# A Polynomial Planning Algorithm that Beats LAMA and FF

**Nir Lipovetzky**
The University of Melbourne
Melbourne, Australia
nir.lipovetzky@unimelb.edu.au

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

### Abstract

It has been shown recently that heuristic and width-based search can be combined to produce planning algorithms with a performance that goes beyond the state-of-the-art. Such algorithms are based on best-first width search (BFWS), a plain best-first search set with evaluations functions combined lexicographically to break ties, some of which express novelty-based preferences. In BFWS($f_5$), for example, the evaluation function $f_5$ weights nodes by a novelty measure, breaking ties by the number of non-achieved goals. BFWS($f_5$) is a best-first algorithm, and hence, it is complete but not polynomial, and its performance doesn't match the state of the art. In this work we show, however, that incomplete versions of BFWS($f_5$) where nodes with novelty greater than $k$ are pruned, are not only polynomial but have an empirical performance that is better than both BFWS($f_5$) and state-of-the-art planners. This is shown by considering all the international planning competition instances. This is the first time where polynomial algorithms with meaningful bounds are shown to achieve state-of-the-art performance in planning. Practical and theoretical implications of this empirical finding are briefly sketched.

## Introduction

Recently, it was shown that many of the common classical planning benchmarks domains can be solved by a polynomial algorithm, called Iterative Width (IW), provided that instances feature a single goal atom (Lipovetzky and Geffner 2012). For problems with multiple atomic goals, however, IW is not effective, and extensions such as Serialized IW (SIW), where IW is used to achieve one atomic goal at a time, do not compete with the best planners. The aim of this paper is to narrow the performance gap between polynomial planning algorithms and state-of-the-art planners. For this, we move away from pure width-based methods such as IW and build on a recent search framework, best-first width search (BFWS), that integrates width-based and heuristic search (Lipovetzky and Geffner 2017). We review width-based search and best-first width-search next, and present then the new polynomial planning algorithms, the evaluation, and the conclusions.



## Width-based Search

Pure width-based search algorithms are exploration algorithms that do not look at the goal at all. The simplest such algorithm is IW(1), which is a plain breadth-first search where newly generated states that do not make an atom $X = x$ true for the first time in the search are pruned. The algorithm IW(2) is similar except that a state $s$ is pruned when there are no atoms $X = x$ and $Y = y$ such that the *pair* of atoms $\langle X = x, Y = y\rangle$ is true in $s$ and false in all the states generated before $s$. IW($k$) is a normal breadth-first except that newly generated states $s$ are pruned when their "novelty" is greater than $k$, where the *novelty* of $s$ is $i$ iff there is a tuple $t$ of $i$ atoms such that $s$ is the first state in the search that makes all the atoms in $t$ true, with no tuple of smaller size having this property (Lipovetzky and Geffner 2012). While simple, it has been shown that IW($k$) manages to solve arbitrary instances of many of the standard benchmark domains in low polynomial time provided that the goal is a single atom. Such domains can be shown to have a small and bounded *width* $w$ that does not depend on the instance size, which implies that they can be solved (optimally) by running IW($w$). Moreover, IW($k$) runs in time and space that are exponential in $k$ and not in the number of problem variables. The procedure IW, that calls the procedures IW(1), IW(2), ... sequentially, has been used to solve instances featuring multiple (conjunctive) atomic goals, in the context of *Serialized IW* (SIW), an algorithm that calls IW for achieving one atomic goal at a time. While SIW is a blind search procedure that is incomplete (it can get trapped into dead-ends), it turns out to perform much better than a greedy best-first search guided by the standard delete relaxation heuristics. Other variations of IW have been used for planning in the Atari games and those of the General Video-Game AI competition (Lipovetzky, Ramirez, and Geffner 2015; Geffner and Geffner 2015; Shleyfman, Tuisov, and Domshlak 2016; Jinnai and Fukunaga 2017).

## Best-First Width Search

Best-first width algorithms (BFWS) have been shown to achieve state-of-the-art performance by combining width and heuristic search (Lipovetzky and Geffner 2017; Katz et al. 2017). BFWS($f$) with $f = \langle h, h_1, \ldots, h_n\rangle$ is a standard best-first search that uses the function $h$ to rank the nodes in OPEN, breaking ties lexicographically with the functions

$h_1, \ldots, h_n$. The primary evaluation function $h$ is given by the novelty measure of the node. The novelty $w(s)$ of a newly generated state $s$ *given the functions* $h_1, \ldots, h_n$ is $i$ iff there is a tuple (set) of $i$ atoms $X = x$ and no tuple of smaller size, that is true in $s$ but false in all previously generated states $s'$ *with the same function values* $h_1(s') = h_1(s)$, $\ldots$, and $h_n(s') = h_n(s)$. For example, a new state $s$ has novelty 1 if there is an atom $X = x$ that is true in $s$ and false in all the states $s'$ generated before $s$ where $h_i(s') = h_i(s)$ for all $i$. We write the measures $w$ as $w_{\langle h_1,\ldots,h_n\rangle}$ sometimes in order to make explicit the functions $h_i$ used in the definition and computation of $w$.

For the planner BFWS($f_5$) (Lipovetzky and Geffner 2017), the evaluation function $f_5$ is given by the tuple $\langle w, \#g\rangle$ where $\#g(s)$ tracks the number of the atomic goals that are not true in $s$. The novelty measure $w$ is computed given two functions: the goal counter $\#g$, and a second counter $\#r(s)$. This second counter tracks the number of atoms in the last relaxed plan computed in the way to $s$ that have been made true. Relaxed plans (such as in FF) are not computed in every state but only in the initial state and in the states that decrease the $\#g$ count. This is because in this "exploratory" setting, computing relaxed plans in every node is not cost-effective. The exact definition of $\#r(s)$ is as follows: if $\pi$ is the set of actions in the last relaxed plan computed in the way to state $s$, say in state $s'$, and $R$ is the set of atoms associated with such a plan, then $\#r(s)$ is the number of atoms in $R$ that have been made true in some state $s''$ in the way from $s'$ to $s$ including these two states (Lipovetzky and Geffner 2014). For STRIPS problems, the set of atoms associated with a plan is given by the preconditions and positive effects of the actions in the plan. In the implementation of BFWS, a default simplification is that novelty measures are computed with 3-value precision, i.e., $w(s)$ is determined to be either 1, 2, or greater than 2. That is, novelty measures greater than 2 are treated as novelty measures equal to 3. This is because determining that a novelty measure for a node is $k$ is exponential in $k-1$, as all potential new tuples of size up to $k$ need to be considered. The algorithm BWFS($f_5$) with the novelty measure $w$ in $f_5$ simplified to $w_{\langle \#g\rangle}$ would be similar to SIW (Lipovetzky and Geffner 2012) in the sense that it would use the goal counter $\#g$ to provide a form of goal serialization while keeping the novelty measures associated with the different subproblems separate. On the other hand, unlike SIW, BWFS($f_5$) is a plain best-first search algorithm that is complete. The additional $\#r$ counter used as part of the novelty measure $w = w_{\langle \#g,\#r\rangle}$ in $f_5$ yields a form of decomposition within each subproblem as well. In particular, BWFS($f_5$) solves delete-free problems by expanding novelty-1 states only, something that wouldn't be true if the novelty measure $w = w_{\langle \#g\rangle}$ was used instead.

## Polynomial $k$-BFWS

BFWS is a complete search algorithm that can be easily turned into a polynomial but incomplete search algorithm by just pruning the states $s$ whose novelty $w(s)$ exceeds a bound $k$. For this, it is assumed that the number of functions $h_i$ in $w = w_{\langle h_1,\ldots,h_n\rangle}$, and their range, are all polynomial. For example, for $w = w_{\langle \#g,\#r\rangle}$ in BFWS($f_5$), there are two functions $\#g$ and $\#r$ such that each takes at most $|F|$ different values, where $F$ is the set of atoms in the problem. The polynomial version of BFWS($f_5$) that is obtained by pruning nodes with novelty $w = w_{\langle \#g,\#r\rangle}$ greater than $k$ never expands more than $|F|^{k+1} \times |G|$ nodes, where $|G|$ represents the number of goals, and never generates more than $|F|^{k+1} \times |G| \times |A|$ nodes, where $A$ is the number of (ground) actions.

We write $k$-BFWS($f$) to refer to the version of BFWS($f$) where nodes with novelty measure greater than $k$ are pruned. We note that unlike BFWS, $k$-BFWS is not a standard best-first search, as it is not complete. The $k$-BFWS($f_5$) algorithm with $k = 1$ was used as the front-end of the complete Dual-BFWS algorithm (Lipovetzky and Geffner 2017), as it terminates quickly, successfully or not. Other front-end algorithms such as Enforced Hill Climbing in FF (Hoffmann and Nebel 2001), are often fast but are not polynomial, as more than one action may be "helpful" at each node and the depth of this search is not bounded in general.

## $k$-BFWS Variants

Below we evaluate $k$-BFWS($f_5$) with $k$ values 1 and 2 and two variations of this algorithm that are still polynomial. The first $k$-BFWS variation results from a simple, polynomial consistency test each time that a state $s$ achieves a new goal. In the consistency-variant, a goal atom $p$ that is true in $s$ but false in its parent is considered unachieved, and hence it's not used for decreasing the count, if it can be shown that $p$ has to be undone in order to reach the other goals. This is determined by computing the $h_{max}$ heuristic from the state $s$ excluding the actions that delete the atom $p$. This is the test used in SIW for discarding some goal serializations (Lipovetzky and Geffner 2012). The test is polynomial and doesn't affect the number of nodes expanded by the algorithm in the worst case. The second $k$-BFWS variation increases the number of expanded nodes by a constant factor of $M$ by allowing some nodes with novelty greater than $k$ to go unpruned. For this, a node $s'$ is regarded as a $k^+$-descendant of a node $s$ if (a) $s'$ is a descendant of node $s$, (b) $s'$ has novelty greater than $k$, (c) $s$ has novelty no greater than $k$, and (d) nodes $s''$ between $s$ and $s'$ have all novelty greater than $k$. In the M-variant of $k$-BFWS, the first $k^+$-descendants of a node are not pruned. The bound on the total number of expanded nodes is then $(M+1) \times |F|^{k+1} \times |G|$. For $M = 0$, this variation of $k$-BFWS reduces to $k$-BFWS.

To summarize, using BFWS($f_5$) as the base algorithm, we consider *three polynomial variations* that result from 1) the pruning bound $k$, 2) the use of a polynomial consistency test in the goal count, and 3) the use of the $M$ parameter for expanding the set of unpruned nodes. These three variations can be *combined*. The version that uses $k$ equal to 2, consistency tests, and $M$ is denoted as 2-C-M, while the version that uses no consistency tests and $M$ equal to 0, is denoted by the number $k$ alone and corresponds to $k$-BFWS($f_5$). Any *sequence* of these polynomial variants defines a *polynomial portfolio* as well. The notation $\langle$1,2,2-M$\rangle$ is used to denote the sequential portfolio where 1-BFWS is followed by 2-

BFWS and then 2-BFWS with $M > 0$. Other polynomial portfolios are obtained by using different values of $M$. For instance, the planner 2-M can be used first with $M$ equal to 1, then 2, 4, 8, ..., until $M_{max}$. Indeed, in the experiments below, each $M$-planner will be used in this manner with $M_{max}$ fixed to 32. In this and in all polynomial portfolios, no timeouts are imposed on the planners; a planner in a sequential portfolio is invoked iff all precedent planners finished with no solution, and time is available. This choice exploits the fact that all the planners are polynomial and finish quickly, whether successfully or unsuccessfully. For example, 1-BFWS finishes with no solution in 374 of the 1676 STRIPS problems below, and in 220 of them, this happens before 0.1 seconds. This explains why portfolios that start with 1-BFWS often solve more problems that 1-BFWS alone even after a few tens of a second (as shown in the plots).

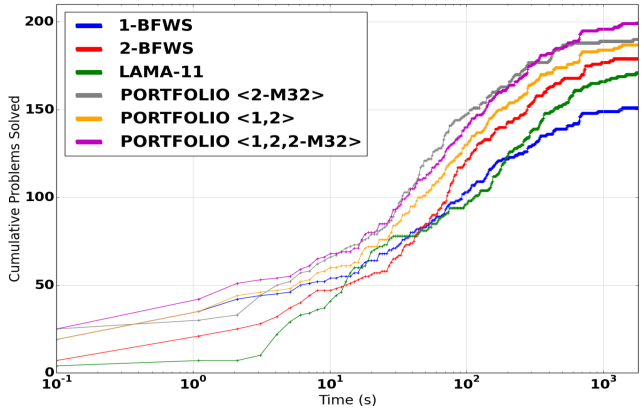


Figure 1: Coverage over IPC-2014 problems as function of time.

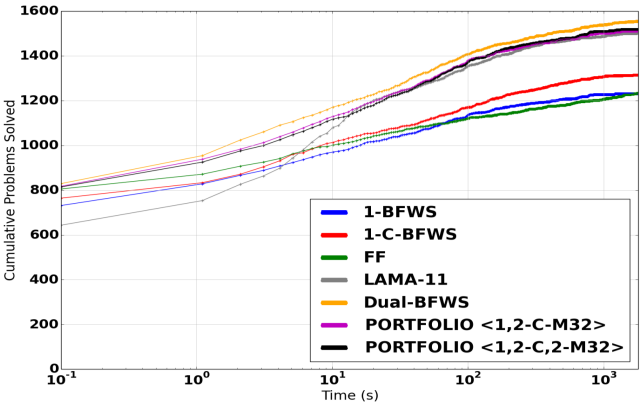


Figure 2: Coverage over all STRIPS problems as function of time.

## Experimental Results

We have evaluated the performance of these polynomial BFWS variants and portfolios over the STRIPS instances of all Int. Planning Competitions so far (1676 instances in total), and separately, over the instances of the 2014 Int. Planning Competition, some of which feature conditional effects as well. These are all instances from the satisficing tracks. The polynomial planners are implemented in LAPKT (Ramirez, Lipovetzky, and Muise 2015). The experiments were performed on a 2.40GHz Intel Processor with time and memory cutoffs of 30 min and 8GB respectively. Table 1 compares some polynomial $k$-BFWS variants with LAMA-11 (Richter and Westphal 2010) over the instances of the 2014 IPC. All the polynomial planners, with the exception of 1-BFWS, solve more problems than LAMA-11, faster, and with better plans (avg plan qualities are 320 vs. 374 for LAMA-11; not shown). Moreover, the polynomial planners solve more problems than LAMA-11 not only at the cutoff time, but at all times as shown by the curves in Figure 1. Some of them solve also more problems than the best IPC-2014 planners like Jasper (Xie, Müller, and Holte 2014), Mercury (Domshlak, Hoffmann, and Katz 2015), and the Ibacop portfolio (Cenamor, De La Rosa, and Fernández 2014), that solve 193, 177, and 198 instances respectively. The polynomial planners 2-M and ⟨1,2,2-M⟩ solve 190 and 199 instances each. In comparison, BWFS($f_5$) solves 192 problems (not shown). Table 2 shows the performance of these planners over the STRIPS benchmarks, along with FF[1] and Dual-BFWS (Lipovetzky and Geffner 2017). Once again, the coverage of the best polynomial $k$-BFWS variants are at level of the best exponential planners, most often with better average times and plan qualities, as shown at the bottom of the table. It is also interesting that 1-BFWS solves as many problems as FF (1234 vs 1235) almost one order of magnitude faster, while 1-C-BFWS, that just adds consistency tests to 1-BFWS, has a higher coverage (1318) and is 4 times faster. LAMA-11 solves 1508 of these instances, outperforming 1-BFWS and 1-C-BFWS but not the ⟨1,2-C-M⟩ and ⟨1,2-C,2-M⟩ portfolios that solve 1511 and 1518 problems with better average times and plan lengths. Curves showing coverage as a function of time are shown in Fig. 2.

| | 1 | 2 | ⟨1,2⟩ | ⟨2-M⟩ | ⟨1,2,2-M⟩ | LAMA11 |
|---|---|---|---|---|---|---|
| Barman (20) | 0 | **20** | **20** | **20** | **20** | 19 |
| CaveDiving (20) | 0 | 0 | 0 | **8** | **8** | 7 |
| Childsnack (20) | 1 | 1 | 2 | **3** | 2 | 0 |
| CityCar (20) | 1 | **5** | **5** | **5** | **5** | 3 |
| Floortile (20) | 0 | **2** | **2** | **2** | **2** | **2** |
| GED (20) | **20** | 19 | **20** | 19 | **20** | **20** |
| Hiking (20) | 0 | 7 | 7 | 8 | 11 | **16** |
| Maintenance (20) | **17** | 12 | **17** | 16 | **17** | 7 |
| Tetris (20) | **17** | 16 | **17** | 12 | **17** | 8 |
| Thoughtful (20) | 15 | **17** | **17** | **17** | **17** | 15 |
| Transport (20) | **20** | **20** | **20** | **20** | **20** | 14 |
| Coverage (280) | 151 | 179 | 187 | 190 | 199 | 171 |
| Average Time | 47 | 64 | 47 | 52 | 47 | 128 |

Table 1: Coverage over STRIPS and ADL IPC-14 problems. Domains solved fully by all planners ommited. Avg. time in seconds over problems solved by all planners. Best coverage shown in red.

## Discussion

We have shown that the performance of the best exponential planners can be approached by means of polynomial algorithms with meaningful bounds. The result is important because the difference between polynomial and exponential

[1]Metric-FF-2.1 in classic-FF mode but ignoring costs, as original FF does not parse action costs.

| | k-BFWS Polynomial Planners | | | | | | Exponential Planners | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 1-C | 2 | 2-C | ⟨1,2-C-M⟩ | ⟨1,2-C,2-M⟩ | FF | LAMA11 | Mercury | JASPER | Dual-BFWS |
| airport (50) | **49** | 47 | **49** | 47 | **49** | **49** | 40 | 31 | 35 | 36 | **49** |
| barman-sat11-strips (20) | 0 | 0 | **20** | **20** | **20** | **20** | 4 | **20** | **20** | **20** | **20** |
| barman-sat14-strips (20) | 0 | 0 | **20** | **20** | **20** | **20** | 3 | 19 | 19 | **20** | **20** |
| blocks (35) | 21 | 34 | **35** | **35** | **35** | **35** | 25 | **35** | **35** | **35** | **35** |
| childsnack-sat14-strips (20) | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 6 | 0 | **10** |
| depot (22) | 19 | 21 | 21 | 21 | **22** | **22** | 20 | **22** | 18 | **22** | **22** |
| driverlog (20) | **20** | **20** | **20** | **20** | **20** | **20** | 18 | **20** | **20** | **20** | **20** |
| floortile-sat11-strips (20) | 0 | 0 | 4 | 2 | 4 | 6 | 6 | 6 | 7 | 6 | **8** |
| floortile-sat14-strips (20) | 0 | 0 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | **4** |
| freecell (80) | 73 | 72 | **80** | **80** | **80** | **80** | 78 | 79 | 79 | **80** | **80** |
| ged-sat14-strips (20) | 19 | 16 | **20** | 16 | 19 | 19 | 4 | **20** | **20** | **20** | **20** |
| grid (5) | 4 | **5** | 4 | **5** | **5** | **5** | **5** | **5** | **5** | **5** | **5** |
| hiking-sat14-strips (20) | 0 | 0 | 7 | 7 | 8 | 11 | 13 | 16 | 12 | **20** | 12 |
| logistics98 (35) | 34 | 34 | 28 | 24 | 34 | 34 | **35** | **35** | **35** | **35** | 34 |
| mprime (35) | 29 | 29 | 29 | 31 | 32 | 32 | 34 | **35** | **35** | **35** | **35** |
| mystery (30) | 14 | 14 | 19 | 19 | 21 | **22** | 15 | 19 | 16 | 19 | 19 |
| nomystery-sat11-strips (20) | 3 | 4 | 14 | 13 | 13 | 14 | 7 | 13 | 13 | **20** | 19 |
| openstacks-sat11-strips (20) | **20** | **20** | **20** | 19 | **20** | **20** | 11 | **20** | **20** | **20** | **20** |
| openstacks-sat14-strips (20) | **20** | **20** | **20** | 15 | **20** | **20** | 9 | **20** | 16 | **20** | **20** |
| openstacks-strips (30) | 28 | 29 | 27 | 29 | 28 | 28 | 26 | **30** | **30** | **30** | 28 |
| parcprinter-08-strips (30) | 6 | **30** | 15 | **30** | **30** | **30** | **30** | **30** | **30** | **30** | **30** |
| parcprinter-sat11-strips (20) | 0 | 18 | 1 | **20** | **20** | **20** | **20** | **20** | **20** | **20** | 19 |
| parking-sat11-strips (20) | **20** | **20** | **20** | **20** | **20** | **20** | 4 | **20** | **20** | **20** | **20** |
| parking-sat14-strips (20) | **20** | 19 | **20** | **20** | **20** | **20** | 0 | **20** | 13 | **20** | **20** |
| pathways-noneg (30) | 27 | 26 | 23 | 26 | 29 | 29 | 14 | **30** | 29 | **30** | **30** |
| pegsol-08-strips (30) | 12 | 25 | 23 | 29 | **30** | 29 | 29 | **30** | **30** | **30** | **30** |
| pegsol-sat11-strips (20) | 2 | 16 | 12 | 19 | **20** | 19 | 19 | **20** | **20** | **20** | **20** |
| pipesworld-notankage (50) | 49 | 48 | 49 | **50** | **50** | **50** | 43 | 44 | 43 | 45 | **50** |
| pipesworld-tankage (50) | 23 | 28 | 33 | 30 | 34 | 33 | 26 | 42 | 40 | **44** | 33 |
| psr-small (50) | 7 | 21 | 22 | 36 | 45 | 43 | 49 | **50** | **50** | **50** | **50** |
| rovers (40) | **40** | **40** | **40** | 35 | **40** | **40** | **40** | **40** | **40** | **40** | **40** |
| satellite (36) | 31 | 32 | 29 | 27 | 31 | 31 | 33 | **36** | **36** | **36** | 32 |
| scanalyzer-08-strips (30) | **30** | 28 | **30** | 28 | **30** | **30** | 27 | **30** | **30** | **30** | **30** |
| scanalyzer-sat11-strips (20) | **20** | 18 | **20** | 18 | **20** | **20** | 17 | **20** | **20** | **20** | **20** |
| sokoban-sat08-strips (30) | 0 | 0 | 11 | 10 | 18 | 21 | 26 | **29** | 28 | 27 | 25 |
| sokoban-sat11-strips (20) | 0 | 0 | 5 | 5 | 11 | 13 | 16 | **19** | 18 | 18 | 16 |
| storage (30) | **30** | 28 | 23 | 27 | **30** | **30** | 18 | 19 | 20 | 26 | 28 |
| tetris-sat14-strips (20) | 17 | 12 | **18** | 12 | 17 | 17 | 3 | 7 | 13 | 13 | 17 |
| thoughtful-sat14-strips (20) | 16 | 16 | 17 | 17 | 17 | 17 | 14 | 15 | 12 | 18 | **20** |
| tidybot-sat11-strips (20) | 9 | 6 | 19 | 19 | **20** | 19 | 16 | 16 | 13 | 18 | 18 |
| tpp (30) | **30** | **30** | 28 | **30** | **30** | **30** | 28 | **30** | **30** | **30** | 29 |
| transport-sat08-strips (30) | **30** | **30** | **30** | **30** | **30** | **30** | 18 | **30** | **30** | **30** | **30** |
| transport-sat11-strips (20) | **20** | **20** | **20** | **20** | **20** | **20** | 1 | 17 | **20** | **20** | **20** |
| transport-sat14-strips (20) | **20** | **20** | **20** | **20** | **20** | **20** | 0 | 14 | **20** | 15 | **20** |
| trucks-strips (30) | 4 | 4 | 11 | 7 | 8 | 9 | 9 | 15 | 18 | **23** | 14 |
| visitall-sat11-strips (20) | **20** | **20** | **20** | **20** | **20** | **20** | 4 | **20** | **20** | **20** | **20** |
| visitall-sat14-strips (20) | **20** | **20** | **20** | **20** | **20** | **20** | 0 | **20** | **20** | **20** | **20** |
| woodworking-sat08-strips (30) | **30** | **30** | 29 | **30** | **30** | **30** | 29 | **30** | **30** | **30** | **30** |
| woodworking-sat11-strips (20) | **20** | **20** | 12 | **20** | **20** | **20** | 19 | **20** | **20** | **20** | **20** |
| Coverage (1676) | 1234 | 1318 | 1387 | 1428 | 1511 | 1518 | 1235 | 1508 | 1504 | 1556 | 1559 |
| Average Time | 10 | 24 | 46 | 33 | 10 | 10 | 88 | 40 | 42 | 38 | 10 |
| Average Length | 118 | 117 | 118 | 117 | 118 | 118 | 71 | 130 | 115 | 132 | 118 |

Table 2: Coverage over all STRIPS benchmarks from IPCs: k-BFWS polynomial vs. exponential planners. Domains solved by all planners ommited. Averages plan quality and time in seconds over problems solved by all planners. Best coverage shown in red.

algorithms is significant both practically and theoretically. The result also highlights the importance of novelty measures and width-based search. Indeed, while these polynomial planners compute relaxed plans when a new goal is achieved, they make no use of heuristic estimators, helpful actions, and landmarks; the techniques that are crucial to modern heuristic search planners. From a practical point of view, the polynomial algorithms are powerful and fast, so that even when they fail to solve a problem, they fail quickly. A consequence of this is that they can be given a quick try before using other polynomial or exponential algorithms. In the paper, we did this in the context of sequential polynomial portfolios but they can be used in combination with existing planners as well. For example, if 1-BWFS is run before the planners FF and LAMA-11, the number of STRIPS benchmarks solved jumps from 1235 to 1493 in the first case, and from 1508 to 1548 in the second. Many other possible ways of using these ideas are possible and worth exploring. From a theoretical point of view, these polynomial algorithms manage to solve some domains fully. This raises the question of whether this is a property of the instances or a property of the domain that will apply to *any* instance. The algorithm IW has been shown to solve in polynomial time *any* instance of many benchmark domains, provided that it features a single atomic goal. Similarly, SIW can be shown to solve instances with multiple goals, such as Visitall, where atomic goals can be achieved in polynomial time, one at a time, in any order. The algorithm $k$-BFWS goes beyond this and can be shown to solve *any* delete-free problem, something that is not true of either IW or SIW. The range of problems that these $k$-BWFS algorithms can provably solve, however, is not clear yet and deserves further study.

## Acknowledgements

The work by N. Lipovetzky is partially supported by the Australian Research Council linkage grant LP11010015. H. Geffner is partially supported by grant TIN2015-67959-P, MEC, Spain.

## References

[Cenamor, De La Rosa, and Fernández 2014] Cenamor, I.; De La Rosa, T.; and Fernández, F. 2014. IBACOP and IBACOP2 planner. In *Proc. of the 8th Int Planning Competition*.

[Domshlak, Hoffmann, and Katz 2015] Domshlak, C.; Hoffmann, J.; and Katz, M. 2015. Red–black planning: A new systematic approach to partial delete relaxation. *Artificial Intelligence* 221:73–114.

[Geffner and Geffner 2015] Geffner, T., and Geffner, H. 2015. Width-based planning for general video-game playing. In *Proc. 11th AI and Interactive Digital Entertainment Conf. (AIIDE)*.

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

[Jinnai and Fukunaga 2017] Jinnai, Y., and Fukunaga, A. 2017. Learning to prune dominated action sequences in online black-box planning. In *Proc. AAAI*.

[Katz et al. 2017] Katz, M.; Lipovetzky, N.; Moshkovich, D.; and Tuisov, A. 2017. Adapting novelty to classical planning as heuristic search. In *Proc. ICAPS*.

[Lipovetzky and Geffner 2012] Lipovetzky, N., and Geffner, H. 2012. Width and serialization of classical planning problems. In *Proc. ECAI*, 540–545.

[Lipovetzky and Geffner 2014] Lipovetzky, N., and Geffner, H. 2014. Width-based algorithms for classical planning: New results. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, 88–90.

[Lipovetzky and Geffner 2017] Lipovetzky, N., and Geffner, H. 2017. Best-first width search: Exploration and exploitation in classical planning. In *Proc. AAAI*.

[Lipovetzky, Ramirez, and Geffner 2015] Lipovetzky, N.; Ramirez, M.; and Geffner, H. 2015. Classical planning with simulators: Results on the atari video games. In *Proc. IJCAI*.

[Ramirez, Lipovetzky, and Muise 2015] Ramirez, M.; Lipovetzky, N.; and Muise, C. 2015. Lightweight Automated Planning ToolKiT. `http://lapkt.org/`. Accessed: 2016-09-15.

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

[Shleyfman, Tuisov, and Domshlak 2016] Shleyfman, A.; Tuisov, A.; and Domshlak, C. 2016. Blind search for atari-like online planning revisited. In *Proc. IJCAI*.

[Xie, Müller, and Holte 2014] Xie, F.; Müller, M.; and Holte, R. 2014. Jasper: the art of exploration in greedy best first search. In *Proc. of the 8th Int Planning Competition*.