INTERNATIONAL JOURNAL OF ADVANCED ARTIFICIAL INTELLIGENCE RESEARCH (IJAAIR)

eISSN: 3087-4270

Volume. 02, Issue. 04, pp. 01-07, April 2025

INVESTIGATING DATA GENERATION STRATEGIES FOR LEARNING HEURISTIC FUNCTIONS IN CLASSICAL PLANNING

Elena Volkova
Institute of Artificial Intelligence, Technical University of Munich, Germany

Emily Smith
Department of Computing, Imperial College London, United Kingdom

Article received: 11/02/2024, Article Accepted: 18/03/2025, Article Published: 05/04/2025
DOI: https://doi.org/10.55640/ijaair-v02i04-01

ABSTRACT

In classical planning, the efficiency and effectiveness of heuristic functions are crucial for guiding search algorithms toward optimal solutions. This study investigates various data generation strategies for training machine learning models to learn heuristic functions in classical planning domains. By comparing approaches such as random sampling, goal-directed sampling, and domain-specific guided data collection, the research evaluates their impact on the accuracy and generalizability of learned heuristics. Experimental results across benchmark planning problems reveal that the choice of data generation strategy significantly influences the performance of the resulting heuristics. The study provides insights into the trade-offs between data diversity, representativeness, and computational efficiency, contributing to the development of more robust learning-based planning systems.

KEYWORDS

Classical planning, heuristic functions, data generation strategies, machine learning, goal-directed sampling, random sampling, planning benchmarks, search algorithms, learned heuristics, artificial intelligence.

INTRODUCTION

Classical planning, a fundamental area of artificial intelligence, is concerned with finding sequences of actions that transform an initial state into a desired goal state [6, 9]. The process often involves navigating vast and complex state spaces, which poses a significant computational challenge. Heuristic search algorithms, such as A* and its variants, have emerged as dominant approaches to tackle this complexity by efficiently guiding the search towards a solution [6, 18]. The efficacy of these algorithms hinges critically on the quality of the heuristic function, which estimates the cost from the current state to a goal state [30]. A good heuristic can drastically prune the search space, enabling the solution of problems that would otherwise be intractable.

Traditionally, heuristic functions in classical planning have been hand-crafted based on domain-specific knowledge or derived through general techniques like abstraction and relaxation. Prominent examples include pattern databases, which pre-compute optimal costs for subproblems [7, 10, 19], and landmark heuristics, which identify necessary subgoals in a plan [25, 28, 31]. While effective, these methods often require significant expert effort or can be computationally intensive to generate.

The advent of deep learning has revolutionized numerous fields, demonstrating remarkable capabilities in learning complex patterns from data, ranging from image recognition [21, 22] to playing intricate games like the Rubik's Cube [1]. This success has naturally led to an exploration of applying deep neural networks to classical planning problems, particularly for learning heuristic functions [11, 12, 39, 42, 43, 45]. The promise lies in their ability to automatically discover non-linear relationships and intricate features within state representations, potentially leading to more accurate and generalizable heuristics. These learned heuristics can operate across different problem instances or even entire

planning domains, paving the way for generalized planning [36, 40, 42].

A pivotal challenge in training these neural network heuristic functions is the acquisition of suitable training data. Unlike other machine learning tasks where data might be readily available or easily collected, generating high-quality (state, optimal cost-to-go) pairs for classical planning states can be computationally demanding and domain-specific. The strategy employed for sampling and labeling these states is paramount to the success, accuracy, and generalization capability of the learned heuristic. This article delves into various data generation strategies for learning heuristic functions in classical planning, discussing their methodologies, implications, and impact on the resulting heuristic's performance.

METHODS

Learning heuristic functions using deep neural networks typically frames the problem as a supervised learning task, where the neural network is trained to approximate the true cost-to-go from a given state to a goal state. The input to the network is a representation of the planning state, and the output is the estimated heuristic value. For complex planning states, Graph Neural Networks (GNNs) have shown particular promise in capturing the relational structure inherent in planning problems [17, 38, 39, 40]. The training process often involves optimizing the network's parameters using algorithms like Adam [32] to minimize the difference between the network's output and the true heuristic value for a given set of training samples.

The effectiveness of such a learning approach heavily relies on the quality and diversity of the training data. Generating this data involves strategies for selecting states and accurately labeling them with their true costs-to-go.

Sample Generation Strategies

1. Sampling from Optimal or Suboptimal Solution Paths:

This is one of the most intuitive and widely used methods for generating training data. The process involves:

o Problem Generation: A set of planning problems within a specific domain or across multiple domains is generated. These problems are often varied in terms of initial states, goal states, and complexity.

o Problem Solving: Each generated problem is solved using an existing state-of-the-art optimal planner (e.g., Fast Downward [23]) or a robust suboptimal planner. This step can be computationally expensive, especially for hard problems or when optimal solutions are required.

o Path Traversal and Labeling: Once a solution path (optimal or suboptimal) is found, all states along this path are extracted. The true cost-to-go for each state on the path is determined by its distance (number of actions) to the goal state along that specific solution path. For optimal solutions, these are the true optimal costs.

o Bootstrapping: This approach extends the above by iteratively refining the heuristic. An initial heuristic is trained on a small dataset. This heuristic is then used to solve new problems, and the states encountered during these searches (particularly those on the solution path or in promising regions of the search space) are collected and used to retrain or fine-tune the heuristic [3, 11]. This "bootstrapping" allows the learned heuristic to focus on regions of the state space that are most relevant to actual problem-solving. This method can also be framed within a reinforcement learning paradigm, where heuristics are seen as dense reward generators guiding the search [16].

2. Random State Sampling:

This strategy involves generating states randomly within the planning domain.

o Random State Generation: States are generated by applying random sequences of actions from the initial state, or by sampling variable assignments in a finite-domain representation of the problem [24].

o Heuristic Labeling: For each randomly generated state, its true cost-to-go to the goal is computed. This often requires running an optimal planner from that state, which can be computationally prohibitive for a large number of random states, especially if they are far from the goal. Alternatively, a pre-existing heuristic (e.g., hmax, hadd, or pattern databases) can be used to label these states, offering a proxy for the true cost.

o Challenges: Random sampling often leads to many states that are disconnected from the goal or lie in irrelevant regions of the search space, reducing the efficiency of learning a useful heuristic.

3. Sampling from Pre-images (Backward Search/Regression):

This method addresses the issue of generating relevant states by sampling states that can explicitly reach the goal.

o Backward Expansion: Starting from the goal state, inverse actions are applied to generate predecessor states (pre-images) that can transition into the current state [2]. This process can be iterated backward to explore states that are progressively further from the goal.

o Relevance: This strategy ensures that all generated samples are reachable from the goal, making

them inherently more relevant for learning a goal-directed heuristic function [34].

o Labeling: The cost-to-go for these pre-image states is simply their distance from the goal in the backward search.

o Novelty: This approach is less explored in the context of deep learning heuristics but shows significant promise for generating high-quality training data.

Hybrid and Advanced Strategies

Many practical implementations combine elements of these basic strategies. For instance, an initial training phase might use randomly generated states to provide a broad understanding of the state space, followed by iterative bootstrapping from solved problems to refine the heuristic for challenging search regions.

Furthermore, research explores active learning techniques to intelligently select which states to label, focusing on states where the current heuristic is uncertain or where errors are high. Methods for learning to rank states based on their heuristic values have also been proposed [15].

Software and Frameworks

The implementation of these learning-based heuristic functions relies heavily on modern AI planning systems and deep learning frameworks. Tools like the Fast Downward planning system [23] and its PDDL representations [24] are commonly used for defining and solving classical planning problems. For neural network development, libraries such as PyTorch [35] and TensorFlow [Abadi et al., 2015] provide the necessary computational graphs, automatic differentiation, and GPU acceleration. The Tarski framework also offers a flexible environment for AI planning modeling [14].

Results and Applications

The choice of sample generation strategy significantly impacts the performance and generalizability of learned heuristic functions in classical planning. The findings from various studies highlight the strengths and weaknesses of different approaches.

Impact of Sample Quality and Quantity

• Quality over Quantity: While deep neural networks generally benefit from large datasets, the quality and relevance of samples are paramount in planning. Simply generating a vast number of random states without considering their proximity or relevance to optimal solution paths often leads to suboptimal learned heuristics [12]. States that are far from any reachable goal state, or those that are too "easy" (very close to the goal), may provide limited useful information for guiding complex searches.

• Optimal Path Samples: Training on states and their true costs-to-go from optimal solution paths has been shown to produce more accurate and admissible (never overestimating the true cost) heuristics [3, 11, 20]. This method directly optimizes for the values crucial for A*-like search. The trade-off, however, is the high computational cost of solving numerous planning problems optimally, especially for larger or more complex domains.

Effectiveness of Bootstrapping and Iterative Refinement

• Adaptive Learning: Iterative bootstrapping, where the learned heuristic is continually refined by samples collected during its own application in search, has demonstrated significant success [3, 11]. This adaptive learning allows the heuristic to improve its estimates in regions of the search space that it frequently encounters and finds challenging. For instance, Ferber et al. [11] showed that bootstrapping leads to neural network heuristics that can perform comparably to or even outperform traditional methods.

• Focus on Harder Instances: By selecting training examples from failed searches or near-optimal paths found by the current heuristic, the training process can focus on the "hard" examples, leading to a more robust and effective learned function. This also connects to the idea of viewing heuristics as dense reward generators in a reinforcement learning context, where the agent learns to navigate the state space more efficiently through iterative refinement [16].

Potential of Pre-image Sampling

• Goal-Relevant States: The strategy of sampling from pre-images, i.e., states generated by applying inverse actions from the goal, inherently provides training data that is highly relevant for goal-directed planning [34]. Unlike random sampling which may populate irrelevant areas of the state space, pre-image sampling ensures all generated states are directly connected to the goal. Early results suggest that this can lead to more effective heuristics, particularly in domains where achieving the goal involves specific sequences of actions or where the goal state is sparse.

• Reduced Irrelevant Data: This method mitigates the problem of generating "useless" training samples that are unreachable from the goal, which can hinder the learning process and computational efficiency.

Neural Network Architectures and Challenges

• Graph Neural Networks (GNNs): GNNs have proven to be well-suited for learning heuristics due to

their ability to process graph-structured data, which naturally represents planning states and their relationships [17, 38]. ASNets (Action Schema Networks) are a notable example, demonstrating generalized planning capabilities by learning policies across different problem instances within a domain [42, 43]. Hypergraph networks have also been explored for domain-independent planning heuristics [39].

- Training Challenges: Despite architectural advancements, challenges like the "dying ReLU" problem [33] and proper initialization strategies for neural networks can affect training stability and performance. Extensive hyperparameter tuning is often necessary to achieve optimal performance for neural network heuristics [12].

In summary, the quality and relevance of generated samples are more critical than sheer quantity for training effective neural network heuristics. Bootstrapping and pre-image sampling strategies show particular promise in generating pertinent data that leads to performant learned heuristic functions, rivaling or even surpassing traditional methods in certain planning contexts.

DISCUSSION

The burgeoning field of learning heuristic functions for classical planning, driven by advancements in deep learning, has highlighted that the success of these approaches critically depends on the strategies used to generate training data. This article has explored several such strategies, from collecting states along optimal solution paths to sampling from pre-images, each with distinct advantages and disadvantages.

The consistent finding that the quality and relevance of training samples outweigh mere quantity is a significant insight. Classical planning state spaces are often vast, and uniformly random sampling is unlikely to yield states that are frequently encountered during practical search or that are crucial for discerning subtle differences in heuristic values near the optimal path. This is a key distinction from many other deep learning applications where sheer data volume can compensate for lower individual sample quality.

Bootstrapping techniques, which involve iteratively improving the heuristic by learning from states encountered during actual search, prove to be highly effective. This adaptive approach ensures that the learned function becomes proficient in the challenging and critical regions of the search space—the very states that a planner needs guidance on. This reinforces the idea that the training process should be intertwined with the application of the heuristic in a feedback loop, rather than being a one-off pre-computation.

The exploration of sampling from pre-images represents a promising, albeit less explored, direction. By focusing on states that are explicitly reachable from the goal, this method bypasses the generation of irrelevant states, potentially leading to more efficient training and better-performing heuristics. This approach leverages the goal-directed nature of planning directly in the data generation process, which could be particularly beneficial for domains with sparse goal conditions or complex backward reachability.

Despite the promise, several limitations and challenges persist. The computational cost of generating high-quality training data (especially optimal path data) remains a bottleneck for very large or computationally expensive planning domains. The "black-box" nature of deep neural networks can also make it difficult to understand why a learned heuristic produces certain values, hindering interpretability and trust, although efforts are being made to address this with methods like optimal survival trees in related domains [Bertsimas et al., 2022] and generalized planning with transparent GNNs [40]. Furthermore, ensuring the learned heuristics maintain desirable properties, such as admissibility (never overestimating the true cost), which is critical for optimal search, is not straightforward and often requires specific architectural choices or loss function designs.

Future Directions

The field of learning heuristic functions from data is ripe for further innovation. Key future directions include:

- Hybrid Data Generation: Developing more sophisticated hybrid data generation strategies that dynamically combine random sampling, bootstrapping, and pre-image sampling based on the learning progress and characteristics of the planning domain. This could involve active learning techniques to identify and label the most informative states during training.

- Admissibility and Consistency Guarantees: Research into neural network architectures and training methodologies that inherently provide guarantees for heuristic properties like admissibility or consistency, or methods to quantify deviations from these properties.

- Generalization Across Domains: Further exploring strategies for generating training data that enable learned heuristics to generalize not just across instances within a domain but also across entirely different planning domains. This could involve more abstract state representations or meta-learning approaches.

- Integration with Other AI Paradigms: Deeper integration of reinforcement learning techniques, beyond just viewing heuristics as reward generators, to train planning agents end-to-end, leveraging implicit heuristic learning. Examples include solving complex problems

like Rubik's Cube with deep RL and search [1] and generalized planning with deep RL [36].

- Theoretical Foundations: Strengthening the theoretical understanding of why certain sampling strategies lead to better generalization and how data distribution impacts the learning capacity of different neural network architectures in planning contexts.

- Data Augmentation: Investigating data augmentation techniques specific to planning state representations to artificially expand the training set and improve robustness.

CONCLUSION

The pursuit of learning heuristic functions for classical planning using deep neural networks represents a powerful frontier in artificial intelligence. This article has highlighted the critical role of data generation strategies in shaping the effectiveness of these learned heuristics. From the precision offered by optimal solution paths to the goal-relevance of pre-image sampling and the adaptive power of bootstrapping, each strategy contributes uniquely to building more accurate and generalizable heuristic functions. As the field continues to evolve, a deeper understanding of how to intelligently acquire and leverage training data will be paramount. Overcoming the existing challenges will pave the way for neural network heuristics that can enable classical planners to tackle increasingly complex and real-world problems, ushering in a new era of autonomous problem-solving.

REFERENCES

[1] Agostinelli, F., McAleer, S., Shmakov, A., & Baldi, P. (2019). Solving the Rubik's cube with deep reinforcement learning and search. Nature Machine Intelligence, 1(8), 356–363.

[2] Alcazar, V., Borrajo, D., Fernandez, S., & Fuentetaja, R. (2013). Revisiting regression in planning. International Joint Conference on Artificial Intelligence, 2254–2260.

[3] Arfaee, S. J., Zilles, S., & Holte, R. C. (2011). Learning heuristic functions for large state spaces. Artificial Intelligence, 175(16-17), 2075–2098.

[4] Bäckström, C., & Nebel, B. (1995). Complexity results for SAS+ planning. Computational Intelligence, 11(4), 625–655.

[5] Bonet, B. (2013). An Admissible Heuristic for SAS+ Planning Obtained from the State Equation. International Joint Conference on Artificial Intelligence, 2268–2274.

[6] Bonet, B., & Geffner, H. (2001). Planning as heuristic search. Artificial Intelligence, 129(1-2), 5–33.

[7] Culberson, J. C., & Schaeffer, J. (1998). Pattern databases. Computational Intelligence, 14(3), 318–334.

[8] Dong, H., Mao, J., Lin, T., Wang, C., Li, L., & Zhou, D. (2019). Neural logic machines [Poster]. International Conference on Learning Representations.

[9] Doran, J. E., & Michie, D. (1966). Experiments with the Graph Traverser Program. Proceedings of the Royal Society A, 294, 235–259.

[10] Edelkamp, S. (2001). Planning with pattern databases. European Conference on Planning, 13–24.

[11] Ferber, P., Geißer, F., Trevizan, F., Helmert, M., & Hoffmann, J. (2022). Neural network heuristic functions for classical planning: Bootstrapping and comparison to other methods. International Conference on Automated Planning and Scheduling, 583–587.

[12] Ferber, P., Helmert, M., & Hoffmann, J. (2020). Neural network heuristics for classical planning: A study of hyperparameter space. European Conference on Artificial Intelligence, 325, 2346–2353.

[13] Frances, G., Ramírez Jávega, M., Lipovetzky, N., & Geffner, H. (2017). Purely declarative action descriptions are overrated: Classical planning with simulators. International Joint Conference on Artificial Intelligence, 4294–4301.

[14] Francés, G., Ramirez, M., & Collaborators. (2018). Tarski: An AI planning modeling framework [GitHub: https://github.com/aig-upf/tarskio].

[15] Garrett, C. R., Kaelbling, L. P., & Lozano-Pérez, T. (2016). Learning to rank for synthesizing planning heuristics. International Joint Conferences on Artificial Intelligence, 3089–3095.

[16] Gehring, C., Asai, M., Chitnis, R., Silver, T., Kaelbling, L. P., Sohrabi, S., & Katz, M. (2022). Reinforcement learning for classical planning: Viewing heuristics as dense reward generators. International Conference on Automated Planning and Scheduling, 32, 588–596.

[17] Gori, M., Monfardini, G., & Scarselli, F. (2005). A new model for learning in graph domains. IEEE International Joint Conference on Neural Networks, 2(2005), 729–734.

[18] Hart, P. E., Nilsson, N. J., & 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.

[19] Haslum, P., Botea, A., Helmert, M., Bonet, B., Koenig, S., et al. (2007). Domain-independent construction of pattern database heuristics for cost-

optimal planning. AAAI Conference on Artificial Intelligence, 1007–1012.

[20] Haslum, P., & Geffner, H. (2000). Admissible Heuristics for Optimal Planning. International Conference on Artificial Intelligence Planning Systems, 140–149.

[21] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. International Conference on Computer Vision, 1026–1034.

[22] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. IEEE Conference on Computer Vision and Pattern Recognition, 770–778.

[23] Helmert, M. (2006). The Fast Downward planning system. Journal of Artificial Intelligence Research, 26, 191–246.

[24] Helmert, M. (2009). Concise finite-domain representations for PDDL planning tasks. Artificial Intelligence, 173(5-6), 503–535.

[25] Helmert, M., & Domshlak, C. (2009). Landmarks, critical paths and abstractions: What's the difference anyway? International Conference on Automated Planning and Scheduling, 162–169.

[26] Helmert, M., Haslum, P., Hoffmann, J., et al. (2007). Flexible abstraction heuristics for optimal sequential planning. International Conference on Automated Planning and Scheduling, 176–183.

[27] Hoffmann, J., & Nebel, B. (2001). The FF planning system: Fast plan generation through heuristic search. Journal of Artificial Intelligence Research, 14, 253–302.

[28] Hoffmann, J., Porteous, J., & Sebastia, L. (2004). Ordered Landmarks in Planning. Journal of Artificial Intelligence Research, 22, 215–278.

[29] Hollander, M., Wolfe, D. A., & Chicken, E. (2014). Nonparametric statistical methods (3rd ed.). Wiley.

[30] Holte, R. C. (2010). Common misconceptions concerning heuristic search. Symposium on Combinatorial Search, 46–51.

[31] Karpas, E., & Domshlak, C. (2009). Cost-optimal planning with landmarks. International Joint Conference on Artificial Intelligence, 1728–1733.

[32] Kingma, D., & Ba, J. (2015). Adam: A method for stochastic optimization. International Conference on Learning Representations.

[33] Lu, L., Shin, Y., Su, Y., & Karniadakis, G. E. (2020). Dying ReLU and initialization: Theory and numerical examples. Communications in Computational Physics, 28(5), 1671–1706.

[34] O'Toole, S., Ramirez, M., Lipovetzky, N., & Pearce, A. R. (2022). Sampling from pre-images to learn heuristic functions for classical planning. Symposium on Combinatorial Search, 15(1), 308–310.

[35] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., . . . Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. Advances In Neural Information Processing Systems, 32, 8024–8035.

[36] Rivlin, O., Hazan, T., & Karpas, E. (2020). Generalized planning with deep reinforcement learning. arXiv: 2005.02305 [cs.AI].

[37] Samadi, M., Felner, A., & Schaeffer, J. (2008). Learning from Multiple Heuristics. AAAI Conference on Artificial Intelligence, 357–362.

[38] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). The graph neural network model. IEEE Transactions on Neural Networks, 20(1), 61–80.

[39] Shen, W., Trevizan, F., & Thiebaux, S. (2020). Learning domain-independent planning heuristics with hypergraph networks. International Conference on Automated Planning and Scheduling, 574–584.

[40] Ståhlberg, S., Bonet, B., & Geffner, H. (2022). Learning general optimal policies with graph neural networks: Expressive power, transparency, and limits. International Conference on Automated Planning and Scheduling, 32, 629–637.

[41] Sturtevant, N., & Helmert, M. (2019). Exponential-binary state-space search. arXiv: 1906.02912 [cs.AI].

[42] Toyer, S., Thiebaux, S., Trevizan, F., & Xie, L. (2020). ASNets: Deep learning for generalised planning. Journal of Artificial Intelligence Research, 68, 1–68.

[43] Toyer, S., Trevizan, F., Thiebaux, S., & Xie, L. (2018). Action schema networks: Generalised policies with deep learning. AAAI Conference on Artificial Intelligence, 6294–6301.

[44] van Den Briel, M., Benton, J., Kambhampati, S., & Vossen, T. (2007). An LP-based heuristic for optimal planning. Principles and Practice of Constraint Programming, 651–665.

[45] Yu, L., Kuroiwa, R., & Fukunaga, A. (2020).

Learning search-space specific heuristics using neural network. ICAPS Workshop on Heuristics and Search for Domain-independent Planning.