Scaling Multi-Agent Epistemic Planning through GNN-Derived Heuristics

Giovanni Briglia*
Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia
Reggio Emilia, Italy
giovanni.briglia@unimore.it

Francesco Fabiano
Department of Computer Science, University of Oxford
Oxford, United Kingdom
francesco.fabiano@cs.ox.ac.uk

Stefano Mariani
Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia
Reggio Emilia, Italy
stefano.mariani@unimore.it

ABSTRACT

Multi-agent Epistemic Planning (MEP) is an autonomous planning framework for reasoning about both the physical world and the beliefs of agents, with applications in domains where information flow and awareness among agents are critical. The richness of MEP requires states to be represented as *Kripke structures*, *i.e.*, directed labeled graphs. This representation limits the applicability of existing heuristics, hindering the scalability of epistemic solvers, which must explore an exponential search space without guidance, resulting often in intractability. To address this, we exploit *Graph Neural Networks* (GNNs) to learn patterns and relational structures within epistemic states, to guide the planning process. GNNs, which naturally capture the graph-like nature of Kripke models, allow us to derive meaningful estimates of state quality—*e.g.*, the distance from the nearest goal—by generalizing knowledge obtained from previously solved planning instances. We integrate these predictive heuristics into an epistemic planning pipeline and evaluate them against standard baselines, showing improvements in the scalability of multi-agent epistemic planning.

KEYWORDS

Multi-Agent Epistemic Planning; Graph-Neural Networks; Learning in Planning; Heuristics

ACM Reference Format:
Giovanni Briglia, Francesco Fabiano, and Stefano Mariani. 2026. Scaling Multi-Agent Epistemic Planning through GNN-Derived Heuristics. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 10 pages. https://doi.org/10.65109/QMRY9661

1 INTRODUCTION

Planning scenarios involving multiple interacting entities, referred to as *multi-agent*, have gained increasing importance due to their relevance in real-world applications, where groups of agents frequently need to interact. However, effectively addressing multi-agent settings poses one of the most interesting challenges in modern AI research: adequately modeling multi-agent interaction while maintaining tractability [10]. This is because such modeling requires accounting not only for the state of the world, but also for the dynamics of information exchange between agents. Such reasoning, which deals with formalizing belief relationships among multiple agents, is referred to as *epistemic reasoning* [26].

Interest in *Multi-agent Epistemic Planning* (MEP)—which integrates epistemic reasoning with automated planning—has surged [2], and several epistemic planners have been proposed [5, 6, 21, 23, 37, 41, 45, 53]. To the best of our knowledge, only a few systems [14, 23, 45] are capable of reasoning over this setting without restrictions. Nonetheless, these systems are severely limited by high computational costs, often making solving impractical. This inefficiency stems mainly from two factors: *(1)* the intrinsic complexity of the underlying representations, which makes applying transitions and evaluating formulas within epistemic states (e-states) substantially harder than in classical planning; and *(2)* the lack of effective heuristics, which results in a blind, combinatorial search as plan length increases. While the aforementioned works in MEP largely address the first issue, few efforts tackle the latter. A notable exception is the $\mathcal{H}$-EFP planner [25], an extension of Le et al. [39], which integrates heuristics guidance to improve scalability. Our work builds upon this direction, sharing the core objective of designing effective heuristics extraction methods. We argue this focus is essential, as *informed search* is what enables scalability in planning systems—from classical heuristics planning [7, 30] to *Monte Carlo Tree Search* (MCTS) in *reinforcement learning* (RL) [9].

The key difference in our approach lies in how heuristics are defined and computed. Unlike Fabiano et al. [25], Le et al. [39], who construct heuristics using traditional planning constructs—such as the *planning graph*—our method adopts a data-driven approach grounded in *Machine Learning* (ML). Specifically, we leverage Graph Neural Networks (GNNs) to extract information from e-states in MEP—modeled as Kripke structures (Definition 2.1)—to estimate the "quality" of these states and derive heuristics functions. The core idea is to use GNNs to approximate the *perfect heuristics*, *i.e.*, to estimate the distance from any epistemic state to the nearest goal. These learned heuristics are then used to guide an *informed search* algorithm [7], enabling efficient traversal of the search space and mitigating its exponential growth. We also introduce techniques for generating the training data required by the GNN-based regressor through a dedicated data generation process. This entire pipeline is implemented in deep (dynamic epistemic logic-based planner),[1] a novel iteration of the state-of-the-art epistemic planner EFP [23, 25].

The key contributions of this work are as follows:

*Also with Department of Computer Science, University of Pisa.

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/QMRY9661

[1] Code available at https://github.com/FrancescoFabiano/deep.

(1) We define and compare three embeddings for Kripke structures to serve as input to a GNN-based regressor.
(2) We propose a fully automated pipeline for efficient data generation and training of the GNN-based regressor to approximate the perfect heuristics in the MEP setting.
(3) We integrate the GNN-regressor into the MEP solving process, where it is used to evaluate epistemic states by assigning heuristics scores that guide the search process.
(4) We provide a comprehensive evaluation of this integration by thoroughly testing several benchmarks.

These contributions, supported by experimental results, represent a foundational step in integrating ML with MEP.

The remainder of this paper is structured as follows. In Section 2, we provide background on MEP and GNNs. Section 3 presents our main theoretical contribution. In particular, Section 3.1 presents the design of the embedding and the dataset generation while Section 3.4 illustrates the training of the GNN-based regressor. Section 4 reports experimental results that evaluate the performance and scalability of our approach. We discuss limitations and related work in Sections 5 and 6, and conclude in Section 7.

2 BACKGROUND

2.1 Dynamic Epistemic Logic

Dynamic Epistemic Logic (DEL) formalizes reasoning about the state of the world and about the dynamic nature of information change, *i.e.*, about higher-order knowledge and/or beliefs. For brevity, this discussion will present only the fundamental intuitions of DEL. Interested readers can explore further details in Moss [40].

Let us denote $\mathcal{AG}$ as a set of agents such that $|\mathcal{AG}| = n$ with $n \geq 1$, and $\mathcal{F}$ as a set of propositional variables, referred to as *fluents literals*, or simply *fluents*. Each *world* is described by a subset of elements from $\mathcal{F}$ intuitively, those deemed True. Furthermore, in epistemic logic, each agent $\mathsf{i} \in \mathcal{AG}$ is associated to an epistemic modal operator $\mathbf{B}_\mathsf{i}$, signifying the belief[2] of the agent. Additionally, the epistemic *group operator* $\mathbf{C}_\alpha$ is introduced. Essentially, this operator represents the *common knowledge* of a group of agents α.

To be more precise, as in Baral et al. [3], we have that a *fluent formula* is a propositional formula built using fluents in $\mathcal{F}$ as propositional variables and the propositional operators $\wedge, \vee, \Rightarrow, \neg$. On the other hand, a *belief formula* is either *(i)* a fluent formula; *(ii)* if φ is a belief formula and $\mathsf{i} \in \mathcal{AG}$, then $\mathbf{B}_\mathsf{i}(\varphi)$ is a belief formula; *(iii)* if φ_1, φ_2 and φ_3 are belief formulae, then $\neg\varphi_3$ and $\varphi_1 \,\mathsf{op}\, \varphi_2$ are belief formulae, where $\mathsf{op} \in \{\wedge, \vee, \Rightarrow\}$; or *(iv)* if φ is a belief formula and $\emptyset \neq \alpha \subseteq \mathcal{AG}$ then $\mathbf{C}_\alpha\varphi$ is a belief formula. $\mathcal{L}^{\mathrm{C}}_{\mathcal{AG}}$ denotes the language of the belief formulae over the set $\mathcal{AG}$.

The classical way of providing semantics for epistemic logic is in terms of *pointed Kripke structures* [38].

Definition 2.1 (Pointed Kripke structure). Let $|\mathcal{AG}| = n$ with $n \geq 1$. A *pointed Kripke structure* is a pair $(M = \langle S, \pi, \mathcal{B}_1, \ldots, \mathcal{B}_\mathsf{n}\rangle, \mathsf{s})$, such that:

- S is a set of worlds;
- $\pi : S \mapsto 2^{\mathcal{F}}$ is a function that associates an interpretation of $\mathcal{F}$ to each element of S;
- for $1 \leq \mathsf{i} \leq \mathsf{n}$, $\mathcal{B}_\mathsf{i} \subseteq S \times S$ is a binary relation over S; and
- $\mathsf{s} \in S$ points at the real world.

To elaborate, the component S encompasses all the possible worlds configurations, while $\mathcal{B}_\mathsf{i}$ specifically represents the beliefs held by each individual agent.

Intuitively, to verify whether a belief formula holds, we need to apply reachability within the Kripke model representing the e-state. By exploring the set of reachable worlds obtained by applying epistemic operators, we determine which configurations of fluents an agent (or group of agents) considers possible. Inconsistencies among these reachable worlds are used to model ignorance. The formal semantics over pointed Kripke structures is provided in [3, 23] and omitted here as it is not integral to understanding the contribution of this paper.

2.2 Multi-Agent Epistemic Planning

We are now ready to introduce the fundamental concepts of MEP, while addressing interested readers to Bolander and Andersen [5], Fagin et al. [26] for a more exhaustive introduction.

Let us begin by providing the notion of a *multi-agent epistemic planning problem* in Definition 2.2. Intuitively, an epistemic planning problem encompasses all the necessary information to frame a planning problem within a multi-agent scenario.

Definition 2.2 (Multi-agent epistemic planning problem). We define a multi-agent epistemic problem as the tuple $P = \langle D = \langle \mathcal{F}, \mathcal{AG}, \mathcal{A}\rangle, \mathcal{I}, \mathcal{G}\rangle$ where:

- $\mathcal{F}$ is the set of all the *fluents* of P;
- $\mathcal{AG}$ is the set of the *agents* of P;
- $\mathcal{A}$ represents the set of all the *actions*;
- $\mathcal{I}$ is the set of belief formulae that describes the *initial conditions* of the planning process; and
- $\mathcal{G}$ is the set of belief formulae that represents the *goal* conditions.

Note that the tuple $D = \langle \mathcal{F}, \mathcal{AG}, \mathcal{A}\rangle$ captures the domain description of which the problem P is an instance.

A solution, or a *plan*, of a MEP problem is a sequence of actions in D that, when executed, transforms the initial e-state into one that satisfies the $\mathcal{G}$.

In this context, an epistemic state—represented by a pointed Kripke structure—encapsulates a problem's "physical" configuration along with the beliefs of the agents.

To the best of our knowledge, the most accepted formalization of a comprehensive action language for multi-agent epistemic planning is $m\mathcal{A}^*$ [3]. Note that other languages capable of reasoning about DEL also exist [6, 41], but they limit their expressiveness in favor of efficiency and are therefore not considered here.

$m\mathcal{A}^*$ serves as a high-level action language facilitating reasoning about agents' beliefs within $\mathcal{L}^{\mathrm{C}}_{\mathcal{AG}}$, where e-states are represented as Kripke structures. It utilizes an English-like syntax, leverages *event models* to define the transition functions, and uses reachability over Kripke models to characterize entailment.

In their work Baral et al. [3] delineate three distinct types of actions within the context of multi-agent domains: *(i) world-altering* actions (or *ontic* actions): employed to modify specific properties, or fluents, within the world—denoted through the statement

[2] We use the terms knowledge and belief interchangeably, as their distinction is beyond this work's scope. See Fagin et al. [26] for a full discussion.

"act_name **causes** ℓ[3]"; *(ii) sensing* actions: used by an agent to refine her beliefs about the world—denoted through the statement "act_name **determines** f"; and *(iii) announcement* actions: utilized by an agent to influence the beliefs held by other agents—denoted through the statement "act_name **announces** f". For brevity, we will not provide further details of $m\mathcal{A}^*$ here. Interested readers can find a comprehensive description in Baral et al. [3].

2.3 Graph Neural Networks (GNNs)

Machine Learning consists of techniques that learn patterns from data and use them to make predictions and generalize, without being explicitly programmed. GNNs [12] extend this idea to graph-structured data: they update node representations by repeatedly exchanging information along edges. This is possible through *message-passing* mechanisms, which allow GNNs to detect structural and semantic regularities across nodes and subgraphs. This makes GNNs particularly suitable for MEP, where epistemic states are represented as directed labeled graphs—*e.g.*, Kripke structures.

In our neural estimator, the GNN forms the first stage of the pipeline: it processes the graph representation of an epistemic state and produces a compact latent embedding that serves as the basis for the heuristic estimate, capturing the structural and logical characteristics of the state. Here, message passing is implemented via GINEConv [31]: given a node v and its neighborhood $N(v)$, the update at layer ℓ is

$$h_v^{(\ell+1)} = \phi\Big(h_v^{(\ell)}, \sum_{u \in N(v)} \psi\big(h_v^{(\ell)}, h_u^{(\ell)}, e_{uv}\big)\Big),$$

where $h_v^{(\ell)}$ is the embedding of node v at layer ℓ, e_{uv} is the embedding of the edge (u, v), and ψ and ϕ are multilayer perceptrons (MLPs). By comparison, the GCN [36] layers update the node features while *ignoring* the edge attributes, and the GAT [52] layers introduce attention coefficients but still assume homogeneous edge types. The inclusion of e_{uv} in ψ makes GINE more expressive for relational structures such as Kripke structures, enabling the model to differentiate heterogeneous edge types (*e.g.*, distinct modal dependencies) that GCN and GAT cannot represent directly.

Reviews of architectures and applications are proposed in Wu et al. [54] and Zhou et al. [55], respectively.

3 LEARNING MEP HEURISTICS WITH GNN

As discussed, planning in MEP is an extremely resource-intensive task. To mitigate this computational burden, we explore alternatives to *Breadth-First Search* (**BFS**), such as *Best-First Search*, which we abbreviate as **HFS** (for *Heuristics-First Search*) to distinguish it from **BFS**. **HFS** is a widely used strategy that prioritizes expanding states with the most favorable heuristics score. Both **BFS** and **HFS** are standard approaches described in Russell and Norvig [46, Ch. 11].

BFS is an *uninformed* search method, exploring the state space uniformly. In contrast, **HFS** aims to guide the search more efficiently by prioritizing e-states that are likely closer to the goal. The key challenge lies in defining an effective evaluation function—also known as a *heuristics*—capable of accurately ranking epistemic states. Heuristics are a central topic in the planning literature and have been extensively studied and proven effective [35, 46]. For this reason, a key contribution of this work is the formalization of heuristics tailored for MEP through the employment of GNNs.

[3] ℓ can be either a fluent or its negation

3.1 Training Data Generation

The first key challenge in employing ML techniques in MEP is determining what kind of information can be meaningfully extrapolated from data. Several approaches have been proposed in automated planning, such as training models end-to-end using full problem descriptions and their associated goals [32, 43]. However, these methods often suffer from low accuracy and require a large number of training instances to generalize effectively [34].

To address these limitations, we adopt a learning approach in which every state represents an independent training signal, rather than relying on complete trajectories. This offers two main advantages. First, although individual predictions may occasionally be inaccurate, their impact on the overall search is limited—as long as the trend of the heuristics is informative, the search remains effective. Second, this approach significantly reduces the amount of data required. Instead of needing complete problems as training instances, we can treat each e-state encountered during exploration as a distinct data point. This allows the generation of tens of thousands of e-states, yielding a large set of training examples in a single run of the planner.

The main objective of this work is to use GNNs to extract information from the data to guide the search. In particular, we aim at approximating the perfect heuristics that assign to each state its distance to the nearest goal.

The need for reasoning over e-states is precisely why we adopt GNNs over other neural architectures. As previously noted, e-states in MEP are represented as labeled, directed graphs, specifically, Kripke structures. These grow unboundedly in size with the length of the plan. In fact, although the set of possible world valuations is finite, *i.e.*, for a planning problem P the number of valuations is exactly $2^{|P(\mathcal{F})|}$, the same valuation may appear multiple times within a Kripke model to capture complex belief relations. This unbounded nature and the inherently relational semantics of epistemic states demand the full expressive power of graph-based models in order to capture the nuances of knowledge and belief.

3.2 e-State Representation

The next challenge we addressed was designing a suitable data representation for integration within a GNN. Each e-state (M, s) is formalized as a pointed Kripke structure (Definition 2.1), namely $(M = \langle S, \pi, \mathcal{B}_1, \ldots, \mathcal{B}_n \rangle, s)$.

Encoding the relational structure $\mathcal{B}_i$ is straightforward, as each agent i can be mapped to a unique integer label. The primary challenge lies in defining a consistent and informative numerical representation of the worlds S suitable for GNN input.

We explored three strategies, each offering a different trade-off between information preservation and efficiency. In Section 4.1, we compare these representations, illustrated below, and show that hash-based encoding achieves the best performance.

Symbolic ID Mapping. Each world $s \in S$ is assigned a symbolic integer identifier through a mapping $\phi : S \to \mathbb{N}$, where $\phi(s_i) = k$ iff s_i is the k-th distinct world encountered. Intuitively, this assigns consecutive integers to newly observed worlds. However,

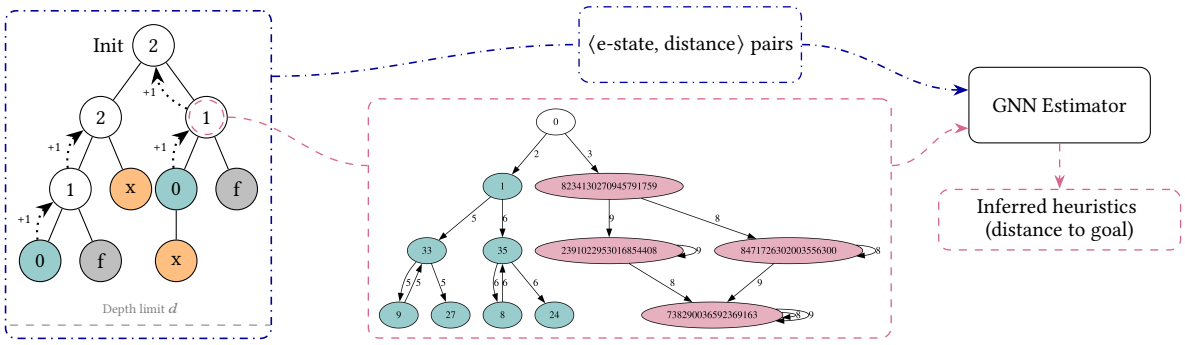

Figure 1: Illustration of the overall training and inference process. On the left, we show dataset generation via DFS: teal nodes represent goals (score 0), black *dotted* arrows show backtracking assigning distances, orange nodes ('x') indicate discarded branches, and gray nodes ('f') are states with no reachable goal. Training is shown by the blue *dashdotted* lines: ⟨e-state, distance⟩ pairs generated by the DFS are fed into the GNN to learn the properties of the e-states. Following the magenta *dashed* lines, we illustrate Inference where a single e-state—shown in its expanded view—is input to the GNN to retrieve its estimated distance to the goal. The teal portion represents the goal encoding, while the magenta portion represents the actual e-state.

since this assignment depends on the order in which worlds are generated, making ϕ not invariant across runs, leading to inconsistencies between datasets. This representation has the lowest computational burden for data generation, training, and inference, but it incurs the greatest information loss, as similarities between worlds are not preserved.

Hash-based Encoding. A hash function $h : S \to \mathbb{N}_{\geq 0}$ is applied to each world: $h(s) = \text{Hash}(\pi(s), r_s)$; where r_s denotes the repetition index distinguishing identical copies of the same world within an e-state. This ensures consistent world identifiers across runs, but relative distances between world evaluations are lost, as, for example, evaluations differing only by a single fluent value may receive completely unrelated hash values. We use the `hash_range` function from the Boost libraries [47] due to its high performance and existing use in `deep` for e-state storage, which minimizes computational overhead. Alternative hashing approaches may offer further improvements and are left for future work.

This encoding represents a middle ground in terms of the trade-off between information loss and the computational power required.

Bitmask Encoding. Each world s is represented as a purely binary vector: $\mathbf{b}(s) = [\, \mathbf{r}_s \mid b_1, b_2, \ldots, b_l \,]$, where $\mathbf{r}_s$ is the binary representation of the repetition index r_s, and $b_j = 1$ if fluent $\mathtt{f}_j \in \mathcal{F}$ holds in $\pi(s)$, and $b_j = 0$ otherwise (or when $j > |\mathcal{F}|$). Note that, by construction, l is assumed to be greater than $|\mathcal{F}|$. Thus, $\mathbf{b}(s)$ encodes both the repetition number of the world within the e-state and its truth assignment over $\mathcal{F}$ as a single binary vector. This embedding preserves most of the logical information about the e-estate, but it comes at the cost of a larger encoding, which requires more data for the model to converge appropriately and results in slower generation, training, and inference.

3.2.1 Goal Encoding within the State Embedding. While the encoded Kripke model—represented through the *dot* language [20]—proved effective, we encountered a second challenge. Training solely on states led the regressor to learn absolute measures of distance. This measure is independent of the underlying goal and therefore generalizes only to problems with identical goal conditions, which limits the applicability of the heuristics.

To mitigate this limitation, we extend the state embedding to include a compact encoding of the goal. This enables the regressor to learn dependencies not only between the e-state and its distance to the goal, but also between structural features of the goal itself. The design objective was to preserve a uniform input dimensionality while introducing minimal overhead. Given that the conversion process is not particularly relevant, we omit its details here and refer the reader to the extended version of this paper [11, Appendix C] for the description of the procedure.

In the resulting formulation, two types of nodes are encoded for each e-state embedding: *(i) state nodes*, corresponding to worlds in the epistemic state; *(ii) goal nodes*, corresponding to symbols describing the goal conditions. This is exemplified in Figure 1, where the left portion of the expanded hashed-based e-state (in teal) shows the goal encoding, and the right portion (in magenta) depicts the e-state itself. The two are connected via a shared graph structure using nodes and edges that use constants and special identifiers throughout the process.

In the hashed- and in the mapping-based representations, the state component remains unchanged, while goal nodes are associated with unique integer identifiers (assigned to the first occurrences of each goal symbol). In the bitmask representation, both node types share the same binary vector structure: $\mathbf{b}(s) = [\, \mathbf{m}_s = [\, \mathbf{r}_s \mid b_1, b_2, \ldots, b_{|l|} \,] \mid \mathbf{g}_s \,]$, where $\mathbf{m}_s$ reflects the bitmask embedding presented above and $\mathbf{g}_s$ is the binary segment reserved for goal encoding. During the experimental evaluation, we fixed the total size of $\mathbf{b}(s)$ to 64 bits, with $|\mathbf{r}_s| = 16$, $l = 32$, and $|\mathbf{g}_s| = 16$.

For *state nodes*, the goal segment is not used ($\mathbf{g}_s = \mathbf{0}$), serving purely as padding to preserve dimensional uniformity across the GNN input space. For *goal nodes*, the segment $\mathbf{g}_s$ encodes the binary representation of the integer identifier associated with the corresponding goal symbol while $\mathbf{m}_s = 0$. Each goal symbol is uniquely represented, while agent-related components reuse the same integer indices as in the e-state encoding.

3.3 Building the Dataset

Having identified the type of data required for training, the next challenge lies in generating such data. To address this, we equipped deep with a *dataset generation* mode that produces pairs of epistemic states and their distances to the nearest goal. This allows training data to be collected from a small set of problems, which is then used to train GNN-based neural regressors.

The generation process works as follows: given a MEP problem, the planner performs a depth-limited *Depth-First Search* (**DFS**) to explore the reachable state space up to a specified depth d (left part of Figure 1). During this traversal, all reachable goal states are identified. deep then backtracks from each goal, assigning to each epistemic state the distance to the closest goal—yielding a dataset that approximates the perfect heuristics. States from which no goal is reachable within depth d are labeled with a special value.

Although conceptually simple, this process suffers from combinatorial explosion: with only 10 actions and depth 25—an overestimate of typical plan length in standard MEP benchmarks—the search space can reach 10^{25} nodes, making exhaustive exploration infeasible. To mitigate this, we draw on ideas from *local search* [28], sampling subregions to maximize coverage. Our **DFS** incorporates probabilistic branch pruning (adaptive to depth and node count), randomized action ordering to avoid prefix bias, a node expansion cap, and duplicate e-state checks. These mechanisms enable diverse yet tractable exploration, allowing informative datasets to be built within minutes per instance.

While data generation and training are handled offline in this work, the structure of our learning pipeline naturally supports an online setup. By slightly extending the search process, the planner could incrementally collect training pairs and update the GNN once a sufficient number of samples is accumulated. Thanks to deep's multithreading support—introduced to emulate the portfolio behavior of $\mathcal{H}$-EFP [25]—this online learning loop could run in parallel, allowing the planner to adapt and improve over time, similar to the cognitive architecture presented in Fabiano et al. [24]. We leave the exploration of this online learning paradigm to future work.

3.4 Training Neural Distance Estimator

Our objective is to train an estimator that, given an e-state, predicts the distance to a goal state. To this end, three design choices were made:

(1) *Discard unreachable nodes.* Distance to the goal is meaningful only for e-states from which the goal can be reached. Keeping unreachable e-states in the training set injects spurious labels that raise the estimator's variance without lowering its bias.
(2) *Limit the number of samples for any distance class.* Raw rollouts produce a strongly skewed distribution, with many more short distance e-states than long distance ones, which can bias the estimator toward the majority class. We therefore limit each distance bin (in percentage) to at most $\mathrm{p_M}$ of the dataset. This balance step *(a)* reduces variance arising from class imbalance while keeping bias low; *(b)* forces the network to allocate capacity uniformly throughout the distance spectrum, lowering worst-case error and improving robustness.
(3) *Linearly normalize the distance target.* To stabilize training, we linearly normalize the true distance $d \in [0, \mathrm{D_{max}}]$ from $\mathrm{min\_val} \in [0, 1)$ to $\mathrm{max\_val} \in (0, 1]$, with $\mathrm{max\_val} \geq \mathrm{min\_val}$. Let $\alpha = \frac{\mathrm{max\_val} - \mathrm{min\_val}}{\mathrm{D_{max}}}$, and $\beta = \mathrm{min\_val}$, then the normalized target is $\tilde{d} = \alpha\, d + \beta \in [0, 1]$. In inference, we recover the original scale through $d = \frac{\tilde{d} - \beta}{\alpha} \in [0, \mathrm{D_{max}}]$
This normalization bounds the regression target, yielding predictable gradient magnitudes, avoiding activation saturation (*e.g.*, sigmoid or tanh), and aligning with common weight-initialization schemes.

Our neural regressor is implemented in `PyTorch` [33], and its graph encoder is built using `PyTorch Geometric` [4]. For each epistemic state (e-state), represented as a graph $G = (V, E)$ with nodes V and edges E, we construct a corresponding data object that encodes the graph structure. This object serves as input to the GNN, which processes it to produce a latent embedding. Specifically, this object includes the following: *(i) Node identifiers:* $V = \{v_i\}_{i=1}^{|V|}$, represented as bit-vectors, in which each node is expressed as a binary vector $b_i \in \{0, 1\}^{d_b}$, where d_b is the bit-length (here $d_b = 64$). The resulting node feature matrix is $X \in \{0, 1\}^{|V| \times d_b}$; *(ii) Edge indices:* $E = \{(u_k, v_k)\}_{k=1}^{|E|}$, encoded as an index tensor $I \in \mathbb{N}^{2 \times |E|}$, whose columns list each source–target pair; *(iii) Edge attributes:* $A = \{a_k\}_{k=1}^{|E|}, \quad a_k \in \mathbb{R}$, stacked into an attribute tensor $A \in \mathbb{R}^{|E| \times 1}$ and normalized to $[0, 1]$. The estimator is trained by minimizing the Mean Squared Error (MSE) between its predicted scalar $\hat{d}$ and the true normalized distance $\tilde{d} \in [0, 1] : \mathrm{MSE} = \frac{1}{B} \sum_{i=1}^{B} (\hat{d}_i - \tilde{d}_i)^2$, using the `AdamW` optimizer with the parameters listed in the extended version of this paper [11, Appendix B], and where B is the mini-batch size (number of samples per gradient update).

The forward pass unfolds in four stages:

(1) *Bitwise embedding of node identifiers and edge attributes:* Each node bit-vector $b_i \in \{0, 1\}^{d_b}$ is projected through a two-layer MLP, $x_i = f_{\mathrm{id}}(b_i) = \mathrm{MLP_{id}}(b_i) \in \mathbb{R}^{d_v}$, producing dense embeddings of dimension $d_v = \mathtt{node\_emb\_dim}$. Similarly, each edge attribute a_k is processed through $e_k = f_{\mathrm{edge}}(a_k) = \mathrm{MLP_{edge}}(a_k) \in \mathbb{R}^{d_e}$, yielding edge_emb_dim-dimensional embeddings. This bitwise encoding provides a compact and lossless representation of node identifiers while supporting continuous optimization and stable gradients.
(2) *Relational message passing with GINEConv:* Two successive GINEConv [31] layers, each followed by a ReLU activation, allow each node to aggregate information from its neighbors while explicitly incorporating edge indices and attributes.
(3) *Graph-level summarization:* The node embeddings from the final GINE layer are aggregated into a fixed-size representation via *mean pooling*: $h_G = \frac{1}{|V|} \sum_{v \in V} h_v$. This operation is permutation-invariant and scale-independent, ensuring stable representations across graphs of different sizes. In contrast, sum pooling scales with $|V|$, potentially biasing larger graphs, while attention pooling introduces learnable weights but increases variance and computational cost. Mean pooling therefore offers the best generalization–variance trade-off for distance regression.

(4) *Deep residual regression head with bounded output:* The pooled graph embedding h_G is processed by a deep residual regressor: $\hat{d} = \sigma(f_{\text{res}}(h_G))$, where f_{res} denotes a sequence of ResidualBlocks (Linear → BatchNorm → ReLU → Dropout → Linear + skip). The sigmoid σ bounds the output in $(0, 1)$, and $\hat{d}$ is finally clamped to $[\texttt{min\_val}, 1 - \texttt{min\_val}]$ to prevent numerical instabilities near the boundaries.

Overall, the combination of *bitwise node embeddings*, *GINEConv* message passing, and *mean pooling* yields a compact and expressive model that generalizes across variable-sized epistemic graphs while remaining numerically stable during training. The full forward-pass is detailed in the extended version of this paper [11, Appendix E].

4 EXPERIMENTS AND EVALUATION

To conduct our experiments, we developed `deep`, a modernized reimplementation of the epistemic planner EFP [23, 25]. `deep` serves as the primary platform for our evaluation and features a modular design that supports the integration of diverse heuristics, including GNN–based ones. Full implementation details and usage instructions are available at https://github.com/FrancescoFabiano/deep.

Here we present a comparative analysis of our primary contribution, `deep` equipped with **HFS***—a depth-augmented variant of **HFS** detailed in Section 4.1—alongside GNN-based heuristics (denoted as `GNN`), and `deep` using Breadth-First Search (denoted as `BFS`). All executions employed bisimulation-based state reduction and visited-state checks, as introduced by Fabiano et al. [23]. For completeness, we also provide comparisons with the different heuristics implemented in $\mathcal{H}$-EFP [25]. All of these heuristics remain available in `deep` and can be combined with the GNN-based one in a portfolio approach. The solver is already equipped to perform this integration easily as a runtime option. We omit the results of such integration, as the focus of this paper is on the use of ML in epistemic planning, rather than on the planner itself.

While several other MEP solvers exist [6, 23, 41], we focus our evaluation on `BFS` to emphasize the impact of incorporating learned heuristics guidance into MEP solving. All experiments were conducted with a timeout of 600 seconds on a 13th-generation Intel® Core™ i9-13900H CPU with 64 GB of system RAM and an NVIDIA RTX 4070 GPU with 8 GB of VRAM.

The evaluation encompasses several standard benchmarks in the MEP setting. For brevity, the definitions of the parameters used for training and the descriptions of the domains, are provided in the extended version of this paper [11, Appendix B and D].

Experimental Setup. To evaluate our contribution, we conducted experiments on standard benchmark domains. We tested two configurations: in the first, each domain had its own model trained solely on data from that domain; in the second, we evaluated the knowledge transfer capabilities of models trained on data pooled from multiple domains and tested on both seen and unseen domains.

We report a subset of experiments to highlight key trends and the impact of our contribution, while full results are provided in the extended version of this paper [11, Appendix G].

Metrics. Our primary evaluation metric is the number of nodes expanded during the search (Nodes), which reflects the informativeness of the heuristics. We focus on this measure because `GNN` currently does not leverage batch computation, making runtime comparisons less meaningful. For completeness, we also report plan length (Length) and solving times in ms (Time).

For aggregate metrics, we report the Interquartile Mean (IQM) and IQR standard deviation (IQR-std) [1], focusing on problems solved by all approaches.

4.1 E-State Representation and Search Strategy

In this section, we analyze how the e-state representation and the search strategy affect the overall performance of our approach.

Specifically, we compare the three e-state representations introduced in Section 3.2 and the two search strategies, **HFS** and its variant **HFS***. The latter, inspired by the classical **A*** algorithm[4] [27, 29], augments the heuristics value $h(s)$ predicted by the GNN with the search depth $d(s)$ to form the total score: $f(s) = h(s) + d(s)$.

We evaluated all six possible combinations, namely the three representations, each paired with both **HFS** and **HFS***. These experiments use GNN models trained on the same domain used during inference. We report aggregate results over both the Training and Test sets. We refer the reader to the extended version of this paper [11, Appendix F] for complete results. We remind that the aggregates are computed only on instances solved by all the approaches.

Table 1 reports results for the three e-state representations—`map`, `hash`, and `mask`—corresponding respectively to the symbolic ID, hashing, and bitmask formulations. As shown in Table 1, `hash` achieves the best overall performance. This can be attributed to the intrinsic richness of the e-state representation, which enables effective information retrieval even when part of the original information is discarded. Although the bitmask representation is only slightly outperformed by the hashed one in specific cases, we consider the trade-off offered by `hash`—lighter training requirements, faster data generation, and reduced inference time—well justifies choosing this as the best encoding.

	Solved Inst.	Nodes	Time [ms]	Length
`map`	74/79 (93.67%)	87 ± 367	2358 ± 9251	6 ± 3
`hash`	75/79 (94.94%)	45 ± 227	871 ± 3225	6 ± 3
`mask`	75/79 (94.94%)	55 ± 224	1121 ± 4151	6 ± 3

Table 1: Comparison of the three data representations.

Next, in Table 2, we illustrate the advantage of using **HFS*** over **HFS**. The results clearly indicate that **HFS*** achieves better performance, solving approximately 32% more instances.

	Solved Inst.	Nodes	Time [ms]	Length
HFS	49/79 (62.03%)	18 ± 27	1290 ± 5528	8 ± 8
HFS*	75/79 (94.94%)	17 ± 42	584 ± 2276	6 ± 4

Table 2: Comparison of HFS and HFS*.

As mentioned, for the sake of space, we presented only aggregate results. We note, however, that the trends shown in Tables 1 and 2

[4] Since the GNN-based heuristics is statistical in nature, its admissibility cannot be guaranteed; hence we do not refer to it as **A***.

were consistently observed across all individual experiments. From this point onward, we therefore adopt hash combined with **HFS*** throughout the paper as our GNN configuration.

4.2 Experimental Results

Table 3 summarizes the aggregate results across all domains, reporting the IQM of Nodes, Time, and Length for each. It compares the GNN-based regressor (GNN) with uninformed search (BFS), where the GNN models are trained on the same domain used at inference time. We report only the results of the Test set (*i.e.*, problems not seen during training).

In Table 4, as a study on the knowledge transfer analysis, we compare GNN with CC-GR, a model trained using the training instances of two domains, namely **CC** and **GR**.

Finally, Table 5 presents aggregated results comparing our approach with existing heuristics in $\mathcal{H}$-EFP. For this evaluation, we use CC-GR, as it represents the most general-purpose configuration. A detailed comparison with individual heuristics (C_PG, L_PG, S_PG, and SUB) is provided in the extended version of this paper [11, Appendix G]. We note that the only metric with significant interpretability is the number of instances solved, since the $\mathcal{H}$-EFP heuristics solve a highly diverse set of instances, making other aggregated measures less informative.

Tables 4 and 5 summarize performance on the full *Test* set across all benchmark domains.

4.3 Discussion

GNN demonstrates informative and robust performance across a range of planning domains, as evidenced by the aggregate metrics reported in Table 3. With the exception of the **GR** domain, GNN consistently reduces the number of explored nodes compared to uninformed search, highlighting the efficiency gains enabled by the learned heuristic.

The **GR** domain presents a particular challenge due to its sparse solution density: even small heuristic inaccuracies can result in poor search guidance, and instances with superficially similar states or goals may require substantially different plans. This variability can mislead GNN and explains its reduced effectiveness in this domain. Importantly, this observation motivates the portfolio-based heuristic selection strategy adopted in deep, where complementary heuristics can compensate when individual ones underperform. A tighter integration of GNN-based heuristics into this portfolio framework is a natural direction for future work.

Further evidence of scalability is provided in Table 4, where GNN outperforms the baseline in terms of explored nodes, demonstrating its ability to generalize across domains, including those unseen during training. We focus on expanded nodes as this metric directly reflects heuristic informativeness rather than implementation dependent overhead. At this stage, GNN inference remains a proof of concept and lacks several engineering optimizations, *e.g.*, batch inference, making runtime performance non-competitive (more information can be found in the extended version [11, Appendix H]). In contrast, heuristics informativeness is a stable property, expected to persist once such optimizations are introduced.

Results for Length are comparable across methods, indicating that GNN preserves near-optimal solution quality. Solving times are likewise similar, although GNN incurs a modest overhead due to the absence of batch computation. A detailed analysis of how batch inference would affect GNN—bringing its per-node overhead closer to that of BFS—is presented in the extended version of this paper [11, Appendix H].

Table 5 further compares GNN against individual heuristics within $\mathcal{H}$-EFP. GNN performs on par with the best individual heuristics, demonstrating its strength and viability as an alternative approach. At the same time, we view GNN-based heuristics as a complementary tool to enhance solver coverage, for example through integration into portfolio-based approaches, rather than as a replacement for existing heuristics.

Overall, GNN achieves consistent reductions in explored nodes and represents a scalable, learning-based step toward effective heuristics for multi-agent epistemic planning, a setting in which heuristic guidance is currently limited.

5 LIMITATIONS AND FUTURE DIRECTIONS

While our approach achieves promising experimental results, several limitations remain. First, we acknowledge that our current implementation does not yet achieve competitive runtime performance compared to existing heuristics methods. This limitation is largely due to engineering considerations. A proper integration of CUDA-based computation and batch processing, combined with a search strategy capable of exploiting these features while minimizing memory exchange, would, in fact, significantly improve inference speed as detailed in the extended version of this paper [11, Appendix H]. Although we recognize this shortcoming, we emphasize that this work is foundational. Addressing the engineering challenges required for an optimized implementation would constitute a substantial effort in its own right—worthy of dedicated study—and represents an important avenue for future research. Nonetheless, our primary goal here is to provide a proof of concept highlighting the potential of heuristics learning in MEP. This is also why we focus on the number of expanded nodes as our primary evaluation metric.

Certain domains, such as **AL** and **GR**, also present unique challenges. In **AL**, problem instances differ only in the nesting depth of belief formulas, which results in weak learning signals. Results for this domain are therefore not very informative as GNN and BFS perform almost identically. In **GR**, the sparsity of valid plans limits the effectiveness of data-driven learning, exposing a current limitation of our data generation pipeline as discussed above.

As mentioned, our current GNN implementation lacks batch inference during planning, which contributes to slower runtime. While enabling batch computation is primarily an engineering task, it also raises design questions regarding when and how to accumulate batches of states for scoring. For instance, the planner could rely on **BFS** or alternate with other heuristics until a sufficient number of candidate states are available for batched GNN evaluation. We leave the systematic study of these strategies for future work.

Finally, an important next step is to integrate our GNN-based heuristics estimates into more advanced and potentially more effective search frameworks, such as MCTS [50]. We believe this direction holds substantial promise for improving the scalability and adaptability of MEP solvers.

	GNN				BFS			
	Solved Inst.	Nodes	Time [ms]	Length	Solved Inst.	Nodes	Time [ms]	Length
AL	6/7 (85.71%)	10 ± 0	274 ± 3142	5 ± 0	6/7 (85.71%)	14 ± 0	82 ± 3851	5 ± 0
CC	18/18 (100.00%)	65 ± 250	996 ± 2006	7 ± 4	18/18 (100.00%)	610 ± 1607	1610 ± 6323	5 ± 3
CB	3/3 (100.00%)	75 ± 860	279 ± 3509	5 ± 2	3/3 (100.00%)	102 ± 1260	118 ± 1603	5 ± 2
GR	8/12 (66.67%)	157 ± 380	6512 ± 14016	6 ± 2	10/12 (83.33%)	448 ± 2068	5094 ± 12086	4 ± 2
SC	19/20 (95.00%)	49 ± 357	157 ± 511	10 ± 6	19/20 (95.00%)	114 ± 372	131 ± 572	8 ± 4
SR	5/6 (83.33%)	6188 ± 17295	38111 ± 106636	8 ± 10	5/6 (83.33%)	7918 ± 22608	42146 ± 120182	8 ± 10
All	59/66 (89.39%)	64 ± 296	1001 ± 4635	7 ± 4	61/66 (92.42%)	242 ± 1080	922 ± 4870	6 ± 4

Table 3: Comparison between GNN and BFS across standard benchmarks. All represents results over the entire Test set.

	Solved Inst.	Nodes	Time [ms]	Length
BFS	55/59 (93.22%)	389 ± 1410	1384 ± 6943	6 ± 4
CC-GR	55/59 (93.22%)	288 ± 1244	4116 ± 13644	6 ± 4

Table 4: Comparison between GNN equipped with CC-GR and BFS across standard benchmarks.

Approach	# Solved	% Solved
GNN	64/75	85.33%
C_PG	37/75	49.33%
L_PG	54/75	72.00%
S_PG	62/75	82.67%
SUB	58/75	77.33%

Table 5: CC-GR against $\mathcal{H}$-EFP's individual heuristics.

6 RELATED WORKS

Machine Learning in Planning. Traditionally, planning heuristics are either hand-crafted or derived from structural features of the search space [46, Ch. 11]. ML-based heuristics offer a scalable alternative, learning meaningful patterns from data [16, 17]. This is exemplified by systems like AlphaGo [48], where learned guidance enables scalable MCTS. Our work builds on these ideas but targets a more structured setting, where planning states are represented as Kripke structures. This introduces challenges, which we address using GNNs to extract semantic features from epistemic states. GNNs have also proven effective in classical planning, where they model relational graphs [49], learn numeric heuristics [8], or guide adaptive search [19].

Recent efforts have also explored using Large Language Models (LLMs) in planning. While LLMs are ineffective as standalone planners [34, 44], they can aid heuristics generation [18, 34] or domain formalization [51]. However, due to the structured nature of MEP, we believe GNNs are a more suitable choice. We leave the integration of LLMs in this context to future work.

Multi-Agent Epistemic Planning. Most work on MEP has focused on foundational problems such as the investigation of DEL fragments [15], the definition of action languages [3, 13, 41], and the development of underlying representations [14, 23]. While these are fundamental contributions, this paper pursues a different goal: enabling efficient exploration via informed search.

To the best of our knowledge, only one line of work addresses this challenge, namely Fabiano et al. [25], Le et al. [39], which derive heuristics using the planning graph structure. Our approach differs by employing data-driven methods. As mentioned, a future direction is to investigate the interplay between these two types of heuristics either in parallel or in conjunction.

Other recent efforts integrate ML and RL with epistemic planning but simplify epistemic state representations. Engesser et al. [22] decompose e-states into bounded feature vectors, bridging epistemic logic and reinforcement learning. Similarly, Nunn et al. [42] use generative models to reason over individual formulae rather than full Kripke structures, enabling different but interesting capabilities.

7 CONCLUSIONS

This work introduces a novel, learning-based approach to heuristics generation for multi-agent epistemic planning, leveraging Graph Neural Networks to guide informed search. By embedding Kripke structures and training a GNN to approximate the perfect heuristics, we enable scalable MEP planning through learning.

We investigated three distinct embeddings of Kripke structures and evaluated their effect on the heuristics accuracy. Through a comprehensive benchmarking, we identified the embedding configurations that most effectively balance expressiveness and scalability.

Our implementation, deep, demonstrates solid performance across standard benchmarks, reducing the number of explored nodes compared to uninformed search. The method also generalizes well to unseen domains and is competitive against existing heuristics.

These results highlight the potential of heuristics learning in MEP, where heuristics are scarce.

For these reasons, this work represents a foundational step toward exploiting machine learning in the context of multi-agent epistemic planning.

ACKNOWLEDGMENTS

This research was partially funded by the EPSRC grant EP/Y028872/1, *Mathematical Foundations of Intelligence: An "Erlangen Programme" for AI.*

REFERENCES

[1] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. 2021. Deep Reinforcement Learning at the Edge of the Statistical Precipice. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. Curran Associates Inc., Red Hook, NY, USA, 29304–29320. https://proceedings.neurips.cc/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html

[2] Chitta Baral, Thomas Bolander, Hans van Ditmarsch, and Sheila A. McIlraith. 2017. Epistemic Planning (Dagstuhl Seminar 17231). *Dagstuhl Reports* 7, 6 (2017), 1–47. https://doi.org/10.4230/DAGREP.7.6.1

[3] Chitta Baral, Gregory Gelfond, Enrico Pontelli, and Tran Cao Son. 2022. An action language for multi-agent domains. *Artif. Intell.* 302 (2022), 103601. https://doi.org/10.1016/J.ARTINT.2021.103601

[4] Piotr Bielak and Tomasz Jan Kajdanowicz. 2022. PyTorch-Geometric Edge - a Library for Learning Representations of Graph Edges. In *The First Learning on Graphs Conference*. OpenReview.net, Virtual. https://openreview.net/forum?id=hM5UIWqZ7d

[5] Thomas Bolander and Mikkel Birkegaard Andersen. 2011. Epistemic planning for single and multi-agent systems. *J. Appl. Non Class. Logics* 21, 1 (2011), 9–34. https://doi.org/10.3166/JANCL.21.9-34

[6] Thomas Bolander, Lasse Dissing, and Nicolai Herrmann. 2021. DEL-based Epistemic Planning for Human-Robot Collaboration: Theory and Implementation. In *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021*. IJCAI Organization, Online, 120–129. https://doi.org/10.24963/KR.2021/12

[7] Blai Bonet and Hector Geffner. 2001. Planning as heuristic search. *Artif. Intell.* 129, 1-2 (2001), 5–33. https://doi.org/10.1016/S0004-3702(01)00108-4

[8] Valerio Borelli, Alfonso Emilio Gerevini, Enrico Scala, and Ivan Serina. 2025. Learning Heuristic Functions with Graph Neural Networks for Numeric Planning. In *Proceedings of the International Symposium on Combinatorial Search*, Vol. 18. AAAI Press, Glasgow, United Kingdom, 251–252.

[9] Bruno Bouzy and Guillaume Chaslot. 2006. Monte-Carlo Go Reinforcement Learning Experiments. In *Proceedings of the 2006 IEEE Symposium on Computational Intelligence and Games (CIG06), University of Nevada, Reno, USA, campus in Reno/Lake Tahoe, 22-24 May, 2006*. IEEE, 187–194. https://doi.org/10.1109/CIG.2006.311699

[10] Ronen I. Brafman and Carmel Domshlak. 2013. On the complexity of planning for agent teams and its implications for single agent planning. *Artif. Intell.* 198 (2013), 52–71. https://doi.org/10.1016/J.ARTINT.2012.08.005

[11] Giovanni Briglia, Francesco Fabiano, and Stefano Mariani. 2025. Scaling Multi-Agent Epistemic Planning through GNN-Derived Heuristics. https://doi.org/10.48550/ARXIV.2508.12840 arXiv:2508.12840

[12] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Process. Mag.* 34, 4 (2017), 18–42. https://doi.org/10.1109/MSP.2017.2693418

[13] Alessandro Burigana and Francesco Fabiano. 2022. The Epistemic Planning Domain Definition Language (Short Paper). In *Proceedings of the 10th Italian workshop on Planning and Scheduling (IPS 2022), University of Udine, Udine, Italy (CEUR Workshop Proceedings, Vol. 3345)*. CEUR-WS.org. https://ceur-ws.org/Vol-3345/paper5_2497.pdf

[14] Alessandro Burigana, Paolo Felli, and Marco Montali. 2023. delphic: Practical DEL Planning via Possibilities. In *JELIA 2023, Dresden, Germany, September 20-22, 2023, Proceedings (Lecture Notes in Computer Science, Vol. 14281)*. Springer, Dresden, Germany, 579–594. https://doi.org/10.1007/978-3-031-43619-2_39

[15] Alessandro Burigana, Paolo Felli, Marco Montali, and Nicolas Troquard. 2023. A Semantic Approach to Decidability in Epistemic Planning. In *Proceedings of AAMAS, London, United Kingdom, 29 May 2023 - 2 June 2023*. ACM, 2361–2363. https://doi.org/10.5555/3545946.3598934

[16] Sergio Jiménez Celorrio, Tomás de la Rosa, Susana Fernández, Fernando Fernández, and Daniel Borrajo. 2012. A review of machine learning for automated planning. *Knowl. Eng. Rev.* 27, 4 (2012), 433–467. https://doi.org/10.1017/S026988891200001X

[17] Dillon Z. Chen, Felipe W. Trevizan, and Sylvie Thiébaux. 2024. Return to Tradition: Learning Reliable Heuristics with Classical Machine Learning. In *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2024, Banff, Alberta, Canada, June 1-6, 2024*. AAAI Press, 68–76. https://doi.org/10.1609/ICAPS.V34I1.31462

[18] Augusto B. Corrêa, André Grahl Pereira, and Jendrik Seipp. 2025. Classical Planning with LLM-Generated Heuristics: Challenging the State of the Art with Python Code. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=UCV21BsuqA

[19] Qiwei Du, Bowen Li, Yi Du, Shaoshu Su, Taimeng Fu, Zitong Zhan, Zhipeng Zhao, and Chen Wang. 2025. Fast Task Planning with Neuro-Symbolic Relaxation. https://doi.org/10.48550/ARXIV.2507.15975 arXiv:2507.15975

[20] John Ellson, Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Gordon Woodhull. 2004. Graphviz and Dynagraph - Static and Dynamic Graph Drawing Tools. In *Graph Drawing Software*. Springer, 127–148. https://doi.org/10.1007/978-3-642-18638-7_6

[21] Thorsten Engesser, Thomas Bolander, Robert Mattmüller, and Bernhard Nebel. 2017. Cooperative Epistemic Multi-Agent Planning for Implicit Coordination. In *Proceedings of the Ninth Workshop on Methods for Modalities, M4M@ICLA 2017, Indian Institute of Technology, Kanpur, India, 8th to 10th January 2017 (EPTCS, Vol. 243)*. 75–90. https://doi.org/10.4204/EPTCS.243.6

[22] Thorsten Engesser, Thibaut Le Marre, Emiliano Lorini, François Schwarzentruber, and Bruno Zanuttini. 2025. A Simple Integration of Epistemic Logic and Reinforcement Learning. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2025, Detroit, MI, USA, May 19-23, 2025*. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 686–694. https://doi.org/10.5555/3709347.3743585

[23] Francesco Fabiano, Alessandro Burigana, Agostino Dovier, and Enrico Pontelli. 2020. EFP 2.0: A Multi-Agent Epistemic Solver with Multiple E-State Representations. In *Proceedings of ICAPS, Nancy, France, October 26-30, 2020*. AAAI Press, 101–109. https://aaai.org/ojs/index.php/ICAPS/article/view/6650

[24] Francesco Fabiano, Marianna B. Ganapini, Andrea Loreggia, Nicholas Mattei, Keerthiram Murugesan, Vishal Pallagani, Francesca Rossi, Biplav Srivastava, and K. Brent Venable. 2025. Thinking Fast and Slow in Human and Machine Intelligence. *Commun. ACM* 68, 8 (July 2025), 72–79. https://doi.org/10.1145/3715709

[25] Francesco Fabiano, Theoderic Platt, Tran Cao Son, and Enrico Pontelli. 2024. $\mathcal{H}$-EFP: Bridging Efficiency in Multi-agent Epistemic Planning with Heuristics. In *PRIMA 2024, Kyoto, Japan, November 18-24, 2024, Proceedings (Lecture Notes in Computer Science, Vol. 15395)*. Springer, 81–86. https://doi.org/10.1007/978-3-031-77367-9_7

[26] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Vardi. 1995. *Reasoning About Knowledge*. The MIT Press. https://doi.org/10.7551/mitpress/5803.001.0001

[27] Daniel Foead, Alifio Ghifari, Marchel Budi Kusuma, Novita Hanafiah, and Eric Gunawan. 2021. A Systematic Literature Review of A* Pathfinding. *Procedia Computer Science* 179 (2021), 507–514. https://doi.org/10.1016/j.procs.2021.01.034 5th International Conference on Computer Science and Computational Intelligence 2020.

[28] Alfonso Gerevini, Alessandro Saetti, and Ivan Serina. 2003. Planning Through Stochastic Local Search and Temporal Action Graphs in LPG. *J. Artif. Intell. Res.* 20 (2003), 239–290. https://doi.org/10.1613/JAIR.1183

[29] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.* 4, 2 (1968), 100–107. https://doi.org/10.1109/TSSC.1968.300136

[30] Malte Helmert. 2006. The Fast Downward Planning System. *J. Artif. Intell. Res.* 26 (2006), 191–246. https://doi.org/10.1613/JAIR.1705

[31] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. 2020. Strategies for Pre-training Graph Neural Networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, Addis Ababa, Ethiopia. https://openreview.net/forum?id=HJlWWJSFDH

[32] Sukai Huang, Nir Lipovetzky, and Trevor Cohn. 2025. Planning in the dark: LLM-symbolic planning pipeline without experts. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence (AAAI'25/IAAI'25/EAAI'25)*. AAAI Press, Article 2957, 9 pages. https://doi.org/10.1609/aaai.v39i25.34855

[33] Sagar Imambi, Kolla Bhanu Prakash, and GR Kanagachidambaresan. 2021. PyTorch. In *Programming with TensorFlow: solution for edge computing applications*. Springer, 87–104.

[34] Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. 2024. Position: LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net. https://openreview.net/forum?id=Th8JPEmH4z

[35] Emil Keyder and Hector Geffner. 2008. Heuristics for Planning with Action Costs Revisited. In *ECAI 2008 - 18th European Conference on Artificial Intelligence, Patras, Greece, July 21-25, 2008, Proceedings (Frontiers in Artificial Intelligence and Applications, Vol. 178)*. IOS Press, 588–592. https://doi.org/10.3233/978-1-58603-891-5-588

[36] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=SJU4ayYgl

[37] Filippos Kominis and Hector Geffner. 2017. Multiagent Online Planning with Nested Beliefs and Dialogue. In *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling, ICAPS 2017, Pittsburgh, Pennsylvania, USA, June 18-23, 2017*. AAAI Press, 186–194. https://aaai.org/ocs/index.php/ICAPS/ICAPS17/paper/view/15748

[38] Saul A. Kripke. 1963. Semantical Analysis of Modal Logic I Normal Modal Propositional Calculi. *Mathematical Logic Quarterly* 9, 5-6 (1963), 67–96. https://doi.org/10.1002/malq.19630090502

arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/malq.19630090502

[39] Tiep Le, Francesco Fabiano, Tran Cao Son, and Enrico Pontelli. 2018. EFP and PG-EFP: Epistemic Forward Search Planners in Multi-Agent Domains. In *Proceedings ICAPS*. AAAI Press, Delft, The Netherlands, 161–170. https://aaai.org/ocs/index.php/ICAPS/ICAPS18/paper/view/17733

[40] Lawrence S. Moss. 2015. Dynamic Epistemic Logic. In *Handbook of Epistemic Logic*. College Publications, Chapter 6, 262–312.

[41] Christian J. Muise, Vaishak Belle, Paolo Felli, Sheila A. McIlraith, Tim Miller, Adrian R. Pearce, and Liz Sonenberg. 2015. Planning Over Multi-Agent Epistemic States: A Classical Planning Approach. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. AAAI Press, 3327–3334. https://doi.org/10.1609/AAAI.V29I1.9665

[42] Pierre Nunn, Marco Sälzer, François Schwarzentruber, and Nicolas Troquard. 2024. A Logic for Reasoning about Aggregate-Combine Graph Neural Networks. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*. ijcai.org, 3532–3540. https://www.ijcai.org/proceedings/2024/391

[43] Vishal Pallagani, Bharath Muppasani, Biplav Srivastava, Francesca Rossi, Lior Horesh, Keerthiram Murugesan, Andrea Loreggia, Francesco Fabiano, Rony Joseph, and Yathin Kethepalli. 2023. Plansformer Tool: Demonstrating Generation of Symbolic Plans Using Transformers. In *Proceedings of IJCAI*. 7158–7162. https://doi.org/10.24963/ijcai.2023/839 Demo Track.

[44] Vishal Pallagani, Bharath C. Muppasani, Kaushik Roy, Francesco Fabiano, Andrea Loreggia, Keerthiram Murugesan, Biplav Srivastava, Francesca Rossi, Lior Horesh, and Amit P. Sheth. 2024. On the Prospects of Incorporating Large Language Models (LLMs) in Automated Planning and Scheduling (APS). In *Proceedings of ICAPS 2024, Banff, Alberta, Canada, June 1-6, 2024*. AAAI Press, 432–444. https://doi.org/10.1609/ICAPS.V34I1.31503

[45] Loc Pham, Tran Cao Son, and Enrico Pontelli. 2023. Planning in Multi-Agent Domains with Untruthful Announcements. *Proceedings of ICAPS* 33, 1 (Jul. 2023), 334–342. https://doi.org/10.1609/icaps.v33i1.27211

[46] Stuart Russell and Peter Norvig. 2020. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson. http://aima.cs.berkeley.edu/

[47] Boris Schling. 2011. *The Boost C++ Libraries*. XML Press.

[48] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nat.* 529, 7587 (2016), 484–489. https://doi.org/10.1038/NATURE16961

[49] Tom Silver, Rohan Chitnis, Aidan Curtis, Joshua B. Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. 2021. Planning with Learned Object Importance in Large Problem Instances using Graph Neural Networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 11962–11971. https://doi.org/10.1609/AAAI.V35I13.17421

[50] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement learning - an introduction, 2nd Edition*. MIT Press.

[51] Marcus Tantakoun, Christian Muise, and Xiaodan Zhu. 2025. LLMs as Planning Formalizers: A Survey for Leveraging Large Language Models to Construct Automated Planning Models. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025 (Findings of ACL, Vol. ACL 2025)*. Association for Computational Linguistics, 25167–25188. https://aclanthology.org/2025.findings-acl.1291/

[52] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. https://openreview.net/forum?id=rJXMpikCZ

[53] Hai Wan, Biqing Fang, and Yongmei Liu. 2021. A general multi-agent epistemic planner based on higher-order belief change. *Artif. Intell.* 301 (2021), 103562. https://doi.org/10.1016/J.ARTINT.2021.103562

[54] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Neural Networks Learn. Syst.* 32, 1 (2021), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386

[55] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81. https://doi.org/10.1016/J.AIOPEN.2021.01.001