The Merge-and-Shrink Planner: Bisimulation-based Abstraction for Optimal Planning

Raz Nissim
Ben-Gurion University
Beer-Sheva, Israel
raznis@cs.bgu.ac.il

Jörg Hoffmann
INRIA
Nancy, France
joerg.hoffmann@inria.fr

Malte Helmert
University of Freiburg
Freiburg, Germany
helmert@informatik.uni-freiburg.de

Abstract

Merge-and-shrink abstraction is a general approach to heuristic design whose key advantage is its capability to make very fine-grained choices in defining abstractions. The Merge-and-shrink planner uses two different strategies for making these choices, both based on the well-known notion of bisimulation. The resulting heuristics are used in two sequential runs of A* search.

Introduction

Many optimal planning systems are based on state-space search using A* and admissible heuristics. Merge-and-shrink abstraction (Dräger *et al.* 2006; Helmert *et al.* 2007), short M&S, uses solution distances in a smaller, *abstract* state space to deliver a consistent and admissible heuristic function.

The abstract state space is built in an incremental fashion, starting with a set of atomic abstractions corresponding to individual variables, then iteratively *merging* two abstractions – replacing them with their synchronized product – and *shrinking* them – aggregating pairs of states into one. Thus, despite the exponential size of the state space, M&S allows to select individual pairs of states to aggregate. This freedom in abstraction design comes with significant advantages. M&S dominates most other known frameworks for computing admissible planning heuristics: for any given state, it can with polynomial overhead compute a larger lower bound (Helmert and Domshlak 2009).

The M&S planner employs two different shrinking strategies, which choose the states to aggregate using the notion of bisimulation. In this paper we briefly describe bisimulation and label reduction, and conclude with a detailed description of the M&S planner.

Background

Our approach is based on the notion of *bisimulation*, a well-known criterion under which an abstract state space "exhibits the same observable behavior" as the original state space (Milner 1990). Two states s, t are bisimilar if: (1) they agree on whether or not the goal is true; and (2) every transition label, i.e., every planning operator, leads into the same abstract state from both s and t. If we aggregate only bisimilar states during M&S, then the heuristic is guaranteed to be perfect. However, bisimulations are exponentially big even in trivial examples. Our key observation is that, for the purpose of computing a heuristic, we can relax bisimulation significantly without losing any information. Namely, we do not need to distinguish the transition labels. Such a *fully label-reduced* bisimulation still preserves solution distance, while often being exponentially smaller.

Unfortunately, while full label reduction does not affect solution distances per se, its application within the M&S framework is problematic. The merging step, in order to synchronize transitions, needs to know which ones share the same label. We tackle this by using *partial* label reductions, ignoring the difference between two labels only if they are equivalent for "the rest" of the M&S construction. We thus obtain, again, a strategy that guarantees to deliver a perfect heuristic.

Even label-reduced bisimulations are sometimes too big, thus for practicality one needs a strategy to approximate further if required. The M&S planner uses two such strategies, each relaxing the strict rules of bisimulation in a different way.

For more details on bisimulation, label reduction, and using their combination to create perfect heuristics in polynomial time in some planning domains, we refer to a conference paper (forthcoming).

The M&S Strategies

The merging strategy we use is linear (meaning only one non-atomic abstraction is maintained), and follows Fast-Downward's "level heuristic" (Helmert 2006). This orders variables so that those "closest to the root of the causal graph" go first (this is beneficial for operator projection because the most influential variables are projected away earlier on).

Our planner uses two shrinking strategies, each having different strengths and weaknesses. After experimenting with many shrinking strategies, we did not find one that greatly outperformed the others. Our choice of strategies for our planner was therefore guided by the relatively high variance of tasks solved by the two. Since these strategies are the core feature of the planner, we describe them in some more detail in what follows.

The Greedy Bisimulation Shrinking Strategy

The bisimulation shrinking strategy computes the coarsest bisimulation, and in the shrinking step, aggregates only bisimilar (abstract) states. In most benchmark domains, however, coarsest bisimulations are still large even under operator projection. Greedy bisimulation is a relaxed variant of bisimulation, which demands the bisimulation property only for transitions $s \to s'$ where the abstract goal distance from s is at most as large as the abstract goal distance from s'. This relaxation forfeits the guarantee of providing a perfect heuristic.

The **greedy bisimulation shrinking strategy** therefore computes the coarsest greedy bisimulation using partial label reduction, aggregating only greedily bisimilar states. Since the (greedy) bisimulation strategy is given no limit on the size of the abstraction, the actual abstraction size depends only on the size of the coarsest greedy bisimulation. We observed that using greedy bisimulation dramatically reduces abstraction size, increasing the number of cases where the abstraction can be built completely. In fact, when using the bisimulation strategy, the abstraction was built completely only in 203 out of 876 IPC tasks we experimented on. Using greedy bisimulation brings this number up to 795. This reduction in size also improves speed, making the abstraction much faster to compute.

The DFP-gop Shrinking Strategy

Motivated by the size of bisimulations, Dräger et al. (2006) propose a more approximate shrinking strategy that we will call the **DFP shrinking strategy**. When building the coarsest bisimulation, the strategy keeps separating states until the size limit N is reached. The latter may happen before a bisimulation is obtained, in which case we may lose information. The strategy prefers to separate states close to the goal, thus attempting to make errors only in more distant states where the errors will hopefully not be as relevant.

The **DFP-gop** shrinking strategy ("g" stands for greedy, "op" for operator projection) enhances the DFP strategy in two ways. First, partial label reduction is used when computing the coarsest bisimulation. Second, if bisimulation breaks the abstraction size limit, the greedy coarsest bisimulation is used to select which states to aggregate. For the abstraction size limit, we chose to set $N = 200,000$. Because DFP-gop aggregates only bisimilar states as long as N is not reached, its high value allows computation of perfect heuristics, in cases where there exist sufficiently small coarsest bisimulations. In all 20 tasks of the IPC benchmark domain Gripper, for example, the final abstraction computed by this strategy is a bisimulation of the original search space, and therefore provides the perfect heuristic.

The high N value comes at a cost – computing the abstraction is slow and requires much memory. Out of the 876 tasks experimented on, only in 490 was the final abstraction computed without running out of time/memory.

The Planner

The M&S planner is implemented on top of the Fast Downward planning system. For further information on Fast Downward's PDDL-to-finite-domain translator, please refer to the paper by Helmert (2009). For details regarding how M&S abstractions are used in the search process, refer to the paper by Helmert et al. (2007). Finally, for general information about the planner, we refer the reader to its original description (Helmert 2006).

The Hybrid Implementation

In order to take advantage of the strengths of both strategies, our planner is designed to divide the given time limit between two sequential runs, using $\frac{4}{9}$ of the available time for the greedy bisimulation shrinking strategy, followed – if no solution is found – with DFP-gop for the remaining time. (The value of $\frac{4}{9}$ was determined experimentally based on data for IPC 1998–2008 benchmarks.) In each run of the planner, the M&S abstraction is computed according to the strategy, and A* search is performed using solution distances in the abstraction as heuristic values.

We chose the hybrid implementation for two reasons. First, cutting the time limit had little effect on coverage results for the two strategies. This allowed us to take advantage of two different M&S strategies. Second, we tried computing both abstractions and using the maximal of the two in one search run, but this turned out to be too costly both in time and in memory. In most cases, construction of the two abstractions either exceeded the 30 minute time limit or the 2GB memory limit.

References

Klaus Dräger, Bernd Finkbeiner, and Andreas Podelski. Directed model checking with distance-preserving abstractions. In *Proc. SPIN 2006*, pages 19–34, 2006.

Malte Helmert and Carmel Domshlak. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proc. ICAPS 2009*, pages 162–169, 2009.

Malte Helmert, Patrik Haslum, and Jörg Hoffmann. Flexible abstraction heuristics for optimal sequential planning. In *Proc. ICAPS 2007*, pages 176–183, 2007.

Malte Helmert. The Fast Downward planning system. *JAIR*, 26:191–246, 2006.

Malte Helmert. Concise finite-domain representations for PDDL planning tasks. *AIJ*, 173:503–535, 2009.

Robin Milner. Operational and algebraic semantics of concurrent processes. In *Handbook of TCS*, pages 1201–1242. 1990.