Efficient Lookahead Encoding and Abstracted Width for Learning General Policies in Classical Planning

Michael Aichmüller
RWTH Aachen University
Aachen, Germany
michael.aichmueller@ml.rwth-aachen.de

Simon Ståhlberg
RWTH Aachen University
Aachen, Germany
simon.stahlberg@gmail.com

Martin Funkquist
Linköping University
Linköping, Sweden
martin.funkquist@liu.se

Hector Geffner
RWTH Aachen University
Aachen, Germany
hector.geffner@ml.rwth-aachen.de

Abstract

Generalized planning aims to learn policies that generalize across large collections of instances within a classical planning domain. Recent approaches using Graph Neural Networks (GNNs) have shown the ability to learn nearly perfect policies for several domains. This work improves on the recently published idea of Iterated Width (IW) policies. Therein, the policy broadens its successor scope through an IW-lookahead search that offers to "jump" over multiple transitions, simplifying the problem structure. Yet, each transition is evaluated individually leading to unscalable compute and expressivity limitations. Furthermore, while an IW(1) search is attractive due to scaling linearly with the number of atoms in a problem, it still becomes inefficient once thousands of objects are considered like in the International Planning Competition (IPC) 2023 benchmark. In this work, we address both limitations. Firstly, we introduce a vastly more efficient holistic encoding of the entire search tree. It jointly represents IW(1)-reachable states only by their relational differences to the current state, which enables Relational GNNs (R-GNNs) to score all transitions in a single forward pass. Secondly, we define Abstracted IW(1) to improve scaling through relational abstraction during novelty checks. Rather than testing fully instantiated atoms, it abstracts each atom by replacing all but one of its arguments with their types. The original atom is then novel if any of its abstracted forms is. This structural compression shifts the scaling of the novelty search to be linear in the number of objects, rather than atoms, still capturing meaningful subgoal structure. Our contributions are evaluated on the hyperscaling IPC 2023 benchmark and across a broad range of domains, including those that require features beyond the C_2 logic fragment. The results show that our policies achieve a new state-of-the-art performance, significantly surpassing prior work, including classical planners like LAMA.

1 Introduction

Generalized planning aims to learn policies that generalize across large collections of planning problems within the same domain, differing only in initial states, goals, and object counts [Srivastava et al., 2011, Hu and De Giacomo, 2011, Jiménez et al., 2019]. A general policy allows for solving diverse planning problems greedily, eliminating the need for search. Various approaches have been pursued to learn such policies from domain descriptions and training instances, ranging from symbolic

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

to deep learning methods. While symbolic approaches offer transparency and potential correctness guarantees, they often struggle with scalability [Khardon, 1999, Martín and Geffner, 2004, Fern et al., 2006, Srivastava et al., 2011, Boutilier et al., 2001, Sanner and Boutilier, 2009, Francès et al., 2021]. Conversely, deep learning approaches, though opaque, tend to scale more effectively [Toyer et al., 2020, Bajpai et al., 2018, Rivlin et al., 2020, Ståhlberg et al., 2023].

State-of-the-art generalized planning methods include supervised transformer-based approaches [Rossetti et al., 2024] and unsupervised reinforcement learning (RL) approaches using graph neural networks (GNNs), specifically relational GNNs. Both paradigms achieve impressive results on benchmarks, despite the known expressivity limitations of GNNs and transformers [Xu et al., 2019, Barceló et al., 2020, Grohe, 2021].

Recently, Aichmüller and Geffner [2025] introduced IW-policies, which use the Iterated-Width (IW) algorithm [Lipovetzky and Geffner, 2012] to perform lookahead searches at each state to augment successor choices. This enables "jumps" over multiple transitions, often simplifying the problem structure and reducing expressivity requirements. These are integrated by replacing 1-step transitions (s, s') with multi-step transitions $[s, s']$ identified by IW(1) from s [Aichmüller and Geffner, 2025]. However, the embeddings for each object o were inferred independently for states s and s', interacting only within the value-function readout. While feasible for small instances, this approach faces key *computational and expressivity limitations* as size increases, because a given state might have many successors, of which each receives a separate and complete encoding of s'.

Our primary contributions address scaling and expressivity through two key innovations. First, we introduce a novel joint encoding approach for an entire tree of state transitions (s, s'), which avoids wastefully re-encoding shared state descriptions. We propose processing atoms from both s and each s' in an *aggregated* encoding within the R-GNN. By recreating the lookahead tree structure within the graph-encoding and representing successors merely by their atom differences to the current state, we compute all values in a single pass, reducing memory usage several-fold.

Second, we propose Abstracted IW(1), a significantly relaxed variant of IW(1). For novelty testing, this variant abstracts state atoms by iteratively replacing all but one argument with their most specific type and tests the abstracted atoms instead. This shifts the worst-case complexity of the search from being linear in the number of atoms to the number of objects. Consequently, Abstracted IW(1) evaluates significantly faster while still capturing meaningful subgoals for task decomposition. Furthermore, this lookahead mitigates the expressivity limitations of models restricted to immediate successors; it simplifies the problem's cost structure and removes the need for features the R-GNN inherently lacks the expressivity to derive.

These improvements are essential for scaling to large state spaces with numerous successors, a necessity highlighted by the massive instances in the IPC 2023 Learning Track (e.g., a 488-block Blocksworld). Scaling to this level requires both highly efficient IW lookahead and excellent model generalization. By achieving this, our approach outperforms prior work on the benchmark, including the strong LAMA baseline. Finally, the proposed lookahead drives substantial improvements in domains requiring expressivity beyond the standard C_2 logic fragment [Drexler et al., 2024].

The paper is organized as follows. We begin with background on classical and generalized planning, and IW, including its use for state-space jumps. We then introduce R-GNNs, followed by the input encodings and the reinforcement learning training procedure. Finally, we present the experimental evaluation, discuss the results, and conclude. Related work is discussed throughout the paper.

2 Planning

Classical Planning. A classical planning instance is a pair $P = \langle D, I \rangle$, where $D = \langle \mathcal{T}, \mathcal{P}, \mathcal{A} \rangle$ is the domain and $I = \langle \mathcal{O}, s_0, g \rangle$ specifies the instance. The domain D consists of a set of types $\mathcal{T}$ organized in a hierarchy, along with a set of predicates $\mathcal{P}$ and actions $\mathcal{A}$. A predicate $P \in \mathcal{P}$ has an arity n and an associated tuple of types $(\tau_1, \ldots, \tau_n) \in \mathcal{T}^n$, and an atom $P(x_1, \ldots, x_n)$ is formed by applying P to a tuple of terms $(x_1, \ldots, x_n)$ of matching types. If all terms are objects from $\mathcal{O}$, the atom is said to be ground. Actions in $\mathcal{A}$ are defined by a name and a typed list of arguments. Each action is associated with a precondition (a set of atoms required to be true) and an effect (sets of atoms to be added or deleted). We denote the set of actions applicable in a state s as $\mathcal{A}[s]$. The instance I defines the set of objects $\mathcal{O}$, where each object $o \in \mathcal{O}$ is assigned a type $\tau \in \mathcal{T}$, the initial

state s_0, and the goal g, where states are sets of ground atoms. A state s is a goal state if $g \subseteq s$. A plan is a sequence of applicable actions that transforms s_0 into a goal state.

Generalized Planning. A generalized planning problem is a set $\mathcal{Q}$ of planning instances P drawn from the same domain, possibly with different object sets, initial states, and goals. Unlike classical planning, where a solution is typically an open-loop sequence of ground actions, a solution to a generalized problem is a closed-loop policy π that selects actions based on the current state and goals. This policy must work for every instance in $\mathcal{Q}$. Formally, π is correct for $\mathcal{Q}$ if for every $P \in \mathcal{Q}$ the trajectory obtained by repeatedly applying π from the initial state reaches a goal state. Generalized planning thus focuses on compact, reusable control policies that generalize across instances.

3 Iterated Width

Iterated Width (IW) is a search algorithm that prunes states based on their *novelty* [Lipovetzky and Geffner, 2012]. For a state s generated in the search, the novelty $w(s)$ is defined as the size of the smallest tuple of atoms true in s that is false in all previously generated states. If no such tuple exists, the novelty is infinite. IW(k) is a standard Breadth-First Search (BFS) that prunes any state s with novelty $w(s) > k$. Since the number of atom tuples of size k is polynomial in the number of ground atoms, IW(k) explores a polynomial number of states. Specifically, IW(1) runs in time linear in the number of reachable atoms and preserves solvability for problems with width 1.

IW is particularly effective because atomic goals in many classical planning domains exhibit a small width, meaning goals can be reached via sequences of states with persistently low novelty, regardless of problem size. For example, in the Delivery domain, picking up n packages and delivering them requires exactly $2n$ IW(1) jumps, regardless of grid size or package locations, as each subgoal is achievable within two jumps[1]. Consequently, IW provides a guarantee that blind search methods, such as bounded-depth BFS, fundamentally lack.

In this work, we use IW(1) as a local lookahead mechanism to define "jumps" in the state space. An IW(1)-jump from s leads to a state s' that is reachable from s using IW(1). This allows the learned policy to reason only over the endpoints of a subproblem that can be solved efficiently by a low-complexity search, focusing learning on high-level decisions. This approach has been explored previously by Aichmüller and Geffner [2025]. An important benefit of IW(1)-jumps is that they often reduce the expressive requirements for learning the Q-value function. This is because IW(1) can solve certain subproblems that would otherwise require more complex features to represent, thus facilitating the learning of a generalized policy.

Abstracted IW(k). Each n-ary predicate $P(x_1, \ldots, x_n)$ grounds $O(|\mathcal{O}|^n)$ many atoms. Thus, although IW(1) is linear in the number of atoms, the number of atoms itself scales polynomially with the object count. This bottleneck materializes in very large instances and cannot be resolved through straightforward parallelization alone. To avoid this polynomial scaling, we relax width-1 novelty by modifying the novelty calculation itself.

For an n-ary ground atom $P(o_1, \ldots, o_n)$ *Abstracted IW(1)* (AIW(1)) iterates over each position i to compress the information content of all but position i's object slots via a reduction function $r(o)$. The original atom is considered novel if any of its abstracted atoms $P(r(o_1), \ldots, r(o_{i-1}), o_i, r(o_{i+1}), \ldots, r(o_n))$ is novel. In this framework, regular IW(1) corresponds to object identity $r(o) = o$, whereas AIW(1) replaces objects by their type in the planning domain, i.e., $r(o) = \texttt{type}(o)$. In domains without a typing hierarchy, AIW treats every object as belonging to the same universal type.

With this modification, the novel atom count is bounded by $O(t^{m-1}|\mathcal{O}|)$, where t is the number of object types and m is the maximum predicate arity. Since both t and m depend only on the domain, the number of novel atoms scales linearly in the problem's object count. However, goal atoms may require full object identities to be preserved to be safely detected in the lookahead. AIW therefore does not abstract an atom $P(o_1, \ldots, o_n)$ if it is mentioned in the goal.

The idea of AIW(1) extends naturally to a general width-k formulation. Let $\ell = (q_i)_{i \in I}$ be a canonically ordered atom-tuple of size $|I| \leq k$, and let $\mathcal{F}(q_i)$ denote the complete set of abstracted

[1]Width-2 goals cannot always be decomposed into width-1 subgoals solvable by IW(1) [Bonet and Geffner, 2024].

forms of atom q_i as described above. Then ℓ has *abstracted novelty at most* k if there exists a non-empty index set $J \subseteq I$ such that the abstracted tuple $(a_j)_{j \in J}$ with $a_j \in \mathcal{F}(q_j)$ has not been seen before in the search. Equivalently, AIW(k) replaces the original subtuples of ℓ considered by standard IW(k) novelty with all abstracted tuples induced by ℓ, and marks ℓ novel if any abstracted tuple is.

AIW(1) records which objects occur in which argument position, but not how objects in different positions are paired. For example, in Spanner the atom `at(bob, loc3)` is tested through `at(bob, location)` and `at(man, loc3)`. The first is always true at the root state, because `at(bob, loc0)` holds, so the atom is novel exactly when no *man*-typed object has been at `loc3` before. A state that is novel for IW(1) can thus be pruned by AIW(1), and AIW(k) generally loses the completeness guarantee of IW(k) for problems of width k. As the lookahead is layered on an IW(0) expansion of the root, however, every primitive successor is always offered to the policy, so abstraction only removes *additional* jumps: what is lost is the guarantee that a width-1 subgoal is reached within one jump, not the reachability of the goal. In exchange, the number of states the lookahead retains and expands is at most linear in the object count, although the per-state branching factor is not reduced. Unless stated otherwise in the remainder, IW and AIW refer to IW(1) and AIW(1), respectively[2].

4 Relational Graph Neural Networks

While standard Graph Neural Networks (GNNs) operate on graphs, planning states are relational structures involving predicates of varying arities. Relational GNNs (R-GNNs) as used in Planning [Ståhlberg et al., 2022a] adapt standard message-passing GNNs to such structures by treating ground atoms as message sources that communicate information to the objects they involve.

Concretely, each object $o \in \mathcal{O}$ has an embedding $\mathrm{f}_i(o) \in \mathbb{R}^k$ at iteration i, initialized to $\mathrm{f}_0(o) = \mathbf{0}$. An R-GNN takes as input a set of atoms, commonly a state s extended with the goal atoms, and produces final embeddings $\mathrm{f}_L(o)$, where L is the number of R-GNN layers. These final embeddings are then used to define the value or Q-functions that implicitly encode the policies.

Let $\mathcal{R}$ be a set of ground atoms. Then the embeddings are updated by message aggregation and updates for a fixed number of iterations as:

$$\mathrm{f}_{i+1}(o) \;=\; \mathrm{f}_i(o) \;+\; \mathrm{comb}_U\big(\mathrm{f}_i(o),\, \mathrm{m}_o\big),$$

where m_o is the aggregated message received by o from all atoms that mention it. Each ground atom $q = P(o_1, \ldots, o_n) \in \mathcal{R}$ computes a set of position-wise messages $\{\mathrm{m}_{q,o_j}\}_{j=1}^{n}$ by applying a predicate-specific function comb_P to the embeddings of the participating objects:

$$\big(\mathrm{m}_{q,o_1}, \ldots, \mathrm{m}_{q,o_n}\big) \;=\; \mathrm{comb}_P\big(\mathrm{f}_i(o_1), \ldots, \mathrm{f}_i(o_n)\big).$$

In other words, comb_P maps the tuple of object embeddings for atom q to a tuple of message vectors, one per argument position. The aggregated message for object o collects all such position-wise messages over atoms that contain o:

$$\mathrm{m}_o \;=\; \mathrm{agg}\big(\{\, \mathrm{m}_{q,o} \;:\; q \in \mathcal{R},\; o \in q \,\}\big).$$

Here, it is important that $\mathrm{agg}(\cdot)$ is a permutation-invariant function (e.g., maximization or summation).

We use the implementation provided by Ståhlberg and Geffner (2026), which uses a smooth maximum (log-sum-exp) as the aggregation function and multi-layer perceptrons (MLPs) as the combination functions. Each MLP consists of two hidden layers with Mish activations [Misra, 2020].

Expressivity. The expressive power of GNNs is tightly linked to the Weisfeiler-Lehman (WL) test [Morris et al., 2019, Xu et al., 2019] and C_2 logic [Cai et al., 1992, Barceló et al., 2020, Grohe, 2021]. This expressivity can be increased by defining vertices as k-tuples, yielding k-GNNs that match the power of k-WL and C_k logic for $k > 2$. However, these more expressive architectures are data-hungry and difficult to train. While architectures with C_3 power appear sufficient for most planning benchmarks [Drexler et al., 2024], recent work suggests that theoretical expressiveness does not always translate to performance. For instance, Edge Transformers, which possess C_3 power [Bergen et al., 2021, Müller et al., 2024], often underperform in non-C_2 domains compared to more efficient architectures that do not aim to capture the full expressivity of C_3 [Ståhlberg et al., 2025].

[2]We also considered a capacity-limited variant (C-AIW) that restricts the retained states per depth, and a version that forces object abstraction to a universal type in Base-Abstracted IW (BAIW). Details and results on these variants are in the appendix.

5 Transition Encodings in R-GNNs

Precisely how states and transitions are encoded as inputs to R-GNNs affects the expressiveness of the resulting architecture. Ståhlberg et al. (2025) showed that the expressiveness of R-GNNs can go beyond C_2 simply by changing the input encoding, and successfully learn general policies for domains that require C_3 expressivity. However, these encodings are not practical for large states as all possible object pairs need to be part of the input. Alternative encodings and their impact on expressiveness have also been studied by Horcík and Sír (2024) and Horcík et al. (2025).

Formally, the R-GNN input consists of a set of ground atoms $\mathcal{R}$ that describe the current state, the goal, and optionally one or more successor states or actions. Different choices for $\mathcal{R}$ yield different encodings and different readout modalities. In our approach we adopt the current state s encoding from Ståhlberg et al. [2022b]. Let $\mathcal{O}$ be the set of objects in s, and let $\bar{o}$ denote a tuple of objects. The atoms representing s are given by:

$$\mathcal{R}^s = \{P(\bar{o}) \mid P(\bar{o}) \in s\} \cup \{P_{G,T}(\bar{o}) \mid P(\bar{o}) \in g \cap s\} \cup \{P_{G,F}(\bar{o}) \mid P(\bar{o}) \in g \setminus s\}.$$

Goal predicates $P_{G,T}$ and $P_{G,F}$ are included to distinguish goal atoms that are true in s from those that are false. This distinction is crucial, as R-GNNs may lack the expressiveness to infer these for predicates of arity greater than one [Francès et al., 2021, Chen et al., 2024a, Drexler et al., 2024].

External Encoding. This encoding serves as a baseline, first used by Ståhlberg et al. [2023] and later adapted for IW-policies by Aichmüller and Geffner [2025]. It processes the current state s and the successor state s' separately, combining them only at the readout stage to produce Q-values. Specifically, the R-GNN generates embeddings for $\mathcal{R}^s$ and $\mathcal{R}^{s'}$ independently. Formally, let $\mathrm{f}(o; \mathcal{R}^s)$ and $\mathrm{f}(o; \mathcal{R}^{s'})$ be the embeddings of object o after the final iteration using $\mathcal{R}^s$ and $\mathcal{R}^{s'}$, respectively. The Q-value for a transition to s' is computed as:

$$Q(s, s') \;=\; \mathrm{MLP}_1\left(\sum\nolimits_{o \in \mathcal{O}} \mathrm{MLP}_2\left(\mathrm{f}(o; \mathcal{R}^s), \mathrm{f}(o; \mathcal{R}^{s'})\right)\right),$$

where MLP_1 and MLP_2 are multi-layer perceptrons. Crucially, since the R-GNN processes states independently, it cannot compute differences during message passing. The inner MLP combines the embeddings of each object from both states, marking their first interaction.

Aggregated-Actions (AA) Encoding. This second baseline, presented by Ståhlberg and Geffner (2026), represents successor states via ground actions. Consequently, this encoding applies only to learning flat policies that select primitive actions. Formally, for a current state s, the input is:

$$\mathcal{R}^{\mathrm{A}} \;=\; \mathcal{R}^s \cup \{P_A(o_a, \bar{o}) \mid a \in \mathcal{A}[s], a = A(\bar{o})\},$$

where o_a is a new object representing ground action a. Here, A is the name of the action schema. The Q-value for ground action a is computed as:

$$Q(s, a) \;=\; \mathrm{MLP}\left(\mathrm{f}(o_a; \mathcal{R}^{\mathrm{A}}), \sum\nolimits_{o \in \mathcal{O}} \mathrm{f}(o; \mathcal{R}^{\mathrm{A}})\right),$$

where $\mathrm{f}(o_a; \mathcal{R}^{\mathrm{A}})$ is the final embedding of the action object o_a, and $\mathrm{f}(o; \mathcal{R}^{\mathrm{A}})$ are the final embeddings of the original objects. Since the effects of ground action a are not explicitly represented, the R-GNN must infer them from the action name and arguments.

Aggregated-Delta (AD) Encoding. In our proposed encoding scheme of Aggregated-Delta, we move from representing each transition $s \to s'$ individually to instead representing the entire tree jointly. Let $\mathcal{T} = (V, E)$ represent an IW (or its variants) directed lookahead tree from root state s.

We represent the root state implicitly via the relation set $\mathcal{R}^s$ again. However, to compactly represent successors we introduce an explicit state object-node $o_{s'}$ for each $s' \in V \setminus \{s\}$ and encode only its added predicates $P^+(o_{s'}, \bar{o})$ for atoms in $s' \setminus s$ and deleted predicates $P^-(o_{s'}, \bar{o})$ for atoms in $s \setminus s'$. Crucially, the added and deleted predicates are anchored to their state object $o_{s'}$. This allows the R-GNN to compute embeddings for each state object $o_{s'}$ based on its differences from the root.

For each edge $(s', s'') \in E(\mathcal{T})$ a new atom $P_{\mathrm{E}}(o_{s'}, o_{s''})$ representing the transition is added that allows the successors to exchange information directly, instead of only indirectly via the problem's object nodes. Another benefit of encoding the tree as a whole is the option to represent relative

search depth of successors in a structural way. To this end, we embed depth nodes o_d for each depth $d \leq d_{\max}(\mathcal{T})$ and imply an order through auxiliary atoms $P_{\mathrm{D}}(o_d, o_{d'})$ whenever $d < d'$. Depth-information flows back into state-nodes via state-depth association atoms $P_{\mathrm{SD}}(o_{s'}, o_d)$ for each successor s' at depth d. The complete set of atoms added to the input is then:

$$
\begin{aligned}
\mathcal{R}^{\mathcal{T}} &= \mathcal{R}^{s} \cup \mathcal{R}^{\mathcal{T}}_{E} \cup \mathcal{R}^{\mathcal{T}}_{D} \cup \bigcup_{s' \in V(\mathcal{T}) \setminus \{s\}} \mathcal{R}^{\mathcal{T}}_{\Delta s'}, \text{ where} \\
\mathcal{R}^{\mathcal{T}}_{E} &= \{P_E(o_{s'}, o_{s''}) \mid (s', s'') \in E(\mathcal{T})\}, \\
\mathcal{R}^{\mathcal{T}}_{D} &= \{P_D(o_d, o_{d'}) \mid d < d'\} \cup \{P_{\mathrm{SD}}(o_{s'}, o_d) \mid s' \in V(\mathcal{T}) \text{ at depth } d\}, \\
\mathcal{R}^{\mathcal{T}}_{\Delta s'} &= \{P^{+}(o_{s'}, \bar{o}) \mid P(\bar{o}) \in s' \setminus s\} \ \cup \ \{P^{-}(o_{s'}, \bar{o}) \mid P(\bar{o}) \in s \setminus s'\} \\
&\quad \cup \{P^{+}_{G}(o_{s'}, \bar{o}) \mid P(\bar{o}) \in g \cap (s' \setminus s)\} \ \cup \ \{P^{-}_{G}(o_{s'}, \bar{o}) \mid P(\bar{o}) \in g \cap (s \setminus s')\}.
\end{aligned}
$$

Finally, the Q-value to state $s' \in V(\mathcal{T})$ is computed similarly as in AA:

$$
Q(s, s') \ = \ \text{MLP}\left(\mathrm{f}(o_{s'}; \mathcal{R}^{\mathcal{T}}), \sum\nolimits_{o \in \mathcal{O}} \mathrm{f}(o; \mathcal{R}^{\mathcal{T}})\right).
$$

Including both the state object embedding $\mathrm{f}(o_{s'}; \mathcal{R}^{\mathcal{T}})$ and the aggregated original object embeddings $\sum_{o \in \mathcal{O}} \mathrm{f}(o; \mathcal{R}^{\mathcal{T}})$ is vital to provide context about s when evaluating the jump to s'. Although using only the first term is possible, representing both states with a single embedding proves difficult in practice. During readout, we compute Q-values for all states $s' \in V(\mathcal{T})$ in a single batch.

AD improves resource usage in two ways. First, root-state embedding reuse reduces the number of required embeddings and the associated memory footprint by a factor of $|V(\mathcal{T})|$. Second, representing states through atom changes avoids message and gradient creation proportional to the number of atoms. As a result, IW-policy models that previously exceeded 24GB of VRAM become trainable with at most 4GB.

Between External and AD encodings, two meaningful interpolations are possible. The first, *internal*, still encodes each transition $s \to s'$ separately, but represents the transition in a single graph over shared object nodes. It uses the full state description $\mathcal{R}^{s}_{\mathcal{P}}$ for s and a duplicated predicate set $\mathcal{R}^{s'}_{\mathcal{P}'}$ for s'. The second, *internal-delta*, also encodes each transition separately, but represents s' by delta predicates $\mathcal{R}^{s}_{\Delta s'}$ without an explicit $o_{s'}$ state-node, while s remains encoded through $\mathcal{R}^{s}$. Neither interpolation matches AD performance, since both lack tree-level communication between successor nodes. Still, both serve as ablations for the benefits of compact internalized inputs. Detailed results are provided in the appendix, which also illustrates all encodings on a small Blocksworld instance (Appendix D) and traces AD through a Spanner example (Appendix E).

6 Learning Generalized Policies

The R-GNN is trained using deep Q-learning [Mnih et al., 2015], following the approach of Ståhlberg and Geffner (2026), which uses a lifted variant of hindsight relabeling. The objective is to learn a Q-value function $Q(s, s')$ that estimates the expected cumulative reward of performing a transition from state s to successor state s'. In this paper, the reward is always defined as -1. Thus, the Q-value approximately represents the negative expected number of transitions to reach the goal from state s.

Additionally, AD encodings introduced depth information of successors as inductive bias which we enforce via an auxiliary loss. This is beneficial whenever the distance of IW-jumps is crucial in an environment. For example, in the Spanner domain, the agent can only move forward and collect items on location, meaning bypassed items cannot be retrieved later. If the agent jumps to pick up the farthest spanner, it often guarantees a dead-end state. This loss ensures that distance information is properly learned, helping the agent acquire a strategy that avoids failure. Depth nodes act as a relational counterpart of positional encodings: rather than attaching depth features to state nodes, they place relative search depth and its order as individual nodes and relations, so that depth participates in message passing. Since absolute label classification or value regression cannot generalize to values unseen during training, we enforce depth order with the pairwise ranking loss of Burges et al. [2005] over (o_k, o_l) pairs with $k < l$. To this end, a small probe matrix W_θ computes a score $z_d = W_\theta f(o_d)$ from each final depth-node embedding. Then, letting $w_{k,l} = 1/(l-k)$ and $Z = \sum_{k<l\leq d_{\max}} w_{k,l}$ each such depth-node pair is compared in a rank-loss $\mathcal{L}_{\mathrm{d-rank}} = \frac{1}{Z} \sum_{k<l\leq d_{\max}} w_{k,l} \log\left(1 + \exp(z_k - z_l)\right)$

Domain	**AIW–AD**	IW–AD	AIW–Ext	IW–Ext	Flat–AA	LAMA*	Distincter*	GOOSE*
blocksworld (90)	**90 (100)**	65 (72)	78 (87)	59 (66)	76 (84)	57 (63)	88 (98)	75 (83)
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	n/a	n/a
medium (30)	**30 (100)**	**30 (100)**	**30 (100)**	29 (97)	**30 (100)**	24 (80)	n/a	n/a
hard (30)	**30 (100)**	5 (17)	18 (60)	0 (0)	16 (53)	3 (10)	n/a	n/a
miconic (90)	**90 (100)**	**90 (100)**	72 (80)	72 (80)	84 (93)	**90 (100)**	**90 (100)**	**90 (100)**
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	n/a	n/a
medium (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	n/a	n/a
hard (30)	**30 (100)**	**30 (100)**	12 (40)	12 (40)	24 (80)	**30 (100)**	n/a	n/a
spanner (90)	**90 (100)**	61 (68)	32 (36)	55 (61)	**90 (100)**	30 (33)	**90 (100)**	73 (81)
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	27 (90)	**30 (100)**	**30 (100)**	n/a	n/a
medium (30)	**30 (100)**	**30 (100)**	2 (7)	27 (90)	**30 (100)**	0 (0)	n/a	n/a
hard (30)	**30 (100)**	1 (3)	0 (0)	1 (3)	**30 (100)**	0 (0)	n/a	n/a
satellite (90)	88 (98)	55 (61)	61 (68)	56 (62)	51 (57)	**90 (100)**	48 (53)	53 (59)
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	n/a	n/a
medium (30)	**30 (100)**	25 (83)	**30 (100)**	26 (87)	21 (70)	**30 (100)**	n/a	n/a
hard (30)	28 (93)	0 (0)	1 (3)	0 (0)	0 (0)	**30 (100)**	n/a	n/a
transport (90)	**86 (96)**	74 (82)	55 (61)	66 (73)	74 (82)	72 (80)	50 (56)	29 (32)
easy (30)	**30 (100)**	**30 (100)**	29 (97)	**30 (100)**	29 (97)	**30 (100)**	n/a	n/a
medium (30)	29 (97)	**30 (100)**	19 (63)	**30 (100)**	25 (83)	**30 (100)**	n/a	n/a
hard (30)	**27 (90)**	14 (47)	7 (23)	6 (20)	20 (67)	12 (40)	n/a	n/a
ferry (90)	80 (89)	76 (84)	71 (79)	67 (74)	62 (69)	74 (82)	**83 (92)**	76 (84)
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	29 (97)	**30 (100)**	n/a	n/a
medium (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	29 (97)	**30 (100)**	n/a	n/a
hard (30)	**20 (67)**	16 (53)	11 (37)	7 (23)	4 (13)	14 (47)	n/a	n/a
rovers (90)	67 (74)	62 (69)	24 (27)	36 (40)	27 (30)	**78 (87)**	42 (47)	37 (41)
easy (30)	**30 (100)**	**30 (100)**	24 (80)	**30 (100)**	27 (90)	**30 (100)**	n/a	n/a
medium (30)	24 (80)	25 (83)	0 (0)	6 (20)	0 (0)	**30 (100)**	n/a	n/a
hard (30)	13 (43)	7 (23)	0 (0)	0 (0)	0 (0)	**18 (60)**	n/a	n/a
childsnack (90)	**64 (71)**	33 (37)	47 (52)	37 (41)	41 (46)	36 (40)	**64 (71)**	29 (32)
easy (30)	**30 (100)**	28 (93)	**30 (100)**	**30 (100)**	27 (90)	**30 (100)**	n/a	n/a
medium (30)	**30 (100)**	5 (17)	17 (57)	7 (23)	12 (40)	5 (17)	n/a	n/a
hard (30)	**4 (13)**	0 (0)	0 (0)	0 (0)	2 (7)	1 (3)	n/a	n/a
sokoban (90)	13 (14)	13 (14)	17 (19)	15 (17)	14 (16)	**44 (49)**	32 (36)	38 (42)
easy (30)	13 (43)	13 (43)	17 (57)	15 (50)	14 (47)	**30 (100)**	n/a	n/a
medium (30)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	**14 (47)**	n/a	n/a
hard (30)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	n/a	n/a
floortile (90)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	**15 (17)**	2 (2)	2 (2)
easy (30)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	**15 (50)**	n/a	n/a
medium (30)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	n/a	n/a
hard (30)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	n/a	n/a
Total (900)	**668 (74)**	529 (59)	457 (51)	463 (51)	519 (58)	586 (65)	589 (65)	502 (56)

Table 1: IPC 2023 benchmark coverage results for AIW with AD encoding (highlighted in bold) against ablations and baselines. Methods marked * perform exponential search at test time. Numbers in parentheses are percentages of row totals.

where $d_{\max}$ is the maximum realized depth in that sample's IW tree. This enforces deeper nodes to receive larger scores relative to shallower ones in a generalizing manner. Stochastic gradient descent updates the R-GNN parameters to minimize the aggregated loss over sampled mini-batches.

During training, exploration is encouraged using a Boltzmann policy, where the probability of selecting a successor state s' from state s is given by $P(s' \mid s) \propto \exp(Q(s, s')/\tau)$, where τ is the temperature parameter that controls the exploration-exploitation trade-off.

7 Experiments

We detail the experimental benchmarks, training and testing setup, and discuss the results.

Domains and Data. Before diving into the specifics, it is important to contextualize these benchmarks, as our evaluation differs from standard practices in deep learning and RL. First, our objective is strictly binary goal-reaching from an initial state in a goal-conditioned RL setting with extremely sparse rewards; unlike typical RL setups, there is no accumulated reward signal or partial degree of success. Second, we focus on zero-shot structural generalization: models are trained on small instances and tested on significantly larger ones, often featuring more than 10–20 times the number of objects seen during training and an exponentially larger state space. For example, in the Blocksworld domain, training is performed on instances with up to 20 blocks and testing on instances with up

to 488 blocks. In contrast, common deep RL protocols train and test on environments of similar scale, a practice criticized by Cobbe et al. [2019]. Finally, we evaluate strictly greedy, deterministic policies that commit to their jumps without the safety net of backtracking. The policy thus acts as a closed-loop controller that fully commits to every chosen jump, not as a search algorithm.

We evaluate our approach on two benchmark sets: hard-to-learn domains from Drexler et al. [2024] that require more than C_2 logic expressivity, and the IPC Learning Track 2023 [Taitler et al., 2024]. This means that both training and testing have been carried out in instances that are much larger than those used in previous works on general policy learning. Detailed domain descriptions and an overview of the train, validation and test sets are available in the appendix, where we also provide a visualization of the immense scaling factors present in the IPC benchmark.

Training. Our implementation is based on PyTorch and extends the codebase of Ståhlberg and Geffner [2026]. The architecture remains unchanged; methods differ only in their input encoding and hyperparameters.[3] We set the embedding size to 32 and used smooth maximum aggregation. The learning rate decayed linearly from 10^{-3} to 10^{-5} over 300 episodes. Similarly, the Boltzmann temperature decayed linearly from 1.0 to 0.1 over 1000 episodes. The discount factor was set to 0.999. Each episode consisted of 32 optimization steps with a batch size of 32. The replay buffer size was 100 for IW-policies and 1000 for flat-policies. We generated 4 trajectories per episode, each capped at 20 IW jumps or 200 action steps. We trained 5 seeds per method and domain on a cluster with an Intel Xeon Platinum 8352M CPU, 20 GB RAM, and an NVIDIA A10/L40S GPU capped at 24GB, with a time limit of 12 hours. Model selection was based on validation coverage; in case of ties, we prioritized the shortest total solution length, followed by the lowest total TD-error, and finally random selection. We test the best 5 checkpoints among all runs and report the best outcome.

Testing. We evaluate policies on the test set using a greedy strategy over Q-value functions. We consider two types of goal-conditioned policies: flat policies that select primitive actions, and IW-policies that select IW-jumps as described in the background. At each step, the policy selects the successor state $s^* = \arg\max_{s' \in \mathcal{S}} Q(s, s')$ that maximizes the Q-value among all valid successors $\mathcal{S}$. To ensure progress and avoid infinite loops, we maintain a history of visited states within the current episode and exclude any transition to a previously visited state from the candidate successors. The evaluation continues until the goal condition is satisfied ($g \subseteq s$), or until a resource limit is exhausted. We enforce a limit of $1,000$ choices and a timeout of 15 and 60 minutes per instance for flat- and IW-policies, respectively. Appendix E walks through a training and an evaluation step. We test the following methods:

(a) *AIW–AD*: Our full proposed method of combining AIW with AD encoding.

(b) *IW–AD*: IW-policies [Aichmüller and Geffner, 2025] using our proposed AD encoding. This method compares the unidimensional improvement made by more efficient encodings.

(c) *AIW–Ext*: AIW-policies using the baseline's External encoding. This method compares the unidimensional improvement made by the relaxed IW-lookahead variant.

(d) *IW–Ext*: IW-policies using External encodings, the direct baseline.

(e) *Flat–AA*: This baseline uses the AA encoding [Ståhlberg and Geffner, 2026] (see Section 5) and relies entirely on the greedy selection of primitive actions that do not result in a cycle.

(f) *LAMA*: The classical planner LAMA [Richter and Westphal, 2008] which uses a best-first search to first find a plan and then improves it using iterated weighted $A^\star$ search. LAMA was given 1 hour and 90 GB of memory per instance.

(g) *Distincter*: Distincter [Bai et al., 2025] uses Greedy Best-First Search (GBFS) with GNNs that provides both heuristic guidance and approximate symmetry detection. To prioritize speed, the GNNs rely on a shallow 3–4 layer architecture; however, this restricted expressivity risks falsely identifying distinct states or actions as symmetric.

(h) *GOOSE*: WL-GOOSE [Chen et al., 2024a] diverges from deep learning approaches by using the WL algorithm alongside classical machine learning models to learn search heuristics from scratch. Rather than relying on GNNs, it colors a graph (specifically, the ILG) and uses the resulting color counts to define its heuristic. This strikes a good balance between computational efficiency and heuristic quality, resulting in competitive performance when used with GBFS.

[3] Code and data: `https://github.com/maichmueller/aggregated-delta`, doi:10.5281/zenodo.24007972.

Domain	**AIW–AD**	IW–AD	AIW–Ext	IW–Ext	Flat–AA	LAMA
depots (86)	**86 (100)**	**86 (100)**	83 (97)	81 (94)	76 (88)	85 (99)
delivery (80)	**80 (100)**	**80 (100)**	**80 (100)**	**80 (100)**	25 (31)	79 (99)
driverlog (123)	120 (98)	**123 (100)**	116 (94)	**123 (100)**	122 (99)	**123 (100)**
goldminer (359)	**359 (100)**	**359 (100)**	133 (37)	69 (19)	132 (37)	222 (62)
grippers (312)	**312 (100)**	**312 (100)**	**312 (100)**	**312 (100)**	**312 (100)**	**312 (100)**
freecell (117)	**117 (100)**	**117 (100)**	**117 (100)**	**117 (100)**	**117 (100)**	**117 (100)**
visitall (100)	**100 (100)**	**100 (100)**	**100 (100)**	**100 (100)**	97 (97)	**100 (100)**
reward (99)	**99 (100)**	**99 (100)**	**99 (100)**	**99 (100)**	97 (98)	**99 (100)**
grid* (100)	**100 (100)**	99 (99)	**100 (100)**	96 (96)	27 (27)	**100 (100)**
logistics* (100)	97 (97)	98 (98)	0 (0)	7 (7)	25 (25)	**100 (100)**
satellite* (100)	**100 (100)**	**100 (100)**	26 (26)	29 (29)	73 (73)	**100 (100)**
Total (1576)	1570 (100)	**1573 (100)**	1166 (74)	1113 (71)	1103 (70)	1437 (91)

Table 2: Coverage results on domains outside the IPC 2023 benchmark for AIW–AD against ablations and baselines. Domains marked * are known to require logic beyond C_2. Numbers in parentheses are percentages of row totals.

We also tested BAIW, C-AIW, and Flat policies using AD encoding in the appendix. We note that all (A)IW and flat policy methods – baseline or not – are trained via the described learning method in this paper, including the usage of lifted HER [Ståhlberg and Geffner, 2026]. Technically, extending the IW(1) policies of Aichmüller and Geffner [2025] with lifted HER is also first presented in this paper, yet we consider this straightforward. The results for Distincter and WL-GOOSE – trained with supervised learning – are taken directly from their original papers, which used a timeout of 30 minutes and 8 GB of memory.

Results. Table 1 reports coverage on the IPC 2023 benchmark. In summary, AIW–AD yields the strongest overall coverage among all methods and improves substantially over every baseline. It also surpasses the classical planning baseline LAMA by a wide margin. To our knowledge, the only previous method to exceed LAMA on this benchmark is Bai et al. [2025], and only by a small margin. Here, the gain over LAMA is obtained *without resorting to an exponential search*: AIW runs in time linear in the number of objects and is invoked only a bounded number of times in sequence.

The main remaining failure cases are *Sokoban* and *Floortile*, where only search-based methods solve a non-trivial subset of instances (LAMA: $44/90$ and $15/90$), although no evaluated method performs convincingly on either domain.

Outside these puzzle domains, performance is consistently strong. In nearly all other domains, AIW–AD solves all easy and medium instances, and most hard instances, with the main remaining gaps concentrated in Rovers and Childsnack. Childsnack exposes a scaling regime that is prohibitive for methods that choose discriminatively from an explicitly enumerated set of options. On the hardest instances, the branching factor reaches tens of millions of applicable actions. Mere action enumeration requires up to 10 minutes, despite highly optimized software and latest hardware, rendering decision-making from fully enumerated options unlikely to scale under reasonable budgets. C-AIW solves six additional hard instances and raises coverage in this domain to $70/90$, the best result to date, to the best of our knowledge. Yet, further progress on Childsnack will likely require models that opt to generate decisions compositionally rather than select them from an enumerated set.

These results show that abstracted width-based lookaheads improve scalability, while also preserving a useful subgoal structure that learned models can exploit. Previous work showed that IW lookaheads can simplify the cost structure and lower the expressive requirements for learned policies [Aichmüller and Geffner, 2025]. Our results show that this effect persists at scale and also under abstracted width-1 lookaheads, despite more aggressive pruning. This is most evident in Satellite and Rovers, two domains known to require features beyond the C_2 logic fragment [Drexler et al., 2024, Ståhlberg et al., 2022a]. Satellite is virtually solved by AIW–AD. On Rovers, some hard instances remain unresolved, but the gains over previous learned approaches are substantial. Moreover, on both domains AD-encoded lookaheads improve over primitive successors, and AIW–AD performs best among the learned policies, underscoring the value of principled lookaheads.

Table 2 extends the comparison to eleven domains outside the IPC 2023 benchmark, including Grid, Logistics, and Satellite, which also lie beyond C_2 logic. AIW–AD and IW–AD solve 1570 and 1573 of the 1576 test instances, ahead of LAMA (1437) and both External variants (1166 and 1113). As these test instances scale more moderately, IW(1) remains affordable, and the encoding rather than

the abstraction makes the difference: on Goldminer, Logistics, and Satellite, AD-encoded policies solve $359/359$, at least $97/100$, and $100/100$ instances, whereas External-encoded policies solve at most 133, 7, and 29. More aggressive abstraction can still fail here: BAIW reaches only 48% test coverage in Logistics due to pruning too aggressively (Table 4), while it still solves Grid in full. In terms of quality, our plans are also on par with LAMA's in aggregate (Appendix G), despite not being optimized for length, but for subgoals (jumps). Together, these results show that AIW–AD is not only effective on the IPC 2023 benchmark, but also on a vast set of structurally diverse domains.

8 Limitations

Failure modes and their remedies span four different categories: (i) AIW can prune useful states when object pairings are lost; retaining $h > 1$ exact arguments in predicates would interpolate between abstract ($h = 1$) and regular novelty ($h = \infty$). (ii) The R-GNN can tie useful and harmful candidates, which requires richer encodings or more expressive architectures [Ståhlberg et al., 2025]. (iii) Greedy selection can commit to dead ends which only (bounded) backtracking or beam search could improve. (iv) Limits of enumeration-based policies due to instance size as in Childsnack, where merely enumerating the applicable actions itself becomes prohibitive (Section 7).

We expect Floortile and Sokoban to mainly reflect (iii), although this remains a hypothesis. Floortile is no failure of symbolic planning per se, as LAMA solves $15/90$. Painting a tile blocks later movement, so success requires precise action sequencing. Failed trajectories with HER goals yield few useful training signals to actually learn such sequencing. Thus, expressivity is not to blame; supervision during training that separates safe from unsafe transitions is required to mitigate it. Sokoban similarly features irreversible pushes, yet is also PSPACE-complete [Culberson, 1997]. Whether a polynomial, generalizing policy exists for this domain's benchmark-set is an open question.

Although AIW gives up the width-1 completeness of IW(1) (Section 3), the lookaheads show no meaningful loss from it in the experiments. However, stronger relaxations do lose solutions: C-AIW drops coverage in Satellite-hard and Rovers-hard from 28 to 3 and from 13 to 2 respectively (Table 3), and BAIW solves only 48% of Logistics.

Lastly, our policies are learned per domain from relational (PDDL) descriptions; changing predicates or action schemas requires retraining. Transfer across domains [Chen et al., 2024b] is an objective out of scope for this paper.

9 Conclusions

We introduced two main contributions. First, we proposed Aggregated-Delta (AD) encodings for R-GNNs, which represent transitions through state differences and combine all lookahead successors into a single relational graph. This enables IW-based successors to be evaluated in one forward pass, greatly reducing memory and runtime costs and removing the main scalability bottleneck of previous IW-policy formulations. Second, we introduced Abstracted IW (AIW), a stricter IW variant that reduces the novelty-based search space by modifying novelty calculation itself.

Together, these contributions yield a new state of the art in generalized planning. On the IPC 2023 benchmark, AIW combined with AD significantly outperforms all previous baselines, including the strong baseline LAMA. The key point is not only that the learned policy improves, but also that AD makes larger lookaheads practical, especially with AIW, whose novelty search scales with object counts rather than atom counts. At the same time, the results show that the stronger abstractions used by AIW still preserve meaningful subgoals that learned models can exploit. On Grid and Logistics, the AD encoding accounts for most of the gains. The gains on Satellite and Rovers – domains that require beyond-C_2 expressiveness – suggest that AIW reduces the expressive burden on the architecture in two ways. First, AIW induces a simpler cost structure by replacing sequences of primitive decisions with jump endpoints. Second, because intermediate states inside such jumps are handled by the lookahead rather than by the network, the learned policy can avoid distinctions that would otherwise require features beyond C_2. Together, these effects show that efficient width-based lookahead enables general policies to scale not only by reducing search cost, but also by reducing what the learned architecture must represent.

Acknowledgments and Disclosure of Funding

We thank Jonas Gösgens and Niklas Jansen for their helpful discussions and insights during the development of this work. The research has been supported by the Alexander von Humboldt Foundation with funds from the Federal Ministry for Education and Research, Germany, and the Wallenberg AI, Autonomous Systems and Software Program (WASP). This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 885107). This project was also funded by the German Federal Ministry of Education and Research (BMBF), by the Knut and Alice Wallenberg Foundation, and the Ministry of Culture and Science of the German State of North Rhine-Westphalia (MKW) under the Excellence Strategy of the Federal Government and the Länder. Additional computational resources were provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), funded by the Swedish Research Council. The authors declare no competing interests.

References

Michael Aichmüller and Hector Geffner. Sketch decompositions for classical planning via deep reinforcement learning. In *Proceedings of the 34th International Joint Conference on Artificial Intelligence (IJCAI 2025)*, pages 8438–8446, 2025.

Yingbin Bai, Sylvie Thiébaux, and Felipe Trevizan. Learning efficiency meets symmetry breaking. *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS 2025)*, 35(1):154–159, 2025.

Aniket Nick Bajpai, Sankalp Garg, et al. Transfer of deep reactive policies for MDP planning. In *Proceedings of the 32nd Annual Conference on Neural Information Processing Systems (NeurIPS 2018)*, pages 10965–10975, 2018.

Pablo Barceló, Egor Kostylev, Mikaël Monet, Jorge Pérez, Juan Reutter, and Juan-Pablo Silva. The logical expressiveness of graph neural networks. In *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*, 2020.

Leon Bergen, Timothy J. O'Donnell, and Dzmitry Bahdanau. Systematic generalization with edge transformers. In *Proceedings of the 35th Annual Conference on Neural Information Processing Systems (NeurIPS 2021)*, pages 1390–1402, 2021.

Blai Bonet and Hector Geffner. General policies, subgoal structure, and planning width. *Journal of Artificial Intelligence Research*, 80:475–516, 2024.

Craig Boutilier, Raymond Reiter, and Bob Price. Symbolic dynamic programming for first-order MDPs. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 690–700, 2001.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005)*, pages 89–96, 2005.

Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

Dillon Z. Chen, Felipe W. Trevizan, and Sylvie Thiébaux. Return to tradition: Learning reliable heuristics with classical machine learning. In *Proceedings of the 34th International Conference on Automated Planning and Scheduling (ICAPS 2024)*, pages 68–76, 2024a.

Dillon Ze Chen, Sylvie Thiébaux, and Felipe W. Trevizan. Learning domain-independent heuristics for grounded and lifted planning. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI 2024)*, pages 20078–20086, 2024b.

Karl Cobbe, Oleg Klimov, Christopher Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, pages 1282–1289, 2019.

Joseph C. Culberson. Sokoban is PSPACE-complete. Technical Report TR 97-02, Department of Computing Science, The University of Alberta, Edmonton, Alberta, Canada, 1997.

Dominik Drexler, Simon Ståhlberg, Blai Bonet, and Hector Geffner. Symmetries and expressive requirements for learning general policies. In *Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning (KR 2024)*, 2024.

Alan Fern, SungWook Yoon, and Robert Givan. Approximate policy iteration with a policy language bias: Solving relational Markov decision processes. *Journal of Artificial Intelligence Research*, 25: 75–118, 2006.

Guillem Francès, Blai Bonet, and Hector Geffner. Learning general planning policies from small examples without supervision. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI 2021)*, pages 11801–11808, 2021.

Martin Grohe. The logic of graph neural networks. In *Proc. LICS*, pages 1–17, 2021.

Rostislav Horcík and Gustav Sír. Expressiveness of graph neural networks in planning domains. In *Proceedings of the 34th International Conference on Automated Planning and Scheduling (ICAPS 2024)*, 2024.

Rostislav Horcík, Gustav Sír, Vítezslav Simek, and Tomás Pevný. State encodings for GNN-based lifted planners. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence (AAAI 2025)*, pages 26525–26533, 2025.

Yuxiao Hu and Giuseppe De Giacomo. Generalized planning: Synthesizing plans that work for multiple environments. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, pages 918–923, 2011.

Sergio Jiménez, Javier Segovia-Aguas, and Anders Jonsson. A review of generalized planning. *The Knowledge Engineering Review*, 34:e5, 2019.

Roni Khardon. Learning action strategies for planning domains. *Artificial Intelligence*, 113:125–148, 1999.

Nir Lipovetzky and Hector Geffner. Width and serialization of classical planning problems. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI 2012)*, pages 540–545, 2012.

Mario Martín and Hector Geffner. Learning generalized policies from planning examples using concept languages. *Applied Intelligence*, 20(1):9–19, 2004.

Diganta Misra. Mish: A self regularized non-monotonic activation function. In *Proceedings of the 31st British Machine Vision Conference (BMVC 2020)*. BMVA Press, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Kirkeby Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI 2019)*, pages 4602–4609, 2019.

Luis Müller, Daniel Kusuma, Blai Bonet, and Christopher Morris. Towards principled graph transformers. In *Proceedings of the 38th Annual Conference on Neural Information Processing Systems (NeurIPS 2024)*, 2024.

Silvia Richter and Matthias Westphal. The LAMA planner — Using landmark counting in heuristic search. IPC 2008 short papers, `http://ipc.informatik.uni-freiburg.de/Planners`, 2008.

Or Rivlin, Tamir Hazan, and Erez Karpas. Generalized planning with deep reinforcement learning. In *ICAPS 2020 Workshop on Bridging the Gap Between AI Planning and Reinforcement Learning (PRL)*, pages 16–24, 2020.

Nicholas Rossetti, Massimiliano Tummolo, Alfonso Emilio Gerevini, Luca Putelli, Ivan Serina, Mattia Chiari, and Matteo Olivato. Learning general policies for planning through GPT models. In *Proceedings of the 34th International Conference on Automated Planning and Scheduling (ICAPS 2024)*, pages 500–508, 2024.

Scott Sanner and Craig Boutilier. Practical solution techniques for first-order MDPs. *Artificial Intelligence*, 173(5-6):748–788, 2009.

Siddharth Srivastava, Neil Immerman, and Shlomo Zilberstein. A new representation and associated algorithms for generalized planning. *Artificial Intelligence*, 175(2):393–401, 2011.

Simon Ståhlberg and Hector Geffner. First-order representation languages for goal-conditioned RL. In *Proceedings of the 40th AAAI Conference on Artificial Intelligence (AAAI 2026)*, 2026.

Simon Ståhlberg, Blai Bonet, and Hector Geffner. Learning general optimal policies with graph neural networks: Expressive power, transparency, and limits. In *Proceedings of the 32nd International Conference on Automated Planning and Scheduling (ICAPS 2022)*, pages 629–637, 2022a.

Simon Ståhlberg, Blai Bonet, and Hector Geffner. Learning generalized policies without supervision using GNNs. In *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning (KR 2022)*, pages 474–483, 2022b.

Simon Ståhlberg, Blai Bonet, and Hector Geffner. Learning general policies with policy gradient methods. In *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning (KR 2023)*, 2023.

Simon Ståhlberg, Blai Bonet, and Hector Geffner. Learning more expressive general policies for classical planning domains. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence (AAAI 2025)*, pages 26697–26706, 2025.

Ayal Taitler, Ron Alford, Joan Espasa, Gregor Behnke, Daniel Fišer, Michael Gimelfarb, Florian Pommerening, Scott Sanner, Enrico Scala, Dominik Schreiber, Javier Segovia-Aguas, and Jendrik Seipp. The 2023 International Planning Competition. *AI Magazine*, 45(2):280–296, 2024.

Sam Toyer, Sylvie Thiébaux, Felipe Trevizan, and Lexing Xie. ASNets: Deep learning for generalised planning. *Journal of Artificial Intelligence Research*, 68:1–68, 2020.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*, 2019.

A Benchmark Results

We discuss here in more detail the test outcomes on our benchmark reported in Table 2 of Section 7. We can clearly read two observations off the table: (1) AD encodings enhance the R-GNN's expressivity, and (2) width-based lookaheads simplify problem structures.

AIW–AD and IW–AD solve virtually all instances (1570 and 1573 of 1576), including those of the domains requiring logic beyond C_2, where each misses only three instances. No other learned method comes close, showing that AD input encodings directly enhance the model's expressivity. We can observe these facts hold for both IW and AIW in our benchmark distinctly, since its test-time scaling is more moderate and, thus, does not render IW(1) itself inefficient, unlike the IPC 2023 benchmark. On Goldminer, only AD encoded lookaheads are able to produce solutions for the entire set, again surpassing even LAMA. AA-encoded and AD-encoded (see Table 4) flat policies are not able to achieve sufficient solve rates on many domains, reaffirming the observation made by Aichmüller and Geffner [2025] that (A)IW lookaheads simplify problem structure indeed and that this simplification is crucial for solving some domains.

B Base-Abstracted IW and Capacity-Limited Abstracted IW

As mentioned in Section 3, we also considered two modifications of AIW. A fallback variant of AIW in Base-Abstracted IW, which is the backup abstraction whenever a domain exhibits no typing information, and a depth-capacity limiting form in C-AIW. The following will further contextualize the methods and then discuss their results on the IPC 2023 benchmark as found in Table 3 alongside Flat AD.

Base-Abstracted IW (BAIW). Reducing every other object information in an atom $P(o_1, \ldots, o_n)$ to the base *object*-type erases the object's information. Alternatively, BAIW can be seen as a novelty check via iteratively testing each of the n position-derived unary atoms $P_i(o_i)$. As a lookahead BAIW is also generally effective, but can be too coarse when several semantically different object types occur in the same predicate. The Spanner domain provides a simple example for this. The first argument x of its predicate

`at(x - locatable, l - location)`

applies both to the *man*-typed object `bob` and to spanner objects, since both are of a subtype of *locatable*. Under the base abstraction, an atom `at(bob, loc)` is tested through the abstracted atoms `at(bob, *)` and `at(*, loc)`, where `*` denotes the object compressed to the base *object*-type. The first abstracted atom, `at(bob, *)`, is already seen in the root through `at(bob, loc0)`. The second abstracted atom, `at(*, loc)`, can also already be seen in the root, because spanners occupy locations and generate atoms such as `at(spanner1, loc)`. Consequently, moving `bob` to a new location may fail to produce any novel atom. The AIW lookahead would be defunct, i.e., not provide any expansion of the root, if not for the primitive successors guaranteed by a prior IW(0) search, as the logic of IW(1) implies. A circumvention of this is to allow the primitive successors to be marked as novel themselves, to ensure BAIW continues novelty checks from these states, effectively combining IW(0) and BAIW(1) lookaheads in one go. This extension also provided meaningful results in preliminary experiments, but we did not evaluate it further.

AIW prevents this failure mode naturally by retaining the type of compressed objects. Then, `at(bob, loc)` is tested through the abstracted atoms `at(bob, location)` and `at(man, loc)`. A location previously seen only through `at(spanner1, loc)` contributes `at(locatable, loc)`, but not the more specific `at(man, loc)`. Hence, `at(man, loc)` remains novel when `bob` first reaches that location. Retaining types in the abstraction thus preserves enough role information to prevent the lookahead from collapsing, while staying meaningful, but coarser than full IW(1).

Capacity-Limited-AIW (C-AIW). To further account for infeasibly large branching factors, we also tried coupling the search with a layer capacity limit C, so that at any depth d in the BFS we may at most retain C novel successors to expand at depth $d + 1$. Selection of the C candidates happens via a scoring heuristic $h(s')$ that counts the number of satisfied goals in s' and prefers those of higher score, breaking ties on a first-seen basis. We experimented with various values for C, finding $C = 1000$ to work well in experiments. Although Capacity-Limited AIW (C-AIW) further

Domain	AIW	BAIW	C-AIW	Flat AD	Flat–AA
blocksworld (90)	**90 (100)**	89 (99)	**90 (100)**	76 (84)	76 (84)
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
medium (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
hard (30)	**30 (100)**	29 (97)	**30 (100)**	16 (53)	16 (53)
miconic (90)	**90 (100)**	**90 (100)**	**90 (100)**	84 (93)	84 (93)
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
medium (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
hard (30)	**30 (100)**	**30 (100)**	**30 (100)**	24 (80)	24 (80)
spanner (90)	**90 (100)**	**90 (100)**	**90 (100)**	73 (81)	**90 (100)**
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
medium (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
hard (30)	**30 (100)**	**30 (100)**	**30 (100)**	13 (43)	**30 (100)**
satellite (90)	**88 (98)**	87 (97)	63 (70)	55 (61)	51 (57)
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
medium (30)	**30 (100)**	**30 (100)**	**30 (100)**	22 (73)	21 (70)
hard (30)	**28 (93)**	27 (90)	3 (10)	3 (10)	0 (0)
transport (90)	86 (96)	**87 (97)**	73 (81)	74 (82)	74 (82)
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	29 (97)
medium (30)	29 (97)	**30 (100)**	**30 (100)**	25 (83)	25 (83)
hard (30)	**27 (90)**	**27 (90)**	13 (43)	19 (63)	20 (67)
ferry (90)	**80 (89)**	**80 (89)**	**80 (89)**	64 (71)	62 (69)
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	29 (97)
medium (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	29 (97)
hard (30)	**20 (67)**	**20 (67)**	**20 (67)**	4 (13)	4 (13)
rovers (90)	**67 (74)**	59 (66)	51 (57)	43 (48)	27 (30)
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	27 (90)
medium (30)	**24 (80)**	17 (57)	19 (63)	13 (43)	0 (0)
hard (30)	**13 (43)**	12 (40)	2 (7)	0 (0)	0 (0)
childsnack (90)	64 (71)	66 (73)	**70 (78)**	59 (66)	41 (46)
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	27 (90)
medium (30)	**30 (100)**	**30 (100)**	**30 (100)**	29 (97)	12 (40)
hard (30)	4 (13)	6 (20)	**10 (33)**	0 (0)	2 (7)
sokoban (90)	13 (14)	11 (12)	12 (13)	**14 (16)**	**14 (16)**
easy (30)	13 (43)	11 (37)	12 (40)	**14 (47)**	**14 (47)**
medium (30)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
hard (30)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
floortile (90)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
easy (30)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
medium (30)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
hard (30)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
Total (900)	**668 (74)**	659 (73)	619 (69)	542 (60)	519 (58)

Table 3: IPC 2023 benchmark coverage results for AIW with AD encoding compared to BAIW, C-AIW, and Flat AD policies, as well as the baseline Flat–AA (repeated from Table 1). Numbers in parentheses are percentages of row totals.

weakens completeness for width-1 goals, we gain an effective search limiter that reduces successor counts based on a fast and simple heuristic. In general, capacity limitation can lose crucial successors as shown in Table 3 with Satellite, Rovers, and Transport, in which performance drops for 'hard' problems, i.e., the largest ones, where capacity limits actually start ejecting successors. In turn, capacity limits can improve the search when there are many equivalent continuation paths to the goal, e.g., due to action commutativity or argument permutations, as is common in Childsnack. Its result of 70/90 solved instances on Childsnack is the highest reported rate on this domain to date.

C Model and Encoding Ablations

We ablate the main model, i.e., AIW lookaheads coupled with AD encoding, along several structural changes in either encoding, IW lookahead behaviour, or both:

- *No Depthnodes* – omitting depth nodes o_d and associated depth atoms $\mathcal{R}^{\mathcal{T}}_{D}$ from the encoding, resulting in the relation set $\mathcal{R}^{\mathcal{T}} \setminus \mathcal{R}^{\mathcal{T}}_{\mathrm{D}}$.
- *No Tree-Atoms* – omitting the encoding step for the parent relation atoms, resulting in the relation set $\mathcal{R}^{\mathcal{T}} \setminus \mathcal{R}^{\mathcal{T}}_{\mathrm{E}}$.
- *No Depthnodes + No Tree-Atoms* – omitting both depth and tree atoms, resulting in the relation set $\mathcal{R}^{\mathcal{T}} \setminus (\mathcal{R}^{\mathcal{T}}_{D} \cup \mathcal{R}^{\mathcal{T}}_{\mathrm{E}})$.

Domain	AIW	BAIW	C-AIW	Flat AD
depots (86)	**86 (100)**	**86 (100)**	**86 (100)**	8 (9)
delivery (80)	**80 (100)**	**80 (100)**	**80 (100)**	63 (79)
driverlog (123)	120 (98)	122 (99)	**123 (100)**	120 (98)
goldminer (359)	**359 (100)**	**359 (100)**	**359 (100)**	155 (43)
grippers (312)	**312 (100)**	**312 (100)**	**312 (100)**	**312 (100)**
freecell (117)	**117 (100)**	**117 (100)**	**117 (100)**	**117 (100)**
visitall (100)	**100 (100)**	**100 (100)**	**100 (100)**	92 (92)
reward (99)	**99 (100)**	**99 (100)**	**99 (100)**	95 (96)
grid* (100)	**100 (100)**	**100 (100)**	**100 (100)**	24 (24)
logistics* (100)	**97 (97)**	48 (48)	93 (93)	70 (70)
satellite* (100)	**100 (100)**	**100 (100)**	**100 (100)**	54 (54)
Total (1576)	**1570 (100)**	1523 (97)	1569 (100)	1110 (70)

Table 4: Benchmark coverage results for AIW compared to its variants BAIW, C-AIW, and Flat policy, all using AD encodings. Domains marked * are known to require logic beyond C_2 expressiveness. Numbers in parentheses are percentages of row totals.

- *Reduced Goals* – Full AD relation set $\mathcal{R}^{\mathcal{T}}$ encoded, yet AIW now also applies the abstracted novelty checks on atoms $P(o_1, \ldots, o_n)$ that match an atom $P_g(o_1, \ldots, o_n)$ in the goal.

Given that Tables 1 and 3 already cover several ablation directions, Table 5 isolates the remaining AD-encoding components. Most of these components were introduced to stabilize test-time behavior. During development, we often observed large variance in test outcomes, which decreased after the full encoding was introduced. The ablation results show no large aggregate performance drop for most removals. However, they indicate policy instabilities: some easy test instances remain unsolved, even when most hard instances are solved, as in Spanner when depth nodes are removed. Values also fluctuate between the ablation setups, resulting in better outcomes, although more components have been deactivated. The only consistently high-impact ablation is the abstraction of atoms that also occur in the goal. Removing this exception substantially reduces coverage on the harder instances of most domains.

Encoding Ablations. As mentioned in Section 5, we consider two interpolating encodings: Internal and Internal-Delta (defined and illustrated in Appendix D). External encodings represent each successor with its full relation set in a separate graph and process state transitions sequentially. At the other end, Aggregated-Delta (AD) represents successors only through their delta atoms and encodes all transitions jointly. Internal and Internal-Delta therefore isolate whether performance gains arise from merging current and successor states into a single graph, from representing successors through state differences, or from combining both mechanisms.

Tables 6 and 7 show that merging current and successor states into a single graph is not sufficient on its own. Internal encodings are worse than External encodings in aggregate coverage and in resource usage. This is likely due to the duplicated relation set, which effectively doubles the number of predicate-specific modules relative to External encodings and increases the number of messages aggregated by the R-GNN. Internal-Delta recovers much of the lost performance while substantially reducing memory usage, showing that explicit state differences are important for exploiting the structure exposed by IW lookaheads. This effect is particularly clear on the extended benchmark: External and Internal encodings fail on several domains where Internal-Delta achieves high coverage, including domains that require logic beyond C_2. In particular, Logistics is solved substantially more often, Satellite becomes almost fully solvable, and Goldminer is solved well only by Internal-Delta and AD. Nevertheless, Internal-Delta still remains below AD, and only AD achieves near-full coverage on the extended benchmark. These results indicate that both components are needed: compact delta-based successor representations and the joint encoding of the full transition set.

Domain	AIW	No Depthnodes	No Tree-Atoms	No Depthnodes No Tree-Atoms	Reduced Goals
blocksworld (90)	**90 (100)**	**90 (100)**	84 (93)	84 (93)	78 (87)
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	28 (93)	**30 (100)**
medium (30)	**30 (100)**	**30 (100)**	**30 (100)**	29 (97)	**30 (100)**
hard (30)	**30 (100)**	**30 (100)**	24 (80)	27 (90)	18 (60)
miconic (90)	**90 (100)**	**90 (100)**	**90 (100)**	**90 (100)**	**90 (100)**
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
medium (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
hard (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
spanner (90)	**90 (100)**	86 (96)	81 (90)	87 (97)	**90 (100)**
easy (30)	**30 (100)**	27 (90)	**30 (100)**	**30 (100)**	**30 (100)**
medium (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
hard (30)	**30 (100)**	29 (97)	21 (70)	27 (90)	**30 (100)**
satellite (90)	**88 (98)**	**88 (98)**	84 (93)	85 (94)	64 (71)
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
medium (30)	**30 (100)**	**30 (100)**	**30 (100)**	29 (97)	28 (93)
hard (30)	**28 (93)**	**28 (93)**	24 (80)	26 (87)	6 (20)
transport (90)	**86 (96)**	83 (92)	82 (91)	**86 (96)**	63 (70)
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
medium (30)	**29 (97)**	28 (93)	27 (90)	**29 (97)**	26 (87)
hard (30)	**27 (90)**	25 (83)	25 (83)	**27 (90)**	7 (23)
ferry (90)	**80 (89)**	**80 (89)**	**80 (89)**	**80 (89)**	69 (77)
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
medium (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
hard (30)	**20 (67)**	**20 (67)**	**20 (67)**	**20 (67)**	9 (30)
rovers (90)	**67 (74)**	57 (63)	43 (48)	56 (62)	36 (40)
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	29 (97)
medium (30)	**24 (80)**	20 (67)	9 (30)	19 (63)	7 (23)
hard (30)	**13 (43)**	7 (23)	4 (13)	7 (23)	0 (0)
childsnack (90)	64 (71)	**66 (73)**	62 (69)	**66 (73)**	65 (72)
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
medium (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
hard (30)	4 (13)	**6 (20)**	2 (7)	**6 (20)**	5 (17)
sokoban (90)	13 (14)	13 (14)	**14 (16)**	11 (12)	11 (12)
easy (30)	13 (43)	13 (43)	**14 (47)**	11 (37)	11 (37)
medium (30)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
hard (30)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
floortile (90)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
easy (30)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
medium (30)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
hard (30)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
Total (900)	**668 (74)**	653 (73)	620 (69)	645 (72)	566 (63)

Table 5: IPC 2023 benchmark ablation coverage results for AIW with AD encoding. Numbers in parentheses are percentages of row totals.

Domain	AD	ID	Int	Ext
blocksworld (90)	**90 (100)**	75 (83)	76 (84)	78 (87)
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
medium (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
hard (30)	**30 (100)**	15 (50)	16 (53)	18 (60)
miconic (90)	**90 (100)**	72 (80)	69 (77)	72 (80)
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
medium (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
hard (30)	**30 (100)**	12 (40)	9 (30)	12 (40)
spanner (90)	**90 (100)**	30 (33)	32 (36)	32 (36)
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
medium (30)	**30 (100)**	0 (0)	2 (7)	2 (7)
hard (30)	**30 (100)**	0 (0)	0 (0)	0 (0)
satellite (90)	**88 (98)**	64 (71)	48 (53)	61 (68)
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
medium (30)	**30 (100)**	**30 (100)**	16 (53)	**30 (100)**
hard (30)	**28 (93)**	4 (13)	2 (7)	1 (3)
transport (90)	**86 (96)**	64 (71)	45 (50)	55 (61)
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	29 (97)
medium (30)	**29 (97)**	28 (93)	14 (47)	19 (63)
hard (30)	**27 (90)**	6 (20)	1 (3)	7 (23)
ferry (90)	**80 (89)**	69 (77)	69 (77)	71 (79)
easy (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
medium (30)	**30 (100)**	**30 (100)**	**30 (100)**	**30 (100)**
hard (30)	**20 (67)**	9 (30)	9 (30)	11 (37)
rovers (90)	**67 (74)**	29 (32)	27 (30)	24 (27)
easy (30)	**30 (100)**	29 (97)	27 (90)	24 (80)
medium (30)	**24 (80)**	0 (0)	0 (0)	0 (0)
hard (30)	**13 (43)**	0 (0)	0 (0)	0 (0)
childsnack (90)	**64 (71)**	32 (36)	38 (42)	47 (52)
easy (30)	**30 (100)**	20 (67)	**30 (100)**	**30 (100)**
medium (30)	**30 (100)**	12 (40)	8 (27)	17 (57)
hard (30)	**4 (13)**	0 (0)	0 (0)	0 (0)
sokoban (90)	13 (14)	12 (13)	7 (8)	**17 (19)**
easy (30)	13 (43)	12 (40)	7 (23)	**17 (57)**
medium (30)	0 (0)	0 (0)	0 (0)	0 (0)
hard (30)	0 (0)	0 (0)	0 (0)	0 (0)
floortile (90)	0 (0)	0 (0)	0 (0)	0 (0)
easy (30)	0 (0)	0 (0)	0 (0)	0 (0)
medium (30)	0 (0)	0 (0)	0 (0)	0 (0)
hard (30)	0 (0)	0 (0)	0 (0)	0 (0)
Total (900)	**668 (74)**	447 (50)	411 (46)	457 (51)

Table 6: IPC 2023 benchmark ablation coverage results for encodings Aggregated-Delta (AD), Internal-Delta (ID), Internal (Int), and External (Ext) paired with AIW lookaheads. Numbers in parentheses are percentages of row totals.

Domain	AD	ID	Int	Ext
depots (86)	**86 (100)**	79 (92)	83 (97)	83 (97)
delivery (80)	**80 (100)**	**80 (100)**	75 (94)	**80 (100)**
driverlog (123)	120 (98)	**122 (99)**	116 (94)	116 (94)
goldminer (359)	**359 (100)**	317 (88)	50 (14)	133 (37)
grippers (312)	**312 (100)**	**312 (100)**	**312 (100)**	**312 (100)**
freecell (117)	**117 (100)**	**117 (100)**	**117 (100)**	**117 (100)**
visitall (100)	**100 (100)**	98 (98)	94 (94)	**100 (100)**
reward (99)	**99 (100)**	**99 (100)**	86 (87)	**99 (100)**
grid* (100)	**100 (100)**	91 (91)	90 (90)	**100 (100)**
logistics* (100)	**97 (97)**	69 (69)	7 (7)	0 (0)
satellite* (100)	**100 (100)**	87 (87)	19 (19)	26 (26)
Total (1576)	**1570 (100)**	1471 (93)	1049 (67)	1166 (74)

Table 7: Encoding ablation coverage results for encodings Aggregated-Delta (AD), Internal-Delta (ID), Internal (Int), and External (Ext) paired with AIW lookaheads on our extended benchmark set. Numbers in parentheses are percentages of row totals.

D Visualizing the Encodings

Figures 1 and 2 draw the R-GNN inputs $\mathcal{R}$ of Section 5 for a small Blocksworld instance, and Figure 3 in Appendix E follows the AD encoding through a training and an evaluation step. In the state s, the blocks a, b, c, and d stand on the table, the goal is $g = \{\texttt{on}(\text{d},\text{c}), \texttt{on}(\text{c},\text{b}), \texttt{on}(\text{b},\text{a})\}$, and the successor s' results from picking up d. Circles are objects and small nodes are atoms, colored by predicate. An atom $P(o_1, o_2)$ is joined to o_1 by a blue edge and to o_2 by an orange edge, which mirrors the position-wise messages of the R-GNN (Section 4). The atoms and edges are those produced by our encoder for this instance. A count such as "+6" gives the number of atoms in a set that are not drawn, and atoms of the nullary predicate `handempty` are omitted. Note that this visualization fails at implying the two phases of relational message passing: the first phase collects the positional object-features in each atom and creates a matrix of outgoing messages that can be viewed as the atom node's feature, while the second phase then sends these messages back to the corresponding objects.

Besides External and AD, Figure 1 shows the two interpolating encodings of the encoding ablation (Tables 6 and 7). The *Internal* encoding adds the atoms of s' to the graph of s as primed atoms over the same objects, so that the R-GNN can compare both states during message passing; the goal atoms stay those of $\mathcal{R}^s$:

$$\begin{aligned} \mathcal{R}^{s,s'} &= \mathcal{R}^s \cup \mathcal{R}^{s'}_{\mathcal{P}'}, \text{ where} \\ \mathcal{R}^{s'}_{\mathcal{P}'} &= \{P'(\bar{o}) \mid P(\bar{o}) \in s'\}. \end{aligned}$$

The *Internal-Delta* encoding keeps only the changes between the two states:

$$\begin{aligned} \mathcal{R}^{\Delta} &= \mathcal{R}^s \cup \mathcal{R}^s_{\Delta s'}, \text{ where} \\ \mathcal{R}^s_{\Delta s'} &= \{P^+(\bar{o}) \mid P(\bar{o}) \in s' \setminus s\} \cup \{P^-(\bar{o}) \mid P(\bar{o}) \in s \setminus s'\} \\ &\quad \cup \{P^+_G(\bar{o}) \mid P(\bar{o}) \in g \cap (s' \setminus s)\} \cup \{P^-_G(\bar{o}) \mid P(\bar{o}) \in g \cap (s \setminus s')\}. \end{aligned}$$

Both compute $Q(s, s') = \text{MLP}\big(\sum_{o \in \mathcal{O}} \text{f}(o; \mathcal{R})\big)$ for their respective $\mathcal{R}$ and, like External, need one graph per transition. AD uses the same added and deleted atoms as Internal-Delta but anchors them to a state node $o_{s'}$, which lets all successors of the lookahead tree share one graph (Figure 2).

E A Walkthrough of Training and Evaluation

Figure 3a shows a Spanner state in which `bob` is at `loc1`, `spanner1` lies at `loc2`, `spanner2` lies at `loc4`, and two loose nuts must be tightened at the gate. Bob can only move forward, and each spanner tightens a single nut. The AIW lookahead from this state offers, among others, jumps to

i) v_1, where bob is at `loc2` (depth 1),

ii) v_2, where bob is at `loc2` carrying `spanner1` (depth 2),

iii) v_3 and v_4, where bob is at `loc3` and `loc4` (depths 2 and 3).

The jumps to v_3 and v_4 are dead-ends: they walk past `spanner1`, which can never be retrieved, so only one nut can be tightened. Walking on from v_2 is pruned, since the abstractions $\texttt{at}(\texttt{bob}, \texttt{location})$ and $\texttt{at}(\texttt{man}, \texttt{loc3})$ of its new atom $\texttt{at}(\texttt{bob}, \texttt{loc3})$ were already seen at the root and at v_3, respectively. In the AD encoding (Figure 3b), the root is represented by $\mathcal{R}^s$ and each candidate by a state node carrying its delta atoms, e.g., $\texttt{at}^+(o_{v_1}, \texttt{bob}, \texttt{loc2})$ and $\texttt{at}^-(o_{v_1}, \texttt{bob}, \texttt{loc1})$ for v_1, and additionally $\texttt{carrying}^+(o_{v_2}, \texttt{bob}, \texttt{spanner1})$ and $\texttt{at}^-(o_{v_2}, \texttt{spanner1}, \texttt{loc2})$ for v_2. The tree atoms $P_E(o_{v_1}, o_{v_2})$, $P_E(o_{v_1}, o_{v_3})$, and $P_E(o_{v_3}, o_{v_4})$ link each candidate to its parent, and P_{SD} atoms associate $o_{v_1}, \ldots, o_{v_4}$ with the depth nodes o_{d_1}, o_{d_2}, and o_{d_3}, which are ordered by P_D atoms. A single forward pass then yields $Q(s, v)$ for all candidates. The depth-ranking loss exists precisely so that the network can separate such candidates by how far the jump travels.

A training step. Figure 3c summarizes both steps. Each episode draws four trajectories from the current policy, each capped at 20 jumps. At a state s, we run the AIW lookahead, encode the *entire* resulting tree of successor states, not only its leaves, as one AD graph, and score all successors in a single forward pass. We then sample $s' \propto \exp(Q(s, s')/\tau)$ instead of taking the argmax, so

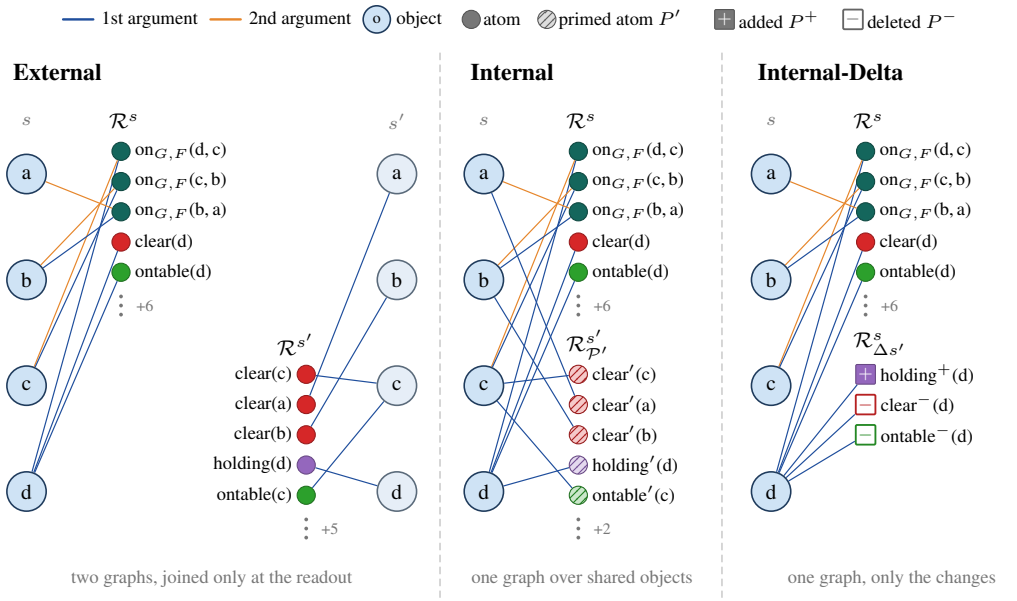


Figure 1: The External, Internal, and Internal-Delta encodings of the transition $s \to s'$, where s' picks up block d. All three contain $\mathcal{R}^s$, whose goal atoms $\texttt{on}_{G,F}$ mark goals that are false in s. *External* builds a second graph for $\mathcal{R}^{s'}$ over a copy of the objects, and the two graphs first interact in the readout $\text{MLP}_2(\text{f}(o; \mathcal{R}^s), \text{f}(o; \mathcal{R}^{s'}))$. *Internal* adds the atoms of s' as primed atoms P' (hatched) over the objects of s. *Internal-Delta* adds only $\texttt{holding}^+(\text{d})$, $\texttt{clear}^-(\text{d})$, and $\texttt{ontable}^-(\text{d})$, the atoms of $s' \setminus s$ and $s \setminus s'$.

exploration happens over jumps rather than primitive actions. The resulting transitions enter the replay buffer, and lifted HER relabels the goal of a failed trajectory with a goal that the trajectory did achieve, which supplies learning signal under the constant reward of -1. Each episode then performs 32 optimization steps with batch size 32 on the TD target plus the auxiliary depth-ranking loss.

An evaluation step. Evaluation is identical up to the readout: we run AIW from s and perform one forward pass over the tree, but exclude successors already visited in the current episode, move to $s^* = \arg\max_{s'} Q(s, s')$, and repeat until $g \subseteq s$ or the budget of $1{,}000$ choices or 60 minutes is exhausted.

Combining the learned policy with a best-first search over jump endpoints is not a drop-in change. Such a search would sit on top of a local lookahead that is itself a search, and how the two interact, and whether the combination pays off, is a research question of its own. It would also confound the measurement: coverage would then reflect the search recovering from value errors and could no longer be attributed to the learned policy. At the scale of IPC 2023, maintaining an open list moreover runs into the same enumeration and memory limits that motivate this work. Most importantly, our claim is that a lookahead whose size is linear in the number of objects, invoked a bounded number of times, suffices to outperform LAMA, an exponential search. Reintroducing such a search would answer a different question.

F Expressivity of the Aggregated-Delta Encoding

This appendix makes the statements of Section 8 precise. Throughout, $\mathcal{T}$ is the lookahead tree rooted at s, O is the set of problem objects, o_v is the state node of a successor $v \in V(\mathcal{T}) \setminus \{s\}$, and R-GNNs are as in Section 4. All statements concern what the architecture can represent, not what training is able to find.

Encoding Loss. For $v \in V(\mathcal{T})$, let $\Delta_v^+ = v \setminus s$ and $\Delta_v^- = s \setminus v$. Since $v = (s \setminus \Delta_v^-) \cup \Delta_v^+$, every state of the tree can be recovered from $\mathcal{R}^{\mathcal{T}}$; the goal follows from the $P_{G,T}$ and $P_{G,F}$ atoms, and the P_E, P_{SD}, and P_D atoms reconstruct the tree and the depth order. The map $(s, \mathcal{T}) \mapsto \mathcal{R}^{\mathcal{T}}$ is therefore

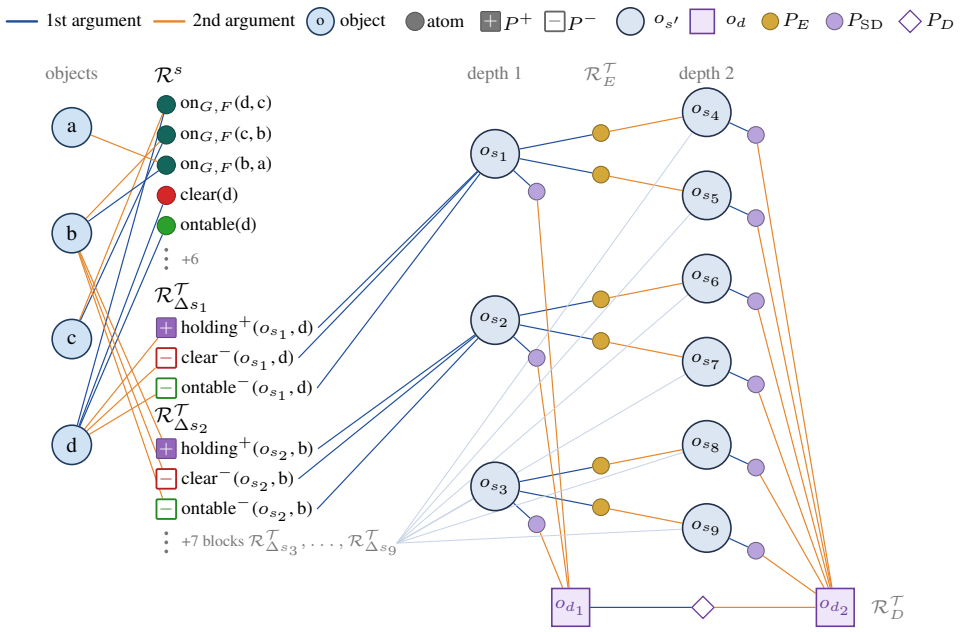


Figure 2: The Aggregated-Delta encoding $\mathcal{R}^{\mathcal{T}}$ of a lookahead tree $\mathcal{T}$ rooted at the state s of Figure 1, truncated for display to depth 2 and three children per state: s_1, s_2, s_3 at depth 1, with s_1 the successor s' of Figure 1, and $s_4, \ldots, s_9$ at depth 2. The root is represented only by its relation set $\mathcal{R}^s$, not by an explicit state node. Each successor s_i receives a state node o_{s_i}, instead, and the delta atoms $\mathcal{R}^{\mathcal{T}}_{\Delta s_i}$, whose first argument is o_{s_i}. Here, only the relations of s_1 and s_2 are drawn, and the links of the other seven are hinted at with faint lines. The tree atoms $P_E(o_{s'}, o_{s''})$ form $\mathcal{R}^{\mathcal{T}}_E$, the depth nodes o_{d_1}, o_{d_2}, the order atom $P_D(o_{d_1}, o_{d_2})$, and the state-depth atoms $P_{\mathrm{SD}}(o_{s'}, o_d)$ form $\mathcal{R}^{\mathcal{T}}_D$. One forward pass over this graph yields $Q(s, s') = \mathrm{MLP}\left(\mathrm{f}(o_{s'}; \mathcal{R}^{\mathcal{T}}), \sum_{o \in \mathcal{O}} \mathrm{f}(o; \mathcal{R}^{\mathcal{T}})\right)$ for all nine successors.

invertible up to a type-preserving renaming of the auxiliary objects. Any information loss occurs during message passing, not in the input.

Expressivity Bound. Let c_L denote L rounds of relational color refinement, i.e., 1-WL on the atom–object incidence graph with edges labeled (P, j): the new color of an object combines its previous color with the multiset of tuples $(P, j, c(o_1), \ldots, c(o_k))$ over all atoms $P(o_1, \ldots, o_k)$ in which it occurs at position j. Positions must be retained because comb_P is position-specific by design. For AD, the signature of a candidate v is $\sigma_L(v) = \left(c_L(o_v; \mathcal{R}^{\mathcal{T}}), \{\!\{c_L(o; \mathcal{R}^{\mathcal{T}}) : o \in O\}\!\}\right)$, matching the readout of Section 5. For External, the counterpart is the paired multiset $\sigma_L^{\mathrm{Ext}}(s, v) = \{\!\{(c_L(o; \mathcal{R}^s), c_L(o; \mathcal{R}^v)) : o \in O\}\!\}$, matching its per-object paired readout. Candidates with equal signatures receive equal scores from every L-layer R-GNN with the respective readout. Within one AD encoding, the multiset over O is shared by all candidates, so any distinction between candidates must appear in $c_L(o_v; \mathcal{R}^{\mathcal{T}})$.

Conversely, on any fixed finite family of inputs, sufficiently wide message, update, and readout functions attain this bound if the aggregation can be injective on multisets of messages. The smooth maximum $\mathrm{LSE}_\tau(z_1, \ldots, z_N) = \tau \log \sum_i \exp(z_i/\tau)$ used in our implementation is a strictly monotone transform of a coordinate-wise sum and, like sum aggregation, preserves multiplicity, since $\mathrm{LSE}_\tau(z, \ldots, z) = z + \tau \log N$. A hard maximum is idempotent and cannot count. This is a guarantee about representability, not learnability. Both separations below need multiplicity only in the first layer, where $\mathrm{f}_0(o) = \mathbf{0}$ makes the counted messages identical, so LSE_τ counts them exactly, however closely it approximates a maximum on distinct messages; all later steps of both constructions need only distinctions that a hard maximum also preserves.

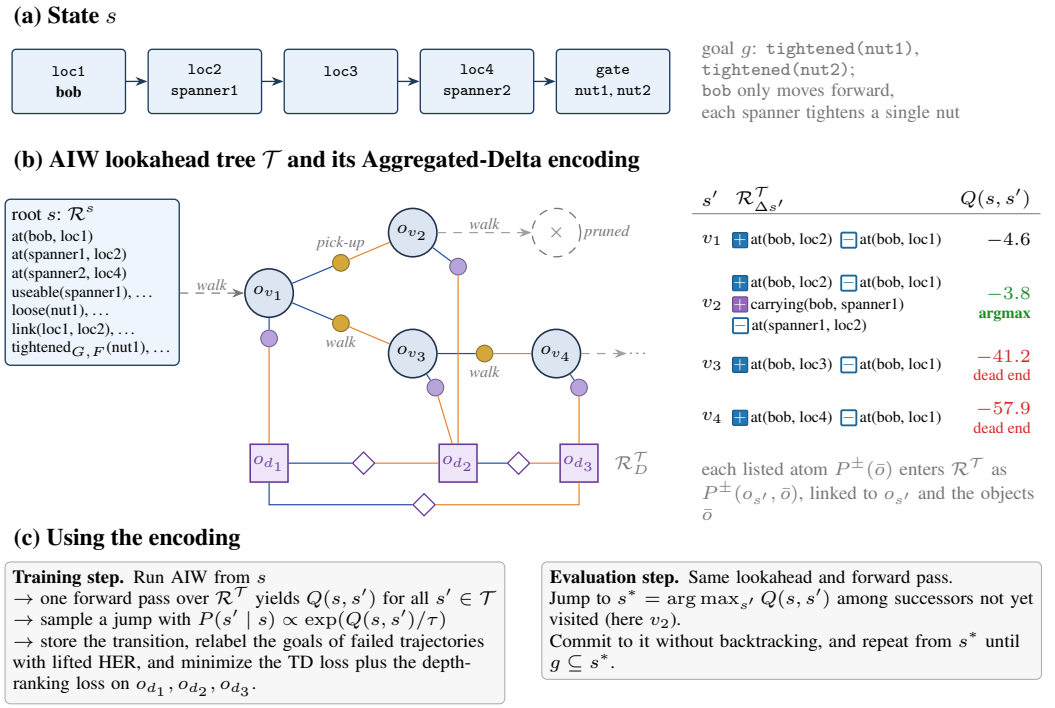


Figure 3: A worked Spanner example, Q-values are illustrative. (a) The state s. (b) The AIW lookahead tree $\mathcal{T}$ from s and its AD encoding $\mathcal{R}^{\mathcal{T}}$, drawn with the symbols of Figure 2: state nodes o_{v_i}, tree atoms P_E labeled with the action that generates the child, and depth nodes with P_{SD} and P_D atoms. The root has no state node, so the jump (walk-action) to v_1 is not actually encoded as a tree atom, and walking on from v_2 is pruned as not novel. For readability, the delta atoms $\mathcal{R}^{\mathcal{T}}_{\Delta s'}$ are listed in the table instead of being drawn as nodes, next to $Q(s, s')$: the network ranks v_2 highest and assigns low values to the dead ends v_3 and v_4. (c) How the forward pass is used in training and evaluation.

F.1 AD vs. External Expressivity Analysis

On Transition Structure. Consider a STRIPS domain with workers w_1, w_2, w_3 and tasks t_1, t_2, t_3, a static predicate *home* with $home(w_i, t_i)$, a fluent $assigned(w, t)$, and two action schemas: one swaps the assignments of two workers, the other rotates the assignments of three. From the root state with $assigned(w_i, t_i)$ for all i, a swap and a rotation yield two successors. Without *home*, both successors would be perfect matchings on the same objects, hence isomorphic with equal optimal value, leaving no ranking to represent. With *home*, *assigned* is a permutation relative to a fixed reference matching, so the swap is a transposition, the rotation is a 3-cycle, and the successors differ. Yet *home* leaves refinement blind: every worker and every task occurs in exactly one *home* and one *assigned* atom, so each state is a disjoint union of cycles (2-cycles at the root, a 4-cycle and a 2-cycle after the swap, a 6-cycle after the rotation), whose lengths color refinement cannot distinguish. All workers, and all tasks, thus remain mutually indistinguishable within each state, and the paired readout of External ties both candidates for every L. AD attaches two added and two deleted *assigned* atoms to the state node of the swap and three of each to that of the rotation, which separates them after one layer by multiplicity. Adding a schema $finish(x, y, t, t')$ with $x \neq y$ and precondition $assigned(x, t') \wedge assigned(y, t) \wedge home(x, t) \wedge home(y, t')$, which becomes applicable only after a mutual swap, yields optimal values $V^* = -1$ and $V^* = -2$ for the two successors, as the 3-cycle first requires a swap. External cannot represent this ranking, whereas AD can.

On Search Context. For every L, there is a Spanner instance whose one-way corridor places the two spanners more than $2L + 1$ steps apart from each other and from both ends of the corridor. Both spanners are needed, so picking up the nearer one is safe, while jumping past it leads to a dead end. The radius-L neighborhoods of the two spanner locations are then disjoint and non-adjacent,

so exchanging them is a color-preserving bijection, and every L-layer External model ties the two candidates. AD separates them within two layers: one layer of multiplicity-preserving aggregation distinguishes all depth nodes by the number of P_D atoms they occur in at either position, and a second layer carries this distinction to the state nodes via P_{SD}. The ablations agree, although not monotonically: AD solves $90/90$ Spanner instances, removing depth nodes reduces this to 86, removing tree atoms to 81, and removing both to 87, while Internal-Delta and External solve $30/90$ and $32/90$ (Tables 5 and 6).

Incomparability. AD is not a universal superset of External. External refines colors within the complete successor encoding $\mathcal{R}^v$ and reads out a candidate-specific multiset over all objects. AD has no $\mathcal{R}^v$: its term $\sum_{o \in O} \mathrm{f}(o; \mathcal{R}^{\mathcal{T}})$ is shared by all candidates of a tree, and its candidate-specific term is the single embedding $\mathrm{f}(o_v; \mathcal{R}^{\mathcal{T}})$, whose L-layer receptive field only reaches objects within $L-1$ hops of the changed atoms. Shared object nodes therefore blur properties of a complete successor that involve objects far from its delta, which External preserves per candidate. We thus expect separations in both directions. An IW-policy, however, ranks alternatives at a common root, where the decision-relevant features are transition-relative (what changed, which goals were achieved or undone) and search-structural (depth and position in the tree). On such families, the two constructions above separate in favor of AD, whereas External has no access to tree or depth information at any depth. AD does not lift the model beyond the 1-WL bound of its own encoding; rather, the encoded structure differs, so the features induced on states exceed what an R-GNN derives from the plain state encoding. Empirically, AD improves the discrimination of the selector: AIW–AD solves 668 IPC 2023 instances against 457 for AIW–Ext (Table 1), and on the extended benchmark 97 versus 0 Logistics, 100 versus 26 Satellite, and 359 versus 133 Goldminer instances (Table 2).

Abstraction Loss. For every ground atom $P(o_1, \ldots, o_n)$ not mentioned in the goal, AIW records the n abstracted atoms that retain one o_i and replace all other arguments by their types. It thus records exactly (i) which goal atoms have been made true and (ii) for each predicate, which objects occur in which argument position, together with the type pattern of the remaining positions. It does not record how objects in different positions are paired: if $\texttt{type}(a) = \texttt{type}(b) = \tau_1$ and $\texttt{type}(c) = \texttt{type}(d) = \tau_2$, then $\{P(a,c), P(b,d)\}$ and $\{P(a,d), P(b,c)\}$ induce the same abstracted atoms $P(a, \tau_2)$, $P(b, \tau_2)$, $P(\tau_1, c)$, and $P(\tau_1, d)$. An atom can hence be novel for IW(1) while all of its abstractions have been seen, in which case AIW(1) prunes the state, so AIW(1) does not inherit the completeness guarantee of IW(1). In return, the number of distinct abstracted atoms is linear in $|O|$ for a fixed domain and goals of linear size.

G Plan Length

Plan length. Our reward counts jumps rather than primitive actions, so plan length is not optimized directly. Table 8 compares the number of primitive actions in AIW–AD plans with those of LAMA on the 523 IPC 2023 test instances solved by both. Averaged over all of them, AIW–AD plans are as long as LAMA's (225 versus 223 actions). They are shorter on Ferry and Blocksworld, equally long on Childsnack and Sokoban, and longer mainly on Transport and Rovers.

Domain	AIW–AD	LAMA
miconic (90)	323	**300**
satellite (88)	172	**157**
ferry (74)	**329**	372
transport (71)	130	**97**
rovers (64)	443	**291**
blocksworld (57)	**164**	359
childsnack (36)	**45**	**45**
spanner (30)	14	**13**
sokoban (13)	**25**	**25**
Total (523)	225	**223**

Table 8: Average number of primitive actions per plan on the IPC 2023 test instances solved by both AIW–AD and LAMA, number of such instances in parentheses. Lower is better. Floortile is omitted, as no instance is solved by both.

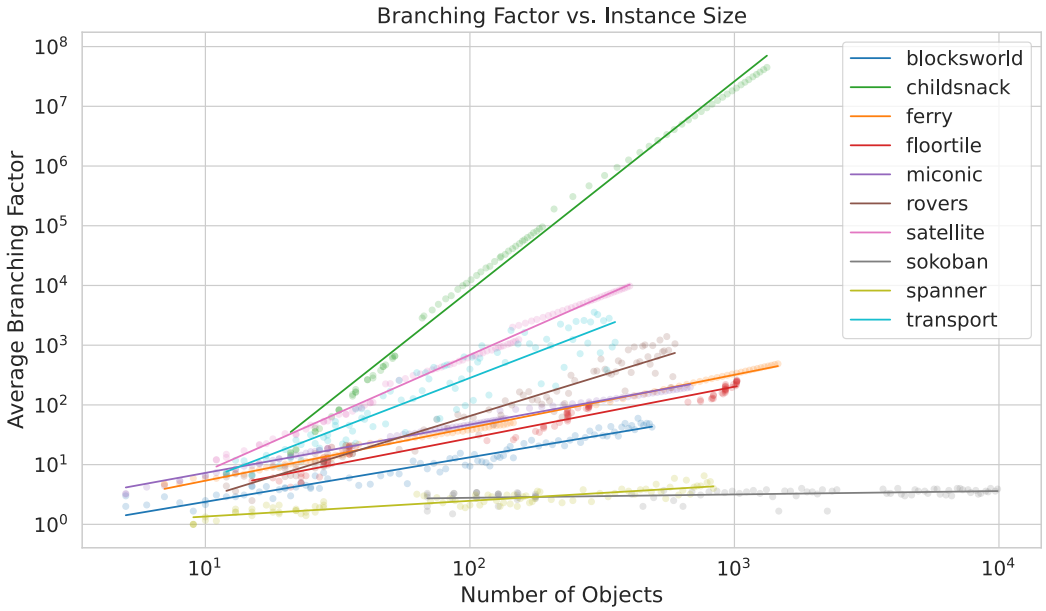


Figure 4: Log-log plot of the branching factor for each IPC benchmark domain as a function of the number of objects.

H IPC 2023 Benchmark Data Scaling Behavior

One way to analyze problem difficulty is to examine how the branching factor scales with instance size. Since the size of the search tree grows exponentially in the branching factor, instances with larger branching factors are generally harder for search-based approaches. We estimate the average branching factor of an instance by running a random walk of length 10 from the initial state, recording the number of applicable actions in each visited state, and averaging these counts. Figure 4 shows the resulting scaling behavior for the IPC domains on a log-log scale. The plot highlights the particular difficulty of Childsnack, whose branching factor grows much faster than in the other domains and reaches values several orders of magnitude larger. Concretely, in Childsnack's hard category, instances #1, #10, #20, and #30 exhibit 0.25, 3.7, 17, and 47 million applicable actions in the initial state, respectively, which scale from requiring 1 second up to 10 minutes simply to generate. This, despite highly optimized software and latest hardware.

Of course, branching factors alone do not explain all of a domain's difficulty. For example, the branching factor in Sokoban remains nearly constant with instance size, although the domain remains challenging for search due to complex long-term consequences of individual action choices.

I Domain Descriptions

This appendix section complements the main text with supplementary content cataloguing every benchmark domain and summarizing the instance statistics highlighted in Table 9.

Blocksworld In Blocksworld, the goal is to stack blocks on top of each other to form specific configurations. The actions involve picking up and putting down blocks, where blocks can only be picked up if they have no other blocks on top of them.

Childsnack In this domain, the task is to serve snacks to children. Some children are gluten allergic and need to be served gluten-free bread. One general policy is to always serve gluten-free bread to gluten allergic children first, then serve the rest to those that are left. Actions include making sandwiches and serving them to children.

	Train					Validation					Test				
		Objects		Atoms			Objects		Atoms			Objects		Atoms	
Domain	#	Min	Max	Min	Max	#	Min	Max	Min	Max	#	Min	Max	Min	Max
delivery	165	6	70	31	447	30	88	129	567	844	80	153	634	1006	4321
depots	589	13	20	62	104	50	21	25	100	130	86	26	60	126	313
driverlog	108	7	15	27	83	48	16	18	71	104	123	19	50	91	291
freecell	787	29	48	124	213	40	31	51	131	225	117	40	56	174	242
goldminer	190	4	20	25	121	40	4	238	25	1609	359	18	361	109	2456
grippers	113	7	42	17	123	40	8	126	19	309	312	22	127	47	308
reward	155	4	144	26	1098	45	169	225	1293	1732	99	256	841	1968	6543
visitall	99	4	144	22	962	27	169	225	1133	1517	100	256	900	1730	6182
grid	128	11	76	48	459	47	108	206	684	1349	100	278	890	1792	6027
logistics	96	6	26	18	78	40	30	52	90	156	100	60	120	180	360
satellite	59	6	34	18	158	66	53	122	216	644	100	140	435	804	3425
blocksworld-ipc	69	2	20	6	45	30	21	29	45	64	90	5	488	13	1019
childsnack-ipc	68	7	41	20	135	30	41	52	135	174	90	21	1327	63	4563
ferry-ipc	69	3	25	9	65	30	26	35	68	92	90	7	1461	18	3898
floortile-ipc	67	5	40	17	238	30	37	45	215	267	90	15	1022	79	6884
miconic-ipc	69	3	21	10	148	30	22	30	164	271	90	5	681	19	21443
rovers-ipc	69	10	28	35	254	30	28	36	193	422	90	12	596	56	294553
satellite-ipc	69	5	32	15	133	30	34	43	139	196	90	11	402	37	2402
sokoban-ipc	69	54	128	134	517	30	128	177	475	793	90	69	9884	219	55872
spanner-ipc	59	6	22	22	88	30	22	28	88	114	90	9	833	31	3961
transport-ipc	69	6	34	20	258	30	34	45	140	382	90	12	354	47	7785

Table 9: Statistics of the benchmarks used in our experiments. For each benchmark, we report the number of instances, as well as the minimum and maximum number of objects and atoms.

Delivery In Delivery, the objective is to deliver packages to their respective destinations using a single truck. The truck can only carry one package at a time, and the actions include loading and unloading packages, as well as driving the truck between locations.

Depots The task in Depots is to move crates onto pallets. The instances contain multiple trucks and hoists that can be used to move crates. The actions include loading and unloading crates onto and from pallets, as well as driving trucks and operating hoists.

Driverlog In Driverlog, the goal is to transport packages and trucks to different locations. In comparison to Delivery and Logistics, this domain also includes drivers that are required to drive the trucks. The actions include loading and unloading packages onto and from trucks, as well as driving trucks and walking drivers to different locations.

Freecell The Freecell domain is based on a card game similar to Solitaire, where the objective is to move all cards to the locations specified in the goal. Actions are to move cards between different cells.

Ferry The Ferry domain is about transporting cars with a ferry. The ferry can only hold a single car at the time, and the goal is to move all cars to their respective destinations. Actions include loading and unloading cars onto and from the ferry, as well as sailing between shores.

Floortile In Floortile, the objective is to paint floor tiles in a specific pattern using a robot. The robot can move around the grid of tiles and paint them in different colors, but can not move to a painted tile. The challenge is to ensure that the final configuration of painted tiles matches the desired pattern, without getting stuck.

Goldminer The goal of Goldminer is for a robot to reach a location that contains gold. The environment is grid based and contains items such as bombs and laser cannons. Actions include moving and using the items to clear a path to the gold.

Grid The task in Grid is to navigate a grid and put keys in specific positions. Some positions are locked and require a key to open. The actions include moving in the grid, picking up keys, and unlocking locked positions.

Grippers In Gripper, the objective is to move balls from one room to another using a robot with two grippers. The robot can carry up to two balls at a time, and the actions include moving between rooms, picking up balls, and dropping them off in the target room. Grippers is similar to Grippers, but instead of a single robot, there are multiple robots, and there can be more than two rooms.

Logistics In Logistics, the goal is to transport packages between different locations using trucks and airplanes. Trucks can only move within a city, while airplanes can fly between cities. The actions include loading and unloading packages onto and from vehicles, as well as driving trucks and flying airplanes to different locations.

Miconic Miconic is about transporting passengers in an elevator to their desired floors. The elevator can move up and down, and passengers can board and leave the elevator at different floors. The goal is to efficiently transport all passengers to their respective floors while minimizing the number of stops.

Reward The goal in Reward is to collect rewards that are placed in different cells. The environment is grid-based and contains obstacles that the agent can not pass. The actions include moving in the grid and collecting rewards.

Rovers Here rovers are tasked with exploring a planetary surface, collecting samples, and taking images. The rovers can move around the terrain, collect samples from specific locations, and take images of points of interest. Different rovers can traverse different types of terrain, which adds complexity to the planning task. The goal is to communicate data from different waypoints.

Satellite In Satellite, the goal is to capture images and point satellites in different directions. The actions include turning the satellites, taking images and calibrating instruments. The challenge is to manage the limited resources of the satellites while ensuring that all required images are captured.

Sokoban This domain is based on the classic puzzle game Sokoban, where the objective is to push boxes onto designated goal locations within a warehouse. The player can move in four directions and push boxes, but cannot pull them. The challenge lies in planning moves carefully to avoid getting boxes stuck in corners or against walls.

Spanner The task in Spanner is to tighten loose nuts using a wrench (spanner). The actions are to move, pickup spanners and tighten nuts. Whenever a wrench is used, it can not be used again. The difficulty is that the agent can only move in one direction, so if it missed to pick up a wrench that it later needs, it will not be able to solve the task.

Transport In Transport, the task is to drive packages to specific locations. Different vehicles have different capacities, so they can carry different number of packages. The actions include loading and unloading packages, as well as driving vehicles between locations.

Visitall Visitall is another grid based domain where the objective is to visit all the cells specified in the goal. The only action is to move between connected cells in the grid.

I.1 Domain statistics

Table 9 summarizes how each benchmark grows across the different splits. The non-IPC domains – Delivery, Depots, Driverlog, Freecell, Goldminer, Grippers, Reward, and Visitall – start with moderate object/atom counts in training (≤ 144 objects, ≤ 1100 atoms) and scale to hundreds of objects and up to 6,543 atoms at test time (e.g., Reward and Visitall).

The IPC-domain test splits explode in complexity. For example, Blocksworld grows from at most 20 in the train set to 488 objects in the test set, Miconic's atom counts reach 21,443, Rovers tops 294,553 atoms, and Sokoban stretches to nearly 10^4 objects.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction are clearly demonstrated with the results in Section 7.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 8 discusses the theoretical limitations (loss of the IW(1) completeness guarantee under AIW, expressivity bounds of the AD-based transition selector), the empirical failure modes (dead ends in Floortile and Sokoban, action enumeration in Childsnack), and the scope of the method (per-domain training, relational inputs). Section 3 characterizes what the abstraction loses, and Appendix F gives the formal statements.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The paper's theoretical statements on the expressivity of the AD encoding and on what AIW records are given in Appendix F, each with its assumptions and a constructive argument. They concern representability, not learnability.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The architecture and encodings are presented in Sections 4 and 5 and the learning target in Section 6. Hyperparameters for training and testing the models are reported in Section 7. Code, data and model checkpoints are publicly available (Section 7).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code is available at `https://github.com/maichmueller/aggregated-delta`, with a README and a container definition for running it. The exact data used for the experiments (training, validation, testing), all model checkpoints and the planner outputs are available at doi:10.5281/zenodo.24007972.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: These are specified in Section 7, and will also be available in the code.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We test the best 5 checkpoints across all the 5 different training runs. Since running statistical tests on such a small sample can give misleading results, we decided not to run statistical significance tests. Running more training runs on each of the datasets

used in our experiments would result in an immense amount of additional compute resource usage. Being mindful about the external costs of compute resources, we choose to include a wider variety of datasets to show our proposed method in more different settings, rather than running more training runs on the same datasets.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: This is reported in Section 7. We required more compute than the sum of the results in the paper suggests, as more architectures were tested before the final one was discovered.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The data used in this paper does not contain information about individuals, no studies involving humans are performed and the published code is documented and easy to run.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This paper is a foundational research paper, with no direct applicable use-cases. We think that providing a discussion of the societal impact of this paper would be too speculative, since this is an early stepping stone to something that could potentially impact society.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not see any clear negative misuses of our data and models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We cite the assets that we use in this paper and have made sure that the licenses are correctly used.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The released code and data are documented in README files provided alongside them. All the assets used in the paper are open to use.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not a component of the proposed methods; they were used only as auxiliary tools for editing, data processing, and implementing standard methods.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.