Tyr: A Ground and Lifted Numeric Planner for the IPC 2026

Dominik Drexler

Linköping University, Sweden
dominik.drexler@liu.se

Abstract

Tyr is a numeric planner submitted to the optimal, agile, and satisficing numeric tracks of IPC 2026. It provides ground and lifted configurations for simple and linear numeric planning. Optimal configurations use eager A* with the blind heuristic; agile and satisficing configurations use lazy greedy best-first search with either goal counting or a lifted additive interval RPG heuristic. In Tyr's lifted configurations, successor generation avoids full grounding by deriving candidate action bindings from substitution-consistency graphs and Datalog facts, then checking each candidate exactly before successor construction.

1 Introduction

Tyr is a C++ planning library with Python bindings for numeric PDDL tasks. The IPC 2026 submission targets the numeric optimal, agile, and satisficing tracks with both ground and lifted configurations for simple numeric planning (SNP) and linear numeric planning (LNP). Tyr supports propositional and numeric fluents, negative preconditions, equality, conditional effects, derived predicates, quantification, disjunction, and action costs. It targets researchers working at the intersection of planning and machine learning: the Python interface supports learning-based workflows, while lifted planning improves robustness on large tasks.

Table 1 summarizes the six submitted configurations. Optimal configurations are baseline cost-based searches: eager A* with the blind heuristic. Agile and satisficing configurations use lazy greedy best-first search. Ground search ranks states by the number of unsatisfied goal conjuncts; lifted search uses an additive cost estimate from a relaxed planning program evaluated by the Tyr bottom-up Datalog machinery.

Tyr compares two finite representations of the same transition system. The ground representation materializes relevant ground actions before search; the lifted representation keeps action schemata in first order and materializes only bindings relevant to the current state. Both share the same action semantics, numeric executor, axiom handling, state storage, and plan extraction.

2 Tyr

Ground configurations instantiate a ground task and use efficient data structures to enumerate applicable ground actions in each state (Helmert 2006). Our lifted configurations keep action schemata in first order and generate state-dependent candidates via substitution-consistency graphs (Ståhlberg 2023). Vertices assign action parameters to objects, and edges encode assignment pairs that are consistent with propositional preconditions. We extended this consistency check also to consider numeric precondition constraints (Drexler 2025). Each k-clique in this graph yields a candidate action binding. However, the resulting ground action instantiations may be inapplicable; a final check filters them out before applying them to generate successor states. The procedure is exact if the precondition literals and numeric constraints have at most two parameters, which holds for many existing benchmarks.

Track	Representation	Search	Heuristic
optimal	ground	eager A*	blind
optimal	lifted	eager A*	blind
satisficing	ground	lazy GBFS	goal count
satisficing	lifted	lazy GBFS	additive RPG
agile	ground	lazy GBFS	goal count
agile	lifted	lazy GBFS	additive RPG

Table 1: Tyr configurations submitted to the IPC 2026 numeric tracks.

The submitted configurations pair these generators with standard search algorithms and lightweight heuristics from the literature. Optimal search configurations use eager A* (Hart, Nilsson, and Raphael 1968) with the blind heuristic. Agile and satisficing configurations use lazy greedy best-first search (Pearl 1984); generated states inherit the parent heuristic value and are re-evaluated when expanded. The open list alternates between a standard and a preferred queue, with weights boosted after heuristic improvements (Richter and Westphal 2010).

Ground agile and satisficing configurations use goal count over propositional and numeric goal constraints. Lifted agile and satisficing configurations use an additive interval RPG heuristic over the relaxed Datalog program. It is a unit-cost interval relaxation for lifted search based on Aldinger and Nebel (2017) and generalizes the lifted additive heuristic for classical planning (Corrêa et al. 2021). Supports are summed across rule bodies, the cheapest derivation is retained, and unsatisfied goal constraints give infinite value.

References

Aldinger, J.; and Nebel, B. 2017. Interval Based Relaxation Heuristics for Numeric Planning with Action Costs. In Kern-Isberner, G.; Fürnkranz, J.; and Thimm, M., eds., *Proceedings of the 40th Annual German Conference on Artificial Intelligence (KI 2017)*, volume 10505 of *Lecture Notes in Artificial Intelligence*, 15–28. Springer-Verlag.

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.

Drexler, D. 2025. Lifted Successor Generation in Numeric Planning (Extended Version). arXiv:2511.00673 [cs.AI].

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107.

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

Pearl, J. 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley.

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

Ståhlberg, S. 2023. Lifted Successor Generation by Maximum Clique Enumeration. In Gal, K.; Nowé, A.; Nalepa, G. J.; Fairstein, R.; and Rădulescu, R., eds., *Proceedings of the 26th European Conference on Artificial Intelligence (ECAI 2023)*, 2194–2201. IOS Press.