Searching with Probes: The Classical Planner PROBE

Nir Lipovetzky
DTIC Universitat Pompeu Fabra
Barcelona, SPAIN
nir.lipovetzky@upf.edu

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

Background

Heuristic search has been the mainstream approach in planning for more than a decade, with planners such as FF, FD, and LAMA being able to solve problems with hundreds of actions and variables in a few seconds (Hoffmann and Nebel 2001; Helmert 2006; Richter and Westphal 2010). The basic idea behind these planners is to search for plans using a search algorithm guided by heuristic estimators derived automatically from the problem (McDermott 1996; Bonet and Geffner 2001). State-of-the-art planners, however, go well beyond this idea, adding a number of techniques that are specific to planning. These techniques, such as helpful actions and landmarks (Hoffmann and Nebel 2001; Hoffmann, Porteous, and Sebastia 2004; Richter, Helmert, and Westphal 2008), are designed to exploit the *propositional structure* of planning problems; a structure that is absent in traditional heuristic search where states and heuristic evaluations are used as *black boxes*. Moreover, new search algorithms have been devised to make use of these techniques. FF, for example, triggers a best-first search when an incomplete but effective greedy search (enforced hill climbing) that uses helpful actions only, fails. In FD and LAMA, the use of helpful or preferred operators is not restricted to the first phase of the search, but to one of the open lists maintained in a multi-queue search algorithm. In both cases, dual search architectures that appeal either to two successive searches or to a single search with multiple open lists, are aimed at solving fast, large problems that are simple, without giving up completeness on problems that are not.

PROBE

The planner PROBE implements a new dual search architecture for planning that is based on the idea of *probes*: single action sequences computed without search from a given state that can quickly go deep into the state space, terminating either in the goal or in failure.

PROBE is a complete, standard greedy best first search (GBFS) STRIPS planner using the standard additive heuristic (Bonet and Geffner 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

The main contribution in PROBE is the design of these probes. A probe is an action sequence computed greedily from a seed state for achieving a *serialization of the problem subgoals* that is computed dynamically along with the probe. The next subgoal to achieve in a probe is chosen among the first unachieved landmarks that are *consistent*. Roughly, a subgoal that must remain true until another subgoal is achieved, is consistent, if once it is made true, it does not have to be undone in order to make the second subgoal achievable. The action sequence to achieve the next subgoal uses standard heuristics and helpful actions, while maintaining and enforcing the reasons for which the previous actions have been selected in the form of *commitments* akin to causal links.

Probes are described in (Lipovetzky and Geffner 2011). As shown in that work, a single probe from the initial state manages to solve by itself 683 out of 980 problems from previous IPCs, a number that compares well with the 627 problems solved by FF in EHC mode, with similar times and plan lengths. Moreover, when a probe is launched from each expanded state in a standard greedy best first search informed by the additive heuristic, the number of problems solved jumps to 900 (92%), which compares well with 827 problems solved by FF (84%), and the 879 problems solved by LAMA (89%). In this note we focus on the changes made to PROBE to adapt it to the Int. Planning Competition 2011.

Improving the First Solution: Anytime Search

In the IPC, planners are given a time window and are rewarded when they compute good quality solutions. Since the time window is often much larger than the time required to find a solution, the IPC version of PROBE follows LAMA in trying to compute a plan fast, and then using the rest of the time to iteratively improve the best plan found so far. The first part is achieved by using PROBE as described in (Lipovetzky and Geffner 2011), i.e. by performing a Greedy Best First Search with probes. The second part in turn is achieved by iteratively triggering a Weighted A* search

without probes, with a reduced weight W on the heuristic term, and by keeping the cost of the best solution as a bound so that plan prefixes whose cost does not improve the bound are pruned. The heuristic used by the WA* phase in PROBE is given by the size of the 'cost sensitive relaxed plan heuristic', which is given by the size of the relaxed plan as produced by the additive heuristic (Keyder 2010),

Dealing with Non-Uniform Costs

The direct approach of replacing length-based heuristics by cost-based heuristics when plan cost is different than plan length is known to run into a problem: if length estimates are ignored, the coverage over many domains is reduced (Richter, Helmert, and Westphal 2008; Keyder 2010). In order to avoid this problem, PROBE treats costs in two ways: in the first stage, for finding the first solution (GBFS with probes), action costs are ignored (i.e., they are all taken to be 1), while in the second stage (WA*), they are taken into account. We found that some problems could be solved in this manner that could not be solved if the real action costs were used in both stages.

An important issue appears with the presence of zero cost actions that can lead to heuristic plateaus in which the application of such operators does not decrease the cost to the goal. In order to avoid these situations, we added a base cost of 0.01 to all zero cost actions (Keyder 2010).

When Probes Fail

In most classical benchmarks a single probe suffices to find a solution, suggesting that most problems admit good landmark serializations On the other hand, in problems with no perfect serializations, such as Sokoban or the 8-puzzle, too many probes turn out to be needed to find a solution, something which rather than boosting the performance of the GBFS loop, slows it down. In order to avoid triggering probes that are not likely to help, we measure the progress that the probes do in solving the problem. Basically, we assume that a probe is useless if the end state of the probe is no better than the first state in the probe, where the notion of better is given as in FF by the heuristic: the final state of the probe is better than the seed state of the probe if its heuristic value is smaller. When a probe is found to be useless in this sense, i.e. it doesn't improve the value of the seed state, then a parameter R, called the *probe ratio* is increased by 1. The meaning of this parameter is that the planner launches a probe every R expanded nodes. The parameter is initially set to 1, and then when probes fail without doing 'useful work', it is increased, so that probes end up being triggered less and less often, thus reducing their overhead.

Experimental Results

We compare PROBE with FF and LAMA over the domains of the last IPC.[1] PROBE is written in C++ and uses Metric-FF as an *ADL* to *Propositional STRIPS* compiler (Hoffmann 2003). LAMA and PROBE are executed without the plan improvement option, reporting the first plan that they find. All experiments were conducted on a dual-processor Xeon 'Woodcrest' running at 2.33 GHz and 8 GB of RAM. Processes time or memory out after 30 minutes or 2 GB. As the first solution of PROBE ignores costs, all action costs are assumed to be 1 so that plan cost is plan length also for LAMA and FF. Table 1 compares PROBE with FF and LAMA over 270 instances from last IPC. In terms of coverage, PROBE solves 7 more problems than LAMA and 37 more than FF.

Domain	I	FF	LAMA	PROBE
Cyber	30	4	30	30
Elevator	30	30	30	30
Openstacks	30	30	30	30
Parc-Printer	30	30	25	30
Pegsol	30	30	30	30
Scanalyzer	30	30	30	28
Sokoban	30	27	25	24
Transport	30	29	30	30
Woodworking	30	17	28	30
Total	270	227	257	264
Percentage		84%	95%	98%

Table 1: Coverage of PROBE vs. FF and LAMA over instances of the last IPC where *I* is the number of instances per domain

[1] FF is FF2.3, while LAMA is the 2008 IPC version. with a more recently fixed parser.

References

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1–2):5–33.

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

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

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered landmarks in planning. *JAIR* 22:215–278.

Hoffmann, J. 2003. The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *JAIR* 20:291–341.

Keyder, E. 2010. *New Heuristics For Planning With Actioin Costs*. Ph.D. Dissertation, Universitat Pompeu Fabra.

Lipovetzky, N., and Geffner, H. 2011. Searching for plans with carefully designed probes. In *Proc. ICAPS-11*.

McDermott, D. 1996. A heuristic estimator for means-ends analysis in planning. In *Proc. AIPS-96*.

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

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks revisited. In *Proc. AAAI-08*.