Fast Downward Dead-End Pattern Database

Florian Pommerening and **Jendrik Seipp**
University of Basel
Basel, Switzerland
{florian.pommerening,jendrik.seipp}@unibas.ch

This paper describes our submission to the Unsolvability International Planning Competition 2016. It uses a *dead-end pattern database* to prune states in a breadth-first search.

Pattern databases (PDBs) (Culberson and Schaeffer 1998; Edelkamp 2001) are usually computed by projecting a planning task onto a subset of its variables (the *pattern*). For every abstract state (i.e., partial state defined on the variables in the pattern) the perfect goal distance in the projection is computed and stored. If an abstract state has no path to an abstract goal in the projection, any concrete state consistent with it cannot have a path to a goal state either. If we reach such a state during the search, it can be pruned.

One simple way to use PDBs for detecting unsolvability is to compute PDBs for a collection of patterns and then use these PDBs for pruning states during a search in the transition system of the original planning task: for every encountered state, retrieve the heuristic value of all PDBs; if any of them is ∞, prune the state.

However, all entries other than ∞ in the PDB can never be used for pruning. Likewise, abstract states that are unreachable in the abstraction are unreachable in the original task and can also never be used for pruning. *Dead-end pattern databases* thus consider only abstract states from a PDB that are reachable in the abstraction and have an infinite goal distance. Viewing each such abstract state as a partial state, we end up with a set of partial states. Any concrete state that is consistent with any partial state in the set can be pruned.

During a preprocessing step, we compute a collection of patterns and generate the PDB for each pattern. After constructing each PDB, we add the partial states that can potentially lead to pruning to our collection of dead ends and destroy the PDB again, so we only have one complete PDB and our growing collection of dead ends in memory at all times. We limit time and memory spent in the preprocessing phase and start searching once the limits are reached or all patterns in our collection have been handled. If any of the partial states is consistent with the initial state, we can stop the preprocessing early and immediately report the task as unsolvable.

The pattern collection we used for the IPC systematically computes all patterns of a certain size. We restrict our attention to *interesting* patterns as defined by Pommerening, Röger, and Helmert (2013). Once all patterns of one size are handled, we continue with the next larger size and repeat this process until either

- the time limit of 900 seconds is reached, or
- the memory limit of 10 million partial states stored in the dead-end PDB is reached, or
- a partial state consistent with the initial state is found, or
- no larger interesting pattern exists.

We implemented dead-end PDBs as a heuristic in the Fast Downward planning system (Helmert 2006) and use it to prune a simple breadth-first search. To efficiently store the set of partial states, we use a match tree data structure, similar to the way the successor generator is stored in Fast Downward. Each inner node of the match tree corresponds to one variable and has a child for each value of the variable and one additional child for a "don't care" value. Leaves determine whether the path leading to them represents a dead-end. A new partial state p can be added to the match tree by following the correct value successor for every variable on which p is defined and the "don't care" successor for other variables until a leaf is reached. If that leaf denotes a dead end, a more general partial state already is contained in the match tree. Otherwise, the leaf is replaced with a sequence of nodes for all remaining variables in the domain of p followed by a leaf denoting a dead-end. A concrete state can be tested against all partial states in the match tree by always following both the matching value successor and the "don't care" successor. If a leaf denoting a dead end is found, the state can be pruned.

Acknowledgments

We would like to thank all Fast Downward contributors, and Malte Helmert in particular.

References

Culberson, J. C., and Schaeffer, J. 1998. Pattern databases. *Computational Intelligence* 14(3):318–334.

Edelkamp, S. 2001. Planning with pattern databases. In *Proc. ECP 2001*, 84–90.

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

Pommerening, F.; Röger, G.; and Helmert, M. 2013. Getting the most out of pattern databases for classical planning. In *Proc. IJCAI 2013*, 2357–2364.