Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICAPS 2021)

Width-Based Backward Search

Chao Lei and Nir Lipovetzky
School of Computing and Information Systems, The University of Melbourne, Australia
clei1@student.unimelb.edu.au, nir.lipovetzky@unimelb.edu.au

Abstract

It has been shown recently that duality mapping is a viable strategy to turn progression (forward search) into regression (backward search), but the experimental results suggest that the dual versions of standard IPC benchmarks are quite difficult to solve for heuristic search planners. We aim to study the performance of width-based planners over regression. Our experiments show that width-based search can solve dual problems efficiently when the goal state is restricted to single fluent, but it becomes challenging when the goal state contains conjunctive fluents. We then show that the backward version of best-first width-search (BFWS) with the evaluation function f_5, BFWS(f_5), and its polynomial variant, *k*-BFWS(f_5), are not competitive with their forward versions, but can be orthogonal over the IPC benchmarks. Hence, we propose a *front-to-end* bidirectional search *k*-BDWS-e and its *front-to-front* variant by integrating forward and backward *k*-BFWS(f_5) with the additional intersection check between expanded states whose novelty is 1 in the opposite *Close* list. Practical findings on the challenges of regression in classical planning are briefly discussed.

Introduction

Forward heuristic search from the initial state towards one of the goal states is one of the most successful search strategies for satisficing classical planning (Bonet and Geffner 2001; Hoffmann and Nebel 2001; Helmert 2006; Richter and Westphal 2010). Backward heuristic search has been explored to a lesser extent due to asymmetries between forward and backward search (Bonet and Geffner 2001; Alcázar et al. 2013), and the lack of informed heuristics for the latter. Recently, forward width-based search algorithms have shown state-of-the-art performance in the last International Planning Competition (IPC) (Frances, Geffner, and Lipovetzky 2018). These algorithms employ an exploration mechanism based on a structural goal-agnostic notion, *novelty*, which assigns value to the states based on how novel they are with respect to the states already visited by the search strategy (Lipovetzky and Geffner 2012). When novelty is used with goal-aware heuristics in a greedy best-first search, the resulting search algorithms are known as best-first width-search (BFWS) (Lipovetzky and Geffner 2017a,b), the backbone of some of the best performing algorithms in the last IPC. The integration of novelty with forward heuristic search algorithms is an active research area (Katz et al. 2017; Fickert 2018, 2020).

Suda (2013) described a duality translation over STRIPS planning tasks to turn progression into regression, but the dual versions of the IPC domains are quite difficult to solve for modern heuristic and SAT planners such as FF (Hoffmann and Nebel 2001), LAMA (Richter and Westphal 2010), and Mp (Rintanen 2010). Backward search has also been exploited in the context of bidirectional search expanding states in both directions with the aim to solve problems "meeting-in-the-middle", where the search succeeds and terminates when there is an intersection between the frontier of each search direction (Politowski and Pohl 1984; Felner et al. 2010; Alcázar et al. 2013; Alcázar, Fernández, and Borrajo 2014; Kuroiwa and Fukunaga 2020).

Width-based search algorithms have not been used in the context of backward and bidirectional search. The aim of this paper is to explore the impacts of duality in the computation of novelty, and the performance of backward and bidirectional width-based search over the IPC domains. For this, we introduce the modifications needed to run width-based algorithms in regression, and explore different strategies to combine BFWS in bidirectional search.

Background

A STRIPS problem $P = \langle F, O, I, G\rangle$ is made up of a set of boolean facts F, a set of actions O each with a triple $\langle pre, add, del\rangle$, an initial $I \subseteq F$ and goal $G \subseteq F$ specification. P represents the state model of a classical planning problem in a compact form (Haslum et al. 2019). The progression state model $S^P = \langle S, s_0, S_G, A, f, c\rangle$ consists of a set of states $S = 2^F$, the initial state $s_0 = I$, the set of goal states $S_G = \{s \mid G \subseteq s \in S\}$, the subset of actions $A(s) = \{a \mid pre(a) \subseteq s,\ a \in O\}$ applicable in s, the transition function $f(s, a) = s \cup add(a) \setminus del(a)$ and the cost function c. A solution is a sequence of actions mapping the initial state s_0 into one of the goal states $s \in S_G$. The regression state model $S^R = \langle S, s_0, S_G, A, f, c\rangle$ of a STRIPS problem P differs from the progression model S^P, as the set of states S can be made of partial states entailing multiple states from the forward model. In the S^R model, the initial state $s_0 = G$ is the goal G; the set of goal states

$S_G = \{s \mid s \subseteq I\}$ are the partial states entailed by I; the subset of actions $A(s) = \{a \mid del(a) \cap s = \emptyset,\ add(a) \cap s \neq \emptyset\}$ applicable in s are actions *consistent* and *relevant* with s, where no fluent in the delete list appears in s and at least one fluent in the add list appears in s. Finally, the transition function $f(s, a) = s \setminus add(a) \cup pre(a)$ regresses a partial state s given an applicable action $a \in A(s)$ (Bonet and Geffner 2001).

Duality The dual of a STRIPS problem P is defined as $P^d = \langle F, O^d, I^d, G^d \rangle$. In the dual problem, the precondition and delete lists of each action $a \in O$ are swapped to form the dual action $a^d \in O^d$ where $a^d = \langle del, add, pre \rangle$. The dual initial state $I^d = F \setminus G$ is set to the complement of the original goal G, and finally the dual goal state $G^d = F \setminus I$ is set to the complement of the initial state I (Suda 2013). The dual mapping transforms S^P into the regression state model S^R by changing the encoding of the problem P directly.

Width-Based Search Lipovetzky and Geffner (2012) introduced a width parameter $w(P)$ which can characterize the complexity of classical planning instances through the structural goal-agnostic notion of novelty. The novelty of a state s is set to the *size* of the *smallest* tuple of fluents $t \subseteq s$ where s is the first state that makes t true in the search. If s does not contain any new tuples t, the novelty of s is $|F|+1$.

The simplest width-based algorithm to exploit novelty is Iterated Width (IW). IW runs a sequence of bounded IW(i), for $i = 1, \ldots, |F|$ until the problem is solved, where each IW(i) is a breadth-first search pruning states whose novelty is greater than i. For a given problem P, the minimum bound i needed to solve it is called the effective width, $w_e(P)$, which is a lower bound of the real width $w(P)$ characterizing the complexity of the problem. IW has been shown to be competitive when G is restricted to $|G| = 1$ single atoms (Lipovetzky and Geffner 2012), and has excelled over classical planning problems specified by simulators (Lipovetzky, Ramirez, and Geffner 2015).

Lipovetzky and Geffner (2017a) introduced best-first width-search (BFWS), a best-first search combining width-based exploration and goal-directed heuristics through the evaluation function $f = \langle w_{h'}, h \rangle$, where h are one or more heuristic functions, and $w_{h'}$ is the novelty measure. Novelty $w_{h'}(s)$ is only computed with respect to states s' seen before whose $h'(s) = h'(s')$. The evaluation function breaks ties lexicographically, preferring novel states first, and then breaking ties by goal-directed heuristics.

One of the best performing BFWS variants is BFWS(f_5) where the heuristic h is the number of unrealized goals in a state ($\#g$), and the novelty measure $w_{h'}$ is computed under $h' = \langle \#g, \#r \rangle$ where $\#r$ is a counter keeping track of how many relevant fluents R have been achieved along the path to the current state s from s_0. The relevant set R is defined as the fluents that appear in a relaxed plan (Hoffmann and Nebel 2001). R is computed in s_0 and in states that achieve one more goal than their parent state. A state s uses the latest R computed in the path from s_0. BFWS(f_5) is complete, but can be easily turned into a polynomial but incomplete search algorithm by just pruning the states s whose novelty $w_{h'}(s)$ exceeds a bound k, named as k-BFWS(f_5). Both planners manage to outperform the state-of-the-art planner LAMA (Lipovetzky and Geffner 2017a,b), while BFWS(f_5) won the agile track of the last IPC.

I		w_e=1		w_e=2		w_e>2	
P	P^D	P	P^D	P	P^D	P	P^D
37,921	88,856	37%	96%	51%	0%	12%	4%

Table 1: Percentage of single-goal instances with effective width w_e 1, 2, or greater than 2 over STRIPS problems P and their Dual version P^D. I stands for the number of instances.

Bidirectional Search Standard bidirectional search based on best-first search uses $Open_f$ ($Open_b$) and $Close_f$ ($Close_b$) to store generated and expanded states in the forward (backward) direction. Bidirectional search can be categorized into one of the following search strategies: 1) *front-to-end* guides the search forward with h_f, which evaluates the distance from states in $Open_f$ to the closest goal state $s' \in S_G$, and searches backwards evaluating the distance from states in $Open_b$ to the initial state s_0; 2) *front-to-front* estimates the heuristic value of a state s according to how close it is to the opposite search frontier (*Open* list). Bidirectional search terminates when the goal or the initial state has been reached. The search can terminate earlier if the frontiers meet-in-the-middle, which requires checking if $Close_f \cap Close_b \neq \emptyset$ (Felner et al. 2010; Kuroiwa and Fukunaga 2020).

Duality in Width-Based Search

There are 42 domains (1095 instances) using the latest version of each domain from the satisficing tracks of IPCs 1998–2018. All following experiments were conducted on a cloud computer with clock speeds of 2.0 GHz Xeon processors, and processes time or memory out after 30 minutes or 8 GB. All search algorithms are implemented on LAPKT using Metric-FF as an ADL to *Propositional* STRIPS compiler (Ramirez, Lipovetzky, and Muise 2015).

We run IW on dual and original instances to compare their average effective width $w_e(P)$. Each original and dual instance with N goal fluents is split into N instances with atomic goal fluents. Instances whose goal fluent is already true in the initial state are excluded. We summarize below the overall $w_e(P)$ of original and dual problems over single goals. $w_e(P)$ provides an approximation of the actual width $w(P)$, where problems are solved generating O($|F|^{w_e(P)}$) states. Lipovetzky and Geffner (2012) indicated that most benchmark domains appear to have a bounded and small width when goals are restricted to single fluent.

Overall, 37,921 original and 88,856 dual single-goal instances are generated. For original instances, there are 37% with w_e = 1, 51% with w_e = 2, and 12% with $w_e > 2$, while for dual instances there are 96% with w_e = 1, 0% with w_e = 2, and 4% with $w_e > 2$ (Table 1). The only dual single-goal instances whose effective width is greater than 2 are from *Agricola* and *Settlers* where all dual instances in *Agricola* have effective width greater than 2, and 76% of 131 dual instances in *Settlers* have effective width greater than 2.

	IW $\|G\| = 1$		BFWS(f_5) $\|G\| > 1$	
	P	P^D	P	P^D
Instances	37,921	88,856	1095	1095
Solved Instances	35,918	87,830	765	67
Percentage	95%	99%	70%	6%

Table 2: Solved instances by IW over original P and dual P^D single-goal $|G| = 1$ instances, as well as BFWS(f_5) over real conjunctive goals $|G| > 1$.

In Table 2, we also compare the number of solved instances by IW over original and dual single-goal instances, and BFWS(f_5) over 1095 original and dual instances with their real conjunctive goal where $|G| > 1$. IW solves 95% of the single-goal original instances and 99% dual; while BFWS(f_5) solves 70% (765 out of 1095) of the original instances but only 6% (67 out of 1095) of the dual instances.

These results suggest that the width of dual instances is changed as the duality transformation alters the structure of the search space. More precisely, the effective width of single-goal dual instances is much lower than the original ones. Such dual instances can be easily solved by IW, as all single dual goals with $w_e(P) = 1$ can be solved with a single action. However, the dual instances with conjunctive goals are difficult to solve for BFWS(f_5) as plans are longer and the size of the dual initial state ($F \setminus G$) is much larger than the original one (I), making it hard to find novel states. In fact, only $O(|G|)$ states can have novelty 1, and in general only $O(|G| \times |F|^{i-1})$ states can have novelty i. The results over conjunctive goals are in line with the degradation reported by Suda (2013) with LAMA, FF and Mp.

Backward Best-First Width-Search

Given the poor results over the dual conjunctive problems, we instead adapt BFWS to solve the regression state model S^R directly. The definition of novelty is the same in both directions, as it only depends on the syntax of the states, i.e. the state variables. We compute the critical paths heuristic h^2 (Haslum and Geffner 2000; Alcázar and Torralba 2015) from s_0 to extract the set of forward mutex fluent pairs (Blum and Furst 1997). Mutexes are used to prune partial states in S^R unreachable from s_0 in S^P, and hence a generated state s is pruned if it contains a mutex pair $h^2(p, q) = \infty$, $p, q \in s$. The goal counter instead of keeping track of the number of unrealized forward goals $g \in G$ in progression, $\#g(s) = |s \cap I| + |s \setminus I|$ keeps track of the number of initial state fluents I achieved, as well as the number of fluents that still have to be removed from s to reach one of the regression goal states. The goal counter is further strengthened by creating an *I-ordering* $p < q$ of fluents $p, q \in I$ when all actions requiring p *edelete* q. In regression, an action a *edeletes* a fluent q if $q \in add(a)$ or $\exists_{p \in pre(a) \cup del(a)} h^2(p, q) = \infty$. This I-ordering graph refines the goal counter $\#g(s)$ by counting as achieved fluents $p \in s, p \in I$ whose precedences are satisfied in s. Finally, in order to compute the counter $\#r$ used in novelty $w_{\#g,\#r}$, the set of h^{max} forward best supporters (Keyder and Geffner 2008) are computed once from s_0 in S^P in contrast with progression, where BFWS recomputes the best supporters every time a new set of relevant fluents R is required. R is defined in a state s by computing a relaxed plan that uses s_0 best supporters to reach s, marking as relevant the fluents that appear in the preconditions of actions in the relaxed plan.

Domain	F-k	B-k	F	B
Childsnack14 (20)	0	**6**	2	**4**
Cybersec (30)	15	10	0	**10**
Floortile14 (20)	2	**20**	2	**20**
Parcprinter11 (20)	13	**18**	13	**18**
Scanalyzer11 (20)	18	**20**	18	**20**
Termes18 (20)	1	**2**	10	8
Trucks (30)	7	**14**	9	**14**
Total coverage (1095)	734	372	765	391
Average time	13.39	46.26	13.10	69.53
Average quality	154.93	176.78	167.13	189.55

Table 3: Solved instances by backward (B) vs. forward (F) k-BFWS(f_5) and BFWS(f_5). Only domains where (B) outperform (F) are shown.

Table 3 compares backward BFWS(f_5) and k-BFWS(f_5) with their forward versions, where k = 2. Backward k-BFWS(f_5) solves 372 problems, a large improvement over the coverage of forward BFWS(f_5) on dual problems, which solves only 67. Forward k-BFWS(f_5) solves 734 of 1095 instances, but significant differences occur in *Childsnack, Parcprinter, Scanalyzer, Termes*, and *Trucks* where backward search solves more instances than forward search. The largest difference occurs in *Floortile*, where forward search solves 2 instances while backward search solves 20. Forward BFWS(f_5) solves 31 more instances than forward k-BFWS(f_5), and backward BFWS(f_5) solves 19 more instances than backward k-BFWS(f_5). In terms of average time and plan quality over the instances solved by forward and backward search, backward search performs worse than forward. The difference in time is large, mainly due to *Visitall* domain.

These results show that backward BFWS algorithms still perform worse than their forward counterparts, but they can be orthogonal over 6 domains that are typically hard for state-of-the-art planners. The question that we address next is how to integrate forward and backward search to offset the weaknesses of each search direction.

Width-Based Bidirectional Search

We propose two k-BFWS(f_5) bidirectional search algorithms, a *front-to-end* bidirectional search strategy (k-BDWS-e) and a *front-to-front* version (k-BDWS-f) where the last expanded state of one direction is regarded as the goal of the opposite direction, triggering a recomputation of the evaluation function over nodes in the *Open* list. Both versions, collectively referred to as k-BDWS, alternate 1-step in each direction, but if one direction stops searching the other takes all the turns. k-BDWS returns no solution if both directions stop with no solution, maintains independent novelty measures for each direction, and terminates with a solution if any direction expands a state $s \in S_G$ in the goal region

Domain	F	B	*FB*		*Dual*	*Dual-FB*		k-BDWS-e			k-BDWS-e *Random*			k-BDWS-e *Head*			k-BDWS-e *Close*		
	S	S	S	F / B	S	S	B	S	F / B / M	MM	S	F / B / M	MM	S	F / B / M	MM	S	F / B / M	MM
Airport (50)	37	25	**46**	37 / 9	49	49	-	27	1 / - / 26	0.18	27	5 / 5 / 17	0.20	27	7 / 15 / 5	0.35	24	- / - / 24	0.17
Barman14 (20)	20	-	20	20 / -	20	20	-	20	19 / - / 1	0.01	3	3 / - / -	-	20	20 / - / -	-	1	- / - / 1	0.03
Blocks World (50)	30	30	**45**	30 / 15	30	30	-	30	- / - / 30	0.11	30	8 / 2 / 20	0.21	24	14 / 5 / 5	0.34	25	- / - / 25	0.17
Childsnack14 (20)	-	**6**	**2**	- / 2	10	10	-	**6**	- / 6 / -	-	**2**	- / 2 / -	-	**6**	- / 6 / -	-	**2**	- / 2 / -	-
Cybersec (30)	15	10	**19**	15 / 4	30	30	-	12	- / 2 / 10	0.03	4	- / 2 / 2	0.08	12	2 / 10 / -	-	4	- / 2 / 2	0.08
Data-netw18 (20)	8	-	8	8 / -	8	8	-	8	7 / - / 1	0.01	5	5 / - / -	-	7	7 / - / -	-	4	2 / - / 2	0.02
Depot (20)	20	3	20	20 / -	20	20	-	20	16 / - / 4	0.15	20	18 / - / 2	0.14	20	20 / - / -	-	20	15 / - / 5	0.08
Driverlog (20)	20	20	20	20 / -	20	20	-	20	15 / - / 5	0.10	20	19 / - / 1	0.36	20	20 / - / -	-	20	15 / - / 5	0.10
Elevators11 (20)	20	13	20	20 / -	20	20	-	20	14 / - / 6	0.01	10	10 / - / -	-	20	20 / - / -	-	10	9 / - / 1	0.02
Floortile14 (20)	2	**20**	**20**	2 / 18	2	20	**18**	**20**	- / 19 / 1	0.01	**20**	- / 20 / -	-	**20**	- / 20 / -	-	**20**	- / 19 / 1	0.01
Ged14 (20)	19	-	19	19 / -	17	17	-	16	16 / - / -	-	15	15 / - / -	-	18	18 / - / -	-	15	15 / - / -	-
Gripper (20)	20	20	20	20 / -	20	20	-	20	- / - / 20	0.02	20	- / 15 / 5	0.03	20	- / 20 / -	-	20	- / - / 20	0.02
Hiking14 (20)	3	2	3	3 / -	11	11	-	3	1 / 2 / -	-	-	- / - / -	-	**5**	3 / 2 / -	-	1	- / - / 1	0.10
Logistics (50)	44	21	44	44 / -	44	44	-	30	28 / - / 2	0.03	30	27 / 1 / 2	0.01	37	35 / 2 / -	-	17	1 / - / 16	0.05
Mprime (35)	33	1	33	33 / -	35	35	-	31	29 / - / 2	0.03	27	26 / - / 1	0.06	31	31 / - / -	-	29	26 / - / 3	0.05
Mystery (30)	19	7	19	19 / -	19	20	**1**	18	11 / - / 7	0.16	15	15 / - / -	-	18	18 / - / -	-	16	11 / - / 5	0.19
Nomystery11 (20)	13	10	13	13 / -	17	17	-	11	- / - / 11	0.05	10	5 / 4 / 1	0.01	11	6 / 5 / -	-	9	- / - / 9	0.08
Openstacks14 (20)	15	-	15	15 / -	19	19	-	-	- / - / -	-	-	- / - / -	-	-	- / - / -	-	-	- / - / -	-
Parcprinter11 (20)	13	**18**	**18**	13 / 5	14	18	**4**	**18**	- / - / 18	0.02	**16**	3 / 11 / 2	-	**18**	3 / 15 / -	-	**16**	- / - / 16	0.02
Parking14 (20)	20	-	20	20 / -	20	20	-	20	20 / - / -	-	20	20 / - / -	-	20	20 / - / -	-	20	19 / - / 1	0.03
Pathways (30)	24	5	24	24 / -	29	29	-	23	- / - / 23	0.07	22	5 / - / 17	0.09	24	21 / 1 / 2	0.25	22	- / - / 22	0.09
Pegsol11 (20)	9	1	9	9 / -	20	20	-	**10**	7 / - / 3	0.06	9	9 / - / -	-	9	9 / - / -	-	**11**	2 / - / 9	0.09
Pipesworld06 (50)	30	6	30	30 / -	32	32	-	30	26 / - / 4	0.22	30	27 / - / 3	0.22	30	29 / 1 / -	-	30	25 / - / 5	0.19
Pipes-notan (50)	50	15	50	50 / -	49	49	-	50	43 / - / 7	0.06	49	49 / - / -	-	49	48 / 1 / -	-	45	38 / - / 7	0.06
Pipes-tan (50)	30	6	30	30 / -	32	32	-	30	26 / - / 4	0.11	30	28 / - / 2	0.20	30	29 / 1 / -	-	30	25 / - / 5	0.12
Rovers (20)	20	19	20	20 / -	20	20	-	20	12 / - / 8	0.10	20	13 / 3 / 4	0.29	20	13 / 6 / 1	0.34	20	12 / - / 8	0.12
Scanalyzer11 (20)	18	**20**	**20**	18 / 2	20	20	-	**20**	7 / 5 / 8	0.02	**20**	5 / 7 / 8	0.19	**20**	5 / 9 / 6	0.25	**20**	4 / 2 / 14	0.13
Snake18 (20)	11	-	11	11 / -	7	7	-	6	6 / - / -	-	3	3 / - / -	-	6	6 / - / -	-	3	2 / - / 1	0.02
Sokoban11 (20)	5	1	5	5 / -	18	18	-	**6**	- / - / 6	0.06	2	- / - / 2	0.18	5	4 / - / 1	0.24	2	- / - / 2	0.18
Spider18 (20)	13	-	13	13 / -	14	14	-	10	9 / - / 1	-	8	8 / - / -	-	9	9 / - / -	-	7	6 / - / 1	-
Storage (30)	29	14	29	29 / -	28	28	-	29	27 / - / 2	0.26	27	25 / - / 2	0.25	29	27 / 1 / 1	0.20	27	24 / - / 3	0.26
Termes18 (20)	1	**2**	**2**	1 / 1	9	9	-	**2**	- / - / 2	0.01	**2**	- / 2 / -	-	**2**	- / 2 / -	-	-	- / - / -	-
Thoughtful14 (20)	20	5	20	20 / -	20	20	-	20	12 / - / 8	0.01	17	17 / - / -	-	20	20 / - / -	-	16	5 / - / 11	0.02
Tpp (30)	30	10	30	30 / -	30	30	-	30	24 / - / 6	0.08	24	20 / - / 4	0.37	30	29 / - / 1	0.07	20	14 / - / 6	0.17
Transport14 (20)	7	-	7	7 / -	8	8	-	6	6 / - / -	-	2	2 / - / -	-	6	6 / - / -	-	2	2 / - / -	-
Trucks (30)	7	**14**	**10**	7 / 3	14	14	-	**13**	2 / - / 11	0.06	**12**	2 / 9 / 1	0.02	**13**	2 / 11 / -	-	**10**	1 / - / 9	0.09
Visitall14 (20)	19	17	**20**	19 / 1	20	20	-	18	- / - / 18	0.34	18	- / - / 18	0.33	18	8 / - / 10	0.33	18	- / - / 18	0.32
Woodwork11 (20)	20	11	20	20 / -	20	20	-	20	19 / 1 / -	-	20	19 / 1 / -	-	20	19 / 1 / -	-	20	19 / 1 / -	-
Zenotravel (20)	20	20	20	20 / -	20	20	-	20	7 / 6 / 7	0.09	20	7 / 11 / 2	0.02	20	8 / 12 / -	-	20	6 / 5 / 9	0.15
Summary (1095)	734	372	794	-/60	841	864	23	713	-/41/262	-	629	-/95/116	-	714	-/146/32	-	596	-/31/267	-

Table 4: Results over *Width-Based Bidirectional Search* planners. S stands for solved instances; F/B stand for forward/backward coverage; M is the number of solved instances where search meets in the middle, and MM is the Meet Metric.

or $s \in \mathit{Close}_x,\ x \in \{f, b\}$ meets the opposite direction, i.e. when the frontiers meet. Checking if frontiers meet is an expensive operation, as every expanded state has to be tested for membership over an increasing set. Therefore, in k-BDWS we only check with respect to the novelty 1 frontier of the opposite direction, states $s' \in \mathit{Close}$ s.t. $w(s') = 1$.

We report experiments with several variants of bidirectional search and omit *Agricola, Caldera, and Settlers* domains as no solutions can be found. In Table 4, F stands for forward k-BFWS(f_5), B for the backward counterpart; *FB* is a simple integration where F is run first and then B runs only if F stops with no solution; k-BDWS-e *Head* checks if the search meets-in-the-middle by only checking intersection with the last expanded state in the opposite direction; k-BDWS-e *Random* checks with a random state in the *Close* list; k-BDWS-e *Close* checks with respect to the full *Close* list. We also test k-BDWS-f, k-BDWS-f *Head* and k-BDWS-f *Close* but omit them from the table. The meet-metric reports the minimum ratio of actions in a plan that belong to either direction (Kuroiwa and Fukunaga 2020). E.g. if both directions contribute equally the metric is 0.5, if one direction contributes 0.2 and the other 0.8, the metric is 0.2.

k-BDWS-e solves 713 instances, one fewer than the *Head* version, 84 more than *Random* and 117 more than the *Close* version. Among the solved instances, k-BDWS-e meets-in-the-middle on 36.75 % of instances, 8 times higher than *Head*, 2 times higher than *Random* while *Close* meets-in-the-middle on 44.80%. These results confirm that checking over only the $w(s) = 1$ states in *Close* is a good trade-off in terms of coverage and meeting-in-the-middle. k-BDWS-e coverage is higher than k-BFWS(f_5) (first column) in 8 domains, carrying the advantage witnessed in previous sec-

tion. k-BDWS-f solves 291 instances, *Head* 271, and *Close* 276. Among the solved instances, k-BDWS-f meets-in-the-middle on 88.66 % of instances, 3 times higher than *Head*, while *Close* meets-in-the-middle on 95.29 % of instances. Clearly, *front-to-front* increases the possibility of meeting-in-the-middle, but the computational overhead decreases the coverage substantially. *FB* solves the most instances, 794, 60 problems more than F over 10 different domains. Considering all variants, if coverage is preferred, *FB* is the best option, whereas k-BDWS-e represents the best trade-off between coverage and meeting-in-the-middle. These results clearly show that forward and backward search can be orthogonal. This is witnessed further by the results over *Dual-BFWS*, runner-up on the last satisficing-track (IPC-2018). *Dual-BFWS* runs first a forward F with $k = 1$, and a second complete BFWS (Lipovetzky and Geffner 2017a) if no solution is found (*Dual* column). We show how running *FB* with $k = 1$ first instead improve the state-of-the-art (*Dual-FB*). *FB* with $k = 1$ can be thought of as a quick preprocessing step that could be integrated in every state-of-the-art planner as it either solves a problem or fails fast.

Discussion

The IPC benchmarks have been modeled with forward search in mind as certain domains are artificially challenging for backward search. The open-world assumption over G means the goal can be left underspecified even when the modeler might be interested in a single-goal state. We experimented with closure techniques over G which decreased the generated nodes over *Blocks World*, *Parking* and *Elevators*. Invariants are extremely important to prune unreachable regression states, but they are seldom specified in PDDL. *Elevators* becomes challenging in backward search, as h^2 mutexes cannot detect unreachable partial states where the total passengers in elevators and at each floor, exceeds the total number of passengers defined in the problem. Finding all mutexes is NP-*hard* (Fišer and Komenda 2018), but specifying useful invariants should be easily achieved through object fluents (Haslum et al. 2019) and Functional STRIPS (Geffner 2000; Francès and Geffner 2015). Finally, action schemas are also underspecified for backward search, in *Parking*, the action *move-car-to-car* has the precondition *(behind ?car1 ?car2)*, but the action does not specify *(not (= ?car1 ?car2))*, which creates plenty of spurious states where a car is behind itself. This does not affect forward search as this fluent cannot be added.

Conclusion

We showed that the effective width of single-goal dual problems is reduced significantly because most plans have length one, however width-based search cannot solve the majority of dual problems with conjunctive goal fluents due to the increased width and longer plans. Then we showed that width-based algorithms can be adapted to work over the regression model directly, and despite solving fewer problems than progression, they are orthogonal when integrated over different bidirectional algorithms and state-of-the-art planners, as the combinatorial structure of certain domains exists only in the forward direction, e.g. *Floortile*. Although current research suggests that regression is harder than progression, regression still plays an important role in classical planning that is absent in satisficing planners since HSPr (Bonet and Geffner 2001), 20 years ago. We hope to investigate further modeling guidelines for backward search, and push for wider adoption in existing state-of-the-art planners.

Acknowledgments

This research was supported by use of The University of Melbourne Research Cloud, a collaborative Australian research platform supported by the National Collaborative Research Infrastructure Strategy (NCRIS).

References

Alcázar, V.; Borrajo, D.; Fernández, S.; and Fuentetaja, R. 2013. Revisiting Regression in Planning. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, IJCAI, 2254–2260.

Alcázar, V.; Fernández, S.; and Borrajo, D. 2014. Analyzing the Impact of Partial States on Duplicate Detection and Collision of Frontiers. In *Proceedings of the 24th International Conference on Automated Planning and Scheduling*, ICAPS, 350–354.

Alcázar, V.; and Torralba, A. 2015. A Reminder about the Importance of Computing and Exploiting Invariants in Planning. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling*, ICAPS, 2–6.

Blum, A. L.; and Furst, M. L. 1997. Fast Planning through Planning Graph Analysis. *Artificial intelligence* 90(1–2): 281–300.

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

Felner, A.; Moldenhauer, C.; Sturtevant, N.; and Schaeffer, J. 2010. Single-Frontier Bidirectional Search. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, AAAI, 59–64.

Fickert, M. 2018. Making Hill-Climbing Great Again through Online Relaxation Refinement and Novelty Pruning. *SOCS* 158–162.

Fickert, M. 2020. A Novel Lookahead Strategy for Delete Relaxation Heuristics in Greedy Best-First Search. In *Proceedings of the 30th International Conference on Automated Planning and Scheduling*, ICAPS, 119–123.

Fišer, D.; and Komenda, A. 2018. Fact-Alternating Mutex Groups for Classical Planning. *Journal of Artificial Intelligence Research* 61(1): 475–521.

Francès, G.; and Geffner, H. 2015. Modeling and Computation in Planning: Better Heuristics from More Expressive Languages. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling*, ICAPS, 70–78.

Frances, G.; Geffner, H.; and Lipovetzky, N. 2018. Best-first width search in the IPC 2018: Complete, simulated, and polynomial variants. *IPC2018–Classical Tracks* 22–26.

Geffner, H. 2000. Functional STRIPS: a more flexible language for planning and problem solving. In *Logic-based artificial intelligence*, 187–209.

Haslum, P.; and Geffner, H. 2000. Admissible Heuristics for Optimal Planning. In *Proceedings of the 5th International Conference on Artificial Intelligence Planning Systems*, AIPS, 140–149.

Haslum, P.; Lipovetzky, N.; Magazzeni, D.; and Muise, C. 2019. An introduction to the planning domain definition language. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 13(2): 1–187.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research* 26(1): 191–246.

Hoffmann, J.; and Nebel, B. 2001. The FF Planning System: Fast Plan Generation through Heuristic Search. *Journal of Artificial Intelligence Research* 14(1): 253–302.

Katz, M.; Lipovetzky, N.; Moshkovich, D.; and Tuisov, A. 2017. Adapting Novelty to Classical Planning as Heuristic Search. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling*, ICAPS, 172–180.

Keyder, E.; and Geffner, H. 2008. Heuristics for Planning with Action Costs Revisited. In *Proceedings of the 18th European Conference on Artificial Intelligence*, ECAI, 588–592.

Kuroiwa, R.; and Fukunaga, A. 2020. Front-to-Front Heuristic Search for Satisficing Classical Planning. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, IJCAI, 4098–4105.

Lipovetzky, N.; and Geffner, H. 2012. Width and Serialization of Classical Planning Problems. In *Proceedings of the 20th European Conference on Artificial Intelligence*, ECAI, 540–545.

Lipovetzky, N.; and Geffner, H. 2017a. Best-First Width Search: Exploration and Exploitation in Classical Planning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, AAAI, 3590–3596.

Lipovetzky, N.; and Geffner, H. 2017b. A Polynomial Planning Algorithm that Beats LAMA and FF. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling*, ICAPS, 195–199.

Lipovetzky, N.; Ramirez, M.; and Geffner, H. 2015. Classical Planning with Simulators: Results on the Atari Video Games. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, IJCAI, 1610–1616.

Politowski, G.; and Pohl, I. 1984. D-Node Retargeting in Bidirectional Heuristic Search. In *Proceedings of the 4th AAAI Conference on Artificial Intelligence*, AAAI, 274–277.

Ramirez, M.; Lipovetzky, N.; and Muise, C. 2015. Lightweight Automated Planning ToolKiT. http://lapkt.org/. Accessed: 2020.

Richter, S.; and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *Journal of Artificial Intelligence Research* 39(1): 127–177.

Rintanen, J. 2010. Heuristics for Planning with SAT. In *Proceedings of the 16th International Conference on Principles and Practice of Constraint Programming*, CP, 414–428.

Suda, M. 2013. Duality in STRIPS planning. *CoRR* abs/1304.0897.