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, 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; lacz, Höller, and Hoffmann 2021, 2022), or extract lifted landmarks (Wichlacz, Höller, and Hoffmann 2021, 2022).

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 Yanakkakis’ 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.

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.
- **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. 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.

To support CPDDL, we added support for sequential portfolios to Powerlifted.

### 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 (Helmer 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.

### Competition Results

Powerlifted did not perform well in either track it participated in. One of the key issues was the lack of PDDL support. Although we used CPDDL to normalize the domains, and the organizers also provided alternative normalized domains, only in Folding the normalization helped. In fact, Powerlifted had the highest score in the Folding domain among the planners participating in the satisficing track.
One problem was that the normalization (both from CPDDL and from the organizers) did not remove negated static preconditions (but non-static ones were removed). As Powerlifted does not support any sort of negation in precondition, two domains failed even in their normalized versions: Labyrinth and Ricochet Robots.

The delete-relaxation heuristics in Powerlifted also produced bugs in the Folding domain. For some yet unknown reason, the goal was always considered unreachable and the heuristic became unsafe. Hence, only configurations using the blind heuristic could solve this domain. By the time of this report, it is unclear what the source of the bug was. It is interesting that Powerlifted was still the best-performing participant in this domain, despite the bug in its more informed configurations.

In total, Powerlifted obtained non-zero scores in only three of the seven domains in the satisficing track: Folding, Quantum Layout, and Slitherlink. In the agile track, it did so in only two domains: Folding and Quantum Layout. It is obvious that Powerlifted must be extended to deal with more expressive features of PDDL in order to be competitive in the IPC.

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.

References


