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*

Abstract

We have recently shown that classical planning problems can be characterized in terms of a width measure that is bounded and small for most planning benchmark domains when goals are restricted to single atoms. Two simple algorithms have been devised for exploiting this structure: Iterated Width (IW) for achieving atomic goals, that runs in time exponential in the problem width by performing a sequence of pruned breadth first searches, and Serialized IW (SIW) that uses IW in a greedy search for achieving conjunctive goals one goal at a time. While SIW does not use heuristic estimators of any sort, it manages to solve more problemas that a Greedy BFS with heuristics like h_{add}. Yet, it does not approach the peformance of more recent planners like LAMA. In addition, ideas like helpful actions and landmarks can be integrated as well, producing a planner with state-of-the-art performance.

Introduction

The main approach for domain independent planning is based on heuristic search with heuristics derived automatically from problems (McDermott 1996; Bonet and Geffner 2001). To this, recent planners add other ideas like helpful actions, landmarks, and multiqueue best-first search for combining different heuristics (Hoffmann and Nebel 2001; Helmert 2006; Richter and Westphal 2010). From a different angle, we have recently shown that most of the benchmark domains are easy when the goals contain single atoms, and that otherwise, they can be easily serialized (Lipovetzky and Geffner 2012). More precisely, we showed that the former problems have a low width, and developed an algorithm, Iterative Width (IW) that runs in time that is exponential in the problem width by performing a sequence of pruned breadth first searches. This algorithm is used in the context of another algorithm, Serialized IW (SIW), that achieves conjunctive goals by using IW greedily for achieving one goal at a time. Surprisingnly, the blind-search algorithm SIW which has no heuristic guidance of any sort, performs better than a greedy best-first search guided by delete-relaxation heuristics.

The aim of this planners is to exploit the new width notion and propose a different approach to search in planning. All planners are based in a new toolkit for automated planning () aiming at ...

The organization of the paper follows this structure

Iterated Width Search

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.

The algorithm, called *Iterated Width* search or *IW*, consists of a sequence of calls *IW*(i) for $i = 0, 1, 2, \ldots$ over a problem P until the problem is solved. Each iteration *IW*(i) is an i-width search that is complete for problems whose width is bounded by i and whose complexity is $O(n^i)$, where n is the number of problem variables. If P is solvable and its width is w, *IW* will solve P in at most $w + 1$ iterations with a complexity $O(n^w)$. *IW*(i) is *a plain forward-state breadth-first search* with just one change: right after a state s is generated, the state is pruned if it doesn't pass a simple *novelty test* that depends on i.

Definition 1 *A newly generated state s* generates a new tuple of atoms t *iff s is the first state generated in the search that makes t true. The size of the smallest new tuple of atoms generated by s is called the* novelty *of s. When s does not generate a new tuple, it's novelty is set to $n + 1$ where n is the number of problem variables.*

In other words, if s is the first state generated in all the search that makes an atom p true, its novelty is 1. If s does not generate a new atom but generates a new pair (p, q), its novelty is 2, and so on. Likewise, if s does not generate a new tuple at all because the same state has been generated before, then its novelty is set to $n + 1$. The higher the novelty measure, the less novel the state. The iterations *IW*(i) are plain *breadth-first searches* that treat newly generated states with novelty measure greater than i as if they were 'duplicate' states:

Definition 2 *IW*(i) *is a breadth-first search that prunes newly generated states when their novelty measure is greater than i.*

Notice that *IW*(n), when n is the number of atoms in the problem, just prunes truly duplicate states and it is therefore complete. On the other hand, *IW*(i) for lower i values prunes many states and is not. Indeed, the number of states *not* pruned in *IW*(1) is $O(n)$ and similarly, the number of states not pruned in *IW*(i) is $O(n^i)$. Likewise, since the novelty of a state is never 0, *IW*(0) prunes all the children states of the initial state s_0, and thus *IW*(0) solves P iff the goal is true in the initial situation. The resulting planning algorithm *IW* is just a series of i-width searches *IW*(i), for increasing values of i:

Definition 3 *Iterated Width (IW) calls IW*(i) *sequentially for* $i = 0, 1, 2, \ldots$ *until the problem is solved or* i *exceeds the number of problem variables.*

Iterated Width (*IW*) is thus a *blind-search algorithm* similar to Iterative Deepening (*ID*) except for two differences. First, each iteration is a pruned *depth-first* search in *ID*, and a pruned *breadth-first* search in *IW*. Second, each iteration increases pruning depth in *ID*, and pruning width or novelty in *IW*.

From the considerations above it is straightforward to show that *IW* like *ID* is *sound* and *complete.* On the other hand, while *IW*(w) is *optimal* for a problem P of width w, *IW* is not necessarily so. The reason is that *IW* may solve P in an iteration *IW*(i) for i smaller than w.

Nonetheless the completeness and optimality of *IW*(w) for problems with width w provides the right complexity bound for *IW*:

Theorem 4 *For solvable problems* P*, the time and space complexity of IW are exponential in* $w(P)$*.*

It's important to realize that this bound is achieved without knowing the actual width of P. This follows from the result below, whose proof we omit for lack of space:

Theorem 5 *For a solvable problem* P *with width* w*, IW*(w) *solves* P *optimally in time exponential in* w*.*

The algorithm *IW*(w) is guaranteed to solve P if $w(P) = w$, yet as discussed above, the algorithm *IW* does not assume that this width is known and thus makes the *IW*(i) calls in order starting from $i = 0$. We refer to the min value of i for which *IW*(i) solves P as the *effective width* of P, $w_e(P)$, which is never higher than the real width $w(P)$.

The effective width $w_e(P)$ provides an approximation of the actual width $w(P)$. While proving formally that most benchmark domains have bounded width *for single atom goals* is tedious, we have run the algorithm *IW* to compute the *effective width* of such goals. Over all domains of Previous IPC: 37% with $w_e = 1$, 51% with $w_e = 2$, and less than 12% with $w_e > 2$. *That is, on average, less than* 12% *of the instances have effective width greater than* 2. Actually, in most domains *all* the instances have effective width at most 2, and in four domains, all the instances have effective width 1. We will see below that a simple extension suffices to make *IW* competitive with a *heuristic* planner over the standard benchmark instances that feature *joint goals*.

Serialized Iterated Width

The fact that single goal atoms can be achieved quite effectively in most benchmarks domains by a pruned breadth-first search that does not look at the goal in any way, suggests that the complexity of benchmarks comes from conjunctive goals. Indeed, this has been the intuition in the field of planning since its beginnings where goal decomposition was deemed as a crucial and characteristic technique. The analysis above formalizes this intuition by showing that the effective width of single atom goals in existing benchmarks is low. This old intuition also suggests that the power of planners that can handle single goals efficiently can be exploited for conjunctive goals through some form of decomposition.

Serialized Iterated Width is a search algorithm that uses the iterated width searches both for constructing a serialization of the problem $P = \langle F, I, O, G\rangle$ and for solving the resulting subproblems. While *IW* is a sequence of i-width searches *IW*(i), $i = 0, 1, \ldots$ over the same problem P, *SIW* is a sequence of *IW* calls over $|G|$ subproblems P_k, $k = 1, \ldots, |G|$. The definition of *SIW* takes advantage of the fact that *IW* is a blind-search procedure that doesn't need to know the goals of the problem in advance; it just needs to recognize them in order to stop. Thanks to this feature *IW* is used both for decomposing P into the sequence of subproblems P_k and for solving each one of them. The plan for P is the concatenation of the plans obtained for the subproblems.

Definition 6 *Serialized Iterated Width (SIW) over* $P = \langle F, I, O, G\rangle$ *consists of a sequence of calls to IW over the problems* $P_k = \langle F, I_k, O, G_k\rangle$*,* $k = 1, \ldots, |G|$*, where*

1. $I_1 = I$,
2. G_k *is first* consistent *set of atoms achieved from* I_k *such that* $G_{k-1} \subset G_k \subseteq G$ *and* $|G_k| = k$*;* $G_0 = \emptyset$
3. I_{k+1} *represents the state where* G_k *is achieved,* $1 < k < |G|$.

In other words, the k-th subcall of *SIW* stops when *IW* generates a state s_k that consistently achieves k goals from G: those achieved in the previous subcall and a new goal from G. The same is required from the next subcall that starts at s_k. The state s_k *consistently* achieves $G_k \subseteq G$ if s_k achieves G_k, and G_k does not need to be undone in order to achieve G. This last condition is checked by testing whether $h_{max}(s_k) = \infty$ is true in P once the actions that delete atoms from G_k are excluded (Bonet and Geffner 2001). Notice that *SIW* does not use heuristic estimators to the goal, and does not even know what goal G_k is when *IW* is invoked on subproblem P_k: it finds this out when *IW* generates a set of atoms G' such that $G_{k-1} \subset G' \subseteq G$ and $|G'| = k$. It then sets G_k to G'. This is how *SIW* manages to use *IW* for both constructing the serialization and solving the subproblems.

The *SIW* algorithm is sound and the solution to P can be obtained by concatenating the solutions to the problems P_1, $\ldots$, P_m, where $m = |G|$. Like *IW*, however, *SIW* does not guarantee the optimality of the plans found. Likewise, while the *IW* algorithm is complete, *SIW* is not. The reason is that the subgoal mechanism implicit in *SIW* commits to intermediate states from which the goal may not be reachable. Of course, if there are no dead-ends in the problem, *SIW* is complete.

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

$$f(n) = novelha(n) \tag{1}$$

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

$$novelha(n) = 2[novel(n) - 1] + help(n)\ . \tag{2}$$

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. Once again, the criterion is simple and follows from performance considerations.

Automated Planning Toolkit

. . . . .

Miquel, Christian, here we can explain the implementation details

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 toolkit. The only difference with respect to PROBE-IPC7 is that the anytime procedure is disabled, as we are only concerned with the first solution.

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...

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.

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

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.