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 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 classical planners that make use of the notion of width (**?**), while PROBE is a standard best-first search planner that augments the expansion of a node by throwing an "intelligent" probe which either reaches the goal or terminates quickly in low polynomial time (**?**). The basic building block of *SIW* is the Iterative Width Procedure (*IW*) for achieving one atomic goal at a time. *IW* 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 from *IW* with helpful-actions, landmarks and delete-relaxation heuristics in a Greedy Best-Fist search.

In the following sections we introduce the basic notions of the algorithms and the implementation.

SIW: Iterated Width Search

The algorithm *Iterated Width*, or *IW*, consists of a sequence of calls *IW*(i) for $i = 0, 1, \dots, |F|$ 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 one of the atoms in the goal (**?**). 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. SIW uses IW to greedily achieve one atomic goal at a time 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 (**?**), but does not approach the performance of the most recent planners. Nonetheless, *SIW* competes well in domains with no dead-ends and simple serializations.

BFS(f): 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. For this, we developed a standard *forward-search best-first planner* guided by an evaluation function that combines the notions of novelty and helpful actions (**?**; **?**). In this planner, called *BFS(f)* (**?**), ties are broken lexicographically by two other measures: (1) the number of subgoals not yet achieved up to a node in the search, and (2) the additive heuristic, h_{add}. The additive heuristic is delayed for non-helpful actions.

PROBE

PROBE is a complete, standard greedy best first search (GBFS) STRIPS planner using the standard additive heuristic (**?**), 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.

PROBE is built using an early-version of an 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.

Implementation Notes

The planners *SIW*, $BFS(f)$ have been implemented using the automated planning toolkit `lwaptk`[1]. The toolkit is an extensible C++ framework that decouples search and heuristic algorithms from PDDL parsing and grounding modules,

[*]firstname.lastname

[1]Source code available from `https://github.com/miquelramirez/lwaptk`.

by relying on planner "agnostic" data structures to represent (ground) fluents and actions. We consider `lwaptk` to be a valuable contribution in itself since it allows to develop, relying on a collection of readily available implementations of search algorithms and planning heuristics, planners which are independent from specific parsing modules and grounding algorithms. If the planner is to be acquiring descriptions of planning tasks from PDDL specifications, the toolkit provides the means to plug into the planner either FF (**?**) or FAST-DOWNWARD (**?**) parsers. Alternatively, and more interestingly, the planner can be embedded into complex applications, *directly*, if the "host" application is written in `C++`, or *indirectly* when the host is written in an interpreted language, such as PYTHON, by wrapping the planner with suitably generated marshalling code.

Acknowledgments This work was partly supported by Australian Research Council Linkage grant LP11010015, and Discovery Projects DP120100332 and DP130102825.