Width and Inference Based Planners: SIW, BFS(f), and PROBE

Nir Lipovetzky
University of Melbourne
Melbourne, Australia
@unimelb.edu.au

Miquel Ramirez
RMIT University
Melbourne, Australia
@rmit.edu.au

Christian Muise
University of Melbourne
Melbourne, Australia
@unimelb.edu.au

Hector Geffner
ICREA & Universitat Pompeu Fabra
Barcelona, SPAIN
@upf.edu*

Introduction

We have entered the planners *SIW*, $BFS(f)$, and $PROBE$ to the *agile-track* of the 2014 International Planning Competition, and an anytime planner for the *satisficing track* that runs both *SIW* and $BFS(f)$. SIW and BFS(f) are planners that make use of width-considerations (Lipovetzky and Geffner 2012), while PROBE is a standard best-first search planner that before expanding a node, it throws a probe that either reaches the goal or terminates in low polynomial time (Lipovetzky and Geffner 2011).

The basic building block of *SIW* is the Iterative Width Procedure (*IW*) for achieving atomic goals, that runs in time exponential in the problem width by performing a sequence of pruned breadth first searches. The planner $BFS(f)$ integrates a *novelty* measure borrowed from *IW* with helpful-actions, landmarks and delete-relaxation heuristics in a Greedy Best-Fist search.

All planners are based in a new toolkit for automated planning () aiming at [.....]

In the following sections we introduce the basic notions of the algortihms and the implementation. We assume a STRIPS problem $P = \langle F, I, O, G\rangle$, where F is the set of atoms, I is the set of atoms characterizing the initial state, O is the set of actions, and G is the set of goal atoms.

Iterated Width Search

The algorithm *Iterated Width* or *IW* consists of a sequence of calls *IW*(i) for $i = 0, 1, \ldots, |F|$ until the problem is solved. Each iteration *IW*(i) is a breadth-first search that prunes right away states that do not pass a *novelty* test; namely, for a state s in *IW*(i) *not* to be pruned there must be a tuple t of at most i atoms such that s is the first state generated in the search that makes t true. The time complexities of *IW*(i) and *IW* are $O(n^i)$ and $O(n^w)$ respectively where n is $|F|$ and w is the problem width. The width of existing domains is low for atomic goals, and indeed, 89% of the benchmarks can be solved by *IW*(2) when the goal is set to any of the atoms in the goal (Lipovetzky and Geffner 2012). The width of the benchmark domains with conjunctive goals, however, is not low in general, yet such problems can be serialized.

The algorithm *Serialized Iterative width* or *SIW* uses *IW* for serializing a problem into subproblems and for solving the subproblems. Basically, SIW uses IW to achieve one atomic goal at a time, greedily, until all atomic goals are achieved jointly. In between, atomic goals may be undone, but after each invocation of IW, each of the previously achieved goals must hold. SIW will thus never call IW more than $|G|$ times where $|G|$ is the number of atomic goals. SIW compares surprisingly well to a baseline heuristic search planner based on greedy best-first search and the h_{add} heuristic (Bonet and Geffner 2001), but doesn't approach the performance of the most recent planners. Still, *SIW* is able to compete well in domains that accept simple serializations and the induced problems can be solved by *IW*(i) with $i = 1, \ldots, 3$.

Novelty Best-First Search

While the blind-search *SIW* procedure competes well with a greedy best-first planner using the additive heuristic, neither planner is state-of-the-art. Since state-of-the-art performance is important in classical planning, we show next that it is possible to deliver such performance by integrating the idea of novelty that arises from width considerations, with known techniques such as helpful actions, landmarks, and heuristics. For this we switch to a *plain forward-search best-first planner* guided by an evaluation function $f(n)$ over the nodes n given by

$$f(n) = novelha(n) \quad (1)$$

where $novelha(n)$ is a measure that combines novelty and helpful actions, as defined below. In addition, ties are broken lexicographically by two other measures: first, $usg(n)$, that counts the number of subgoals not yet achieved up to n, and second, $h_{add}(n)$, that is the additive heuristic.

The subgoals are the problem landmarks (Hoffmann, Porteous, and Sebastia 2004) derived using a standard polynomial algorithm over the delete-relaxation (Zhu and Givan 2003; Keyder, Richter, and Helmert 2010).

The count $usg(n)$ is similar to the landmark heuristic in LAMA (Richter, Helmert, and Westphal 2008), simplified a bit: we use only atomic landmarks (no disjunctions), sound orderings, and count a top goal p as achieved when goals q that must be established before q have been achieved (Lipovetzky and Geffner 2011).

The $novelha(n)$ measure combines the novelty of n and whether the action leading to n is helpful or not (Hoffmann and Nebel 2001). The novelty of n is defined as the size of the smallest tuple t of atoms that is true in n and false in *all previously generated nodes n' in the search with the same number of unachieved goals* $usg(n') = usg(n)$. Basically, nodes n and n' in the search with different number of unachieved goals, $usg(n) \neq usg(n')$, are treated as being about different subproblems, and are not compared for determining their novelty. The novelty of a node $novel(n)$ is computed approximately, being set to 3 when it's neither 1 nor 2. Similarly, if $help(n)$ is set to 1 or 2 according to whether the action leading to n was helpful or not, then $novelha(n)$ is set to a number between 1 and 6 defined as

$$novelha(n) = 2[novel(n) - 1] + help(n) \; . \qquad (2)$$

That is, $novelha(n)$ is 1 if the novelty of n is 1 and the action leading to n is helpful, 2 if the novelty is 1 and the action is not helpful, 3 if the novelty is 2 and the action is helpful, and so on. Basically, novel states (lower $novel(n)$ measure) are preferred to less novel states, and helpful actions are preferred to non-helpful, with the former criterion carrying more weight. Nodes generated by non-helpful actions delay the evaluation of $h_{adad}(n)$. Once again, the criterion is simple and follows from performance considerations.

PROBE: The Planner

PROBE is a complete, standard greedy best first search (GBFS) STRIPS planner using the standard additive heuristic (Bonet and Geffner 2001) (h_{ff} if conditional effects are present (Hoffmann and Nebel 2001)), with just *one change*: when a state is selected for expansion, it first launches a *probe* from the state to the goal. If the probe reaches the goal, the problem is solved and the solution is returned. Otherwise, the states expanded by probe are added to the open list, and control returns to the GBFS loop. *The crucial and only novel part in the planning algorithm is the definition and computation of the probes* (Lipovetzky and Geffner 2011).

PROBE is based on an early-version of the automated planning toolkit that supports the implementation details of the width-based algorithms. The only difference with respect to PROBE-IPC7 is that the anytime procedure is disabled, as we are only concerned with the first solution.

Automated Planning Toolkit

. . . . .

Miquel, Christian, here we can explain the implementation details

Experiments

We call the resulting best-first search planner, BFS(f), and compare it with three state-of-the-art planners: FF, LAMA, and PROBE (Hoffmann and Nebel 2001; Richter, Helmert, and Westphal 2008; Lipovetzky and Geffner 2011).[1] Like LAMA, BFS(f) uses delayed evaluation, a technique that is useful for problems with large branching factors (Richter and Helmert 2009).

. . . . I can generate a table with results over the last IPC taking 5min max, (agile track) and keeping just the time score.

On the other hand, runing the 30min anytime-algorithm will take ages..., and we should run lama-anytime as well for compare comparison. we may skip this, unless you think it's really meaningful...

[1] FF is FF2.3, while PROBE and LAMA are from the 2011 IPC.

Discussion

. . . .

Acknowledgments

References

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129:5–33.

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

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered landmarks in planning. *Journal of Artificial Intelligence Research* 22(1):215–278.

Keyder, E.; Richter, S.; and Helmert, M. 2010. Sound and complete landmarks for and/or graphs. In *Proceedings of the Nineteenth European Conference on Artificial Intelligence (ECAI 2010)*, 335–340.

Lipovetzky, N., and Geffner, H. 2011. Searching for plans with carefully designed probes. In *Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling (ICAPS 2011)*, 154–161.

Lipovetzky, N., and Geffner, H. 2012. Width and serialization of classical planning problems. In *Proceedings of the Twentieth European Conference on Artificial Intelligence (ECAI 2012)*, 540–545.

McDermott, D. 1996. A heuristic estimator for means-ends analysis in planning. In *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems (AIPS-96)*.

Richter, S., and Helmert, M. 2009. Preferred operators and deferred evaluation in satisficing planning. In *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research* 39:122–177.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks revisited. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI 2008)*, 975–982.

Zhu, L., and Givan, R. 2003. Landmark extraction via planning graph propagation. In *ICAPS 2003 Doctoral Consortium*, 156–160.