Technical Appendix

[1] **Anubhav Singh,** [1] **Nir Lipovetzky,** [2] **Miquel Ramirez,** [3] **Javier Segovia-Aguas**
[1] School of Computing and Information Systems, University of Melbourne, Australia
[2] Electrical and Electronic Engineering, University of Melbourne, Australia
[3] Dept. Information and Communication Technologies, Universitat Pompeu Fabra, Spain
anubhavs@student.unimelb.edu.au, {nir.lipovetzky, miquel.ramirez}@unimelb.edu.au, javier.segovia@upf.edu

1 Implementation details

Hash Functions We used the *std::hash* method from the standard C++ library to compute the hash values. The hash value were combined using the method *hash_combine* described in the proposal for C++ standard *N3876* (Josuttis 2014).

Seeds used for multiple runs of planners using *randomization* We used the sequence 101, 204, 307 and 410, formed by concatenating two AP sequences of the form $x_1 = 1 + (n-1)$ and $x_2 = 01 + 3(n-1)$, for $n \in [1, 4]$.

2 Additional results

2.1 P_2 (BFWS(f_5)) vs. pI-$P_{\bar{\omega}}$AC : Coverage with the Agile track constraints on time and memory

From Table. 1, we observe that the coverage of pI-$P_{\bar{\omega}}$AC is 59 higher than P_2 (BFWS(f_5)), one of the best performing to-date in the Agile track (IPC 2018). The experiments were performed with the *time* and *memory limit* of 300 sec and 8 GB respectively.

	BFWS(f_5)	pI-$P_{\bar{\omega}}$AC
BFWS(f_5)	0	78
pI-$P_{\bar{\omega}}$AC	19	0

Table 1: Comparing BFWS(f_5) and pI-$P_{\bar{\omega}}$AC. The value in a cell represents the number of instances which were solved by the *col* planner but not the *row* planner, with the *time limit* of 300 sec and memory limit of 8 GB.

2.2 Pairwise comparison of plan cost between planners using Approximate Novelty Search and P_2 (BFWS(f_5))

From Fig. 1 and 2, we note that the plan length remain similar for different configuration of BFWS($\hat{f}_5$) planners using approximate novelty search.

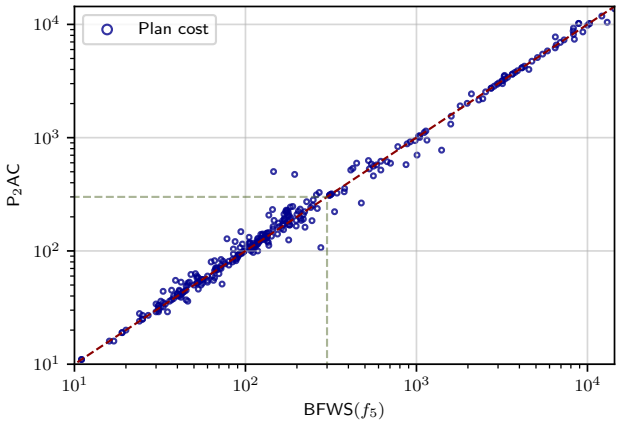

Figure 1: Pairwise comparison of cost between BFWS(f_5) and P_2AC on instances from every IPC satisficing benchmark.

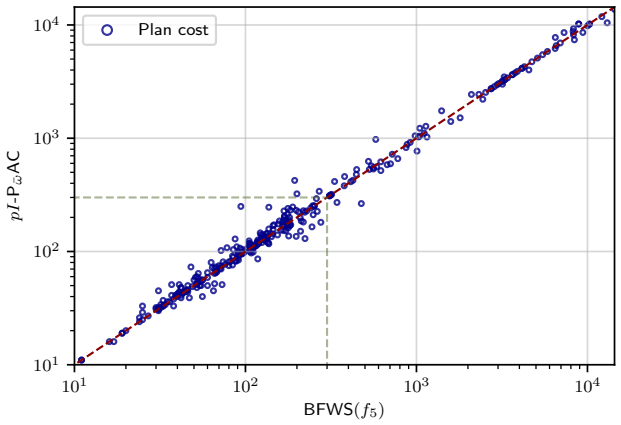

Figure 2: Pairwise comparison of cost between BFWS(f_5) and pI-$P_{\bar{\omega}}$AC on instances from every IPC satisficing benchmark.

2.3 Complete table of results

In Table. 2, we show the *coverage* on all the domains which were part of our experiments. We included all the domains

domain	B1	B2	P_2	P_2A	P_2AC	P_3AC	P_3AC	p-P_2	B3	p-P_2A	p-P_3A	pI-$P_{\bar{\omega}}$AC	pI-$\mathcal{L}_{\bar{\omega}}$AC
agricola-sat18	12	8	11	11±1.0	12±1.7	15±0.8	16±1.3	10	15	10±0.6	15±0.8	16±1.3	12±0.5
airport	34	47	46	46±0.6	46±0.6	44±0.0	44±0.6	46	47	46±0.6	45±1.0	46±0.6	46±0.5
assembly	30	30	30	30±0.6	29±1.0	30±0.6	30±0.6	30	30	30±0.5	30±0.0	30±0.0	30±0.0
barman-sat14-strips	20	20	20	20±0.0	20±0.0	20±0.0	20±0.0	20	20	20±0.0	20±0.0	20±0.0	20±0.0
blocks	35	35	35	35±0.0	35±0.5	35±0.0	35±0.0	35	35	35±0.0	35±0.0	35±0.0	35±0.5
caldera-sat18	16	20	15	16±1.0	20±0.5	16±1.0	18±0.5	19	20	20±0.5	18±0.5	20±0.5	20±0.0
cavediving	7	7	7	7±0.0	8±0.6	8±0.0	8±0.5	1	8	2±2.9	8±0.5	9±1.0	8±0.5
childsnack-sat14-strips	6	10	0	4±1.3	5±1.7	3±1.9	6±0.6	0	2	5±0.5	6±0.6	8±1.3	8±1.3
citycar-sat14-adl	5	20	5	5±0.0	20±0.0	5±0.0	20±0.6	20	20	20±0.5	5±0.0	20±0.0	20±0.0
data-network-sat18	13	11	9	12±1.7	19±1.0	11±2.5	18±0.5	16	14	17±1.0	16±0.6	18±0.6	18±0.6
depot	20	22	22	22±0.0	22±0.0	22±0.0	22±0.0	22	22	22±0.0	22±0.0	22±0.0	22±0.0
driverlog	20	20	20	20±0.0	20±0.0	20±0.0	20±0.0	20	20	20±0.0	20±0.0	20±0.0	20±0.0
elevators-sat11-strips	20	20	20	20±0.0	20±0.0	20±0.0	20±0.0	20	20	20±0.0	20±0.0	20±0.0	20±0.0
flashfill-sat18	14	16	12	14±2.4	14±0.6	14±2.2	14±1.0	15	9	14±2.4	14±1.9	14±1.0	14±1.0
floortile-sat14-strips	2	2	1	2±0.5	2±0.0	2±0.0	2±0.0	0	1	1±0.0	2±0.5	2±0.0	2±0.0
freecell	80	80	80	80±0.0	80±0.0	80±0.0	80±0.0	80	80	80±0.0	80±0.0	80±0.0	80±0.0
ged-sat14-strips	20	20	20	20±0.0	20±0.0	20±0.0	20±0.0	20	20	20±0.0	20±0.0	20±0.0	20±0.0
grid	5	5	5	5±0.0	5±0.0	5±0.0	5±0.0	5	5	5±0.0	5±0.0	5±0.0	5±0.0
gripper	20	20	20	20±0.0	20±0.0	20±0.0	20±0.0	20	20	20±0.0	20±0.0	20±0.0	20±0.0
hiking-sat14-strips	20	12	12	14±2.1	8±0.8	18±1.0	20±0.5	9	13	12±1.8	20±0.0	20±0.0	19±0.5
logistics00	28	28	28	28±0.0	28±0.0	28±0.0	28±0.0	28	28	28±0.0	28±0.0	28±0.0	28±0.0
maintenance-sat14-adl	11	17	17	16±0.5	16±0.6	16±0.5	17±0.5	17	17	16±0.5	16±0.5	17±0.5	17±0.5
miconic	150	150	150	150±0.0	150±0.0	150±0.0	150±0.0	150	150	150±0.0	150±0.0	150±0.0	150±0.0
movie	30	30	30	30±0.0	30±0.0	30±0.0	30±0.0	30	30	30±0.0	30±0.0	30±0.0	30±0.0
mprime	35	35	32	30±0.6	35±0.0	31±0.5	34±0.8	35	35	35±0.0	32±0.8	35±0.0	35±0.0
mystery	19	19	19	19±0.0	19±0.0	19±0.5	19±0.5	19	18	19±0.0	19±0.5	19±0.5	19±0.5
nomystery-sat11-strips	11	19	13	14±1.0	12±1.0	13±0.5	14±1.0	13	13	12±1.7	14±0.6	15±1.0	18±1.4
nurikabe-sat18	9	14	16	14±0.6	15±1.3	14±0.6	15±2.1	16	16	14±0.5	14±0.6	15±1.0	14±0.6
openstacks-sat14-strips	20	20	20	20±0.0	20±0.0	20±0.0	20±0.5	20	20	20±0.0	20±0.0	20±0.0	12±1.0
organic-synthesis-split-sat18	12	11	5	6±0.5	3±0.8	7±1.0	5±1.4	4	3	4±0.5	6±1.0	7±0.0	6±0.5
parcprinter-sat11-strips	20	16	9	5±1.0	5±1.9	5±0.8	6±1.0	9	16	6±1.0	5±1.0	8±0.0	6±0.5
parking-sat14-strips	20	20	20	20±0.0	20±0.0	20±0.0	20±0.0	20	20	20±0.0	20±0.0	20±0.0	20±0.0
pathways-noneg	24	30	23	30±0.6	29±1.5	30±0.6	29±0.5	24	27	30±0.5	30±0.6	30±0.0	30±0.0
pegsol-sat11-strips	20	20	20	20±0.0	20±0.5	20±0.0	20±0.0	5	20	12±1.5	18±0.5	20±0.0	20±0.0
pipesworld-notankage	43	50	50	50±0.0	50±0.0	50±0.0	50±0.0	50	50	50±0.0	50±0.0	50±0.0	50±0.0
pipesworld-tankage	43	38	43	42±0.5	42±0.6	42±0.5	43±0.8	41	39	41±1.5	42±0.5	42±1.2	43±1.5
psr-small	50	50	48	49±0.5	49±0.5	50±0.0	50±0.0	31	46	34±1.3	43±0.8	49±0.5	48±0.6
rovers	40	37	39	40±0.0	40±0.0	40±0.0	40±0.0	39	38	40±0.0	40±0.0	40±0.0	40±0.0
satellite	36	31	27	30±0.8	32±0.6	30±0.5	30±0.0	27	31	32±0.5	30±0.5	34±0.6	34±0.6
scanalyzer-sat11-strips	20	20	20	20±0.0	20±0.0	20±0.0	20±0.0	20	20	20±0.0	20±0.0	20±0.0	20±0.5
schedule	150	149	149	149±1.0	149±0.8	149±1.0	150±0.6	149	149	149±1.0	149±1.0	149±1.0	149±1.0
settlers-sat18	18	8	7	6±1.0	12±0.6	6±1.3	10±0.0	10	11	9±1.0	6±1.0	12±0.6	17±1.3
snake-sat18	5	12	19	16±0.5	15±0.8	17±0.5	17±0.5	18	3	16±0.5	17±0.5	20±0.5	20±0.5
sokoban-sat11-strips	19	17	14	15±0.5	10±1.0	16±0.5	14±0.8	6	13	4±0.6	11±0.5	16±0.6	16±0.6
spider-sat18	16	14	13	15±1.0	14±1.0	15±1.0	14±1.3	13	11	15±1.0	15±1.0	14±1.0	15±0.5
storage	20	28	29	30±0.6	30±0.6	30±0.6	30±0.5	30	29	30±0.6	30±0.6	30±0.5	30±0.6
termes-sat18	16	9	9	10±0.0	8±0.6	9±1.0	8±1.3	1	6	2±0.5	6±1.9	7±1.4	10±1.3
tetris-sat14-strips	16	16	20	20±0.0	20±0.0	20±0.0	20±0.0	20	18	20±0.0	20±0.0	20±0.0	20±0.0
thoughtful-sat14-strips	15	20	20	20±0.0	20±0.0	20±0.0	20±0.0	20	20	20±0.0	20±0.0	20±0.0	20±0.0
tidybot-sat11-strips	17	18	19	20±0.0	20±0.5	20±0.0	20±0.0	20	20	20±0.0	20±0.0	20±0.0	19±0.5
tpp	30	29	29	30±0.6	30±0.0	29±0.0	30±0.0	30	30	30±0.0	30±0.0	30±0.0	30±0.0
transport-sat14-strips	16	20	20	20±0.0	20±0.0	20±0.0	20±0.0	20	20	20±0.0	20±0.0	20±0.0	20±0.0
trucks-strips	18	16	9	9±0.8	9±1.3	9±0.8	10±1.4	11	8	12±1.8	11±1.3	12±1.3	12±0.8
visitall-sat14-strips	20	20	20	20±0.5	20±0.0	20±0.5	20±0.0	20	20	20±0.5	20±0.5	20±0.0	20±0.0
woodworking-sat11-strips	20	20	20	20±0.0	20±0.0	20±0.0	20±0.0	20	20	20±0.0	20±0.0	20±0.0	20±0.0
zenotravel	20	20	20	20±0.0	20±0.0	20±0.0	20±0.0	20	20	20±0.0	20±0.0	20±0.0	20±0.0
Total	1456	1496	1436	1455±8.7	1476±4.2	1463±8.9	1502±4.9	1414	1456	1438±5.9	1462±8.0	1524±2.5	1516±5.0

Table 2: Coverage over all satisficing benchmarks from IPCs: *complete* - B1: LAMA-*first*, B2: DUAL-BFWS, 'P...', and *polynomial incomplete* - B3: $\langle 1, 2\text{-C-M} \rangle$ and 'p-P...'. $P_{\bar{\omega}}$ refers to BFWS(f_5) planner with $\mathcal{W} = [1, \bar{\omega} + 1]$, $\mathcal{L}_{\bar{\omega}}$ is BFWS($f_{\mathcal{L}}$) , which uses *Landmarks* , 'I-' stands for *Iterative*, 'A' for *approximate*, and 'C' for *control* over *open list*. The *mean coverage* is shown along with the *standard deviation* for the planners using random sampling over 4 different seeds.

from IPC satisficing benchmarks, chosing the one from latest IPC in case a domain occurred in multiple IPCs.

3 About Benchmarks

We used the collection of PDDL files from the IPC's *github* repository (IPC 1998-2018), and generated the list of satisficing domains using *Downward Benchmarks'* script (Dow 2018). The PDDL files were updated to enforce compatibility with *Tarski's* (Frances and Ramirez 2019) parser in the following manner

1. Lowercased the letters in the PDDL files.
2. *:action-costs* added to *:requirements* in *floortile* and *settlers* domains.
3. Keyword 'max' changed to 'max1' in *pathways-noneg*.
4. The 'type' *pieces* defined as *object* type in *tetris*.
5. In *tidybot* domain, renamed the object *cart* as *cart1* to avoid conflict with the 'type' *cart*. Also, removed the 'effect' *(not (base-obstacle ?cx ?cy2))* from the actions *base-cart-up* and *base-cart-down* as it caused conflict with the 'effect' *(base-obstacle ?cx ?cy2)* in *LAPKT* (Ramirez, Lipovetzky, and Muise 2015) . This change does not affect the search tree.

The *PDDL files* and *scripts*, used to run experiments with *Downward Lab's* (Seipp et al. 2017) experiment module, are included in the source code zip.

References

1998-2018. International Planning Competition – Classical Tracks. https://github.com/AI-Planning/classical-domains/. Accessed: 19-01-2020.

2018. Downward Benchmarks. https://github.com/aibasel/downward-benchmarks. Accessed: 19-01-2020.

Frances, G.; and Ramirez, M. 2019. Tarski - An AI Planning Modeling Framework. https://github.com/aig-upf/tarski. Accessed: 2019-11-11.

Josuttis, N. 2014. Convenience Functions to Combine Hash Values. http://open-std.org/JTC1/SC22/WG21/docs/papers/2014/n3876.pdf. Accessed: 2019-11-01.

Ramirez, M.; Lipovetzky, N.; and Muise, C. 2015. Lightweight Automated Planning ToolKiT. http://lapkt.org/. Accessed: 2020-01-20.

Seipp, J.; Pommerening, F.; Sievers, S.; and Helmert, M. 2017. Downward Lab. https://doi.org/10.5281/zenodo.790461. doi:10.5281/zenodo.790461. URL https://doi.org/10.5281/zenodo.790461.