Frontier Search and Plan Reconstruction in Oversubscription Planning

Daniel Muller

The Faculty of Industrial Engineering and Management
Technion - Israel Institute of Technology, Haifa, Israel
mullerdm@gmail.com, +972508120806, https://mullerd.webgr.technion.ac.il

Introduction

Oversubscription planning (OSP) (Smith 2004) is a problem of choosing an action sequence which reaches a state with a high utility, given a budget for total action cost. This formulation allows to handle situations with underconstrained resources, which do not allow to achieve all possible goal propositions. In optimal OSP, the task is further constrained to finding a path which achieves a state with maximal utility. Best-First-Branch-and-Bound (BFBB) is a heuristic search algorithm which is widely used for solving OSP problems. BFBB relies on an admissible utility-upper-bounding heuristic function (with budget restrictions) \( h : S \times \mathbb{R}^{0+} \to \mathbb{R} \) to estimate the true utility \( h^*(s, b) \). An incremental BFBB search algorithm with landmark-based approximations (inc-compile-and-BFBB) was proposed for OSP heuristic search (Domshlak and Mirkis 2015) to address tasks with non-negative and 0-binary utility functions. inc-compile-and-BFBB maintains the best solution so far and a set of reference states, extended with all the non-redundant value-carrying states discovered during the search. Each iteration requires search re-start in order to exploit the new information obtained along the search. Recent work presented a relative estimation of achievements with value-driven landmarks (Muller and Karpas 2018a) addressing arbitrary additive utility functions, which incrementally improves the best solution so far eliminating the need to maintain a set of reference states. This paper proposes a progressive frontier search algorithm, which alleviates the computational cost of search restart once new information is acquired. Our technique allows the new search iteration to continue from any state on the frontier of the previous search iteration, leading to improved efficiency of the search. An extended version of this abstract is available online (Muller and Karpas 2018b).

Background

We represent OSP model in a language close to SAS+ for classical planning (Bäckström and Klein 1991; Bäckström and Nebel 1995), an oversubscription planning (OSP) task is given by a sextuple \( \Pi = \langle V, s_0, u; O, c, b \rangle \), where \( V = \{v_1, \ldots, v_n\} \) is a finite set of finite-domain state variables, with each complete assignment to \( V \) representing a state, and \( S = \text{dom}(v_1) \times \cdots \times \text{dom}(v_n) \) being the state space of the task; \( s_0 \in S \) is a designated initial state; \( u \) is an efficiently computable state utility function \( u : S \to \mathbb{R} \); \( O \) is a finite set of actions, with each action \( o \in O \) being represented by a pair \( \langle \text{pre}(o) , \text{eff}(o) \rangle \) of partial assignments to \( V \), called preconditions and effects of \( o \) respectively; \( c : O \to \mathbb{R}^{0+} \) is an action cost function; \( b \in \mathbb{R}^{0+} \) is a cost budget allowed for the task. An assignment of a variable \( v \) to value \( d \) is denoted by \( \langle v/d \rangle \) and referred as a proposition. For a partial assignment \( p \to V \), let \( \mathcal{V}(p) \subseteq V \) denote the subset of variables instantiated by \( p \), and, for \( v \in \mathcal{V}(p) \), \( p[v] \) denote the value provided by \( p \) to the variable \( v \). Action \( o \) is applicable in a state \( s \) if \( s[e] = \text{pre}(o)[v] \) for all \( v \in \mathcal{V}(\text{pre}(o)) \). Applying \( o \) changes the value of each \( v \in \mathcal{V}(\text{eff}(o)) \) to \( \text{eff}(o)[v] \), and the resulting state is denoted by \( s[o] \). A sequence of actions \( \langle o_1, \ldots, o_m \rangle \) denoted by \( \pi \), called a plan for \( s \) if it is applicable in \( s \) and \( c(\pi) \leq b \). We assume an arbitrary additive utility function with multi-valued variables, defined as \( u(s) = \sum_{\langle v/d \rangle \in s} u_v(d) \), with \( u_v(d) \in \mathbb{R} \) for all variable-value pairs \( \langle v/d \rangle \).

OSP Frontier Search

Let \( \Pi = \langle V, s_0, u; O, c, b \rangle \) be an OSP task, given a set of plans \( \pi_1, \ldots, \pi_n \) for \( \Pi \) with the corresponding set of end-states \( S_{\text{ref}} = \{s_1, \ldots, s_n\} \) of \( \Pi \), \( s_i = s_0 \pi_i \). Let a plan \( \pi \) with end-state \( s \) be a newly discovered plan that achieves something beyond what \( \pi_1, \ldots, \pi_n \) already achieve. To exploit the information about the newly discovered valuable state, we must restart the search from the initial state. To avoid re-exploration of states that have been discovered in current iteration, we modify \( \Pi \) to hold the acquired information compactly for next iteration.

Definition 1. Given an OSP task \( \Pi = \langle V, s_0, u; O, c, b \rangle \), a set of states and costs \( S_{\text{front}} = \{s_1, c_1), \ldots, (s_n, c_n)\} \) of \( \Pi \), where \( \{s_1, \ldots, s_n\} \) are the states at the frontier and \( \{c_1, \ldots, c_n\} \) are the budgets invested in reaching the respective frontier states from the initial state \( s_0 \), the frontier-compilation of \( \Pi \) is an OSP task \( \Pi = \langle V, s_0, u; O, c, b \rangle \) with \( V = \{v' \mid v \in V\} \) where \( \text{dom}(v') = \text{dom}(v) \cup \{\bot\} \), \( s_0 = s_{\text{dum}} = \{\langle v/\bot \rangle \mid \forall v \in V\} \), \( u \) is the same as \( u \), except that \( u(\langle v/\bot \rangle) = -\infty \) for each \( v \in V \); \( O = O \cup \bigcup_{i=1}^{n} \text{ref}_{s_i} \) with \( \text{pre}(o_s) = \{\langle v/\bot \rangle \mid \forall v \in V\} \), \( \text{eff}(o_s) = s_i \), \( c(o_s) = c_i \),
Figure 1: An example of prog-inc-compile-and-BFBB procedure, with (a)-(c) depicting snapshots of the search at the end of each iteration. Undiscovered states are uncolored; discovered ones are green; the frontier of each iteration is circles in red; (a) shows the graphical skeleton of the search space of each iteration. (b) and (c) depict the relevant information for reconstruction when the solution is reached.

\[
c(\alpha) = c(\alpha), \quad b = b.
\]

In plain words, frontier-compilation extends \(\Pi\) with dummy initial state \(s_{\text{dummy}}\), and a set of dummy operators \(o_{s_1}, \ldots, o_{s_j}\), which applicable only at \(s_{\text{dummy}}\). For each plan \(\pi_j\) and state \(s_j\), dummy operator \(o_{s_j}\) constructed to achieve \(s_j\) at total cost of \(\pi_j\), \(s_{\text{dummy}}[o_{s_j}] = s_j\), and the cost \(c(o_{s_j}) = c(\pi_j)\). This modification preserves the semantics of \(\Pi\) with regard to states that not discovered yet, while the growth in the description of the obtained task is linear in the size of the memory that is already consumed by the search. We then perform value driven landmarks compilation (Muller and Karpas 2018a) on the frontier compilation of \(\Pi\). The extracted set of value driven landmarks is compiled into \(\Pi\), obtaining a new OSP task \(\Pi_L\) integrating the direct “guidance” from initial state towards the frontier of the last iteration break point. This allows to avoid restart the search from scratch and re-generate nodes, as it occur in the inc-compile-and-BFBB procedure, leading to an improved runtime efficiency of the search.

**Theorem 1.** Let \(\Pi = \langle V, s_0, u; O, c, b \rangle\) be an OSP task, \(S_{\text{front}} = \{s_1, \ldots, s_n\}\) be a subset of \(\Pi\)'s states comprising the search frontier at node expansion \(i\), \(\{c_1, \ldots, c_n\}\) be the costs of the respective plans towards the frontier nodes, and \(\Pi\) be the respective frontier-compilation of \(\Pi\). For any optimal plan \(\pi\) for \(\Pi\), revealed at node expansion \(i \geq t\) with cost of \(c(\pi) = b', \) where \(b' \leq b\), and \(u(\pi) = \alpha\), there exists a plan \(\bar{\pi}\) for \(\Pi\), such that \(c(\pi) = c(\bar{\pi}) = b'\) and \(u(\pi) = u(\bar{\pi}) = \alpha\). Correspondingly for each \(\bar{\pi}\) for \(\Pi\) there exists a plan \(\pi\) for \(\Pi\).

The frontier is a snapshot of the OPEN list with additional details of spent budget to achieve each node. The proof of correctness of frontier search is by the construction of \(S_{\text{front}} := S_{\text{front}} \cup \{(s(n), g(n))\}\) and the anytime “output improvement” property of the BFBB forward search: BFBB maintains the best-so-far solution, and prunes all branches that promise value lower or equal to \(u(\pi)\). All the generated nodes \(n\) with cost-so-far \(g(n)\) higher than the problem’s budget \(b\) are also immediately pruned. With each update of best so far solution \(n^*\), it is added to \(S_{\text{front}}\). By maintaining for each node on the frontier a respective best so far budget spent to reach it, for any optimal solution \(\pi^*\) of \(\Pi\), there is a corresponding solution \(\bar{\pi}\) in \(\Pi\).

We introduce a **progressive frontier search with plan reconstruction** algorithm, called prog-inc-compile-and-BFBB, which dynamic depicted in Figure 1. prog-inc-compile-and-BFBB exploits Theorem 1 to directly extend inc-compile-and-BFBB with a continuous progression procedure to avoid restart from scratch. At each iteration \(i\), prog-inc-compile-and-BFBB generates a search tree \(t_{\text{tree}_i}\) from \(\text{front}_{i-1}\) to \(\text{front}_i\), as depicted in Figure 1 (a). Once the end-state of the optimal plan is discovered, we need to reconstruct the corresponding plan. While this step is straightforward in inc-compile-and-BFBB, in prog-inc-compile-and-BFBB it requires maintaining the trees \(\{t_{\text{tree}_i}\}\) generated during the search in order to backtrack the plan. At the same time, the overall size of the tree set \(\{t_{\text{tree}_i}\}\) roughly equals the size of the single search tree that is generated and maintained by the inc-compile-and-BFBB. At each iteration of prog-inc-compile-and-BFBB the frontier defines a cross-line in the graph, illustrated in Figure 1 (b). This allows us to use (Korf and Zhang 2000; Korf et al. 2005) technique for plan reconstruction. Figure 1 (c) aims to show the backtracking effort and information. For each node on \(\text{front}_i\) we hold the coordinates of its ancestor on \(\text{front}_{i-1}\), this is adequate to backtrack a plan.

**References**


