF planning and there is a straightforward transformation from one to the other. While the approach taken by Nakhost, Hoffmann, and Müller (2012) does not guarantee optimality and therefore cannot be compared to here, a follow up work (Wilhelm, Steinmetz, and Hoffmann 2018) did consider optimality, albeit while ignoring resource consumption limitations. Thus, these approaches are somewhat equivalent to the baselines we compare to in our work.

Conclusions and Future Work

We have shown that a previously introduced compilation to multiple cost function classical planning allows the A^* algorithm to be used to solve oversubscription planning problems, with out-of-the-box classical planning heuristics. Further, we introduced a family of bound-sensitive heuristics that are much more informed than their classical counterparts in this setting. Our experiments show that this approach results in a state-of-the-art method for some bound settings and domains.

For future work, we would like to adapt existing search pruning techniques, such as partial order reduction (Alkhazraji et al. 2012) and symmetry pruning (Pochter, Zohar, and Rosenschein 2011) to MCF planning to be used with A^* search for OSP. Stubborn sets were recently defined for RCP (Wilhelm, Steinmetz, and Hoffmann 2018). It is worth investigating how the definition would carry over to MCF, and how our definition of bound-sensitive heuristics can be carried over to RCP. Further, our methods may be applicable to numeric planning problems in which the variables describe resources that are strictly decreasing and can be expressed in terms of secondary cost functions and associated bounds. Bound-sensitive heuristics could provide a principled way of reasoning about numeric variables in this context.

Another future research direction we would like to explore that builds on the methods introduced here is the use of non-admissible heuristics for satisficing OSP. The method by which bound-sensitive h^{max} is obtained is fairly general and should be equally applicable for h^{add} or general relaxed plan heuristics (Keyder and Geffner 2008). A second direction is the use of these heuristics in other planning settings in which tradeoffs must be made between different cost functions, e.g. minimizing fuel use in the presence of bounds on time or vice versa in logistics problems.

Acknowledgments

We thank Malte Helmert for his helpful comments regarding the complexity of the shortest paths problem with secondary cost bounds.

References

Alkhazraji, Y.; Wehrle, M.; Mattmüller, R.; and Helmert, M. 2012. A Stubborn Set Algorithm for Optimal Planning. In De Raedt, L.; Bessiere, C.; Dubois, D.; Doherty, P.; Frasconi, P.; Heintz, F.; and Lucas, P., eds., *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI 2012)*, 891–892. IOS Press.

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

Bonet, B.; and Geffner, H. 2001. Planning as Heuristic Search. *Artificial Intelligence*, 129(1): 5–33.

Dobson, S.; and Haslum, P. 2017. Cost-Length Tradeoff Heuristics for Bounded-Cost Search. 58.

Edelkamp, S. 2001. Planning with Pattern Databases. In Cesta, A.; and Borrajo, D., eds., *Proceedings of the Sixth European Conference on Planning (ECP 2001)*, 84–90. AAAI Press.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107.

Haslum, P. 2013. Heuristics for Bounded-Cost Search. In Borrajo, D.; Kambhampati, S.; Oddi, A.; and Fratini, S., eds., *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling (ICAPS 2013)*, 312–316. AAAI Press.

Haslum, P.; and Geffner, H. 2000. Admissible Heuristics for Optimal Planning. In Chien, S.; Kambhampati, S.; and Knoblock, C. A., eds., *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2000)*, 140–149. AAAI Press.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26: 191–246.

Helmert, M.; and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*, 162–169. AAAI Press.

Helmert, M.; Haslum, P.; Hoffmann, J.; and 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.

Karp, R. M. 1972. Reducibility among combinatorial problems. In Miller, R. E.; and Thatcher, J. W., eds., *Complexity of Computer Computations*, 85–103. Plenum Press.

Katz, M.; Keyder, E.; Pommerening, F.; and Winterer, D. 2019a. Oversubscription Planning as Classical Planning with Multiple Cost Functions. In *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS 2019)*. AAAI Press.

Katz, M.; Keyder, E.; Pommerening, F.; and Winterer, D. 2019b. PDDL benchmarks for oversubscription planning. https://doi.org/10.5281/zenodo.2576024.

Keyder, E.; and Geffner, H. 2008. Heuristics for Planning with Action Costs Revisited. In *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI 2008)*, 588–592.

Keyder, E.; and Geffner, H. 2009. Soft Goals Can Be Compiled Away. *Journal of Artificial Intelligence Research*, 36: 547–556.

Mirkis, V.; and Domshlak, C. 2013. Abstractions for Oversubscription Planning. In Borrajo, D.; Kambhampati, S.; Oddi, A.; and Fratini, S., eds., *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling (ICAPS 2013)*, 153–161. AAAI Press.

Mirkis, V.; and Domshlak, C. 2014. Landmarks in Oversubscription Planning. In Schaub, T.; Friedrich, G.; and O'Sullivan, B., eds., *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI 2014)*, 633–638. IOS Press.

Muller, D.; and Karpas, E. 2018. Value Driven Landmarks for Oversubscription Planning. In de Weerdt, M.; Koenig, S.; Röger, G.; and Spaan, M., eds., *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling (ICAPS 2018)*, 171–179. AAAI Press.

Nakhost, H.; Hoffmann, J.; and Müller, M. 2012. Resource-Constrained Planning: A Monte-Carlo Random Walk Approach. In McCluskey, L.; Williams, B.; Silva, J. R.; and Bonet, B., eds., *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*, 181–189. AAAI Press.

Pochter, N.; Zohar, A.; and Rosenschein, J. S. 2011. Exploiting Problem Symmetries in State-Based Planners. In Burgard, W.; and Roth, D., eds., *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2011)*, 1004–1009. AAAI Press.

Sievers, S.; Wehrle, M.; and Helmert, M. 2014. Generalized Label Reduction for Merge-and-Shrink Heuristics. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI 2014)*, 2358–2366. AAAI Press.

Smith, D. E. 2004. Choosing Objectives in Over-Subscription Planning. In Zilberstein, S.; Koehler, J.; and Koenig, S., eds., *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004)*, 393–401. AAAI Press.

Speck, D.; and Katz, M. 2021. Symbolic Search for Oversubscription Planning. In Leyton-Brown, K.; and Mausam, eds., *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2021)*, 11972–11980. AAAI Press.

Stern, R. T.; Puzis, R.; and Felner, A. 2011. Potential Search: A Bounded-Cost Search Algorithm. In Bacchus, F.; Domshlak, C.; Edelkamp, S.; and Helmert, M., eds., *Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling (ICAPS 2011)*, 234–241. AAAI Press.

Thayer, J. T.; and Ruml, W. 2011. Bounded Suboptimal Search: A Direct Approach Using Inadmissible Estimates. In Walsh, T., ed., *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, 674–679. AAAI Press.

Trevizan, F. W.; Thiébaux, S.; and Haslum, P. 2017. Occupation Measure Heuristics for Probabilistic Planning. In Barbulescu, L.; Frank, J.; Mausam; and Smith, S. F., eds., *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS 2017)*, 306–315. AAAI Press.

Wilhelm, A.; Steinmetz, M.; and Hoffmann, J. 2018. On Stubborn Sets and Planning with Resources. In de Weerdt, M.; Koenig, S.; Röger, G.; and Spaan, M., eds., *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling (ICAPS 2018)*, 288–297. AAAI Press.