The Powerlifted Planning System in the IPC 2023

Augusto B. Corrêa[1], Guillem Francès[2], Markus Hecher[3], Davide Mario Longo[4], Jendrik Seipp[5]
[1]University of Basel, Switzerland
[2]Independent Researcher
[3]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, USA
[4]TU Wien, Institute of Logic and Computation, Austria
[5]Linköping University, Sweden
augusto.blaascorrea@unibas.ch, guillem.frances@gmail.com, hecher@mit.edu,
davidem.longo@gmail.com, jendrik.seipp@liu.se

In this planner abstract, we introduce the version of the *Powerlifted* (Corrêa et al. 2020) planning system used in the IPC 2023. Powerlifted is a lifted planner that works directly on the PDDL representation of planning tasks (i.e., it does not ground the tasks). It is a heuristic search planner,[1] and it contains several heuristics and search engines.

At its core, Powerlifted uses database and logic programming techniques to search more efficiently. It relies on conjunctive queries to generate successor states (Corrêa et al. 2020), Datalog programs to compute relaxed-plans (Corrêa et al. 2021, 2022), and fast on-demand indexing to evaluate states (Corrêa and Seipp 2022). Other works also used Powerlifted to study how to produce very fast heuristics (Lauer 2020; Lauer et al. 2021), or extract lifted landmarks (Wichlacz, Höller, and Hoffmann 2021, 2022).

In contrast to previous usages, the Powerlifted version used in the IPC 2023 is a sequential portfolio. It runs a sequence of different configurations, each with a specified timeout. Sequential portfolios have worked very well for ground planners in previous IPCs (e.g., Helmert et al. 2011; Röger, Pommerening, and Seipp 2014; Seipp and Röger 2018), so we extend the idea to lifted planners here. Powerlifted participated in the satisficing and agile tracks.

Next, we highlight the techniques used in our IPC 2023 version of Powerlifted. We also discuss the new features implemented within Powerlifted exclusively for the IPC. To keep the abstract at an appropriate length, we refer to the original papers for more details. We also refer to Ullman (1988, 1989) for a comprehensive explanation of the database and logic programming terms used here.

Main Techniques

As mentioned above, Powerlifted relies on database techniques for several aspects of its design. The key usages are for state representation and successor generation.

In our submission, we use the *sparse* state representation of Powerlifted. It represents a state as a database, where each predicate is a table, and a ground atom that is true in this state corresponds to a tuple in its associated table. Powerlifted also supports an extensional representation, where all (relaxed) reachable atoms are computed in advance, and states are represented as a simple evaluation of true/false to each atom — which is essentially the same representation as some ground planners (e.g., Helmert 2006) use for STRIPS tasks. Earlier experiments showed that the extensional representation does not pay-off (Corrêa 2019), so we do not use it in our portfolio.

Powerlifted also contains several different successor generators (Corrêa et al. 2020). We use the one based on Yannakakis' algorithm 1981 that exploits acyclicity of preconditions and implicitly existentially quantified variables. The version implemented in Powerlifted also takes inequalities into account, which are originally not considered by Yannakakis (1981) but are studied by later algorithms (Papadimitriou and Yannakakis 1999). The support for inequalities in Powerlifted is an ad-hoc modification to Yannakakis' algorithm and has no efficiency guarantees.

We use a combination of different heuristics and search engines to construct our portfolio. For heuristics, we use the following:

- h^{blind}: the blind heuristic evaluating non-goal states to 1, and goal states to 0.
- h^{add}: the lifted implementation of the additive heuristic (Bonet and Geffner 2001; Corrêa et al. 2021).
- h^{FF}: the lifted implementation of the FF heuristic (Hoffmann and Nebel 2001; Corrêa et al. 2022).
- h^{RFF}: the rule-based FF heuristic (Corrêa et al. 2022).

The last three heuristics are all based on delete-relaxation, and hence they compute preferred operators (POs; Richter and Helmert 2009) as a side-effect.

For search engines, we use different variants of greedy best-first search (GBFS) and best-first width search (BFWS; Lipovetzky and Geffner 2017):

- GBFS: a regular (eager) greedy-best first search search.
- Lazy GBFS: a GBFS with lazy state evaluation (Richter and Helmert 2009). We always combine it with preferred operators (POs) and use two versions: *(i)* Lazy-Prune, where the search prunes states produced by non-preferred operators; *(ii)* Lazy-PO, where the search gives priority to states generated by preferred operators.
- BFWS: a regular BFWS search (without pruning). It has a width parameter w defining the size of the atom conjunctions.

[1]Although Powerlifted has been used as a lifted SAT-planner as well (Höller and Behnke 2022).

- Alt-BFWS: the alternation between BFWS and (lazy) GBFS introduced by Corrêa and Seipp (2022). It also has the width w as a parameter.

As commonly done (e.g., Francès et al. 2017), we limit the choice of w to 1 or 2.

IPC 2023 Features

All techniques listed above have already been studied and evaluated. Next, we introduce the novel ideas of our IPC submission.

PDDL Support

Originally, Powerlifted only supported the fragment of PDDL consisting of STRIPS with inequalities. To support the more expressive fragment used in the competition, we use CPDDL.[2] CPDDL rewrites PDDL files to remove more sophisticated features. It can also be used as a lifted planner, or as a tool to compute information in the lifted setting (e.g., Fišer, Torralba, and Shleyfman 2019; Fišer 2020; Fišer et al. 2021; Horčík and Fišer 2021). However, we only use the PDDL rewriting machinery of CPDDL in our pipeline.

Overall, our submitted version of Powerlifted supports *almost* the full PDDL language: some negated preconditions are not removed by CPDDL, and are still not supported by Powerlifted — unless they are nullary. The planner just aborts if it finds negated preconditions with arity higher than 0. Functional action costs are also not supported, and they are simply ignored (i.e., the domain is transformed into unit cost if they are present).

Sequential Portfolios

We added support for sequential portfolios to Powerlifted. In a nutshell, one can provide a sequence of different search configurations, each with a specific time limit. Powerlifted then performs each search iteratively, based on the time limit given. Time limits are adjusted every time a configuration finishes before reaching its pre-defined limit. We use a total of 22 configurations and 1943 instances to learn the portfolio. All learned configurations transform the input tasks into tasks with unit cost actions.

For the satisficing track, we use the Stone Soup algorithm (Helmert et al. 2011; Röger, Pommerening, and Seipp 2014; Seipp and Röger 2018) to learn a 30 minute portfolio. We refer to the paper by Seipp and Röger (2018) for an explanation of how this learning algorithm works. The learned portfolio uses 10 configurations in total, which we show in the top part of Table 1. The individual configuration with highest coverage was Alt-BFWS (with $w = 1$) using h^{RFF}. It solved 1387 tasks. Our learned portfolio had a coverage of 1793 tasks.

For the agile track, we use the Greedy approximation algorithm (Streeter and Smith 2008; Seipp 2018). Once again, we refer to the referenced papers for details. The learned portfolio for this track is described in the bottom part of Table 1. This portfolio is much longer than the satisficing track one, having 23 configurations in total, even though it runs for only 5 minutes. As for the satisficing case, the individual configuration with highest coverage was Alt-BFWS (with $w = 1$) using h^{RFF}, with a coverage of 1297 tasks. Our learned portfolio had a coverage of 1372 tasks. The coverage increase is not as significant as for the satisficing track, mostly due to the higher overlap of solved tasks in the shorter time limit.

	Search	**Heuristic**	**Time**
Satisficing Track	Alt-BFWS ($w = 1$)	h^{RFF}	476
	Alt-BFWS ($w = 1$)	h^{add}	38
	Alt-BFWS ($w = 2$)	h^{FF}	74
	Alt-BFWS ($w = 2$)	h^{add}	359
	Lazy-PO	h^{FF}	234
	BFWS ($w = 2$)	h^{blind}	278
	Lazy-PO	h^{add}	80
	BFWS ($w = 1$)	h^{blind}	116
	GBFS	h^{RFF}	80
	GBFS	h^{add}	29
Agile Track	Alt-BFWS ($w = 1$)	h^{FF}	1
	BFWS ($w = 1$)	h^{blind}	1
	Lazy-PO	h^{add}	2
	BFWS ($w = 1$)	h^{blind}	2
	Lazy-PO	h^{FF}	3
	Alt-BFWS ($w = 1$)	h^{FF}	9
	BFWS ($w = 2$)	h^{blind}	9
	Lazy-PO	h^{FF}	9
	Alt-BFWS ($w = 1$)	h^{add}	17
	BFWS ($w = 2$)	h^{blind}	35
	Lazy-Prune	h^{FF}	23
	Lazy-PO	h^{add}	5
	Alt-BFWS ($w = 1$)	h^{FF}	40
	GBFS	h^{add}	3
	Lazy-Prune	h^{add}	30
	BFWS ($w = 1$)	h^{blind}	47
	Lazy-PO	h^{RFF}	9
	GBFS	h^{RFF}	6
	Alt-BFWS ($w = 2$)	h^{RFF}	7
	GBFS	h^{add}	7
	Lazy-PO	h^{add}	7
	GBFS	h^{RFF}	9
	Alt-BFWS ($w = 1$)	h^{RFF}	10

Table 1: Ordered list of configurations used in the sequential portfolio for each track. Time in seconds.

Acknowledgments

We thank all Powerlifted contributors for their help to improve the planning system. We especially thank Dillon Chen, Daniel Fišer, Daniel Gnad, Malte Helmert, Pascal Lauer, Florian Pommerening, Philipp Sauer, and Julia Wichlacz for their input, bug reporting, and discussions. We are also grateful to Daniel Fišer for allowing us to use CPDDL, and for showing us how to use it.

[2]https://gitlab.com/danfis/cpddl

References

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

Corrêa, A. B. 2019. *Planning using Lifted Task Representations*. Master's thesis, University of Basel.

Corrêa, A. B.; Francès, G.; Pommerening, F.; and Helmert, M. 2021. Delete-Relaxation Heuristics for Lifted Classical Planning. In Goldman, R. P.; Biundo, S.; and Katz, M., eds., *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICAPS 2021)*, 94–102. AAAI Press.

Corrêa, A. B.; Pommerening, F.; Helmert, M.; and Francès, G. 2020. Lifted Successor Generation using Query Optimization Techniques. In Beck, J. C.; Karpas, E.; and Sohrabi, S., eds., *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling (ICAPS 2020)*, 80–89. AAAI Press.

Corrêa, A. B.; Pommerening, F.; Helmert, M.; and Francès, G. 2022. The FF Heuristic for Lifted Classical Planning. In Honavar, V.; and Spaan, M., eds., *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI 2022)*, 9716–9723. AAAI Press.

Corrêa, A. B.; and Seipp, J. 2022. Best-First Width Search for Lifted Classical Planning. In Thiébaux, S.; and Yeoh, W., eds., *Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling (ICAPS 2022)*, 11–15. AAAI Press.

Fišer, D. 2020. Lifted Fact-Alternating Mutex Groups and Pruned Grounding of Classical Planning Problems. In Conitzer, V.; and Sha, F., eds., *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2020)*, 9835–9842. AAAI Press.

Fišer, D.; Gnad, D.; Katz, M.; and Hoffmann, J. 2021. Custom-Design of FDR Encodings: The Case of Red-Black Planning. In Zhou, Z.-H., ed., *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI 2021)*, 4054–4061. IJCAI.

Fišer, D.; Torralba, Á.; and Shleyfman, A. 2019. Operator Mutexes and Symmetries for Simplifying Planning Tasks. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI 2019)*, 7586–7593. AAAI Press.

Francès, G.; Ramírez, M.; Lipovetzky, N.; and Geffner, H. 2017. Purely Declarative Action Representations are Overrated: Classical Planning with Simulators. In Sierra, C., ed., *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI 2017)*, 4294–4301. IJCAI.

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

Helmert, M.; Röger, G.; Seipp, J.; Karpas, E.; Hoffmann, J.; Keyder, E.; Nissim, R.; Richter, S.; and Westphal, M. 2011. Fast Downward Stone Soup. In *IPC 2011 Planner Abstracts*, 38–45.

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

Höller, D.; and Behnke, G. 2022. Encoding Lifted Classical Planning in Propositional Logic. In Thiébaux, S.; and Yeoh, W., eds., *Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling (ICAPS 2022)*, 134–144. AAAI Press.

Horčík, R.; and Fišer, D. 2021. Endomorphisms of Lifted Planning Problems. In Goldman, R. P.; Biundo, S.; and Katz, M., eds., *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICAPS 2021)*, 174–183. AAAI Press.

Lauer, P. 2020. *Unary Relaxation*. Bachelor's thesis, Saarland University.

Lauer, P.; Torralba, Á.; Fišer, D.; Höller, D.; Wichlacz, J.; and Hoffmann, J. 2021. Polynomial-Time in PDDL Input Size: Making the Delete Relaxation Feasible for Lifted Planning. In Zhou, Z.-H., ed., *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI 2021)*, 4119–4126. IJCAI.

Lipovetzky, N.; and Geffner, H. 2017. Best-First Width Search: Exploration and Exploitation in Classical Planning. In Singh, S.; and Markovitch, S., eds., *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI 2017)*, 3590–3596. AAAI Press.

Papadimitriou, C. H.; and Yannakakis, M. 1999. On the Complexity of Database Queries. *Journal of Computer and System Sciences*, 58(3): 407–427.

Richter, S.; and Helmert, M. 2009. Preferred Operators and Deferred Evaluation in Satisficing Planning. In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*, 273–280. AAAI Press.

Röger, G.; Pommerening, F.; and Seipp, J. 2014. Fast Downward Stone Soup 2014. In *Eighth International Planning Competition (IPC-8): Planner Abstracts*, 28–31.

Seipp, J. 2018. Fast Downward Remix. In *Ninth International Planning Competition (IPC-9): Planner Abstracts*, 74–76.

Seipp, J.; and Röger, G. 2018. Fast Downward Stone Soup 2018. In *Ninth International Planning Competition (IPC-9): Planner Abstracts*, 80–82.

Streeter, M. J.; and Smith, S. F. 2008. New Techniques for Algorithm Portfolio Design. In *Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence (UAI 2008)*, 519–527.

Ullman, J. D. 1988. *Principles of Database and Knowledge-Base Systems. Volume I: Classical Database Systems*. Computer Science Press.

Ullman, J. D. 1989. *Principles of Database and Knowledge-Base Systems. Volume II: The New Technologies*. Computer Science Press.

Wichlacz, J.; Höller, D.; and Hoffmann, J. 2021. Landmark Heuristics for Lifted Planning – Extended Abstract. In Ma, H.; and Serina, I., eds., *Proceedings of the 14th Annual Symposium on Combinatorial Search (SoCS 2021)*, 242–244. AAAI Press.

Wichlacz, J.; Höller, D.; and Hoffmann, J. 2022. Landmark Heuristics for Lifted Classical Planning. In De Raedt, L., ed., *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI 2022)*, 4665–4671. IJCAI.

Yannakakis, M. 1981. Algorithms for Acyclic Database Schemes. In *Proceedings of the 7th International Conference on Very Large Data Bases (VLDB 1981)*, 82–94. IEEE Press.