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

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 [11]. 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* [27].

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, 22, 24, 39, 43, 47, 56]. To the best of our knowledge, only a few systems [14, 24, 47] 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 [26], an extension of Le et al. [41], 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 [8, 31] to *Monte Carlo Tree Search* (MCTS) in *reinforcement learning* (RL) [10].

The key difference in our approach lies in how heuristics are defined and computed. Unlike Fabiano et al. [26], Le et al. [41], 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 [8], 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 [24, 26].

The key contributions of this work are as follows:

(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.

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

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

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 [42].

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

- a fluent formula;
- if φ is a belief formula and $\mathsf{i} \in \mathcal{AG}$, then $\mathbf{B}_\mathsf{i}(\varphi)$ is a belief formula;
- 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
- 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* [40].

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, 24] 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. [27] 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 widely 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, 43], 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 "`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

[2] We use the terms knowledge and belief interchangeably, as their distinction is beyond this work's scope. See Fagin et al. [27] for a full discussion.

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

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 [32]: 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 [37] layers update the node features while *ignoring* the edge attributes, and the GAT [55] 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. [57] and Zhou et al. [58], 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 [48, 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 [36, 48]. For this reason, a key contribution of this work is the formalization of heuristics tailored for MEP through the employment of GNNs.

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 [33, 45]. However, these methods often suffer from low accuracy and require a large number of training instances to generalize effectively [35].

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 address is 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, since this assignment depends on the order in which worlds are generated, ϕ is 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.

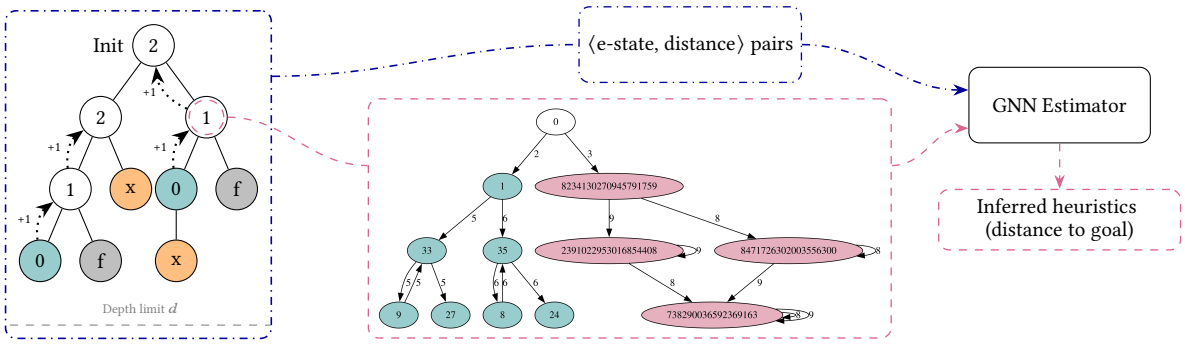


Figure 1: Illustration of the overall training and inference process. On the left, we show dataset generation via DFS: teal nodes represent goals (score 0), black *dotted* arrows show backtracking assigning distances, orange nodes ('x') indicate discarded branches, and gray nodes ('f') are states with 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.

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 [49] 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 $\mathsf{f}_j \in \mathcal{F}$ 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 [21]—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 Appendix C for the description of the procedure.

In the resulting formulation, two types of nodes are encoded for each e-state embedding:

- *state nodes*, corresponding to worlds in the epistemic state;
- *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* [29], 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 [26]—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. [25]. 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 $\mathsf{p_M}$ 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 $d \in [0, \mathsf{D_{max}}]$ from $\mathsf{min\_val} \in [0, 1)$ to $\mathsf{max\_val} \in (0, 1]$, with $\mathsf{max\_val} \geq \mathsf{min\_val}$. Let $\alpha = \frac{\mathsf{max\_val} - \mathsf{min\_val}}{\mathsf{D_{max}}}$, and $\beta = \mathsf{min\_val}$, then the normalized target is $\tilde{d} = \alpha\, d + \beta \in [0, 1]$. In inference, we recover the original scale through $d = \frac{\tilde{d} - \beta}{\alpha} \in [0, \mathsf{D_{max}}]$

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` [34], 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:

- *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 $X \in \{0, 1\}^{|V| \times d_b}$;
- *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;
- *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 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,
$$x_i = f_{\mathrm{id}}(b_i) = \mathrm{MLP}_{\mathrm{id}}(b_i) \in \mathbb{R}^{d_v},$$
producing dense embeddings of dimension $d_v = \mathsf{node\_emb\_dim}$. Similarly, each edge attribute a_k is processed through
$$e_k = f_{\mathrm{edge}}(a_k) = \mathrm{MLP}_{\mathrm{edge}}(a_k) \in \mathbb{R}^{d_e},$$
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 [32] 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 Appendix E.

4 EXPERIMENTS AND EVALUATION

To conduct our experiments, we developed deep, a modernized reimplementation of the epistemic planner EFP [24, 26]. 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. [24]. For completeness, we also provide comparisons with the different heuristics implemented in $\mathcal{H}$-EFP [26]. 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, 24, 43], 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 Appendix B and D, respectively.

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 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] [28, 30], 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 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.

As mentioned, for the sake of space, we presented only aggregate results. We note, however, that the trends shown in Tables 1 and 2 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]Since the GNN-based heuristics is statistical in nature, its admissibility cannot be guaranteed; hence we do not refer to it as **A***.

	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*.

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**.

Table 5 compares 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) and their description is provided in 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 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 its efficiency gains.

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 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 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 viability as an alternative approach. At the same time, we view GNN-based heuristics as a complementary tool to enhance solver coverage, *e.g.*, 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 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 heuristic estimates into more advanced search frameworks, such as Monte Carlo Tree Search [53]. We believe this approach has strong potential to improve 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 [48, Ch. 11]. ML-based heuristics offer a scalable alternative, learning meaningful patterns from data [16, 17]. This is exemplified by systems like AlphaGo [50], 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 [51], learn numeric heuristics [9], or guide adaptive search [20].

Recent efforts have also explored using Large Language Models (LLMs) in planning. While LLMs are ineffective as standalone planners [35, 46], they can aid heuristics generation [19, 35] or domain formalization [54]. 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, 43], and the development of underlying representations [14, 24]. 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. [26], Le et al. [41], 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. [23] decompose e-states into bounded feature vectors, bridging epistemic logic and reinforcement learning. Similarly, Nunn et al. [44] 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] Thomas Bolander, Martin Holm Jensen, and Francois Schwarzentruber. 2015. Complexity results in epistemic planning. In *Proceedings of the 24th International Conference on Artificial Intelligence* (Buenos Aires, Argentina) *(IJCAI'15)*. AAAI Press, 2791–2797.

[8] 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

[9] 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.

[10] 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

[11] 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

[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] Martin C. Cooper, Andreas Herzig, Faustine Maffre, Frédéric Maris, and Pierre Régnier. 2019. The epistemic gossip problem. *Discret. Math.* 342, 3 (2019), 654–663. https://doi.org/10.1016/J.DISC.2018.10.041

[19] 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

[20] 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

[21] 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

[22] 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

[23] 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

[24] 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

[25] 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

[26] 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

[27] 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

[28] 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.

[29] 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

[30] 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

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

[32] 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

[33] 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

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

[35] 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

[36] 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

[37] 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

[38] Filippos Kominis and Hector Geffner. 2015. Beliefs In Multiagent Planning: From One Agent to Many. In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, ICAPS 2015, Jerusalem, Israel, June 7-11,*

2015. AAAI Press, 147–155. http://www.aaai.org/ocs/index.php/ICAPS/ICAPS15/paper/view/10617

[39] 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

[40] 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

[41] 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

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

[43] 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

[44] 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

[45] 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.

[46] 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

[47] 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

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

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

[50] 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

[51] 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

[52] Tran Cao Son, Enrico Pontelli, Chitta Baral, and Gregory Gelfond. 2014. Finitary S5-Theories. In *Logics in Artificial Intelligence - 14th European Conference, JELIA 2014, Funchal, Madeira, Portugal, September 24-26, 2014. Proceedings (Lecture Notes in Computer Science, Vol. 8761)*. Springer, 239–252. https://doi.org/10.1007/978-3-319-11558-0_17

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

[54] 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/

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

[56] 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

[57] 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

[58] 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

Technical Appendix

A COMPUTATIONAL RESOURCE USED

All experiments were performed on a 13^{th} Intel(R) Core(TM) i9-13900H with 20 CPU cores and 64 GB of system RAM, alongside an NVIDIA RTX 4070 GPU with 8 GB of dedicated VRAM. Note that the experiments do not need multiple repetitons as the inference and the planning process does not depend on randomness.

Overall, for the final results, sample generation, model training and inference across all experimental batches required approximately 180 hours of high-performance computing (HPC) time. Since the project began, however, simulation trials have consumed nearly 400 hours.

Illustrative timing breakdown for a single model:

- *Sample generation:* Approximately 15 minutes per experimental setup.
- *Data-loader construction:* 15 minutes for fewer than 10k samples; up to 2 hours for more than 100k samples.
- *Model training:* 10 minutes for very small data-loaders; up to 3 hours for very large ones.

B PARAMETERS AND HYPERPARAMETERS

In this section, we summarize all model and optimizer settings used in our experiments. Table 6 lists the architectural parameters of the GNN-based-Regressor; while Table 7 gives the key AdamW optimizer hyperparameters.

Parameter Description	Value
Size of hidden feature dimension in GINEConvs and regressor input.	128
Dimension of initial node embeddings (ID MLP output).	64
Dimension of initial edge embeddings (edge MLP output).	32
Hidden size in the residual regressor head.	128
Number of ResidualBlock layers in the regressor.	3
Dropout probability inside each ResidualBlock.	0.2
Lower clamp bound on final sigmoid output.	1e-3
Upper clamp bound on final sigmoid output.	1 - 1e-3

Table 6: Values and descriptions of the GNN-based-Regressor **parameters.**

Hyperparameter Description	Value
Learning rate for parameter updates.	10^{-3}
Exponential decay rates for the first and second moment estimates.	(0.9, 0.999)
Term added to the denominator to improve numerical stability.	10^{-8}
Coefficient for decoupled weight decay (L2 regularization).	10^{-2}
Flag to enable the AMSGrad variant of AdamW.	False

Table 7: Key hyperparameters of the AdamW optimizer.

B.1 Case studies

Here, an overview of the experimental configuration parameters is provided.

Parameter	Description	Value
D_{max}	Maximum goal distance considered	50
d_b	Length of the input bitmask vector	64
min_val	Lower bound for linear normalization	10^{-3}
max_val	Upper bound for linear normalization	$1 - 10^{-3}$
p_M	Maximum class imbalance threshold in the target distribution	0.5

Table 8: Experiment configuration parameters: overview of each parameter and its respective value.

C GOAL ENCODING INTO E-STATE EMBEDDING

We first note that generating a Kripke structure corresponding to the goal state is not a viable solution. As discussed by Son et al. [52], there exist infinitely many variations of epistemic states that satisfy a given arbitrary belief formula—*i.e.*, a goal description—making it impossible to generate all of them. Generating only one or a few such states would bias the training process toward those specific representations, favoring e-states that structurally resemble the generated ones. This is problematic, as the structure of valid e-states can vary significantly.

Moreover, as shown in Bolander et al. [7], the computational complexity of generating such structures is exponential in complexity with respect to the size of the formula.

To address this, we developed a custom syntax tree that parses the problem's goal into an ad-hoc graph-based representation. This representation highlights key components such as belief operators and logical connectives, while maintaining consistent naming conventions with the state representation. Specifically, the only shared identifiers between the goal and the e-state are those that denote the same underlying entity—in our case, the agent IDs.

The procedure is presented in Algorithm 1

Algorithm 1 Recursive Goal Formula Graph Construction

Input: Belief formula φ, goal ID g, node counter `next_id`, parent node p
Output: The graph is printed recursively on the output file

function GenerateGoalEmbedding(φ, g, `next_id`, p)
  `id` ← `increment(next_id)`
  `n` ← `string(id)`
  if φ is a fluent formula **then**
    for all subformula `S` in φ **do** ▷ Handles logic OR
      if `S` has multiple fluents **then** ▷ Handles logic AND
        `n` ← `new_node_id`
        connect `p` → `n` with label g
        `p` ← `n`
      for all fluent `f` in `S` **do**
        connect `p` → `get_f_id(f)` with label g ▷ Retrieves unique fluent ID
  else if $\varphi = \mathbf{B}_a(\psi)$ **then**
    connect `p` → `n` with label g
    connect `n` ↔ `get_a_id(a)` with label g ▷ Retrieves unique agent ID
    GenerateGoalEmbedding(ψ, g, `next_id`, `n`)
  else if $\varphi = \mathbf{C}_G(\psi)$ **then**
    connect `p` → `n` with label g
    for all agent `a` in group `G` **do**
      connect `n` ↔ `get_a_id(a)` with label g
    GenerateGoalEmbedding(ψ, g, `next_id`, `n`)
  else if φ is a propositional formula **then**
    connect `p` → `n` with label g
    GenerateGoalEmbedding(ψ_1, g, `next_id`, `n`)
    if ψ_2 exists **then**
      GenerateGoalEmbedding(ψ_2, g, `next_id`, `n`)

D DOMAINS

Here we present the complete description of the benchmarks used to evaluate our contributions. These have been collected from the literature [18, 24, 38, 56].

- *Assembly Line* (**AL**). This domain involves two agents, each responsible for processing a separate part of a product. Each processing step may fail, and agents can inform one another of their task's outcome. Based on this shared knowledge, the agents decide whether to *assemble* the product or *restart*. The goal is fixed—assembling the product—but the complexity varies depending on the *depth* of the belief formulas used in the executability conditions.
- *Collaboration and Communication* (**CC**). In this domain, $n \geq 2$ agents move along a corridor with $k \geq 2$ rooms, in which $m \geq 1$ boxes can be located. Whenever an agent enters a room, she can observe whether a specific box is present. Additionally, agents can communicate information about the boxes' positions to other *attentive* agents. The goals involve both agents' physical positions and their beliefs about the boxes.
- *Coin in the Box* (**CB**). Here, $n \geq 3$ agents are in a room with a locked box containing a coin, which lies either heads or tails up. Initially, no agent knows the coin's orientation. One agent holds the key to open the box. Typical goals involve some agents learning the coin's status, while others may need to know that someone else knows it—or remain ignorant of this fact entirely.
- *Grapevine* (**GR**). In this setting, $n \geq 2$ agents are distributed across $k \geq 2$ rooms. Agents can freely move between rooms and share "secrets" with any other agents present in the same room. This domain supports diverse goal types, ranging from secret sharing to creating misconceptions about others' beliefs.
- *Selective Communication* (**SC**). This domain features $n \geq 2$ agents, each starting in one of the $k \geq 2$ rooms arranged along a corridor. An agent may broadcast information, which is heard by all agents in the same or adjacent rooms. Each agent can move between neighboring rooms. The goals often require certain agents to know specific facts while ensuring that others remain unaware of them.
- *Selective Communication Enriched* (**SCR**). This domain is a replica of **SC** but enriched with extra, useless actions, to decrease the density of the solutions in the search space.
- *Epistemic Gossip* (**EG** – Only used in experiments presented in the Appendix). This domain extends the classic gossip problem, where information ("secrets") must be disseminated among $n \geq 2$ agents using the minimum number of calls. Unlike the traditional formulation, which requires some agents to know some secrets (epistemic depth 1), we consider arbitrary epistemic depths. For example, goals may require that all agents know that all agents know all secrets (depth 2), and so on.

E FORWARD PASS NEURAL DISTANCE REGRESSOR

Algorithm 2 Batch-wise Distance Estimation (bit-vector IDs, no goal, no depth)

Input: `BatchDict` with entry:

- `state_graph` = (B, $\mathtt{E_i}$, $\mathtt{E_a}$, batch)
  - $\mathtt{B} \in \{0,1\}^{N \times d_b}$: node bit-vectors ($d_b$ bits per node, typically $d_b = 48$);
  - $\mathtt{E_i} \in \mathbb{N}^{2 \times |E|}$: edge indices in **COO format** (Coordinate format), where each column (u, v) identifies a directed edge $u \to v$;
  - $\mathtt{E_a} \in \mathbb{R}^{|E| \times 1}$: edge attributes (scalar per edge);
  - $\mathtt{batch} \in \mathbb{N}^N$: batch assignment of each node to its graph.

Output: $\mathbf{d} \in \mathbb{R}^b$, estimated distances per batch (b = number of graphs)

function REGRESSOR(r)
  $\mathtt{h}^{(0)} \leftarrow \mathrm{ReLU}\big(W^{(1)}\mathtt{r} + b^{(1)}\big)$
  for $j = 1$ to BL **do** ▷ BL = number of residual blocks
    $\mathtt{u} \leftarrow W^{(j,1)}\mathtt{h}^{(\mathtt{j}-1)} + b^{(j,1)}$
    $\mathtt{u} \leftarrow \mathrm{BN}_{j,1}(\mathtt{u})$
    $\mathtt{u} \leftarrow \mathrm{ReLU}(\mathtt{u})$
    $\mathtt{u} \leftarrow \mathrm{Dropout}(\mathtt{u})$
    $\mathtt{v} \leftarrow W^{(j,2)}\mathtt{u} + b^{(j,2)}$
    $\mathtt{v} \leftarrow \mathrm{BN}_{j,2}(\mathtt{v})$
    $\mathtt{h}^{(\mathtt{j})} \leftarrow \mathrm{ReLU}\big(\mathtt{v} + \mathtt{h}^{(\mathtt{j}-1)}\big)$ ▷ residual skip connection
  $\mathtt{z} \leftarrow W^{(\mathrm{out})}\mathtt{h}^{(\mathtt{BL})} + b^{(\mathrm{out})}$
  return $\mathrm{Clamp}\big(\mathrm{Sigmoid}(\mathtt{z}),\ \mathrm{min} = \mathtt{min\_val},\ \mathrm{max} = 1 - \mathtt{min\_val}\big)$
function ENCODE(G, conv1, conv2)
  $(\mathtt{B}, \mathtt{E_i}, \mathtt{E_a}, \mathtt{batch}) \leftarrow \mathtt{G}$
  $\tilde{\mathtt{X}} \leftarrow \mathtt{MLP_{id}}(\mathtt{B})$ ▷ $\tilde{\mathtt{X}} \in \mathbb{R}^{N \times d_v}$
  $\mathtt{E_{emb}} \leftarrow \mathtt{MLP_{edge}}(\mathtt{E_a})$ ▷ $\mathtt{E_{emb}} \in \mathbb{R}^{|E| \times d_e}$
  $\mathtt{H} \leftarrow \mathrm{ReLU}\big(\mathtt{conv1}(\tilde{\mathtt{X}}, \mathtt{E_i}, \mathtt{E_{emb}})\big)$
  $\mathtt{H} \leftarrow \mathrm{ReLU}\big(\mathtt{conv2}(\mathtt{H}, \mathtt{E_i}, \mathtt{E_{emb}})\big)$
  return $\mathtt{GlobalMeanPool}(\mathtt{H}, \mathtt{batch})$ ▷ $\in \mathbb{R}^{b \times d_h}$
function FORWARDPASS(BatchDict)
  $\mathtt{G_s} \leftarrow \mathtt{BatchDict.get("state\_graph")}$
  $\mathtt{s} \leftarrow \mathrm{ENCODE}(\mathtt{G_s},\ \mathtt{s\_conv1},\ \mathtt{s\_conv2})$
  $\mathtt{r} \leftarrow \mathtt{s}$ ▷ no goal, no depth concatenation
  $\mathtt{distance} \leftarrow \mathrm{REGRESSOR}(\mathtt{r})$
  return distance

Notation: d_b = bit length of node identifiers, d_v = node embedding dimension, d_e = edge embedding dimension, d_h = hidden dimension in GINE layers, and b = number of graphs in the batch.

F E-STATE REPRESENTATION AND SEARCH STRATEGY

In what follows, we present the results for all the experiments of the e-state representation and search strategy study.

We will use the following abbreviations:

- `mask`: e-state representation that uses symbolic ID-based embedding.
- `hash`: e-state representation that uses hashing-based embedding.
- `map`: e-state representation that uses bitmask-based embedding.
- **HFS**: **Best-First Search**, which we abbreviate as **HFS** (for Heuristics-First Search) to distinguish it from **BFS**
- **HFS***.: variant of **HFS** that augments the heuristics value $h(s)$ predicted by the GNN with the search depth $d(s)$ to form the total score: $f(s) = h(s) + d(s)$
- Nodes: the number of nodes expanded during search, reflecting the informativeness of the heuristics.
- Length: the length of the plan found.
- Time: the solving time (in milliseconds).
- IQM: Interquartile Mean, used as an aggregate performance metric.
- IQR-std: Interquartile Range, reported as a measure of variability.
- avg: arithmetic mean, reported as a baseline aggregate metric.
- std: standard deviation, used to quantify variability in the data.
- all: indicates that the aggregate value is computed over all instances solved by that approach.
- comm: indicates that the aggregate value is computed only over instances solved by all approaches in the comparison.
- TO: indicates a timeout (after 600 seconds).
- −: indicates a missing value due to the problem not being solved within the timeout.

F.1 e-State Representations Comparison

Instance Name	mask			hash			map		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
Assemble_B3-pl_5	5	14	164	5	14	154	5	14	308
Assemble_B5-pl_5	5	14	223	5	14	371	5	14	475
Assemble_B7-pl_5	5	14	4368	5	14	4790	5	14	9894
CC_2_2_3-pl_4	4	13	183	5	9	141	4	14	171
CC_2_2_3-pl_6	6	62	424	6	10	106	6	178	1532
CC_2_2_4-pl_5	7	43	938	9	77	2507	5	94	3483
CC_2_3_4-pl_3	5	6	2517	3	3	1381	3	19	6067
CC_2_3_4-pl_7	11	2054	88913	9	22	3244	7	4153	585278
CC_3_2_3-pl_4	5	13	219	5	7	160	4	47	689
CC_3_2_3-pl_5	6	51	888	6	45	463	5	136	2088
CC_3_3_3-pl_4	6	19	1730	4	4	495	4	39	1337
CoinBox-pl_3	3	17	220	3	17	129	3	17	228
CoinBox-pl_6	6	381	3534	6	381	2748	6	381	3760
SC_4_1-pl_3	3	4	51	3	4	52	3	4	96
SC_4_2-pl_7	7	120	609	7	120	545	7	97	533
SC_4_3-pl_5	5	17	116	5	17	62	5	17	98
SC_8_10-pl_6	6	49	315	6	49	343	6	113	882
SC_9_11-pl_6	6	26	179	6	26	161	6	26	198
SC_9_11-pl_8	8	66	388	8	66	271	8	66	500
SC_10_8-pl_10	10	10	174	10	10	97	10	10	108
SC_10_8-pl_15	15	130	824	15	130	543	15	130	822
SC_10_10-pl_6	6	309	8676	6	309	5998	6	298	8135
SC_10_10-pl_7	7	647	17999	7	647	11495	7	589	16064
SC_10_10-pl_10	10	3096	80030	10	3096	58382	10	1977	51904
SC_10_10-pl_13	13	51	431	13	51	360	13	51	358
avg ± std (all)	7 ± 3	289 ± 705	8565 ± 22733	7 ± 3	206 ± 608	3800 ± 11431	6 ± 3	340 ± 871	27800 ± 114266
IQM ± IQR-std (all)	6 ± 2	40 ± 106	748 ± 2298	6 ± 3	31 ± 67	596 ± 2353	6 ± 2	66 ± 119	1290 ± 3452
avg ± std (comm)	7 ± 3	289 ± 705	8565 ± 22733	7 ± 3	206 ± 608	3800 ± 11431	6 ± 3	340 ± 871	27800 ± 114266
IQM ± IQR-std (comm)	6 ± 2	40 ± 106	748 ± 2298	6 ± 3	31 ± 67	596 ± 2353	6 ± 2	66 ± 119	1290 ± 3452
Solved Instances	25/25 (100.00%)			25/25 (100.00%)			25/25 (100.00%)		

Table 9: Comparison of execution on the ablation study over the Train instances. The model used has been trained using the instances reported in this table.

Instance Name	mask			hash			map		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
Assemble_B10-pl_5	–	–	TO	–	–	TO	–	–	TO
Assemble_B2-pl_5	5	14	264	5	14	264	5	14	285
Assemble_B4-pl_5	5	14	275	5	14	332	5	14	330
Assemble_B6-pl_5	5	14	804	5	14	826	5	14	1127
Assemble_B8-pl_5	5	14	24655	5	14	25636	5	14	48340
Assemble_B9-pl_5	5	14	295842	5	14	296625	–	–	TO
Assemble_C-pl_5	5	14	124	5	14	237	5	14	238
CC_2_2_3-pl_3	3	9	111	4	11	108	3	9	181
CC_2_2_3-pl_5	5	17	272	5	9	134	5	77	684
CC_2_2_3-pl_7	7	163	2390	8	33837	214811	7	558	4358
CC_2_2_3-pl_8	8	1327	18235	8	247	2434	8	2539	27312
CC_2_2_4-pl_3	3	3	217	3	3	279	3	11	373
CC_2_2_4-pl_4	6	7	324	6	9	212	4	23	778
CC_2_2_4-pl_6	6	21	1251	7	26	296	6	363	9536
CC_2_2_4-pl_7	8	129	3989	8	498	4701	7	2084	45668
CC_2_3_4-pl_4	5	5	1995	6	10	1655	4	42	12130
CC_2_3_4-pl_5	10	132	20181	12	61	5848	5	127	45028
CC_2_3_4-pl_6	11	1269	65775	8	11	3383	6	675	124726
CC_3_2_3-pl_3	3	4	228	3	3	196	3	14	318
CC_3_2_3-pl_6	6	74	1201	6	113	1092	6	367	6443
CC_3_2_3-pl_7	8	185	3058	7	1835	24464	7	1548	26110
CC_3_3_3-pl_3	3	3	218	3	4	367	3	15	790
CC_3_3_3-pl_5	6	26	1975	7	22	534	5	253	8316
CC_3_3_3-pl_6	8	226	13395	–	–	TO	6	1942	73680
CC_3_3_3-pl_7	8	1476	59481	8	79	1255	7	11726	405444
CoinBox-pl_2	2	2	52	2	2	50	2	2	109
CoinBox-pl_5	5	77	425	5	77	552	5	78	728
CoinBox-pl_7	7	1817	10395	7	1817	13110	7	1793	17758
SC_4_1-pl_5	5	17	125	5	17	99	5	17	93
SC_4_2-pl_5	5	15	132	5	15	99	5	30	184
SC_4_2-pl_8	8	339	1639	8	339	1316	8	184	923
SC_4_3-pl_6	6	21	162	6	21	113	6	21	93
SC_4_3-pl_8	8	59	186	8	59	156	8	59	177
SC_4_4-pl_5	5	17	125	5	17	87	5	17	91
SC_8_10-pl_8	8	542	5157	8	542	3841	8	485	4121
SC_8_10-pl_9	9	1660	17268	9	1660	14557	9	1772	16936
SC_8_10-pl_12	12	31662	199243	12	31662	203157	12	35063	281914
SC_9_11-pl_4	4	8	143	4	8	141	4	8	139
SC_9_11-pl_5	5	11	152	5	11	96	5	11	122
SC_9_11-pl_7	7	41	431	7	41	299	7	41	326
SC_9_11-pl_9	9	238	1895	9	238	2158	9	237	2041
SC_9_11-pl_10	10	663	6151	10	663	5467	10	657	6013
SC_9_11-pl_11	11	1262	15073	11	1262	10960	11	1272	12372
SC_10_8-pl_9	–	–	TO	–	–	TO	–	–	TO
SC_10_8-pl_14	14	93	481	14	93	378	14	93	572
SC_10_10-pl_2	2	6	280	2	6	173	2	6	311
SC_10_10-pl_3	3	10	239	3	10	248	3	10	347
SC_10_10-pl_9	9	10	188	9	10	120	9	10	117
SC_10_10-pl_9	9	886	20614	9	886	17444	9	1248	32644
SC_10_10-pl_10	10	15	191	10	15	158	10	15	161
SC_10_10-pl_13	13	15826	285398	13	15826	330014	13	17305	472785
SC_10_10-pl_14	–	–	TO	14	30967	582175	–	–	TO
SC_10_10-pl_17	17	946	7240	17	946	5175	17	942	7546
SC_10_10-pl_17	–	–	TO	–	–	TO	–	–	TO
avg ± std (all)	7 ± 3	1228 ± 4886	21793 ± 62537	7 ± 3	2481 ± 7827	35557 ± 106042	7 ± 3	1711 ± 5628	34711 ± 94709
IQM ± IQR-std (all)	6 ± 4	66 ± 300	1758 ± 9389	6 ± 4	61 ± 447	1335 ± 5215	6 ± 3	156 ± 661	3908 ± 16651
avg ± std (comm)	7 ± 3	1274 ± 4981	16258 ± 49770	7 ± 3	1939 ± 6818	18730 ± 61527	7 ± 3	1706 ± 5687	33899 ± 95522
IQM ± IQR-std (comm)	7 ± 4	73 ± 376	1465 ± 6213	6 ± 3	63 ± 368	1126 ± 4650	6 ± 3	137 ± 648	3365 ± 13240
Solved Instances	50/54 (92.59%)			50/54 (92.59%)			49/54 (90.74%)		

Table 10: Comparison of execution on the ablation study over the Test instances. The model used has been trained using the instances reported in the previous Train Table.

Instance Name	mask			hash			map		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
Assemble_B10-pl_5	–	–	TO	–	–	TO	–	–	TO
Assemble_B2-pl_5	5	14	264	5	14	264	5	14	285
Assemble_B3-pl_5	5	14	164	5	14	154	5	14	308
Assemble_B4-pl_5	5	14	275	5	14	332	5	14	330
Assemble_B5-pl_5	5	14	223	5	14	371	5	14	475
Assemble_B6-pl_5	5	14	804	5	14	826	5	14	1127
Assemble_B7-pl_5	5	14	4368	5	14	4790	5	14	9894
Assemble_B8-pl_5	5	14	24655	5	14	25636	5	14	48340
Assemble_B9-pl_5	5	14	295842	5	14	296625	–	–	TO
Assemble_C-pl_5	5	14	124	5	14	237	5	14	238
CC_2_2_3-pl_3	3	9	111	4	11	108	3	9	181
CC_2_2_3-pl_4	4	13	183	5	9	141	4	14	171
CC_2_2_3-pl_5	5	17	272	5	9	134	5	77	684
CC_2_2_3-pl_6	6	62	424	6	10	106	6	178	1532
CC_2_2_3-pl_7	7	163	2390	8	33837	214811	7	558	4358
CC_2_2_3-pl_8	8	1327	18235	8	247	2434	8	2539	27312
CC_2_2_4-pl_3	3	3	217	3	3	279	3	11	373
CC_2_2_4-pl_4	6	7	324	6	9	212	4	23	778
CC_2_2_4-pl_5	7	43	938	9	77	2507	5	94	3483
CC_2_2_4-pl_6	6	21	1251	7	26	296	6	363	9536
CC_2_2_4-pl_7	8	129	3989	8	498	4701	7	2084	45668
CC_2_3_4-pl_3	5	6	2517	3	3	1381	3	19	6067
CC_2_3_4-pl_4	5	5	1995	6	10	1655	4	42	12130
CC_2_3_4-pl_5	10	132	20181	12	61	5848	5	127	45028
CC_2_3_4-pl_6	11	1269	65775	8	11	3383	6	675	124726
CC_2_3_4-pl_7	11	2054	88913	9	22	3244	7	4153	585278
CC_3_2_3-pl_3	3	4	228	3	3	196	3	14	318
CC_3_2_3-pl_4	5	13	219	5	7	160	4	47	689
CC_3_2_3-pl_5	6	51	888	6	45	463	5	136	2088
CC_3_2_3-pl_6	6	74	1201	6	113	1092	6	367	6443
CC_3_2_3-pl_7	8	185	3058	7	1835	24464	7	1548	26110
CC_3_3_3-pl_3	3	3	218	3	4	367	3	15	790
CC_3_3_3-pl_4	6	19	1730	4	4	495	4	39	1337
CC_3_3_3-pl_5	6	26	1975	7	22	534	5	253	8316
CC_3_3_3-pl_6	8	226	13395	–	–	TO	6	1942	73680
CC_3_3_3-pl_7	8	1476	59481	8	79	1255	7	11726	405444
CoinBox-pl_2	2	2	52	2	2	50	2	2	109
CoinBox-pl_3	3	17	220	3	17	129	3	17	228
CoinBox-pl_5	5	77	425	5	77	552	5	78	728
CoinBox-pl_6	6	381	3534	6	381	2748	6	381	3760
CoinBox-pl_7	7	1817	10395	7	1817	13110	7	1793	17758
SC_4_1-pl_3	3	4	51	3	4	52	3	4	96
SC_4_1-pl_5	5	17	125	5	17	99	5	17	93
SC_4_2-pl_5	5	15	132	5	15	99	5	30	184
SC_4_2-pl_7	7	120	609	7	120	545	7	97	533
SC_4_2-pl_8	8	339	1639	8	339	1316	8	184	923
SC_4_3-pl_5	5	17	116	5	17	62	5	17	98
SC_4_3-pl_6	6	21	162	6	21	113	6	21	93
SC_4_3-pl_8	8	59	186	8	59	156	8	59	177
SC_4_4-pl_5	5	17	125	5	17	87	5	17	91
SC_8_10-pl_6	6	49	315	6	49	343	6	113	882
SC_8_10-pl_8	8	542	5157	8	542	3841	8	485	4121
SC_8_10-pl_9	9	1660	17268	9	1660	14557	9	1772	16936
SC_8_10-pl_12	12	31662	199243	12	31662	203157	12	35063	281914
SC_9_11-pl_4	4	8	143	4	8	141	4	8	139

SC_9_11-pl_5	5	11	152	5	11	96	5	11	122
SC_9_11-pl_6	6	26	179	6	26	161	6	26	198
SC_9_11-pl_7	7	41	431	7	41	299	7	41	326
SC_9_11-pl_8	8	66	388	8	66	271	8	66	500
SC_9_11-pl_9	9	238	1895	9	238	2158	9	237	2041
SC_9_11-pl_10	10	663	6151	10	663	5467	10	657	6013
SC_9_11-pl_11	11	1262	15073	11	1262	10960	11	1272	12372
SC_10_8-pl_9	–	–	TO	–	–	TO	–	–	TO
SC_10_8-pl_10	10	10	174	10	10	97	10	10	108
SC_10_8-pl_14	14	93	481	14	93	378	14	93	572
SC_10_8-pl_15	15	130	824	15	130	543	15	130	822
SC_10_10-pl_2	2	6	280	2	6	173	2	6	311
SC_10_10-pl_3	3	10	239	3	10	248	3	10	347
SC_10_10-pl_6	6	309	8676	6	309	5998	6	298	8135
SC_10_10-pl_7	7	647	17999	7	647	11495	7	589	16064
SC_10_10-pl_9	9	10	188	9	10	120	9	10	117
SC_10_10-pl_9	9	886	20614	9	886	17444	9	1248	32644
SC_10_10-pl_10	10	15	191	10	15	158	10	15	161
SC_10_10-pl_10	10	3096	80030	10	3096	58382	10	1977	51904
SC_10_10-pl_13	13	15826	285398	13	15826	330014	13	17305	472785
SC_10_10-pl_13	13	51	431	13	51	360	13	51	358
SC_10_10-pl_14	–	–	TO	14	30967	582175	–	–	TO
SC_10_10-pl_17	17	946	7240	17	946	5175	17	942	7546
SC_10_10-pl_17	–	–	TO	–	–	TO	–	–	TO
avg ± std (all)	7 ± 3	915 ± 4034	17383 ± 53089	7 ± 3	1723 ± 6489	24971 ± 88115	6 ± 3	1248 ± 4653	32376 ± 101790
IQM ± IQR-std (all)	6 ± 3	54 ± 218	1255 ± 5436	6 ± 4	44 ± 232	971 ± 4112	6 ± 3	87 ± 445	2415 ± 9514
avg ± std (comm)	7 ± 3	937 ± 4087	13624 ± 42651	7 ± 3	1346 ± 5601	13617 ± 50834	7 ± 3	1238 ± 4684	31810 ± 102369
IQM ± IQR-std (comm)	6 ± 3	55 ± 224	1121 ± 4151	6 ± 3	45 ± 227	871 ± 3225	6 ± 3	87 ± 367	2358 ± 9251
Solved Instances	75/79 (94.94%)			75/79 (94.94%)			74/79 (93.67%)		

Table 11: Comparison of execution on the ablation study over the Train and Test instances combined.

F.2 Symbolic Mapping Representation Search Strategies Comparison

Instance Name	HFS*			HFS		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
Assemble_B3-pl_5	5	14	308	5	14	302
Assemble_B5-pl_5	5	14	475	5	14	448
Assemble_B7-pl_5	5	14	9894	5	14	10020
CC_2_2_3-pl_4	4	14	171	4	9	182
CC_2_2_3-pl_6	6	178	1532	7	25	291
CC_2_2_4-pl_5	5	94	3483	12	70	2787
CC_2_3_4-pl_3	3	19	6067	24	199	49682
CC_2_3_4-pl_7	7	4153	585278	–	–	TO
CC_3_2_3-pl_4	4	47	689	5	16	254
CC_3_2_3-pl_5	5	136	2088	8	39	500
CC_3_3_3-pl_4	4	39	1337	4	10	599
CoinBox-pl_3	3	17	228	4	12	201
CoinBox-pl_6	6	381	3760	–	–	TO
SC_4_1-pl_3	3	4	96	3	4	109
SC_4_2-pl_7	7	97	533	–	–	TO
SC_4_3-pl_5	5	17	98	–	–	TO
SC_8_10-pl_6	6	113	882	1951	24489	384947
SC_9_11-pl_6	6	26	198	509	5116	48319
SC_9_11-pl_8	8	66	500	1099	12447	152051
SC_10_8-pl_10	10	10	108	10	10	137
SC_10_8-pl_15	15	130	822	–	–	TO
SC_10_10-pl_6	6	298	8135	–	–	TO
SC_10_10-pl_7	7	589	16064	–	–	TO

SC_10_10-pl_10	10	1977	51904	–	–	TO
SC_10_10-pl_13	13	51	358	17	49	400
avg ± std (all)	6 ± 3	340 ± 871	27800 ± 114266	216 ± 513	2502 ± 6299	38308 ± 94330
IQM ± IQR-std (all)	6 ± 2	66 ± 119	1290 ± 3452	8 ± 12	28 ± 58	1733 ± 9766
avg ± std (comm)	6 ± 3	50 ± 49	1671 ± 2543	216 ± 513	2502 ± 6299	38308 ± 94330
IQM ± IQR-std (comm)	5 ± 2	29 ± 52	701 ± 1304	8 ± 12	28 ± 58	1733 ± 9766
Solved Instances	25/25 (100.00%)			17/25 (68.00%)		

Table 12: Comparison of execution on the ablation study with `map` over the Train instances. The model used has been trained using the instances reported in this Table.

Instance Name	HFS*			HFS		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
Assemble_B10-pl_5	–	–	TO	–	–	TO
Assemble_B2-pl_5	5	14	285	5	14	287
Assemble_B4-pl_5	5	14	330	5	14	357
Assemble_B6-pl_5	5	14	1127	5	14	1190
Assemble_B8-pl_5	5	14	48340	5	14	49503
Assemble_B9-pl_5	–	–	TO	–	–	TO
Assemble_C-pl_5	5	14	238	5	14	257
CC_2_2_3-pl_3	3	9	181	4	10	184
CC_2_2_3-pl_5	5	77	684	7	28	255
CC_2_2_3-pl_7	7	558	4358	44	363	2249
CC_2_2_3-pl_8	8	2539	27312	–	–	TO
CC_2_2_4-pl_3	3	11	373	4	9	370
CC_2_2_4-pl_4	4	23	778	7	28	866
CC_2_2_4-pl_6	6	363	9536	13	79	2764
CC_2_2_4-pl_7	7	2084	45668	–	–	TO
CC_2_3_4-pl_4	4	42	12130	24	201	49087
CC_2_3_4-pl_5	5	127	45028	13	85	27215
CC_2_3_4-pl_6	6	675	124726	29	256	56262
CC_3_2_3-pl_3	3	14	318	9	48	533
CC_3_2_3-pl_6	6	367	6443	6	21	323
CC_3_2_3-pl_7	7	1548	26110	–	–	TO
CC_3_3_3-pl_3	3	15	790	5	17	828
CC_3_3_3-pl_5	5	253	8316	18	137	3073
CC_3_3_3-pl_6	6	1942	73680	31	279	5961
CC_3_3_3-pl_7	7	11726	405444	–	–	TO
CoinBox-pl_2	2	2	109	2	2	106
CoinBox-pl_5	5	78	728	5	16	265
CoinBox-pl_7	7	1793	17758	–	–	TO
SC_4_1-pl_5	5	17	93	–	–	TO
SC_4_2-pl_5	5	30	184	–	–	TO
SC_4_2-pl_8	8	184	923	–	–	TO
SC_4_3-pl_6	6	21	93	–	–	TO
SC_4_3-pl_8	8	59	177	–	–	TO
SC_4_4-pl_5	5	17	91	–	–	TO
SC_8_10-pl_8	8	485	4121	–	–	TO
SC_8_10-pl_9	9	1772	16936	–	–	TO
SC_8_10-pl_12	12	35063	281914	–	–	TO
SC_9_11-pl_4	4	8	139	6	13	150
SC_9_11-pl_5	5	11	122	8	21	189
SC_9_11-pl_7	7	41	326	1286	15107	202732
SC_9_11-pl_9	9	237	2041	–	–	TO
SC_9_11-pl_10	10	657	6013	–	–	TO
SC_9_11-pl_11	11	1272	12372	–	–	TO

SC_10_8-pl_9		–	–	TO	–	–	TO
SC_10_8-pl_14		14	93	572	16	33	252
SC_10_10-pl_2		2	6	311	3	8	315
SC_10_10-pl_3		3	10	347	–	–	TO
SC_10_10-pl_9		9	10	117	9	10	132
SC_10_10-pl_9		9	1248	32644	12	57	1589
SC_10_10-pl_10		10	15	161	11	17	176
SC_10_10-pl_13		13	17305	472785	–	–	TO
SC_10_10-pl_14		–	–	TO	–	–	TO
SC_10_10-pl_17		17	942	7546	–	–	TO
SC_10_10-pl_17		–	–	TO	–	–	TO
avg ± std	(all)	7 ± 3	1711 ± 5628	34711 ± 94709	55 ± 233	583 ± 2746	14051 ± 39043
IQM ± IQR-std	(all)	6 ± 3	156 ± 661	3908 ± 16651	8 ± 8	27 ± 65	830 ± 2509
avg ± std	(comm)	5 ± 2	209 ± 424	12858 ± 27539	55 ± 233	583 ± 2746	14051 ± 39043
IQM ± IQR-std	(comm)	5 ± 2	40 ± 113	1716 ± 8031	8 ± 8	27 ± 65	830 ± 2509
Solved Instances			49/54 (90.74%)			29/54 (53.70%)	

Table 13: Comparison of execution on the ablation study with `map` over the Test instances. The model used has been trained using the instances reported in the previous Train Table.

F.3 Hashed-Based Representation Search Strategies Comparison

Instance Name	HFS*			HFS		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
Assemble_B3-pl_5	5	14	154	5	14	249
Assemble_B5-pl_5	5	14	371	5	14	245
Assemble_B7-pl_5	5	14	4790	5	14	4660
CC_2_2_3-pl_4	5	9	141	7	12	123
CC_2_2_3-pl_6	6	10	106	9	18	246
CC_2_2_4-pl_5	9	77	2507	13	38	1221
CC_2_3_4-pl_3	3	3	1381	3	3	918
CC_2_3_4-pl_7	9	22	3244	9	19	4247
CC_3_2_3-pl_4	5	7	160	6	8	315
CC_3_2_3-pl_5	6	45	463	–	–	TO
CC_3_3_3-pl_4	4	4	495	4	4	424
CoinBox-pl_3	3	17	129	4	12	142
CoinBox-pl_6	6	381	2748	–	–	TO
SC_4_1-pl_3	3	4	52	3	4	125
SC_4_2-pl_7	7	120	545	–	–	TO
SC_4_3-pl_5	5	17	62	–	–	TO
SC_8_10-pl_6	6	49	343	483	4776	58158
SC_9_11-pl_6	6	26	161	509	5116	55844
SC_9_11-pl_8	8	66	271	1098	12438	186999
SC_10_8-pl_10	10	10	97	10	10	112
SC_10_8-pl_15	15	130	543	–	–	TO
SC_10_10-pl_6	6	309	5998	–	–	TO
SC_10_10-pl_7	7	647	11495	–	–	TO
SC_10_10-pl_10	10	3096	58382	–	–	TO
SC_10_10-pl_13	13	51	360	17	49	519
avg ± std (all)	7 ± 3	206 ± 608	3800 ± 11431	129 ± 289	1326 ± 3197	18503 ± 45821
IQM ± IQR-std (all)	6 ± 3	31 ± 67	596 ± 2353	8 ± 8	17 ± 28	932 ± 4002
avg ± std (comm)	6 ± 3	23 ± 22	868 ± 1321	129 ± 289	1326 ± 3197	18503 ± 45821
IQM ± IQR-std (comm)	6 ± 3	15 ± 17	273 ± 354	8 ± 8	17 ± 28	932 ± 4002
Solved Instances		25/25 (100.00%)			17/25 (68.00%)	

Table 14: Comparison of execution on the ablation study with hash over the Train instances. The model used has been trained using the instances reported in this Table.

Instance Name	HFS*			HFS		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
Assemble_B10-pl_5	–	–	TO	–	–	TO
Assemble_B2-pl_5	5	14	264	5	14	125
Assemble_B4-pl_5	5	14	332	5	14	167
Assemble_B6-pl_5	5	14	826	5	14	689
Assemble_B8-pl_5	5	14	25636	5	14	25949
Assemble_B9-pl_5	5	14	296625	5	14	379798
Assemble_C-pl_5	5	14	237	5	14	165
CC_2_2_3-pl_3	4	11	108	22	146	2225
CC_2_2_3-pl_5	5	9	134	10	24	563
CC_2_2_3-pl_7	8	33837	214811	–	–	TO
CC_2_2_3-pl_8	8	247	2434	11	36	763
CC_2_2_4-pl_3	3	3	279	3	3	290
CC_2_2_4-pl_4	6	9	212	8	11	689
CC_2_2_4-pl_6	7	26	296	–	–	TO
CC_2_2_4-pl_7	8	498	4701	–	–	TO
CC_2_3_4-pl_4	6	10	1655	13	19	4652
CC_2_3_4-pl_5	12	61	5848	13	24	5774
CC_2_3_4-pl_6	8	11	3383	8	10	5843
CC_3_2_3-pl_3	3	3	196	3	3	330
CC_3_2_3-pl_6	6	113	1092	–	–	TO
CC_3_2_3-pl_7	7	1835	24464	–	–	TO
CC_3_3_3-pl_3	3	4	367	3	3	295
CC_3_3_3-pl_5	7	22	534	8	11	670
CC_3_3_3-pl_6	–	–	TO	–	–	TO
CC_3_3_3-pl_7	8	79	1255	–	–	TO
CoinBox-pl_2	2	2	50	2	2	64
CoinBox-pl_5	5	77	552	5	16	220
CoinBox-pl_7	7	1817	13110	–	–	TO
SC_4_1-pl_5	5	17	99	–	–	TO
SC_4_2-pl_5	5	15	99	–	–	TO
SC_4_2-pl_8	8	339	1316	–	–	TO
SC_4_3-pl_6	6	21	113	–	–	TO
SC_4_3-pl_8	8	59	156	–	–	TO
SC_4_4-pl_5	5	17	87	–	–	TO
SC_8_10-pl_8	8	542	3841	1586	19087	326379
SC_8_10-pl_9	9	1660	14557	1586	19087	319399
SC_8_10-pl_12	12	31662	203157	–	–	TO
SC_9_11-pl_4	4	8	141	6	13	172
SC_9_11-pl_5	5	11	96	8	21	296
SC_9_11-pl_7	7	41	299	1286	15107	195915
SC_9_11-pl_9	9	238	2158	1099	12457	150920
SC_9_11-pl_10	10	663	5467	1100	12470	176047
SC_9_11-pl_11	11	1262	10960	1100	12472	173729
SC_10_8-pl_9	–	–	TO	–	–	TO
SC_10_8-pl_14	14	93	378	16	33	273
SC_10_10-pl_2	2	6	173	4	13	720
SC_10_10-pl_3	3	10	248	–	–	TO
SC_10_10-pl_9	9	10	120	9	10	225
SC_10_10-pl_9	9	886	17444	13	65	946
SC_10_10-pl_10	10	15	158	11	17	273

SC_10_10-pl_13	13	15826	330014	–	–	TO
SC_10_10-pl_14	14	30967	582175	–	–	TO
SC_10_10-pl_17	17	946	5175	–	–	TO
SC_10_10-pl_17	–	–	TO	–	–	TO
avg ± std (all)	7 ± 3	2481 ± 7827	35557 ± 106042	249 ± 510	2851 ± 6028	55455 ± 108611
IQM ± IQR-std (all)	6 ± 4	61 ± 447	1335 ± 5215	8 ± 9	19 ± 31	1488 ± 10596
avg ± std (comm)	7 ± 3	187 ± 391	12358 ± 51387	249 ± 510	2851 ± 6028	55455 ± 108611
IQM ± IQR-std (comm)	7 ± 4	22 ± 71	882 ± 3307	8 ± 9	19 ± 31	1488 ± 10596
Solved Instances	50/54 (92.59%)			32/54 (59.26%)		

Table 15: Comparison of execution on the ablation study with hash over the Test instances. The model used has been trained using the instances reported in the previous Train Table.

Instance Name	HFS*			HFS		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
Assemble_B10-pl_5	–	–	TO	–	–	TO
Assemble_B2-pl_5	5	14	264	5	14	125
Assemble_B3-pl_5	5	14	154	5	14	249
Assemble_B4-pl_5	5	14	332	5	14	167
Assemble_B5-pl_5	5	14	371	5	14	245
Assemble_B6-pl_5	5	14	826	5	14	689
Assemble_B7-pl_5	5	14	4790	5	14	4660
Assemble_B8-pl_5	5	14	25636	5	14	25949
Assemble_B9-pl_5	5	14	296625	5	14	379798
Assemble_C-pl_5	5	14	237	5	14	165
CC_2_2_3-pl_3	4	11	108	22	146	2225
CC_2_2_3-pl_4	5	9	141	7	12	123
CC_2_2_3-pl_5	5	9	134	10	24	563
CC_2_2_3-pl_6	6	10	106	9	18	246
CC_2_2_3-pl_7	8	33837	214811	–	–	TO
CC_2_2_3-pl_8	8	247	2434	11	36	763
CC_2_2_4-pl_3	3	3	279	3	3	290
CC_2_2_4-pl_4	6	9	212	8	11	689
CC_2_2_4-pl_5	9	77	2507	13	38	1221
CC_2_2_4-pl_6	7	26	296	–	–	TO
CC_2_2_4-pl_7	8	498	4701	–	–	TO
CC_2_3_4-pl_3	3	3	1381	3	3	918
CC_2_3_4-pl_4	6	10	1655	13	19	4652
CC_2_3_4-pl_5	12	61	5848	13	24	5774
CC_2_3_4-pl_6	8	11	3383	8	10	5843
CC_2_3_4-pl_7	9	22	3244	9	19	4247
CC_3_2_3-pl_3	3	3	196	3	3	330
CC_3_2_3-pl_4	5	7	160	6	8	315
CC_3_2_3-pl_5	6	45	463	–	–	TO
CC_3_2_3-pl_6	6	113	1092	–	–	TO
CC_3_2_3-pl_7	7	1835	24464	–	–	TO
CC_3_3_3-pl_3	3	4	367	3	3	295
CC_3_3_3-pl_4	4	4	495	4	4	424
CC_3_3_3-pl_5	7	22	534	8	11	670
CC_3_3_3-pl_6	–	–	TO	–	–	TO
CC_3_3_3-pl_7	8	79	1255	–	–	TO
CoinBox-pl_2	2	2	50	2	2	64
CoinBox-pl_3	3	17	129	4	12	142
CoinBox-pl_5	5	77	552	5	16	220
CoinBox-pl_6	6	381	2748	–	–	TO
CoinBox-pl_7	7	1817	13110	–	–	TO

SC_4_1-pl_3	3	4	52	3	4	125
SC_4_1-pl_5	5	17	99	–	–	TO
SC_4_2-pl_5	5	15	99	–	–	TO
SC_4_2-pl_7	7	120	545	–	–	TO
SC_4_2-pl_8	8	339	1316	–	–	TO
SC_4_3-pl_5	5	17	62	–	–	TO
SC_4_3-pl_6	6	21	113	–	–	TO
SC_4_3-pl_8	8	59	156	–	–	TO
SC_4_4-pl_5	5	17	87	–	–	TO
SC_8_10-pl_6	6	49	343	483	4776	58158
SC_8_10-pl_8	8	542	3841	1586	19087	326379
SC_8_10-pl_9	9	1660	14557	1586	19087	319399
SC_8_10-pl_12	12	31662	203157	–	–	TO
SC_9_11-pl_4	4	8	141	6	13	172
SC_9_11-pl_5	5	11	96	8	21	296
SC_9_11-pl_6	6	26	161	509	5116	55844
SC_9_11-pl_7	7	41	299	1286	15107	195915
SC_9_11-pl_8	8	66	271	1098	12438	186999
SC_9_11-pl_9	9	238	2158	1099	12457	150920
SC_9_11-pl_10	10	663	5467	1100	12470	176047
SC_9_11-pl_11	11	1262	10960	1100	12472	173729
SC_10_8-pl_9	–	–	TO	–	–	TO
SC_10_8-pl_10	10	10	97	10	10	112
SC_10_8-pl_14	14	93	378	16	33	273
SC_10_8-pl_15	15	130	543	–	–	TO
SC_10_10-pl_2	2	6	173	4	13	720
SC_10_10-pl_3	3	10	248	–	–	TO
SC_10_10-pl_6	6	309	5998	–	–	TO
SC_10_10-pl_7	7	647	11495	–	–	TO
SC_10_10-pl_9	9	10	120	9	10	225
SC_10_10-pl_9	9	886	17444	13	65	946
SC_10_10-pl_10	10	15	158	11	17	273
SC_10_10-pl_10	10	3096	58382	–	–	TO
SC_10_10-pl_13	13	15826	330014	–	–	TO
SC_10_10-pl_13	13	51	360	17	49	519
SC_10_10-pl_14	14	30967	582175	–	–	TO
SC_10_10-pl_17	17	946	5175	–	–	TO
SC_10_10-pl_17	–	–	TO	–	–	TO
avg ± std (all)	7 ± 3	1723 ± 6489	24971 ± 88115	207 ± 450	2322 ± 5273	42635 ± 93496
IQM ± IQR-std (all)	6 ± 4	44 ± 232	971 ± 4112	8 ± 8	18 ± 27	1290 ± 5528
avg ± std (comm)	6 ± 3	130 ± 326	8372 ± 41892	207 ± 450	2322 ± 5273	42635 ± 93496
IQM ± IQR-std (comm)	6 ± 4	17 ± 42	584 ± 2276	8 ± 8	18 ± 27	1290 ± 5528
Solved Instances		75/79 (94.94%)			49/79 (62.03%)	

Table 16: Comparison of execution on the ablation study with hash over the Train and Test instances combined.

Instance Name	HFS*			HFS		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
Assemble_B10-pl_5	–	–	TO	–	–	TO
Assemble_B2-pl_5	5	14	285	5	14	287
Assemble_B3-pl_5	5	14	308	5	14	302
Assemble_B4-pl_5	5	14	330	5	14	357
Assemble_B5-pl_5	5	14	475	5	14	448
Assemble_B6-pl_5	5	14	1127	5	14	1190
Assemble_B7-pl_5	5	14	9894	5	14	10020
Assemble_B8-pl_5	5	14	48340	5	14	49503

Assemble_B9-pl_5	–	–	TO	–	–	TO
Assemble_C-pl_5	5	14	238	5	14	257
CC_2_2_3-pl_3	3	9	181	4	10	184
CC_2_2_3-pl_4	4	14	171	4	9	182
CC_2_2_3-pl_5	5	77	684	7	28	255
CC_2_2_3-pl_6	6	178	1532	7	25	291
CC_2_2_3-pl_7	7	558	4358	44	363	2249
CC_2_2_3-pl_8	8	2539	27312	–	–	TO
CC_2_2_4-pl_3	3	11	373	4	9	370
CC_2_2_4-pl_4	4	23	778	7	28	866
CC_2_2_4-pl_5	5	94	3483	12	70	2787
CC_2_2_4-pl_6	6	363	9536	13	79	2764
CC_2_2_4-pl_7	7	2084	45668	–	–	TO
CC_2_3_4-pl_3	3	19	6067	24	199	49682
CC_2_3_4-pl_4	4	42	12130	24	201	49087
CC_2_3_4-pl_5	5	127	45028	13	85	27215
CC_2_3_4-pl_6	6	675	124726	29	256	56262
CC_2_3_4-pl_7	7	4153	585278	–	–	TO
CC_3_2_3-pl_3	3	14	318	9	48	533
CC_3_2_3-pl_4	4	47	689	5	16	254
CC_3_2_3-pl_5	5	136	2088	8	39	500
CC_3_2_3-pl_6	6	367	6443	6	21	323
CC_3_2_3-pl_7	7	1548	26110	–	–	TO
CC_3_3_3-pl_3	3	15	790	5	17	828
CC_3_3_3-pl_4	4	39	1337	4	10	599
CC_3_3_3-pl_5	5	253	8316	18	137	3073
CC_3_3_3-pl_6	6	1942	73680	31	279	5961
CC_3_3_3-pl_7	7	11726	405444	–	–	TO
CoinBox-pl_2	2	2	109	2	2	106
CoinBox-pl_3	3	17	228	4	12	201
CoinBox-pl_5	5	78	728	5	16	265
CoinBox-pl_6	6	381	3760	–	–	TO
CoinBox-pl_7	7	1793	17758	–	–	TO
SC_4_1-pl_3	3	4	96	3	4	109
SC_4_1-pl_5	5	17	93	–	–	TO
SC_4_2-pl_5	5	30	184	–	–	TO
SC_4_2-pl_7	7	97	533	–	–	TO
SC_4_2-pl_8	8	184	923	–	–	TO
SC_4_3-pl_5	5	17	98	–	–	TO
SC_4_3-pl_6	6	21	93	–	–	TO
SC_4_3-pl_8	8	59	177	–	–	TO
SC_4_4-pl_5	5	17	91	–	–	TO
SC_8_10-pl_6	6	113	882	1951	24489	384947
SC_8_10-pl_8	8	485	4121	–	–	TO
SC_8_10-pl_9	9	1772	16936	–	–	TO
SC_8_10-pl_12	12	35063	281914	–	–	TO
SC_9_11-pl_4	4	8	139	6	13	150
SC_9_11-pl_5	5	11	122	8	21	189
SC_9_11-pl_6	6	26	198	509	5116	48319
SC_9_11-pl_7	7	41	326	1286	15107	202732
SC_9_11-pl_8	8	66	500	1099	12447	152051
SC_9_11-pl_9	9	237	2041	–	–	TO
SC_9_11-pl_10	10	657	6013	–	–	TO
SC_9_11-pl_11	11	1272	12372	–	–	TO
SC_10_8-pl_9	–	–	TO	–	–	TO
SC_10_8-pl_10	10	10	108	10	10	137
SC_10_8-pl_14	14	93	572	16	33	252

SC_10_8-pl_15		15	130	822	–	–	TO
SC_10_10-pl_2		2	6	311	3	8	315
SC_10_10-pl_3		3	10	347	–	–	TO
SC_10_10-pl_6		6	298	8135	–	–	TO
SC_10_10-pl_7		7	589	16064	–	–	TO
SC_10_10-pl_9		9	10	117	9	10	132
SC_10_10-pl_9		9	1248	32644	12	57	1589
SC_10_10-pl_10		10	15	161	11	17	176
SC_10_10-pl_10		10	1977	51904	–	–	TO
SC_10_10-pl_13		13	17305	472785	–	–	TO
SC_10_10-pl_13		13	51	358	17	49	400
SC_10_10-pl_14		–	–	TO	–	–	TO
SC_10_10-pl_17		17	942	7546	–	–	TO
SC_10_10-pl_17		–	–	TO	–	–	TO
avg ± std	(all)	6 ± 3	1248 ± 4653	32376 ± 101790	115 ± 371	1292 ± 4503	23015 ± 66231
IQM ± IQR-std	(all)	6 ± 3	87 ± 445	2415 ± 9514	7 ± 10	26 ± 63	808 ± 2747
avg ± std	(comm)	6 ± 3	150 ± 346	8724 ± 22575	115 ± 371	1292 ± 4503	23015 ± 66231
IQM ± IQR-std	(comm)	5 ± 2	32 ± 80	831 ± 3890	7 ± 10	26 ± 63	808 ± 2747
Solved Instances			74/79 (93.67%)			46/79 (58.23%)	

Table 17: Comparison of execution on the ablation study with `map` over the Train and Test instances combined.

F.4 Bitmask Representation Search Strategies Comparison

Instance Name	HFS*			HFS		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
Assemble_B3-pl_5	5	14	164	5	14	378
Assemble_B5-pl_5	5	14	223	5	14	323
Assemble_B7-pl_5	5	14	4368	5	14	8500
CC_2_2_3-pl_4	4	13	183	6	12	116
CC_2_2_3-pl_6	6	62	424	7	17	228
CC_2_2_4-pl_5	7	43	938	14	43	847
CC_2_3_4-pl_3	5	6	2517	7	9	4534
CC_2_3_4-pl_7	11	2054	88913	–	–	TO
CC_3_2_3-pl_4	5	13	219	8	12	406
CC_3_2_3-pl_5	6	51	888	21	3442	92075
CC_3_3_3-pl_4	6	19	1730	7	8	788
CoinBox-pl_3	3	17	220	4	12	160
CoinBox-pl_6	6	381	3534	–	–	TO
SC_4_1-pl_3	3	4	51	3	4	57
SC_4_2-pl_7	7	120	609	–	–	TO
SC_4_3-pl_5	5	17	116	–	–	TO
SC_8_10-pl_6	6	49	315	483	4776	50264
SC_9_11-pl_6	6	26	179	–	–	TO
SC_9_11-pl_8	8	66	388	–	–	TO
SC_10_8-pl_10	10	10	174	10	10	153
SC_10_8-pl_15	15	130	824	–	–	TO
SC_10_10-pl_6	6	309	8676	–	–	TO
SC_10_10-pl_7	7	647	17999	–	–	TO
SC_10_10-pl_10	10	3096	80030	–	–	TO
SC_10_10-pl_13	13	51	431	17	49	406
avg ± std (all)	7 ± 3	289 ± 705	8565 ± 22733	40 ± 118	562 ± 1412	10616 ± 25045
IQM ± IQR-std (all)	6 ± 2	40 ± 106	748 ± 2298	7 ± 7	14 ± 19	482 ± 2496
avg ± std (comm)	6 ± 2	25 ± 19	856 ± 1148	40 ± 118	562 ± 1412	10616 ± 25045
IQM ± IQR-std (comm)	5 ± 1	18 ± 33	389 ± 712	7 ± 7	14 ± 19	482 ± 2496
Solved Instances	25/25 (100.00%)			15/25 (60.00%)		

Table 18: Comparison of execution on the ablation study with `mask` over the Train instances. The model used has been trained using the instances reported in this Table.

Instance Name	HFS*			HFS		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
Assemble_B10-pl_5	–	–	TO	–	–	TO
Assemble_B2-pl_5	5	14	264	5	14	269
Assemble_B4-pl_5	5	14	275	5	14	376
Assemble_B6-pl_5	5	14	804	5	14	1025
Assemble_B8-pl_5	5	14	24655	5	14	35914
Assemble_B9-pl_5	5	14	295842	5	14	465844
Assemble_C-pl_5	5	14	124	5	14	174
CC_2_2_3-pl_3	3	9	111	4	7	103
CC_2_2_3-pl_5	5	17	272	5	14	156
CC_2_2_3-pl_7	7	163	2390	11	2150	6290
CC_2_2_3-pl_8	8	1327	18235	–	–	TO
CC_2_2_4-pl_3	3	3	217	3	3	202
CC_2_2_4-pl_4	6	7	324	6	6	390
CC_2_2_4-pl_6	6	21	1251	55	1062	10165
CC_2_2_4-pl_7	8	129	3989	–	–	TO
CC_2_3_4-pl_4	5	5	1995	5	5	1126
CC_2_3_4-pl_5	10	132	20181	–	–	TO
CC_2_3_4-pl_6	11	1269	65775	–	–	TO
CC_3_2_3-pl_3	3	4	228	3	3	236
CC_3_2_3-pl_6	6	74	1201	19	311	12446
CC_3_2_3-pl_7	8	185	3058	–	–	TO
CC_3_3_3-pl_3	3	3	218	3	3	212
CC_3_3_3-pl_5	6	26	1975	8	12	2513
CC_3_3_3-pl_6	8	226	13395	10	18	1864
CC_3_3_3-pl_7	8	1476	59481	–	–	TO
CoinBox-pl_2	2	2	52	2	2	71
CoinBox-pl_5	5	77	425	5	16	156
CoinBox-pl_7	7	1817	10395	–	–	TO
SC_4_1-pl_5	5	17	125	–	–	TO
SC_4_2-pl_5	5	15	132	–	–	TO
SC_4_2-pl_8	8	339	1639	–	–	TO
SC_4_3-pl_6	6	21	162	–	–	TO
SC_4_3-pl_8	8	59	186	–	–	TO
SC_4_4-pl_5	5	17	125	–	–	TO
SC_8_10-pl_8	8	542	5157	1586	19087	329262
SC_8_10-pl_9	9	1660	17268	1586	19087	321429
SC_8_10-pl_12	12	31662	199243	–	–	TO
SC_9_11-pl_4	4	8	143	6	13	192
SC_9_11-pl_5	5	11	152	8	21	315
SC_9_11-pl_7	7	41	431	–	–	TO
SC_9_11-pl_9	9	238	1895	–	–	TO
SC_9_11-pl_10	10	663	6151	–	–	TO
SC_9_11-pl_11	11	1262	15073	–	–	TO
SC_10_8-pl_9	–	–	TO	–	–	TO
SC_10_8-pl_14	14	93	481	16	33	390
SC_10_10-pl_2	2	6	280	4	13	386
SC_10_10-pl_3	3	10	239	–	–	TO
SC_10_10-pl_9	9	10	188	9	10	114
SC_10_10-pl_9	9	886	20614	13	65	1442
SC_10_10-pl_10	10	15	191	11	17	220

SC_10_10-pl_13		13	15826	285398	–	–	TO
SC_10_10-pl_14		–	–	TO	–	–	TO
SC_10_10-pl_17		17	946	7240	–	–	TO
SC_10_10-pl_17		–	–	TO	–	–	TO
avg ± std	(all)	7 ± 3	1228 ± 4886	21793 ± 62537	118 ± 400	1450 ± 4819	41148 ± 114657
IQM ± IQR-std	(all)	6 ± 4	66 ± 300	1758 ± 9389	7 ± 6	15 ± 11	731 ± 2311
avg ± std	(comm)	6 ± 3	136 ± 342	13465 ± 53762	118 ± 400	1450 ± 4819	41148 ± 114657
IQM ± IQR-std	(comm)	5 ± 2	18 ± 66	681 ± 1778	7 ± 6	15 ± 11	731 ± 2311
Solved Instances			50/54 (92.59%)			29/54 (53.70%)	

Table 19: Comparison of execution on the ablation study with `mask` over the Test instances. The model used has been trained using the instances reported in the previous Train Table.

Instance Name	HFS*			HFS		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
Assemble_B10-pl_5	–	–	TO	–	–	TO
Assemble_B2-pl_5	5	14	264	5	14	269
Assemble_B3-pl_5	5	14	164	5	14	378
Assemble_B4-pl_5	5	14	275	5	14	376
Assemble_B5-pl_5	5	14	223	5	14	323
Assemble_B6-pl_5	5	14	804	5	14	1025
Assemble_B7-pl_5	5	14	4368	5	14	8500
Assemble_B8-pl_5	5	14	24655	5	14	35914
Assemble_B9-pl_5	5	14	295842	5	14	465844
Assemble_C-pl_5	5	14	124	5	14	174
CC_2_2_3-pl_3	3	9	111	4	7	103
CC_2_2_3-pl_4	4	13	183	6	12	116
CC_2_2_3-pl_5	5	17	272	5	14	156
CC_2_2_3-pl_6	6	62	424	7	17	228
CC_2_2_3-pl_7	7	163	2390	11	2150	6290
CC_2_2_3-pl_8	8	1327	18235	–	–	TO
CC_2_2_4-pl_3	3	3	217	3	3	202
CC_2_2_4-pl_4	6	7	324	6	6	390
CC_2_2_4-pl_5	7	43	938	14	43	847
CC_2_2_4-pl_6	6	21	1251	55	1062	10165
CC_2_2_4-pl_7	8	129	3989	–	–	TO
CC_2_3_4-pl_3	5	6	2517	7	9	4534
CC_2_3_4-pl_4	5	5	1995	5	5	1126
CC_2_3_4-pl_5	10	132	20181	–	–	TO
CC_2_3_4-pl_6	11	1269	65775	–	–	TO
CC_2_3_4-pl_7	11	2054	88913	–	–	TO
CC_3_2_3-pl_3	3	4	228	3	3	236
CC_3_2_3-pl_4	5	13	219	8	12	406
CC_3_2_3-pl_5	6	51	888	21	3442	92075
CC_3_2_3-pl_6	6	74	1201	19	311	12446
CC_3_2_3-pl_7	8	185	3058	–	–	TO
CC_3_3_3-pl_3	3	3	218	3	3	212
CC_3_3_3-pl_4	6	19	1730	7	8	788
CC_3_3_3-pl_5	6	26	1975	8	12	2513
CC_3_3_3-pl_6	8	226	13395	10	18	1864
CC_3_3_3-pl_7	8	1476	59481	–	–	TO
CoinBox-pl_2	2	2	52	2	2	71
CoinBox-pl_3	3	17	220	4	12	160
CoinBox-pl_5	5	77	425	5	16	156
CoinBox-pl_6	6	381	3534	–	–	TO
CoinBox-pl_7	7	1817	10395	–	–	TO

SC_4_1-pl_3	3	4	51	3	4	57
SC_4_1-pl_5	5	17	125	–	–	TO
SC_4_2-pl_5	5	15	132	–	–	TO
SC_4_2-pl_7	7	120	609	–	–	TO
SC_4_2-pl_8	8	339	1639	–	–	TO
SC_4_3-pl_5	5	17	116	–	–	TO
SC_4_3-pl_6	6	21	162	–	–	TO
SC_4_3-pl_8	8	59	186	–	–	TO
SC_4_4-pl_5	5	17	125	–	–	TO
SC_8_10-pl_6	6	49	315	483	4776	50264
SC_8_10-pl_8	8	542	5157	1586	19087	329262
SC_8_10-pl_9	9	1660	17268	1586	19087	321429
SC_8_10-pl_12	12	31662	199243	–	–	TO
SC_9_11-pl_4	4	8	143	6	13	192
SC_9_11-pl_5	5	11	152	8	21	315
SC_9_11-pl_6	6	26	179	–	–	TO
SC_9_11-pl_7	7	41	431	–	–	TO
SC_9_11-pl_8	8	66	388	–	–	TO
SC_9_11-pl_9	9	238	1895	–	–	TO
SC_9_11-pl_10	10	663	6151	–	–	TO
SC_9_11-pl_11	11	1262	15073	–	–	TO
SC_10_8-pl_9	–	–	TO	–	–	TO
SC_10_8-pl_10	10	10	174	10	10	153
SC_10_8-pl_14	14	93	481	16	33	390
SC_10_8-pl_15	15	130	824	–	–	TO
SC_10_10-pl_2	2	6	280	4	13	386
SC_10_10-pl_3	3	10	239	–	–	TO
SC_10_10-pl_6	6	309	8676	–	–	TO
SC_10_10-pl_7	7	647	17999	–	–	TO
SC_10_10-pl_9	9	10	188	9	10	114
SC_10_10-pl_9	9	886	20614	13	65	1442
SC_10_10-pl_10	10	15	191	11	17	220
SC_10_10-pl_10	10	3096	80030	–	–	TO
SC_10_10-pl_13	13	15826	285398	–	–	TO
SC_10_10-pl_13	13	51	431	17	49	406
SC_10_10-pl_14	–	–	TO	–	–	TO
SC_10_10-pl_17	17	946	7240	–	–	TO
SC_10_10-pl_17	–	–	TO	–	–	TO
avg ± std (all)	7 ± 3	915 ± 4034	17383 ± 53089	91 ± 334	1147 ± 4021	30739 ± 95330
IQM ± IQR-std (all)	6 ± 3	54 ± 218	1255 ± 5436	7 ± 6	14 ± 14	652 ± 2819
avg ± std (comm)	6 ± 3	98 ± 283	9167 ± 44059	91 ± 334	1147 ± 4021	30739 ± 95330
IQM ± IQR-std (comm)	5 ± 1	18 ± 40	529 ± 1581	7 ± 6	14 ± 14	652 ± 2819
Solved Instances		75/79 (94.94%)			44/79 (55.70%)	

Table 20: Comparison of execution on the ablation study with `mask` over the Train and Test instances combined.

G EXPERIMENTAL RESULTS

In what follows, we present the results for all the experiments we conducted.

We will use the following abbreviations:

- GNN: denotes our primary contribution, deep equipped with $\mathbf{A}^*$ search and GNN-based heuristics.
- BFS: deep using breadth-first search.
- $\mathcal{H}$-EFP: a planner that also incorporates heuristics, albeit not derived through learning techniques [26]. Following Fabiano et al. [26], we will use the following to distinguish between the various heuristics used by $\mathcal{H}$-EFP
  - SUB: Heuristics that associates a higher evaluation to e-states that satisfy more sub-goals.
  - C_PG: Heuristics that emulates the classical Planning Graph by deriving the "importance" of each belief formula (its distance from the goal level) and then each e-state is characterized by the sum of the derived belief formulae scores. In particular C_PG reflects the *hAdd* heuristics in MEP.
  - L_PG: Heuristics that calculates the score of an e-state by constructing a Planning Graph from it (as initial state) and calculating its length—the shorter the better—of the constructed *e*-PG. This behavior is similar to the one adopted by the heuristics *hFF* in classical planning.
  - S_PG: This heuristics is simply an execution of C_PG on every e-state.
- Nodes: the number of nodes expanded during search, reflecting the informativeness of the heuristics.
- Length: the length of the plan found.
- Time: the solving time (in milliseconds).
- IQM: Interquartile Mean, used as an aggregate performance metric.
- IQR-std: Interquartile Range, reported as a measure of variability.
- avg: arithmetic mean, reported as a baseline aggregate metric.
- std: standard deviation, used to quantify variability in the data.
- all: indicates that the aggregate value is computed over all instances solved by that approach.
- comm: indicates that the aggregate value is computed only over instances solved by all approaches in the comparison.
- TO: indicates a timeout (after 600 seconds).
- −: indicates a missing value due to the problem not being solved within the timeout.

Since each set of experiments includes both "Train" and "Test" problems, we report results for each in separate tables. This distinction is indicated in the table captions using either **Train** or **Test**.

G.1 Experimental Setup #1 – Standard Benchmarks

Instance Name	GNN			BFS		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
Assemble_B3-pl_5	5	10	151	5	14	39
Assemble_B5-pl_5	5	10	124	5	14	55
Assemble_B7-pl_5	5	10	891	5	14	1057
avg ± std (all)	5 ± 0	10 ± 0	389 ± 355	5 ± 0	14 ± 0	384 ± 476
IQM ± IQR-std (all)	5 ± 0	10 ± 0	151 ± 384	5 ± 0	14 ± 0	55 ± 509
avg ± std (comm)	5 ± 0	10 ± 0	389 ± 355	5 ± 0	14 ± 0	384 ± 476
IQM ± IQR-std (comm)	5 ± 0	10 ± 0	151 ± 384	5 ± 0	14 ± 0	55 ± 509
Solved Instances	3/3 (100.00%)			3/3 (100.00%)		

Table 21: Comparison of execution on the Assembly Line domain over the Train instances. The model used by GNN has been trained using the instances reported in this Table.

Instance Name	GNN			BFS		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
Assemble_B2-pl_5	5	10	364	5	14	37
Assemble_B4-pl_5	5	10	122	5	14	40
Assemble_B6-pl_5	5	10	185	5	14	123
Assemble_B8-pl_5	5	10	4270	5	14	5145
Assemble_B9-pl_5	5	10	49596	5	14	65910
Assemble_B10-pl_5	–	–	TO	–	–	TO
Assemble_C-pl_5	5	10	140	5	14	38
avg ± std (all)	5 ± 0	10 ± 0	9113 ± 18166	5 ± 0	14 ± 0	11882 ± 24233
IQM ± IQR-std (all)	5 ± 0	10 ± 0	274 ± 3142	5 ± 0	14 ± 0	82 ± 3851
avg ± std (comm)	5 ± 0	10 ± 0	9113 ± 18166	5 ± 0	14 ± 0	11882 ± 24233
IQM ± IQR-std (comm)	5 ± 0	10 ± 0	274 ± 3142	5 ± 0	14 ± 0	82 ± 3851
Solved Instances	6/7 (85.71%)			6/7 (85.71%)		

Table 22: Comparison of execution on the Assembly Line domain over the Test instances. The model used by GNN has been trained using the instances reported in the previous Train Table.

Instance Name	GNN			BFS		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
CC_2_2_3-pl_4	5	8	120	4	17	20
CC_2_2_3-pl_6	8	67	284	6	287	300
CC_2_2_4-pl_5	7	22	217	5	170	632
CC_2_3_4-pl_3	3	19	2640	3	6	253
CC_2_3_4-pl_7	–	–	TO	7	8907	93302
CC_3_2_3-pl_4	5	7	99	4	30	60
CC_3_2_3-pl_5	8	23	179	5	178	317
CC_3_3_3-pl_4	7	17	634	4	39	120
avg ± std (all)	6 ± 2	23 ± 19	596 ± 851	5 ± 1	1204 ± 2913	11876 ± 30777
IQM ± IQR-std (all)	6 ± 2	19 ± 10	227 ± 310	4 ± 1	104 ± 178	248 ± 291
avg ± std (comm)	6 ± 2	23 ± 19	596 ± 851	4 ± 1	104 ± 100	243 ± 192
IQM ± IQR-std (comm)	6 ± 2	19 ± 10	227 ± 310	4 ± 1	80 ± 150	224 ± 218
Solved Instances	7/8 (87.50%)			8/8 (100.00%)		

Table 23: Comparison of execution on the Collaboration and Communication domain over the Train instances. The model used by GNN has been trained using the instances reported in this Table.

Instance Name	GNN			BFS		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]

CC_2_2_3-pl_3		4	7	155	3	9	13
CC_2_2_3-pl_5		8	226	1030	5	78	92
CC_2_2_3-pl_7		9	271	972	7	1065	1111
CC_2_2_3-pl_8		8	2373	7418	8	2065	2126
CC_2_2_4-pl_3		3	11	300	3	6	42
CC_2_2_4-pl_4		5	10	209	4	24	115
CC_2_2_4-pl_6		7	10	185	6	822	1505
CC_2_2_4-pl_7		8	280	1857	7	4154	7004
CC_2_3_4-pl_4		4	16	2064	4	29	1173
CC_2_3_4-pl_5		13	53	2250	5	275	7629
CC_2_3_4-pl_6		9	20	1213	6	1601	22473
CC_3_2_3-pl_3		3	3	104	3	8	20
CC_3_2_3-pl_6		6	8	127	6	469	830
CC_3_2_3-pl_7		8	42	325	7	3345	7256
CC_3_3_3-pl_3		3	3	194	3	8	68
CC_3_3_3-pl_5		10	745	14505	5	538	1347
CC_3_3_3-pl_6		8	200	2970	6	1643	4670
CC_3_3_3-pl_7		9	456	5932	7	12149	30566
avg ± std	(all)	7 ± 3	263 ± 548	2323 ± 3566	5 ± 2	1572 ± 2824	4891 ± 8173
IQM ± IQR-std	(all)	7 ± 4	65 ± 250	996 ± 2006	5 ± 3	610 ± 1607	1610 ± 6323
avg ± std	(comm)	7 ± 3	263 ± 548	2323 ± 3566	5 ± 2	1572 ± 2824	4891 ± 8173
IQM ± IQR-std	(comm)	7 ± 4	65 ± 250	996 ± 2006	5 ± 3	610 ± 1607	1610 ± 6323
Solved Instances		18/18 (100.00%)			18/18 (100.00%)		

Table 24: Comparison of execution on the Collaboration and Communication domain over the Test instances. The model used by GNN has been trained using the instances reported in the previous Train Table.

Instance Name		GNN			BFS		
		Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
CoinBox-pl_3		3	17	139	3	11	22
CoinBox-pl_6		6	359	1425	6	912	1057
avg ± std	(all)	4 ± 2	188 ± 171	782 ± 643	4 ± 2	462 ± 450	540 ± 518
IQM ± IQR-std	(all)	4 ± 2	188 ± 171	782 ± 643	4 ± 2	462 ± 450	540 ± 518
avg ± std	(comm)	4 ± 2	188 ± 171	782 ± 643	4 ± 2	462 ± 450	540 ± 518
IQM ± IQR-std	(comm)	4 ± 2	188 ± 171	782 ± 643	4 ± 2	462 ± 450	540 ± 518
Solved Instances		2/2 (100.00%)			2/2 (100.00%)		

Table 25: Comparison of execution on the Coin in the Box domain over the Train instances. The model used by GNN has been trained using the instances reported in this Table.

Instance Name		GNN			BFS		
		Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
CoinBox-pl_2		2	2	84	2	2	8
CoinBox-pl_5		5	75	279	5	102	118
CoinBox-pl_7		7	1721	7102	7	2523	3214
avg ± std	(all)	5 ± 2	599 ± 794	2488 ± 3263	5 ± 2	876 ± 1166	1113 ± 1486
IQM ± IQR-std	(all)	5 ± 2	75 ± 860	279 ± 3509	5 ± 2	102 ± 1260	118 ± 1603
avg ± std	(comm)	5 ± 2	599 ± 794	2488 ± 3263	5 ± 2	876 ± 1166	1113 ± 1486
IQM ± IQR-std	(comm)	5 ± 2	75 ± 860	279 ± 3509	5 ± 2	102 ± 1260	118 ± 1603
Solved Instances		3/3 (100.00%)			3/3 (100.00%)		

Table 26: Comparison of execution on the Coin in the Box domain over the Test instances. The model used by GNN has been trained using the instances reported in the previous Train Table.

Instance Name	GNN			BFS		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
Grapevine_3-pl_3	3	18	162	3	31	117
Grapevine_3-pl_4	6	38	200	4	160	667
Grapevine_4-pl_2	2	3	131	2	7	101
Grapevine_5-pl_4	7	68	58927	4	616	61732
avg ± std (all)	4 ± 2	32 ± 24	14855 ± 25445	3 ± 1	204 ± 245	15654 ± 26604
IQM ± IQR-std (all)	4 ± 4	28 ± 31	181 ± 14728	4 ± 1	96 ± 249	392 ± 15820
avg ± std (comm)	4 ± 2	32 ± 24	14855 ± 25445	3 ± 1	204 ± 245	15654 ± 26604
IQM ± IQR-std (comm)	4 ± 4	28 ± 31	181 ± 14728	4 ± 1	96 ± 249	392 ± 15820
Solved Instances	4/4 (100.00%)			4/4 (100.00%)		

Table 27: Comparison of execution on the Grapevine domain over the Train instances. The model used by GNN has been trained using the instances reported in this Table.

Instance Name	GNN			BFS		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
Grapevine_3-pl_2	2	4	114	2	6	22
Grapevine_3-pl_5	–	–	TO	5	823	2028
Grapevine_3-pl_6	–	–	TO	6	2118	4353
Grapevine_3-pl_7	9	10414	144193	7	12008	19763
Grapevine_4-pl_3	6	332	5342	3	40	612
Grapevine_4-pl_4	7	190	10743	4	233	1429
Grapevine_4-pl_5	6	90	6738	5	1449	7077
Grapevine_4-pl_6	7	581	33904	6	4055	18234
Grapevine_5-pl_2	3	3	398	2	8	815
Grapevine_5-pl_3	6	17	3224	3	71	11056
Grapevine_5-pl_5	–	–	TO	–	–	TO
Grapevine_5-pl_6	–	–	TO	–	–	TO
avg ± std (all)	6 ± 2	1454 ± 3392	25582 ± 45969	4 ± 2	2081 ± 3531	6539 ± 7028
IQM ± IQR-std (all)	6 ± 2	157 ± 380	6512 ± 14016	4 ± 3	644 ± 1903	3722 ± 9093
avg ± std (comm)	6 ± 2	1454 ± 3392	25582 ± 45969	4 ± 2	2234 ± 3919	7376 ± 7609
IQM ± IQR-std (comm)	6 ± 2	157 ± 380	6512 ± 14016	4 ± 2	448 ± 2068	5094 ± 12086
Solved Instances	8/12 (66.67%)			10/12 (83.33%)		

Table 28: Comparison of execution on the Grapevine domain over the Test instances. The model used by GNN has been trained using the instances reported in the previous Train Table.

Instance Name	GNN			BFS		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
SC_4_1-pl_3	3	3	56	3	4	2
SC_4_2-pl_7	7	729	549	7	136	106
SC_4_3-pl_5	5	31	81	5	11	4
SC_8_10-pl_6	10	24	131	6	113	148
SC_9_11-pl_6	6	19	99	6	17	20
SC_9_11-pl_8	10	15	96	8	65	91
SC_10_8-pl_10	10	10	60	10	10	8
SC_10_8-pl_15	17	23	88	15	96	92
SC_10_10-pl_13	14	33	165	13	52	49
avg ± std (all)	9 ± 4	99 ± 223	147 ± 146	8 ± 4	56 ± 47	58 ± 50
IQM ± IQR-std (all)	9 ± 4	22 ± 16	99 ± 50	7 ± 4	48 ± 85	52 ± 84
avg ± std (comm)	9 ± 4	99 ± 223	147 ± 146	8 ± 4	56 ± 47	58 ± 50
IQM ± IQR-std (comm)	9 ± 4	22 ± 16	99 ± 50	7 ± 4	48 ± 85	52 ± 84
Solved Instances	9/9 (100.00%)			9/9 (100.00%)		

Table 29: Comparison of execution on the Selective Communication domain over the Train instances. The model used by GNN has been trained using the instances reported in this Table.

Instance Name	GNN			BFS		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
SC_4_1-pl_5	5	26	100	5	11	4
SC_4_2-pl_5	5	549	431	5	22	24
SC_4_2-pl_8	11	1116	942	8	318	258
SC_4_3-pl_6	6	26	77	6	21	6
SC_4_3-pl_8	8	75	112	8	66	16
SC_4_4-pl_5	5	13	64	5	11	4
SC_8_10-pl_8	12	802	3550	8	413	571
SC_8_10-pl_9	12	1175	5247	9	1399	2267
SC_8_10-pl_12	13	12336	42301	12	22486	34954
SC_9_11-pl_4	4	4	65	4	6	8
SC_9_11-pl_5	6	6	64	5	10	12
SC_9_11-pl_7	7	9	75	7	32	30
SC_9_11-pl_9	11	23	141	9	146	216
SC_9_11-pl_10	12	30	199	10	355	590
SC_9_11-pl_11	12	40	197	11	907	1617
SC_10_8-pl_9	–	–	TO	–	–	TO
SC_10_8-pl_14	16	19	83	14	51	40
SC_10_10-pl_9	9	9	65	9	10	7
SC_10_10-pl_10	11	11	72	10	12	9
SC_10_10-pl_17	18	189	738	17	1075	1567
avg ± std (all)	10 ± 4	866 ± 2729	2870 ± 9389	9 ± 3	1440 ± 4977	2221 ± 7743
IQM ± IQR-std (all)	10 ± 6	49 ± 357	157 ± 511	8 ± 4	114 ± 372	131 ± 572
avg ± std (comm)	10 ± 4	866 ± 2729	2870 ± 9389	9 ± 3	1440 ± 4977	2221 ± 7743
IQM ± IQR-std (comm)	10 ± 6	49 ± 357	157 ± 511	8 ± 4	114 ± 372	131 ± 572
Solved Instances	19/20 (95.00%)			19/20 (95.00%)		

Table 30: Comparison of execution on the Selective Communication domain over the Test instances. The model used by GNN has been trained using the instances reported in the previous Train Table.

Instance Name	GNN			BFS		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
SC_R_10_10-pl_6	6	298	1987	6	333	1635
SC_R_10_10-pl_7	7	589	3870	7	792	3951
SC_R_10_10-pl_10	10	1977	11786	10	2431	11831
avg ± std (all)	8 ± 2	955 ± 733	5881 ± 4246	8 ± 2	1185 ± 901	5806 ± 4364
IQM ± IQR-std (all)	7 ± 2	589 ± 840	3870 ± 4900	7 ± 2	792 ± 1049	3951 ± 5098
avg ± std (comm)	8 ± 2	955 ± 733	5881 ± 4246	8 ± 2	1185 ± 901	5806 ± 4364
IQM ± IQR-std (comm)	7 ± 2	589 ± 840	3870 ± 4900	7 ± 2	792 ± 1049	3951 ± 5098
Solved Instances	3/3 (100.00%)			3/3 (100.00%)		

Table 31: Comparison of execution on the Selective Communication Enriched domain over the Train instances. The model used by GNN has been trained using the instances reported in this Table.

Instance Name	GNN			BFS		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
SC_R_10_10-pl_2	2	6	125	2	5	45
SC_R_10_10-pl_3	3	10	109	3	17	88
SC_R_10_10-pl_9	9	1248	7447	9	1112	6079
SC_R_10_10-pl_13	13	17305	106761	13	22625	120270

SC_R_10_10-pl_14	14	38714	239252	14	47255	239197
SC_R_10_10-pl_17	–	–	TO	–	–	TO
avg ± std (all)	8 ± 5	11457 ± 15123	70739 ± 93461	8 ± 5	14203 ± 18642	73136 ± 94840
IQM ± IQR-std (all)	8 ± 10	6188 ± 17295	38111 ± 106636	8 ± 10	7918 ± 22608	42146 ± 120182
avg ± std (comm)	8 ± 5	11457 ± 15123	70739 ± 93461	8 ± 5	14203 ± 18642	73136 ± 94840
IQM ± IQR-std (comm)	8 ± 10	6188 ± 17295	38111 ± 106636	8 ± 10	7918 ± 22608	42146 ± 120182
Solved Instances	5/6 (83.33%)			5/6 (83.33%)		

Table 32: Comparison of execution on the Selective Communication Enriched domain over the Test instances. The model used by GNN has been trained using the instances reported in the previous Train Table.

Instance Name	GNN			BFS		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
Assemble_B3-pl_5	5	10	151	5	14	39
Assemble_B5-pl_5	5	10	124	5	14	55
Assemble_B7-pl_5	5	10	891	5	14	1057
CC_2_2_3-pl_4	5	8	120	4	17	20
CC_2_2_3-pl_6	8	67	284	6	287	300
CC_2_2_4-pl_5	7	22	217	5	170	632
CC_2_3_4-pl_3	3	19	2640	3	6	253
CC_2_3_4-pl_7	–	–	TO	7	8907	93302
CC_3_2_3-pl_4	5	7	99	4	30	60
CC_3_2_3-pl_5	8	23	179	5	178	317
CC_3_3_3-pl_4	7	17	634	4	39	120
CoinBox-pl_3	3	17	139	3	11	22
CoinBox-pl_6	6	359	1425	6	912	1057
Grapevine_3-pl_3	3	18	162	3	31	117
Grapevine_3-pl_4	6	38	200	4	160	667
Grapevine_4-pl_2	2	3	131	2	7	101
Grapevine_5-pl_4	7	68	58927	4	616	61732
SC_R_10_10-pl_6	6	298	1987	6	333	1635
SC_R_10_10-pl_7	7	589	3870	7	792	3951
SC_R_10_10-pl_10	10	1977	11786	10	2431	11831
SC_4_1-pl_3	3	3	56	3	4	2
SC_4_2-pl_7	7	729	549	7	136	106
SC_4_3-pl_5	5	31	81	5	11	4
SC_8_10-pl_6	10	24	131	6	113	148
SC_9_11-pl_6	6	19	99	6	17	20
SC_9_11-pl_8	10	15	96	8	65	91
SC_10_8-pl_10	10	10	60	10	10	8
SC_10_8-pl_15	17	23	88	15	96	92
SC_10_10-pl_13	14	33	165	13	52	49
avg ± std (all)	7 ± 3	159 ± 393	3046 ± 10989	6 ± 3	534 ± 1652	6131 ± 19990
IQM ± IQR-std (all)	6 ± 3	20 ± 35	228 ± 584	5 ± 3	72 ± 164	207 ± 618
avg ± std (comm)	7 ± 3	159 ± 393	3046 ± 10989	6 ± 3	234 ± 483	3017 ± 11523
IQM ± IQR-std (comm)	6 ± 3	20 ± 35	228 ± 584	5 ± 2	65 ± 158	174 ± 594
Solved Instances	28/29 (96.55%)			29/29 (100.00%)		

Table 33: Comparison of execution over all domains over the Train instances. The model used by GNN has been trained using the instances reported in this Table.

Instance Name	GNN			BFS		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
Assemble_B10-pl_5	–	–	TO	–	–	TO
Assemble_B2-pl_5	5	10	364	5	14	37

Assemble_B4-pl_5	5	10	122	5	14	40
Assemble_B6-pl_5	5	10	185	5	14	123
Assemble_B8-pl_5	5	10	4270	5	14	5145
Assemble_B9-pl_5	5	10	49596	5	14	65910
Assemble_C-pl_5	5	10	140	5	14	38
CC_2_2_3-pl_3	4	7	155	3	9	13
CC_2_2_3-pl_5	8	226	1030	5	78	92
CC_2_2_3-pl_7	9	271	972	7	1065	1111
CC_2_2_3-pl_8	8	2373	7418	8	2065	2126
CC_2_2_4-pl_3	3	11	300	3	6	42
CC_2_2_4-pl_4	5	10	209	4	24	115
CC_2_2_4-pl_6	7	10	185	6	822	1505
CC_2_2_4-pl_7	8	280	1857	7	4154	7004
CC_2_3_4-pl_4	4	16	2064	4	29	1173
CC_2_3_4-pl_5	13	53	2250	5	275	7629
CC_2_3_4-pl_6	9	20	1213	6	1601	22473
CC_3_2_3-pl_3	3	3	104	3	8	20
CC_3_2_3-pl_6	6	8	127	6	469	830
CC_3_2_3-pl_7	8	42	325	7	3345	7256
CC_3_3_3-pl_3	3	3	194	3	8	68
CC_3_3_3-pl_5	10	745	14505	5	538	1347
CC_3_3_3-pl_6	8	200	2970	6	1643	4670
CC_3_3_3-pl_7	9	456	5932	7	12149	30566
CoinBox-pl_2	2	2	84	2	2	8
CoinBox-pl_5	5	75	279	5	102	118
CoinBox-pl_7	7	1721	7102	7	2523	3214
Grapevine_3-pl_2	2	4	114	2	6	22
Grapevine_3-pl_5	–	–	TO	5	823	2028
Grapevine_3-pl_6	–	–	TO	6	2118	4353
Grapevine_3-pl_7	9	10414	144193	7	12008	19763
Grapevine_4-pl_3	6	332	5342	3	40	612
Grapevine_4-pl_4	7	190	10743	4	233	1429
Grapevine_4-pl_5	6	90	6738	5	1449	7077
Grapevine_4-pl_6	7	581	33904	6	4055	18234
Grapevine_5-pl_2	3	3	398	2	8	815
Grapevine_5-pl_3	6	17	3224	3	71	11056
Grapevine_5-pl_5	–	–	TO	–	–	TO
Grapevine_5-pl_6	–	–	TO	–	–	TO
SC_R_10_10-pl_2	2	6	125	2	5	45
SC_R_10_10-pl_3	3	10	109	3	17	88
SC_R_10_10-pl_9	9	1248	7447	9	1112	6079
SC_R_10_10-pl_13	13	17305	106761	13	22625	120270
SC_R_10_10-pl_14	14	38714	239252	14	47255	239197
SC_R_10_10-pl_17	–	–	TO	–	–	TO
SC_4_1-pl_5	5	26	100	5	11	4
SC_4_2-pl_5	5	549	431	5	22	24
SC_4_2-pl_8	11	1116	942	8	318	258
SC_4_3-pl_6	6	26	77	6	21	6
SC_4_3-pl_8	8	75	112	8	66	16
SC_4_4-pl_5	5	13	64	5	11	4
SC_8_10-pl_8	12	802	3550	8	413	571
SC_8_10-pl_9	12	1175	5247	9	1399	2267
SC_8_10-pl_12	13	12336	42301	12	22486	34954
SC_9_11-pl_4	4	4	65	4	6	8
SC_9_11-pl_5	6	6	64	5	10	12
SC_9_11-pl_7	7	9	75	7	32	30
SC_9_11-pl_9	11	23	141	9	146	216

SC_9_11-pl_10		12	30	199	10	355	590
SC_9_11-pl_11		12	40	197	11	907	1617
SC_10_8-pl_9		–	–	TO	–	–	TO
SC_10_8-pl_14		16	19	83	14	51	40
SC_10_10-pl_9		9	9	65	9	10	7
SC_10_10-pl_10		11	11	72	10	12	9
SC_10_10-pl_17		18	189	738	17	1075	1567
avg ± std	(all)	7 ± 4	1559 ± 5721	12150 ± 38359	6 ± 3	2462 ± 7298	10425 ± 34654
IQM ± IQR-std	(all)	7 ± 4	64 ± 296	1001 ± 4635	6 ± 3	287 ± 1098	1068 ± 4632
avg ± std	(comm)	7 ± 4	1559 ± 5721	12150 ± 38359	6 ± 3	2495 ± 7417	10671 ± 35210
IQM ± IQR-std	(comm)	7 ± 4	64 ± 296	1001 ± 4635	6 ± 4	242 ± 1080	922 ± 4870
Solved Instances			59/66 (89.39%)			61/66 (92.42%)	

Table 34: Comparison of execution over all domains over the Test instances. The model used by GNN has been trained using the instances reported in the previous Train Table.

G.2 Experimental Setup #2 – Knowledge Transfer

Experimental Setup #2 serves as the Knowledge Transfer experimental results.

To accomplish this, we trained the model CC-GR on the **Collaboration and Communication** and **Grapevine** domains, and tested it against all other domains. To evaluate this model against the other heuristics, we also included the **Epistemic Gossip** domain to expand the testing set for the various heuristics.

Instance Name	GNN			BFS		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
CC_2_2_3-pl_4	4	14	129	4	17	35
CC_2_2_3-pl_6	6	178	997	6	287	705
CC_2_2_4-pl_5	5	94	1915	5	170	1262
CC_2_3_4-pl_3	3	19	2510	3	6	446
CC_2_3_4-pl_7	7	4153	341064	7	8907	171616
CC_3_2_3-pl_4	4	47	418	4	30	91
CC_3_2_3-pl_5	5	136	1331	5	178	549
CC_3_3_3-pl_4	4	39	690	4	39	197
Grapevine_3-pl_3	3	21	266	3	31	177
Grapevine_3-pl_4	4	113	1863	4	160	1374
Grapevine_4-pl_2	2	4	398	2	7	160
Grapevine_5-pl_4	4	430	191371	4	616	120092
avg ± std (all)	4 ± 1	437 ± 1126	45246 ± 103440	4 ± 1	871 ± 2429	24725 ± 55183
IQM ± IQR-std (all)	4 ± 1	75 ± 126	1202 ± 1651	4 ± 1	101 ± 178	556 ± 1117
avg ± std (comm)	4 ± 1	437 ± 1126	45246 ± 103440	4 ± 1	871 ± 2429	24725 ± 55183
IQM ± IQR-std (comm)	4 ± 1	75 ± 126	1202 ± 1651	4 ± 1	101 ± 178	556 ± 1117
Solved Instances	12/12 (100.00%)			12/12 (100.00%)		

Table 35: Comparison of execution on the Train instances of the domains used for the model CC-GR**.**

Instance Name	GNN			BFS		
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]
CC_2_2_3-pl_3	3	9	123	3	9	12
CC_2_2_3-pl_5	5	77	424	5	78	117
CC_2_2_3-pl_7	7	558	2502	7	1065	1789
CC_2_2_3-pl_8	8	2539	15968	8	2065	3288
CC_2_2_4-pl_3	3	11	184	3	6	39
CC_2_2_4-pl_4	4	23	332	4	24	155
CC_2_2_4-pl_6	6	363	5495	6	822	2139
CC_2_2_4-pl_7	7	2084	29932	7	4154	9578
CC_2_3_4-pl_4	4	42	5574	4	29	1757
CC_2_3_4-pl_5	5	127	19082	5	275	11988
CC_2_3_4-pl_6	6	675	73622	6	1601	32067
CC_3_2_3-pl_3	3	14	139	3	8	20
CC_3_2_3-pl_6	6	367	3152	6	469	1287
CC_3_2_3-pl_7	7	1548	14035	7	3345	10559
CC_3_3_3-pl_3	3	15	368	3	8	77
CC_3_3_3-pl_5	5	253	5069	5	538	1832
CC_3_3_3-pl_6	6	1942	48781	6	1643	7964
CC_3_3_3-pl_7	7	11726	424997	7	12149	46251
CoinBox-pl_2	2	2	154	2	2	4
CoinBox-pl_5	5	78	578	5	102	127
CoinBox-pl_7	7	1793	13609	7	2523	4710
Grapevine_3-pl_2	2	4	173	2	6	25
Grapevine_3-pl_5	5	569	9163	5	823	3279
Grapevine_3-pl_6	6	1621	23871	6	2118	6000
Grapevine_3-pl_7	7	6781	132590	7	12008	26646
Grapevine_4-pl_3	3	13	632	3	40	911

Grapevine_4-pl_4	4	141	7522	4	233	2077
Grapevine_4-pl_5	5	433	19923	5	1449	10105
Grapevine_4-pl_6	6	6916	408771	6	4055	25563
Grapevine_5-pl_2	2	4	1814	2	8	1184
Grapevine_5-pl_3	3	27	13589	3	71	13375
Grapevine_5-pl_5	–	–	TO	–	–	TO
Grapevine_5-pl_6	–	–	TO	–	–	TO
SC_R_10_10-pl_2	2	6	132	2	5	63
SC_R_10_10-pl_3	3	10	123	3	17	142
SC_R_10_10-pl_9	9	1248	7513	9	1112	9769
SC_R_10_10-pl_13	13	17305	134216	13	22625	210204
SC_R_10_10-pl_14	14	38714	278706	14	47255	483958
SC_R_10_10-pl_17	–	–	TO	–	–	TO
SC_4_1-pl_5	5	17	92	5	11	3
SC_4_2-pl_5	5	30	141	5	22	20
SC_4_2-pl_8	8	184	533	8	318	314
SC_4_3-pl_6	6	21	96	6	21	6
SC_4_3-pl_8	8	59	135	8	66	17
SC_4_4-pl_5	5	17	71	5	11	4
SC_8_10-pl_8	8	485	2414	8	413	635
SC_8_10-pl_9	9	1772	10314	9	1399	2377
SC_8_10-pl_12	12	35063	238864	12	22486	32874
SC_9_11-pl_4	4	8	105	4	6	9
SC_9_11-pl_5	5	11	110	5	10	12
SC_9_11-pl_7	7	41	203	7	32	39
SC_9_11-pl_9	9	237	1111	9	146	241
SC_9_11-pl_10	10	657	3543	10	355	689
SC_9_11-pl_11	11	1272	7150	11	907	1756
SC_10_8-pl_9	–	–	TO	–	–	TO
SC_10_8-pl_14	14	93	502	14	51	48
SC_10_10-pl_9	9	10	143	9	10	9
SC_10_10-pl_10	10	15	379	10	12	11
SC_10_10-pl_17	17	942	7457	17	1075	1684
avg ± std (all)	6 ± 3	2527 ± 7312	35931 ± 91296	6 ± 3	2729 ± 7638	17633 ± 69781
IQM ± IQR-std (all)	6 ± 4	288 ± 1244	4116 ± 13644	6 ± 4	389 ± 1410	1384 ± 6943
avg ± std (comm)	6 ± 3	2527 ± 7312	35931 ± 91296	6 ± 3	2729 ± 7638	17633 ± 69781
IQM ± IQR-std (comm)	6 ± 4	288 ± 1244	4116 ± 13644	6 ± 4	389 ± 1410	1384 ± 6943
Solved Instances	55/59 (93.22%)			55/59 (93.22%)		

Table 36: Comparison of execution on the Test instances of the domains used for the model CC-GR**. The model used by GNN is** CC-GR **and it has been trained on the instances reported in the previous Train Table.**

$\mathcal{H}$-EFP's Individual Heuristics against Model CC-GR

Instance Name	GNN			BFS			$\mathcal{H}$-EFP C_PG only			
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]	Heur.
Assemble_B2-pl_5	5	14	140	5	14	47	5	5	97	C_PG
Assemble_B4-pl_5	5	14	243	5	14	77	5	5	114	C_PG
Assemble_B6-pl_5	5	14	760	5	14	461	5	5	1852	C_PG
Assemble_B8-pl_5	5	14	31750	5	14	24112	5	5	147496	C_PG
Assemble_B9-pl_5	5	14	585443	5	14	390542	–	–	TO	–
Assemble_B10-pl_5	–	–	TO	–	–	TO	–	–	TO	–
Assemble_C-pl_5	5	14	132	5	14	49	5	5	87	C_PG
CC_2_2_3-pl_3	3	9	62	3	9	17	5	5	64	C_PG
CC_2_2_3-pl_5	5	76	362	5	77	176	5	5	49	C_PG
CC_2_2_3-pl_7	7	451	2245	7	1052	2776	–	–	TO	–
CC_2_2_3-pl_8	8	1934	11128	8	2045	5186	–	–	TO	–
CC_2_2_4-pl_3	3	11	182	3	6	60	4	4	162	C_PG
CC_2_2_4-pl_4	4	23	378	4	24	232	–	–	TO	–
CC_2_2_4-pl_6	6	432	4769	6	824	5030	–	–	TO	–
CC_2_2_4-pl_7	7	1911	21141	7	4171	16361	–	–	TO	–
CC_2_3_4-pl_4	4	42	8451	4	29	2504	–	–	TO	–
CC_2_3_4-pl_5	5	127	30013	5	277	18700	–	–	TO	–
CC_2_3_4-pl_6	6	756	94259	6	1644	50540	–	–	TO	–
CC_3_2_3-pl_3	3	14	123	3	8	49	5	5	143	C_PG
CC_3_2_3-pl_6	6	368	4453	6	471	2205	–	–	TO	–
CC_3_2_3-pl_7	7	1596	25601	7	3329	17005	–	–	TO	–
CC_3_3_3-pl_3	3	15	699	3	8	180	5	5	590	C_PG
CC_3_3_3-pl_5	5	253	6863	5	538	3136	–	–	TO	–
CC_3_3_3-pl_6	6	1955	52929	6	1643	15139	–	–	TO	–
CC_3_3_3-pl_7	7	11769	304660	7	12143	86997	–	–	TO	–
CoinBox-pl_2	2	2	61	2	2	11	4	4	144	C_PG
CoinBox-pl_5	5	77	425	5	101	377	5	5	139	C_PG
CoinBox-pl_7	7	1816	18684	7	2490	7580	–	–	TO	–
Grapevine_3-pl_2	2	4	111	2	6	60	3	3	100	C_PG
Grapevine_3-pl_5	5	568	10866	5	821	4755	–	–	TO	–
Grapevine_3-pl_6	6	1599	21958	6	2113	9928	–	–	TO	–
Grapevine_3-pl_7	7	6561	92670	7	12014	61716	–	–	TO	–
Grapevine_4-pl_3	3	13	824	3	40	1351	–	–	TO	–
Grapevine_4-pl_4	4	141	7062	4	233	5094	–	–	TO	–
Grapevine_4-pl_5	5	434	17301	5	1445	16673	–	–	TO	–
Grapevine_4-pl_6	6	6790	310258	6	4066	40681	–	–	TO	–
Grapevine_5-pl_2	2	4	1344	2	8	1906	3	3	1592	C_PG
Grapevine_5-pl_3	3	27	17341	3	71	26331	–	–	TO	–
Grapevine_5-pl_5	–	–	TO	–	–	TO	–	–	TO	–
Grapevine_5-pl_6	–	–	TO	–	–	TO	–	–	TO	–
SC_4_1-pl_5	5	17	73	5	11	21	–	–	TO	–
SC_4_2-pl_5	5	15	215	5	21	49	–	–	TO	–
SC_4_2-pl_8	8	339	2484	8	306	627	–	–	TO	–
SC_4_3-pl_6	6	21	162	6	21	19	–	–	TO	–
SC_4_3-pl_8	8	59	273	8	66	53	–	–	TO	–
SC_4_4-pl_5	5	17	119	5	11	11	–	–	TO	–
SC_8_10-pl_8	8	542	4565	8	388	1412	–	–	TO	–
SC_8_10-pl_9	9	1660	20349	9	1332	4478	–	–	TO	–
SC_8_10-pl_12	12	18541	187834	12	21604	35400	–	–	TO	–
SC_9_11-pl_4	4	8	89	4	6	23	4	4	74	C_PG
SC_9_11-pl_5	5	11	157	5	10	32	6	6	120	C_PG
SC_9_11-pl_7	7	41	424	7	32	96	8	8	143	C_PG
SC_9_11-pl_9	9	238	2497	9	146	439	12	12	285	C_PG
SC_9_11-pl_10	10	663	6789	10	356	1364	13	13	272	C_PG
SC_9_11-pl_11	11	1262	14034	11	913	2892	14	14	373	C_PG
SC_10_8-pl_9	–	–	TO	–	–	TO	–	–	TO	–
SC_10_8-pl_14	14	93	544	14	51	127	15	24	160	C_PG
SC_10_10-pl_9	9	10	151	9	10	35	9	9	92	C_PG
SC_10_10-pl_10	10	15	223	10	12	29	10	10	87	C_PG
SC_10_10-pl_17	17	946	6582	17	1078	3180	–	–	TO	–
gossip_3_3_3	4	35	1257	4	41	170	5	5	179	C_PG
gossip_3_3_6	2	3	195	2	4	50	4	4	152	C_PG
gossip_3_3_9	6	356	3296	6	356	479	6	6	172	C_PG
gossip_4_3_9	–	–	TO	–	–	TO	9	9	972	C_PG
gossip_4_4_1	2	2	249	2	2	128	2	2	563	C_PG
gossip_4_4_3	–	–	TO	–	–	TO	10	10	4486	C_PG
gossip_4_4_7	–	–	TO	–	–	TO	6	6	1984	C_PG
gossip_5_3_3	4	168	16236	4	179	2572	7	7	1400	C_PG
gossip_5_3_5	0	0	114	0	0	6	–	–	TO	–
gossip_5_3_7	–	–	TO	–	–	TO	8	8	1241	C_PG
gossip_5_4_3	–	–	TO	–	–	TO	12	12	10860	C_PG
gossip_5_4_4	1	1	267	1	1	68	1	1	207	C_PG
gossip_5_4_9	–	–	TO	–	–	TO	16	16	13997	C_PG
gossip_5_5_4	1	1	731	1	1	172	1	1	733	C_PG
gossip_5_5_8	–	–	TO	–	–	TO	8	8	30331	C_PG
avg ± std (all)	6 ± 3	1014 ± 2909	30556 ± 92026	6 ± 3	1230 ± 3419	13625 ± 50476	7 ± 4	42 ± 168	4928 ± 30250	
IQM ± IQR-std (all)	5 ± 1	133 ± 162	3791 ± 4395	5 ± 1	169 ± 226	1518 ± 1652	6 ± 1	8 ± 2	235 ± 170	
avg ± std (comm)	6 ± 3	1031 ± 2930	31039 ± 92684	6 ± 3	1250 ± 3443	13841 ± 50851	7 ± 4	46 ± 179	5170 ± 32344	
IQM ± IQR-std (comm)	5 ± 1	133 ± 162	3906 ± 4419	5 ± 1	180 ± 230	1518 ± 1652	6 ± 1	8 ± 3	164 ± 107	
Solved Instances		64/75 (85.33%)			64/75 (85.33%)			37/75 (49.33%)		

Table 37: Comparison of execution on all the Test problem instances of all the domains. The model used by GNN is CC-GR and it has been trained on the instances reported in the previous Train Table.

Instance Name	GNN			BFS			$\mathcal{H}$-EFP L_PG only			
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]	Heur.
Assemble_B2-pl_5	5	14	140	5	14	47	5	8	164	L_PG
Assemble_B4-pl_5	5	14	243	5	14	77	5	8	218	L_PG
Assemble_B6-pl_5	5	14	760	5	14	461	5	8	2995	L_PG
Assemble_B8-pl_5	5	14	31750	5	14	24112	5	8	241692	L_PG
Assemble_B9-pl_5	5	14	585443	5	14	390542	–	–	TO	–
Assemble_B10-pl_5	–	–	TO	–	–	TO	–	–	TO	–
Assemble_C-pl_5	5	14	132	5	14	49	5	8	179	L_PG
CC_2_2_3-pl_3	3	9	62	3	9	17	3	3	67	L_PG
CC_2_2_3-pl_5	5	76	362	5	77	176	5	7	123	L_PG
CC_2_2_3-pl_7	7	451	2245	7	1052	2776	11	35	676	L_PG
CC_2_2_3-pl_8	8	1934	11128	8	2045	5186	9	16	443	L_PG
CC_2_2_4-pl_3	3	11	182	3	6	60	3	3	180	L_PG
CC_2_2_4-pl_4	4	23	378	4	24	232	4	4	196	L_PG
CC_2_2_4-pl_6	6	432	4769	6	824	5030	7	19	1110	L_PG
CC_2_2_4-pl_7	7	1911	21141	7	4171	16361	10	39	2253	L_PG
CC_2_3_4-pl_4	4	42	8451	4	29	2504	4	4	1566	L_PG
CC_2_3_4-pl_5	5	127	30013	5	277	18700	8	18	6684	L_PG
CC_2_3_4-pl_6	6	756	94259	6	1644	50540	8	23	10986	L_PG
CC_3_2_3-pl_3	3	14	123	3	8	49	3	3	126	L_PG
CC_3_2_3-pl_6	6	368	4453	6	471	2205	6	13	493	L_PG
CC_3_2_3-pl_7	7	1596	25601	7	3329	17005	12	57	2394	L_PG
CC_3_3_3-pl_3	3	15	699	3	8	180	3	3	312	L_PG
CC_3_3_3-pl_5	5	253	6863	5	538	3136	6	34	3583	L_PG
CC_3_3_3-pl_6	6	1955	52929	6	1643	15139	9	28	3029	L_PG
CC_3_3_3-pl_7	7	11769	304660	7	12143	86997	54	682	61837	L_PG
CoinBox-pl_2	2	2	61	2	2	11	2	2	66	L_PG
CoinBox-pl_5	5	77	425	5	101	377	5	10	763	L_PG
CoinBox-pl_7	7	1816	18684	7	2490	7580	8	15	1319	L_PG
Grapevine_3-pl_2	2	4	111	2	6	60	2	4	215	L_PG
Grapevine_3-pl_5	5	568	10866	5	821	4755	53	493	34373	L_PG
Grapevine_3-pl_6	6	1599	21958	6	2113	9928	–	–	TO	–
Grapevine_3-pl_7	7	6561	92670	7	12014	61716	–	–	TO	–
Grapevine_4-pl_3	3	13	824	3	40	1351	3	10	2114	L_PG
Grapevine_4-pl_4	4	141	7062	4	233	5094	10	54	10507	L_PG
Grapevine_4-pl_5	5	434	17301	5	1445	16673	–	–	TO	–
Grapevine_4-pl_6	6	6790	310258	6	4066	40681	–	–	TO	–
Grapevine_5-pl_2	2	4	1344	2	8	1906	2	4	4751	L_PG
Grapevine_5-pl_3	3	27	17341	3	71	26331	–	–	TO	–
Grapevine_5-pl_5	–	–	TO	–	–	TO	–	–	TO	–
Grapevine_5-pl_6	–	–	TO	–	–	TO	–	–	TO	–
SC_4_1-pl_5	5	17	73	5	11	21	5	6	45	L_PG
SC_4_2-pl_5	5	15	215	5	21	49	5	6	83	L_PG
SC_4_2-pl_8	8	339	2484	8	306	627	–	–	TO	–
SC_4_3-pl_6	6	21	162	6	21	19	–	–	TO	–
SC_4_3-pl_8	8	59	273	8	66	53	–	–	TO	–
SC_4_4-pl_5	5	17	119	5	11	11	5	6	51	L_PG
SC_8_10-pl_8	8	542	4565	8	388	1412	–	–	TO	–
SC_8_10-pl_9	9	1660	20349	9	1332	4478	–	–	TO	–
SC_8_10-pl_12	12	18541	187834	12	21604	35400	–	–	TO	–
SC_9_11-pl_4	4	8	89	4	6	23	4	5	154	L_PG
SC_9_11-pl_5	5	11	157	5	10	32	5	8	201	L_PG
SC_9_11-pl_7	7	41	424	7	32	96	7	12	323	L_PG
SC_9_11-pl_9	9	238	2497	9	146	439	–	–	TO	–
SC_9_11-pl_10	10	663	6789	10	356	1364	–	–	TO	–
SC_9_11-pl_11	11	1262	14034	11	913	2892	–	–	TO	–
SC_10_8-pl_9	–	–	TO	–	–	TO	–	–	TO	–
SC_10_8-pl_14	14	93	544	14	51	127	14	14	289	L_PG
SC_10_10-pl_9	9	10	151	9	10	35	9	9	170	L_PG
SC_10_10-pl_10	10	15	223	10	12	29	10	10	212	L_PG
SC_10_10-pl_17	17	946	6582	17	1078	3180	–	–	TO	–
gossip_3_3_3	4	35	1257	4	41	170	4	10	380	L_PG
gossip_3_3_6	2	3	195	2	4	50	2	3	152	L_PG
gossip_3_3_9	6	356	3296	6	356	479	6	21	646	L_PG
gossip_4_3_9	–	–	TO	–	–	TO	9	50	6289	L_PG
gossip_4_4_1	2	2	249	2	2	128	2	2	470	L_PG
gossip_4_4_3	–	–	TO	–	–	TO	8	45	26285	L_PG
gossip_4_4_7	–	–	TO	–	–	TO	12	80	40528	L_PG
gossip_5_3_3	4	168	16236	4	179	2572	6	27	7133	L_PG
gossip_5_3_5	0	0	114	0	0	6	0	0	17	L_PG
gossip_5_3_7	–	–	TO	–	–	TO	12	80	23547	L_PG
gossip_5_4_3	–	–	TO	–	–	TO	10	69	38632	L_PG
gossip_5_4_4	1	1	267	1	1	68	1	1	239	L_PG
gossip_5_4_9	–	–	TO	–	–	TO	16	129	61902	L_PG
gossip_5_5_4	1	1	731	1	1	172	1	1	689	L_PG
gossip_5_5_8	–	–	TO	–	–	TO	–	–	TO	–
avg ± std (all)	6 ± 3	1014 ± 2909	30556 ± 92026	6 ± 3	1230 ± 3419	13625 ± 50476	8 ± 10	41 ± 111	11182 ± 34781	
IQM ± IQR-std (all)	5 ± 1	133 ± 162	3791 ± 4395	5 ± 1	169 ± 226	1518 ± 1652	6 ± 5	12 ± 24	1043 ± 4275	
avg ± std (comm)	5 ± 2	526 ± 1732	13855 ± 45504	5 ± 2	684 ± 1896	5755 ± 14755	8 ± 10	37 ± 117	8472 ± 35467	
IQM ± IQR-std (comm)	5 ± 3	51 ± 268	1584 ± 7234	5 ± 3	66 ± 376	750 ± 3492	5 ± 5	9 ± 14	645 ± 2112	
Solved Instances		64/75 (85.33%)			64/75 (85.33%)			54/75 (72.00%)		

Table 38: Comparison of execution on all the Test problem instances of all the domains. The model used by GNN is CC-GR and it has been trained on the instances reported in the previous Train Table.

Instance Name	GNN			BFS			$\mathcal{H}$-EFP S_PG only			
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]	Heur.
Assemble_B2-pl_5	5	14	140	5	14	47	5	8	176	S_PG
Assemble_B4-pl_5	5	14	243	5	14	77	5	8	209	S_PG
Assemble_B6-pl_5	5	14	760	5	14	461	5	8	3140	S_PG
Assemble_B8-pl_5	5	14	31750	5	14	24112	5	8	239144	S_PG
Assemble_B9-pl_5	5	14	585443	5	14	390542	–	–	TO	–
Assemble_B10-pl_5	–	–	TO	–	–	TO	–	–	TO	–
Assemble_C-pl_5	5	14	132	5	14	49	5	8	178	S_PG
CC_2_2_3-pl_3	3	9	62	3	9	17	3	3	89	S_PG
CC_2_2_3-pl_5	5	76	362	5	77	176	5	5	92	S_PG
CC_2_2_3-pl_7	7	451	2245	7	1052	2776	7	18	389	S_PG
CC_2_2_3-pl_8	8	1934	11128	8	2045	5186	–	–	TO	–
CC_2_2_4-pl_3	3	11	182	3	6	60	3	3	147	S_PG
CC_2_2_4-pl_4	4	23	378	4	24	232	4	4	209	S_PG
CC_2_2_4-pl_6	6	432	4769	6	824	5030	6	6	296	S_PG
CC_2_2_4-pl_7	7	1911	21141	7	4171	16361	10	26	1542	S_PG
CC_2_3_4-pl_4	4	42	8451	4	29	2504	4	4	1531	S_PG
CC_2_3_4-pl_5	5	127	30013	5	277	18700	5	5	2390	S_PG
CC_2_3_4-pl_6	6	756	94259	6	1644	50540	6	6	1731	S_PG
CC_3_2_3-pl_3	3	14	123	3	8	49	3	3	167	S_PG
CC_3_2_3-pl_6	6	368	4453	6	471	2205	6	6	284	S_PG
CC_3_2_3-pl_7	7	1596	25601	7	3329	17005	10	27	1077	S_PG
CC_3_3_3-pl_3	3	15	699	3	8	180	3	3	337	S_PG
CC_3_3_3-pl_5	5	253	6863	5	538	3136	5	5	471	S_PG
CC_3_3_3-pl_6	6	1955	52929	6	1643	15139	6	8	737	S_PG
CC_3_3_3-pl_7	7	11769	304660	7	12143	86997	–	–	TO	–
CoinBox-pl_2	2	2	61	2	2	11	2	2	72	S_PG
CoinBox-pl_5	5	77	425	5	101	377	5	5	350	S_PG
CoinBox-pl_7	7	1816	18684	7	2490	7580	8	9	705	S_PG
Grapevine_3-pl_2	2	4	111	2	6	60	2	2	91	S_PG
Grapevine_3-pl_5	5	568	10866	5	821	4755	7	15	867	S_PG
Grapevine_3-pl_6	6	1599	21958	6	2113	9928	6	7	536	S_PG
Grapevine_3-pl_7	7	6561	92670	7	12014	61716	11	26	2193	S_PG
Grapevine_4-pl_3	3	13	824	3	40	1351	3	3	537	S_PG
Grapevine_4-pl_4	4	141	7062	4	233	5094	4	4	553	S_PG
Grapevine_4-pl_5	5	434	17301	5	1445	16673	–	–	TO	–
Grapevine_4-pl_6	6	6790	310258	6	4066	40681	–	–	TO	–
Grapevine_5-pl_2	2	4	1344	2	8	1906	2	2	1226	S_PG
Grapevine_5-pl_3	3	27	17341	3	71	26331	3	3	1750	S_PG
Grapevine_5-pl_5	–	–	TO	–	–	TO	7	13	10355	S_PG
Grapevine_5-pl_6	–	–	TO	–	–	TO	6	7	12755	S_PG
SC_4_1-pl_5	5	17	73	5	11	21	5	6	55	S_PG
SC_4_2-pl_5	5	15	215	5	21	49	5	6	99	S_PG
SC_4_2-pl_8	8	339	2484	8	306	627	–	–	TO	–
SC_4_3-pl_6	6	21	162	6	21	19	–	–	TO	–
SC_4_3-pl_8	8	59	273	8	66	53	–	–	TO	–
SC_4_4-pl_5	5	17	119	5	11	11	5	6	68	S_PG
SC_8_10-pl_8	8	542	4565	8	388	1412	–	–	TO	–
SC_8_10-pl_9	9	1660	20349	9	1332	4478	–	–	TO	–
SC_8_10-pl_12	12	18541	187834	12	21604	35400	13	13	996	S_PG
SC_9_11-pl_4	4	8	89	4	6	23	4	4	127	S_PG
SC_9_11-pl_5	5	11	157	5	10	32	6	7	220	S_PG
SC_9_11-pl_7	7	41	424	7	32	96	7	11	337	S_PG
SC_9_11-pl_9	9	238	2497	9	146	439	10	16	737	S_PG
SC_9_11-pl_10	10	663	6789	10	356	1364	20	52	3455	S_PG
SC_9_11-pl_11	11	1262	14034	11	913	2892	16	23	1294	S_PG
SC_10_8-pl_9	–	–	TO	–	–	TO	–	–	TO	–
SC_10_8-pl_14	14	93	544	14	51	127	14	14	395	S_PG
SC_10_10-pl_9	9	10	151	9	10	35	9	9	126	S_PG
SC_10_10-pl_10	10	15	223	10	12	29	10	10	169	S_PG
SC_10_10-pl_17	17	946	6582	17	1078	3180	22	22	763	S_PG
gossip_3_3_3	4	35	1257	4	41	170	4	10	475	S_PG
gossip_3_3_6	2	3	195	2	4	50	2	3	149	S_PG
gossip_3_3_9	6	356	3296	6	356	479	6	21	696	S_PG
gossip_4_3_9	–	–	TO	–	–	TO	9	50	6988	S_PG
gossip_4_4_1	2	2	249	2	2	128	2	2	541	S_PG
gossip_4_4_3	–	–	TO	–	–	TO	8	45	28848	S_PG
gossip_4_4_7	–	–	TO	–	–	TO	12	80	42795	S_PG
gossip_5_3_3	4	168	16236	4	179	2572	6	27	7701	S_PG
gossip_5_3_5	0	0	114	0	0	6	0	0	18	S_PG
gossip_5_3_7	–	–	TO	–	–	TO	12	80	23180	S_PG
gossip_5_4_3	–	–	TO	–	–	TO	10	69	35226	S_PG
gossip_5_4_4	1	1	267	1	1	68	1	1	198	S_PG
gossip_5_4_9	–	–	TO	–	–	TO	16	129	58642	S_PG
gossip_5_5_4	1	1	731	1	1	172	1	1	663	S_PG
gossip_5_5_8	–	–	TO	–	–	TO	–	–	TO	–
avg ± std (all)	6 ± 3	1014 ± 2909	30556 ± 92026	6 ± 3	1230 ± 3419	13625 ± 50476	7 ± 4	16 ± 23	8085 ± 31482	
IQM ± IQR-std (all)	5 ± 1	133 ± 162	3791 ± 4395	5 ± 1	169 ± 226	1518 ± 1652	6 ± 5	8 ± 12	639 ± 1501	
avg ± std (comm)	5 ± 3	766 ± 2638	12944 ± 30900	5 ± 3	1054 ± 3336	6024 ± 12442	6 ± 4	10 ± 9	5231 ± 32153	
IQM ± IQR-std (comm)	5 ± 4	86 ± 403	2383 ± 10062	5 ± 4	97 ± 511	1046 ± 4309	5 ± 4	7 ± 8	456 ± 793	
Solved Instances		64/75 (85.33%)			64/75 (85.33%)			62/75 (82.67%)		

Table 39: Comparison of execution on all the Test problem instances of all the domains. The model used by GNN is CC-GR and it has been trained on the instances reported in the previous Train Table.

Instance Name	GNN			BFS			$\mathcal{H}$-EFP SUB only			
	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]	Length	Nodes	Time [ms]	Heur.
Assemble_B2-pl_5	5	14	140	5	14	47	5	10	274	SUB
Assemble_B4-pl_5	5	14	243	5	14	77	5	10	287	SUB
Assemble_B6-pl_5	5	14	760	5	14	461	5	10	921	SUB
Assemble_B8-pl_5	5	14	31750	5	14	24112	5	10	40687	SUB
Assemble_B9-pl_5	5	14	585443	5	14	390542	5	10	594127	SUB
Assemble_B10-pl_5	–	–	TO	–	–	TO	–	–	TO	–
Assemble_C-pl_5	5	14	132	5	14	49	5	10	272	SUB
CC_2_2_3-pl_3	3	9	62	3	9	17	4	4	77	SUB
CC_2_2_3-pl_5	5	76	362	5	77	176	5	6	84	SUB
CC_2_2_3-pl_7	7	451	2245	7	1052	2776	11	42	416	SUB
CC_2_2_3-pl_8	8	1934	11128	8	2045	5186	–	–	TO	–
CC_2_2_4-pl_3	3	11	182	3	6	60	3	3	188	SUB
CC_2_2_4-pl_4	4	23	378	4	24	232	5	10	404	SUB
CC_2_2_4-pl_6	6	432	4769	6	824	5030	9	18	634	SUB
CC_2_2_4-pl_7	7	1911	21141	7	4171	16361	7	18	1263	SUB
CC_2_3_4-pl_4	4	42	8451	4	29	2504	5	10	4796	SUB
CC_2_3_4-pl_5	5	127	30013	5	277	18700	5	8	7774	SUB
CC_2_3_4-pl_6	6	756	94259	6	1644	50540	9	22	8670	SUB
CC_3_2_3-pl_3	3	14	123	3	8	49	3	3	137	SUB
CC_3_2_3-pl_6	6	368	4453	6	471	2205	7	8	188	SUB
CC_3_2_3-pl_7	7	1596	25601	7	3329	17005	7	13	578	SUB
CC_3_3_3-pl_3	3	15	699	3	8	180	3	3	471	SUB
CC_3_3_3-pl_5	5	253	6863	5	538	3136	6	8	789	SUB
CC_3_3_3-pl_6	6	1955	52929	6	1643	15139	6	8	770	SUB
CC_3_3_3-pl_7	7	11769	304660	7	12143	86997	–	–	TO	–
CoinBox-pl_2	2	2	61	2	2	11	2	2	89	SUB
CoinBox-pl_5	5	77	425	5	101	377	7	9	223	SUB
CoinBox-pl_7	7	1816	18684	7	2490	7580	–	–	TO	–
Grapevine_3-pl_2	2	4	111	2	6	60	2	2	94	SUB
Grapevine_3-pl_5	5	568	10866	5	821	4755	5	6	322	SUB
Grapevine_3-pl_6	6	1599	21958	6	2113	9928	6	7	516	SUB
Grapevine_3-pl_7	7	6561	92670	7	12014	61716	–	–	TO	–
Grapevine_4-pl_3	3	13	824	3	40	1351	3	3	475	SUB
Grapevine_4-pl_4	4	141	7062	4	233	5094	4	4	426	SUB
Grapevine_4-pl_5	5	434	17301	5	1445	16673	6	9	1246	SUB
Grapevine_4-pl_6	6	6790	310258	6	4066	40681	6	7	2320	SUB
Grapevine_5-pl_2	2	4	1344	2	8	1906	2	2	1491	SUB
Grapevine_5-pl_3	3	27	17341	3	71	26331	3	3	1811	SUB
Grapevine_5-pl_5	–	–	TO	–	–	TO	5	6	4094	SUB
Grapevine_5-pl_6	–	–	TO	–	–	TO	7	11	15754	SUB
SC_4_1-pl_5	5	17	73	5	11	21	–	–	TO	–
SC_4_2-pl_5	5	15	215	5	21	49	–	–	TO	–
SC_4_2-pl_8	8	339	2484	8	306	627	–	–	TO	–
SC_4_3-pl_6	6	21	162	6	21	19	–	–	TO	–
SC_4_3-pl_8	8	59	273	8	66	53	–	–	TO	–
SC_4_4-pl_5	5	17	119	5	11	11	–	–	TO	–
SC_8_10-pl_8	8	542	4565	8	388	1412	–	–	TO	–
SC_8_10-pl_9	9	1660	20349	9	1332	4478	–	–	TO	–
SC_8_10-pl_12	12	18541	187834	12	21604	35400	–	–	TO	–
SC_9_11-pl_4	4	8	89	4	6	23	4	4	59	SUB
SC_9_11-pl_5	5	11	157	5	10	32	6	6	66	SUB
SC_9_11-pl_7	7	41	424	7	32	96	8	8	96	SUB
SC_9_11-pl_9	9	238	2497	9	146	439	12	12	176	SUB
SC_9_11-pl_10	10	663	6789	10	356	1364	13	13	206	SUB
SC_9_11-pl_11	11	1262	14034	11	913	2892	14	14	239	SUB
SC_10_8-pl_9	–	–	TO	–	–	TO	–	–	TO	–
SC_10_8-pl_14	14	93	544	14	51	127	16	33	205	SUB
SC_10_10-pl_9	9	10	151	9	10	35	9	10	51	SUB
SC_10_10-pl_10	10	15	223	10	12	29	11	12	77	SUB
SC_10_10-pl_17	17	946	6582	17	1078	3180	–	–	TO	–
gossip_3_3_3	4	35	1257	4	41	170	4	10	250	SUB
gossip_3_3_6	2	3	195	2	4	50	2	3	100	SUB
gossip_3_3_9	6	356	3296	6	356	479	6	21	408	SUB
gossip_4_3_9	–	–	TO	–	–	TO	9	50	4497	SUB
gossip_4_4_1	2	2	249	2	2	128	2	2	343	SUB
gossip_4_4_3	–	–	TO	–	–	TO	8	45	20623	SUB
gossip_4_4_7	–	–	TO	–	–	TO	12	80	54239	SUB
gossip_5_3_3	4	168	16236	4	179	2572	6	27	6392	SUB
gossip_5_3_5	0	0	114	0	0	6	0	0	15	SUB
gossip_5_3_7	–	–	TO	–	–	TO	12	80	18720	SUB
gossip_5_4_3	–	–	TO	–	–	TO	10	69	23396	SUB
gossip_5_4_4	1	1	267	1	1	68	1	1	196	SUB
gossip_5_4_9	–	–	TO	–	–	TO	16	129	33129	SUB
gossip_5_5_4	1	1	731	1	1	172	1	1	596	SUB
gossip_5_5_8	–	–	TO	–	–	TO	–	–	TO	–
avg ± std (all)	6 ± 3	1014 ± 2909	30556 ± 92026	6 ± 3	1230 ± 3419	13625 ± 50476	6 ± 4	16 ± 23	14772 ± 77443	
IQM ± IQR-std (all)	5 ± 1	133 ± 162	3791 ± 4395	5 ± 1	169 ± 226	1518 ± 1652	6 ± 4	9 ± 9	573 ± 2003	
avg ± std (comm)	5 ± 3	414 ± 1040	26116 ± 91820	5 ± 3	504 ± 985	13305 ± 54917	6 ± 3	10 ± 8	13646 ± 83133	
IQM ± IQR-std (comm)	5 ± 3	77 ± 354	2742 ± 13014	5 ± 3	89 ± 433	1193 ± 4899	5 ± 4	8 ± 7	370 ± 605	
Solved Instances	64/75 (85.33%)			64/75 (85.33%)			58/75 (77.33%)			

Table 40: Comparison of execution on all the Test problem instances of all the domains. The model used by GNN is CC-GR and it has been trained on the instances reported in the previous Train Table.

H OVERHEAD OF SINGLE-QUERY GNN-BASED ESTIMATION

Although our current implementation evaluates the GNN heuristic one state at a time, we can estimate the potential impact of batching using aggregate runtime statistics.

From Table 4, BFS requires 1384/389 ≈ 3.56 ms (time per expanded node), whereas the GNN-based configuration requires 4116/288 ≈ 14.29 ms per node. The difference, $\beta \approx 10.73$ ms, represents the effective overhead of a single-query GNN evaluation.

Assuming that batching affects only this scoring component, we model the per-node cost under batch size B as

$$r(B) = r_{\text{BFS}} + \frac{\beta}{B},$$

where $r_{\text{BFS}} = 3.56$ ms denotes the planner-dominated per-node cost. For $B = 1$, this yields $r(1) = r_{\text{BFS}} + \beta \approx 14.29$ ms, matching the observed single-query runtime.

The total projected runtime under batching is then

$$T(B) = N_{\text{GNN}} \cdot r(B),$$

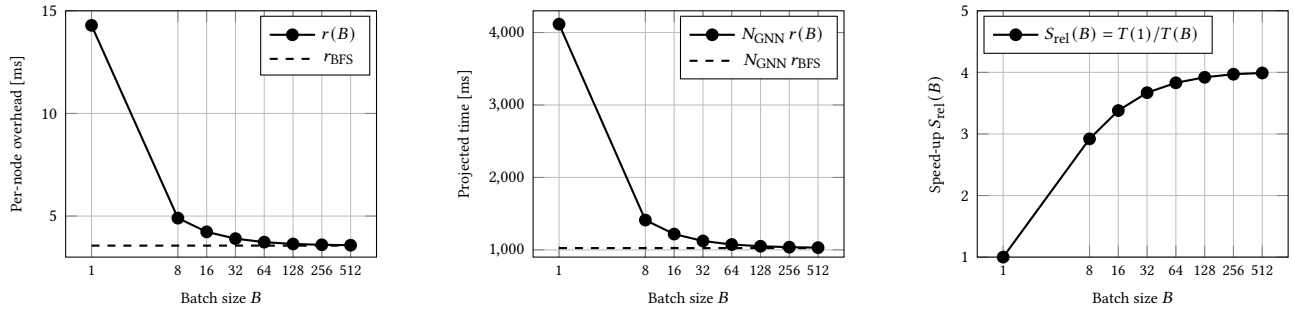
where N_{GNN} is the number of nodes expanded by GNN.

Figure 2 reports the resulting estimates for $B \in \{1, 8, 16, 32, 64, 128, 256, 512\}$. The results show that batching rapidly amortizes the GNN overhead: with $B = 32$, the projected runtime decreases from 4116 ms to approximately 1123 ms, approaching the planner-dominated regime. This supports our choice of expanded nodes as the primary evaluation metric, as heuristic informativeness remains invariant under such engineering optimizations.

Finally, we quantify the relative speed-up induced by batching. Given the projected runtime $T(B)$, we define the speed-up with respect to single-query inference ($B = 1$) as

$$S_{\text{rel}}(B) := \frac{T(1)}{T(B)} = \frac{r(1)}{r(B)} = \frac{r_{\text{BFS}} + \beta}{r_{\text{BFS}} + \beta/B}.$$

Note that $S_{\text{rel}}(B) \leq B$, with diminishing returns as $r(B)$ approaches the planner-dominated baseline r_{BFS}.



(a) Per-node cost. **(b) Projected runtime.** **(c) Relative speed-up.**

Figure 2: Projected batching scalability for the GNN-based heuristic evaluator. Per-node cost is modeled as $r(B) = r_{\text{BFS}} + \beta/B$, and total runtime as $T(B) = N_{\text{GNN}} \cdot r(B)$. The resulting relative speed-up is $S_{\text{rel}}(B) = \frac{T(1)}{T(B)} = \frac{r_{\text{BFS}}+\beta}{r_{\text{BFS}}+\beta/B}$. Batching amortizes single-query GNN overhead, with diminishing returns as costs approach the planner-dominated regime.