Learning Heuristic Functions with Graph Neural Networks for Numeric Planning (Extended Abstract)

Valerio Borelli[1,2], Alfonso Emilio Gerevini[1], Enrico Scala[1], Ivan Serina[1]

[1] University of Brescia
[2] La Sapienza University of Rome
valerio.borelli@unibs.it, valerio.borelli@uniroma1.it, alfonso.gerevini@unibs.it, enrico.scala@unibs.it, ivan.serina@unibs.it

Abstract

In this paper, we investigate the application of heuristics based on Graph Neural Networks (GNNs) to lifted numeric planning problems, an area that is still relatively unexplored. Building upon the GNN approach for learning general policies proposed by Ståhlberg, Bonet, and Geffner (2022), we propose an architecture sensitive to the numeric components inherent in the planning problems we address. We achieve this by observing that, although the state space of a numeric planning problem is infinite, the finite subgoal structure of the problem can be incorporated into the architecture, allowing for the construction of only a finite structure. Instead of learning general policies, we train our models to function as a heuristic within a best-first search algorithm. We explore various configurations of this architecture and demonstrate that the resulting heuristics are highly informative and, in certain domains, offer a better trade-off between guidance and computational cost compared to other heuristics.

Introduction

Deep learning techniques have increasingly been applied to planning problems, with applications in heuristic learning and value-function approximation(Ståhlberg, Bonet, and Geffner 2022). However, while classical planning has seen success with GNN-based policy and heuristics, numeric planning is still under-explored. One of the challenges in this context is the presence of an infinite state space together with numeric structures that are still unclear how to handle. Our work builds upon the GNN-based approach of Ståhlberg, Bonet, and Geffner (2022) for learning value functions and adapts it for numeric planning by integrating numeric conditions into the network. Other works have proposed neural architectures for numeric planning, i.e., ASNets Wang and Thiébaux (2024) and graph-based heuristics Chen and Thiébaux (2024). Our work is however different. Indeed, with the idea to better reflect the numeric structure of the problem in the GNN, we not only integrate the numeric information stored in the numeric fluents but also integrate numeric (sub)goals (i.e., preconditions and goals) into the relational structure of the GNN. We design the GNN in order to use it as a heuristic in a state-space search planner. Therefore, given a state s, the GNN will be used to estimate how far s is from the goal.

Methodology

Informally speaking, a numeric planning problem is the task of finding a course of actions given some initial state, goal and a formal specification of the actions. (More details in Fox and Long (2003).) Numeric planning differs from classical planning in that action preconditions and goals can have numeric constraints, and actions can assign and modify numeric state variables too. This makes it undecidable in general (Helmert 2002). Our approach represents numeric planning states as relational structures, therefore working at a lifted level. A relational structure for a state s is defined by the set of objects $o \in O$, the set of domain predicates $p \in P$, and the atoms $q = p(o_1, ..., o_m)$, that are true in s. Where objects serve as nodes, predicates as possible types of arches, and atoms as the adjacency matrix.

Numeric Goals and Preconditions The most straightforward extension to the numeric case would be to fully specify each value of a numeric fluent as an atom. However, for a given problem specification, there is an infinite number of values and we won't be able to distinguish them all. To address this issue, we observed that where in classical planning preconditions and goals are always represented by Boolean atoms, in Numeric Planning those can be represented by numeric conditions of the form $exp\{<, \leq\}0$ where exp is a numeric expression of numeric fluents. We focus on capturing numeric conditions of the problem; this lets us focus on the truth of such conditions. Also they retain the interactions between numeric fluents. Each numeric condition is encoded as a new type of arc in the GNN architecture.

The main idea of our approach is to let the GNN compute heuristic values by reasoning on which subgoal is achieved and which is not. And we can do this on a state-by-state basis. The *weight* of a subgoal will be a function of the objects it consists of, and of domain independent information established through learning the "impact" of its lifted representation.

Numeric Fluents Without a way to handle numeric fluents, different states may be perceived as identical by the architecture; this can happen, for instance, whenever a single execution of action will not make true a numeric condition. For example, let's consider a problem with a single numeric

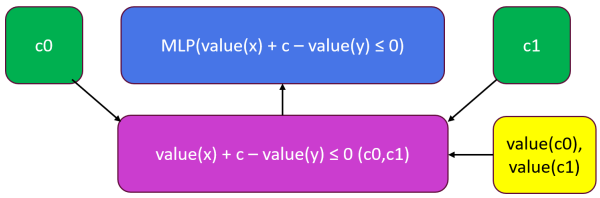

Figure 1: GNN architecture example for the domain Counters. The yellow node transmits the numeric values of the fluents $value(c0), value(c1)$ to the purple node, which represents the atom. This node encodes them with the object embedding of $c0$ and $c1$. The resulting vector is then passed to the blue node, which represents the Multi-Layer Perceptron (**MLP**) for the numeric condition. Each condition is handled by a specific MLP, allowing us to define fixed input and output dimensions for each MLP.

Domain	h^{ccWLF}_{rank}	h^0	h^c	h^n
Counters	3	0	2	**4**
Fo-Counters	4	0	2	**7**
Sailing	**5**	0	0	1
Fo-Sailing	**8**	1	2	**8**
Mprime	**15**	7	9	8
Expedition	3	1	1	**6**

Table 1: Coverage analysis among three variants of our architecture (h^0,h^c,h^n) and h^{ccWLF}_{rank}.

condition in the form $x > 5$. Suppose the current state is $x = 1$, and an action increments x by 1. The new state will have $x = 2$, but the numeric condition remains false in both states. As a result, our GNN will perceive these states as identical. To address this issue, we complement the GNN with state-dependent numeric values. We attach the numeric values of our fluents directly to the atoms representing the numeric subgoals of our problem (Figure 1).

Similarly to Ståhlberg, Bonet, and Geffner (2022), our architecture is constructed and trained for the lifted structure of numeric planning problem. In order to become operative for a specific instance, we add the objects of our problem. The network takes a state of a planning problem as input, and gives a heuristic value as output.

Experimental Evaluation

The heuristic function is learned via supervised training on optimal plan datasets generated using an A* search with the h^{rmax} heuristic (Scala et al. 2020). The learned heuristic is then employed within a greedy best-first search algorithm. We compare the heuristic learned by our architecture with one in which the numeric structure of the problem is completely ignored, namely h^0 – this corresponds to Ståhlberg, Bonet, and Geffner (2022) formulation, and with h^{ccWLF}_{rank}, the Graph-based heuristic proposed by Chen and Thiébaux (2024), the state of the art in learning-based heuristics for numeric planning problems. To shed lights on the effects of making the architecture sensitive to the numeric structure of the problem, we experimented two variants: h^c, which is the version where numeric conditions are abstracted as Boolean conditions, and h^n, which is our more advanced solution where numeric values are provided to the MLP associated with each numeric condition. As numeric planning problems, we selected six domains from the International Planning Competition (IPC) 2023 Numeric Track (Taitler et al. 2024): `Expedition`, `Fo-Sailing`, `Fo-Counters`, `Counters`, `Mprime`, `Sailing`. Each domain consists of 20 instances, with increasing difficulty scaling from the first instance to the last. Table 1 reports results on the problem coverage obtained by all the considered heuristics. From these results it emerges quite clearly that our architecture benefits quite a lot from making it sensitive to the numeric information in the problem.

Conclusion

We have proposed a new approach to tackle numeric planning problems using GNNs as heuristic functions. The approach extends the architecture of (Ståhlberg, Bonet, and Geffner 2022) to handle numeric problems, and uses the learned heuristics inside a numeric planner written in python. This work is a step in the direction of understanding how deep learning tools can be used in a numeric planner, and what are the main issues that must be overcome.

Acknowledgments

This work has been supported by MUR projects PRIN2020 RIPER and PNRR PE0000013-FAIR BAC resilientplans.

References

Chen, D.; and Thiébaux, S. 2024. Graph learning for numeric planning. *Advances in Neural Information Processing Systems*, 37: 91156–91183.

Fox, M.; and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *J. Artif. Intell. Res.*, 20: 61–124.

Helmert, M. 2002. Decidability and Undecidability Results for Planning with Numerical State Variables. In *AIPS*, 44–53. AAAI.

Scala, E.; Haslum, P.; Thiébaux, S.; and Ramírez, M. 2020. Subgoaling Techniques for Satisficing and Optimal Numeric Planning. *J. Artif. Intell. Res.*, 68: 691–752.

Ståhlberg, S.; Bonet, B.; and Geffner, H. 2022. Learning Generalized Policies without Supervision Using GNNs. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 19, 474–483.

Taitler, A.; Alford, R.; Espasa, J.; Behnke, G.; Fiser, D.; Gimelfarb, M.; Pommerening, F.; Sanner, S.; Scala, E.; Schreiber, D.; Segovia-Aguas, J.; and Seipp, J. 2024. The 2023 International Planning Competition. *AI Mag.*, 45(2): 280–296.

Wang, R. X.; and Thiébaux, S. 2024. Learning Generalised Policies for Numeric Planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, 633–642.