SCONE: SYMMETRY-AWARE NEURAL COMBINATORIAL OPTIMIZATION FOR GRAPH COLORING

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph coloring, which entails assigning colors to nodes such that no adjacent nodes have the same color, is a fundamental combinatorial optimization problem. In general, however, finding a node coloring with no monochromatic edges given a fixed palette is NP-hard. This motivates learning-based methods that can learn to exploit recurring structure across instances to efficiently produce high-quality colorings. Crucially, the validity of a coloring is invariant to permutations of the colors, and we show that exploiting this symmetry reduces the size of the effective state space over which the model has to learn its policy. At the same time, the coloring algorithm must be able to break some symmetries, for example to assign different colors to two adjacent nodes belonging to the same automorphism orbit. In this paper, we introduce SCONE, a model that uses a *color-equivariant* graph neural network to construct colorings sequentially, allowing information from previous assignments to propagate and break node symmetries. SCONE achieves perfect solution quality with low inference times across several graph coloring tasks, including on `sudoku_extreme` and register allocations, and showcases generalization to graph families not encountered during training.

1 INTRODUCTION

Graph coloring is a fundamental combinatorial optimization problem, which, given a graph and a palette of colors, asks for an assignment of colors to the nodes of the graph such that adjacent nodes receive different colors. Given a palette of k colors, deciding whether such an assignment exists is NP-complete for every fixed $k \geq 3$ (Garey et al., 1974). Moreover, computing the smallest number of colors required to obtain a conflict-free assignment, known as the chromatic number, is NP-hard. Graph coloring has applications in several practical domains, such as register allocation, where a compiler has to choose how to assign the variables of a program to the computer registers (Chaitin et al., 1981), and timetabling problems, where conflicting activities must occupy distinct timeslots (Welsh & Powell, 1967; Carter et al., 1996). Sudoku puzzles can also be seen as graph coloring instances, with cells represented as nodes and digits as colors (Herzberg & Murty, 2007).

Traditional exact approaches include integer programming (Méndez-Díaz & Zabala, 2006), as well as SAT and constraint satisfaction problem (CSP) solvers (Heule et al., 2022). Moreover, classical heuristics, which are generally faster but provide no guarantee of finding a valid coloring, include constructive algorithms such as greedy coloring (Welsh & Powell, 1967) and DSATUR (Brélaz, 1979), as well as local-search methods (Hertz & de Werra, 1987; Malaguti & Toth, 2010).

Intuitively, when instances are drawn from a distribution or feature recurring structure, machine-learning-based algorithms can learn to exploit these regularities and produce high-quality solutions efficiently. More generally, neural combinatorial optimization uses learned models as heuristics for constrained optimization problems like the traveling salesman (Bello et al., 2016; Khalil et al., 2017) and knapsack (Drakulic et al., 2023). When trained on instance distributions, these models amortize the cost of learning a heuristic and enable fast inference on new instances (Bengio et al., 2021).

Related Work For graph coloring, neural approaches can optimize instances directly or learn across a distribution. PI-SAGE (Schuetz et al., 2022) casts coloring as unsupervised node classification, using a graph neural network (GNN) to produce soft color assignments, which are then rounded. Building on PI-SAGE, Vanderbush & Weber (2026) improve approximate k-coloring using recursive warm starts obtained from $(k-1)$-colorings. ROS (Qiu et al., 2025) similarly relaxes assign-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

ments to continuous ones, optimizes a continuous objective, and samples discrete solutions. Other methods learn general procedures across instances. ANYCSP (Tönshoff et al., 2023) trains a GNN with reinforcement learning for solving general CSPs by performing global stochastic updates. Similarly, CONSFORMER (Xu et al., 2025) trains a Transformer to improve complete assignments using differentiable relaxations of the constraints, as a learned form of local search. Instead, Le et al. (2026) learn color representations with a supervised contrastive loss and decode them by clustering.

Graph tasks often exhibit symmetries (Gilmer et al., 2017). Most commonly, if two graphs are isomorphic via a node permutation, node-level outputs should permute accordingly. In graph coloring, applying the same node permutation to a graph and to a valid coloring preserves validity. Importantly, however, an individual coloring isn't *invariant* under node permutations, since nodes in the same automorphism orbit may receive different colors. A solver must therefore be able to break node symmetries. Moreover, graph coloring exhibits a symmetry over its *output*. Color labels are interchangeable, since permuting them transforms any coloring into an equivalent one. These observations suggest that a learned solver should exploit both forms of symmetry to learn efficiently.

Contributions We introduce SCONE (**S**ymmetry-aware **CO**loring via **N**eural **E**quivariance), a model that constructs colorings sequentially, one node at a time. At each step, it takes as input the current partial coloring, allowing information from previous assignments to propagate through the graph, which distinguishes any nodes that were previously symmetric. Node colors are scored by a *color-equivariant* GNN, so relabeling the colors permutes the predictions accordingly without changing the behavior of the model. We show theoretically that equivariance reduces the state space that the model must learn (Theorem 1), and that sequential decoding yields expressive models (Theorems 3, 4, 5). We evaluate SCONE on real-world coloring applications, including Sudoku, register allocation, and exam timetabling. Here, it achieves state-of-the-art solution quality among neural methods, being the only one to reliably solve all instances, while also having the lowest solve times.

2 PRELIMINARIES

A graph is a tuple $G = (V_G, E_G)$, where $V_G = \{1, \dots, |V_G|\}$ is a set of nodes and $E_G \subseteq \{\{u, v\} : u, v \in V_G,\ u \neq v\}$ is a set of undirected edges. We denote by $\mathcal{N}_G(u)$ the set of neighbors of u.

Graph coloring Let $\mathcal{C} = \{1, \dots, k\}$ be a palette of k colors, and let ε denote the absence of a color. A partial k-coloring of G is a map $c : V_G \to \mathcal{C} \cup \{\varepsilon\}$. We refer to (G, c) as a k-coloring instance. A complete coloring $c^\star : V_G \to \mathcal{C}$ is a *proper coloring* if $c^\star(u) \neq c^\star(v)\ \forall \{u, v\} \in E_G$. It is a valid solution of (G, c) if it is proper and extends the coloring c, i.e., $c(u) \neq \varepsilon \Rightarrow c^\star(u) = c(u)$. A graph is k-colorable if it has a proper k-coloring, and its chromatic number $\chi(G)$ is the smallest such k.

Groups and permutations For group Γ, let $\gamma \cdot x$ be the action of $\gamma \in \Gamma$ on x. Its Γ-*orbit* is $\Gamma \cdot x = \{\gamma \cdot x : \gamma \in \Gamma\}$, and $\mathcal{X}/\Gamma$ denotes the set of orbits. A function is Γ-*equivariant* if $f(\gamma \cdot x) = \gamma \cdot f(x)$ for all $\gamma \in \Gamma$ and $x \in \mathcal{X}$. We denote by S_m the group of permutations of $\{1, \dots, m\}$. For $\tau \in S_m$, let $P_\tau \in \{0, 1\}^{m \times m}$ be its permutation matrix, such that $(P_\tau x)_i = x_{\tau^{-1}(i)}$. A node permutation π acts on graphs by relabeling their nodes as $\pi \cdot G = (V_G, \{\{\pi(u), \pi(v)\} : \{u, v\} \in E_G\})$, and on a coloring as $(\pi \cdot c)(\pi(u)) = c(u), \forall u$. Two coloring instances are isomorphic if one is obtained from the other by such a node permutation π, written $(G, c) \simeq (G', c')$. If such a π satisfies $\pi(u) = u'$, then $u \in V_G$ and $u' \in V_{G'}$ have the same *rooted isomorphism* type. We denote by $\mathrm{Aut}(G, c) = \{\pi : \pi \cdot G = G,\ \pi \cdot c = c\}$ the automorphism group of (G, c). Its orbits on V_G are called node orbits.

3 GRAPH COLORING AND SYMMETRIES

Our goal is to design a machine learning model that colors a graph from a fixed palette while minimizing monochromatic edges, ideally producing a proper coloring. This requires exploiting the symmetries of the task for sample efficiency, while breaking them when necessary for expressivity.

3.1 NODE PERMUTATION SYMMETRIES AND WHEN TO BREAK THEM

Consider a graph coloring instance (G, c) and a valid solution (i.e., a valid coloring completing c) $c^* : V_G \to \mathcal{C}$. Then, if another graph (G', c') is isomorphic to (G, c) via an isomorphism π, the coloring $\pi \cdot c^*$ is a valid solution for (G', c'). We therefore want to use an architecture that is equivariant to permutations of the nodes, like a message-passing GNN (Gilmer et al., 2017).

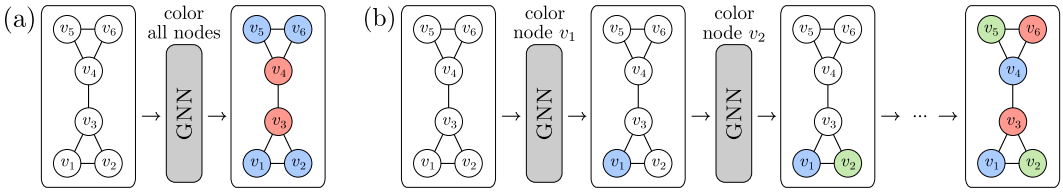

Figure 1: **(a)** Nodes v_1 and v_2 are symmetric and thus indistinguishable, so a deterministic model that predicts colors simultaneously for each node must assign them the same color, even if adjacent. **(b)** SCONE sequentially colors one node at a time, conditioned on previous assignments. Once v_1 has been colored, v_2 can be distinguished from it, allowing the model to assign a distinct color to v_2.

The most immediate strategy for graph coloring with a model is to treat color assignment as a multiclass classification task with labels in $\mathcal{C}$. We would predict the color scores simultaneously for each node, and then select the argmax color for each node independently from each other, as if it was a node classification task. However, as noted also in Li et al. (2022), this strategy has a fundamental limitation, as nodes in the same node orbit of (G, c) are assigned the same color even if neighboring.

Consider, for example, the adjacent nodes v_1 and v_2 in Figure 1 (a). Since they belong to the same orbit, any node-equivariant predictor must assign them the same vector of color scores, even if it can distinguish arbitrary rooted-graph isomorphism types. Therefore, an adversary can always break any score ties so that these nodes are assigned the same color, thus failing to yield a proper coloring.

Therefore, although the predictor itself remains permutation-equivariant, the decoding procedure must be able to *break* symmetries. We do so by constructing the solution sequentially, coloring one node at a time and recomputing the scores after every assignment, see Fig. 1 (b). When automorphic nodes are tied, selecting any one of them breaks the symmetry of the resulting state, allowing previously indistinguishable nodes to become distinguishable. In particular, given the graph and a partial coloring, the model produces color scores to every uncolored node, selects one node, and updates the coloring by assigning it the highest-scoring color. The process ends when all nodes are colored.

More formally, given a k-coloring instance (G, c_0), we define the state space $\mathcal{S}_{(G,c_0)}$ as the set of instances (up to isomorphism) that can be reached by coloring some uncolored nodes of G, and the set of extendable states $\mathcal{S}^+_{(G,c_0)}$ as states that can lead to a proper coloring $c^\star$ of (G, c_0), i.e.

$$\mathcal{S}_{(G,c_0)} = \big\{(G, c) : \, c_0(v) \neq \varepsilon \Rightarrow c(v) = c_0(v) \; \forall v \in V_G\big\}/\simeq$$

$$\mathcal{S}^+_{(G,c_0)} = \big\{(G, c) : \, \exists \text{ valid solution } c^\star \text{ of } (G, c_0) \text{ s.t. } c(v) \neq \varepsilon \Rightarrow c(v) = c^\star(v) \; \forall v \in V_G\big\}/\simeq .$$

From each state $(G, c) \in \mathcal{S}_{(G,c_0)}$, the policy assigns to each uncolored node a score for each color. Based on these scores the policy selects a single uncolored node $u \in V_G$ and assigns a color i to this node (breaking ties arbitrarily), updating the state to (G, c'), with $c'(u) = i$ and $c'(v) = c(v) \; \forall v \neq u$.

Supposing that the color predictor can distinguish any *rooted isomorphism*, a single parameter setting can then express every k-colorable graph of bounded size. At each extendable state, the predictor can memorize, for every uncolored node, the colors leading to an extendable successor. Since every uncolored node has at least one such color, selecting any node and any such color preserves extendability, so induction produces a proper coloring. (Thm. 5). While practical models generally cannot distinguish arbitrary rooted-graph isomorphism types (Morris et al., 2019), this shows that, at least in principle, sequential coloring has enough expressivity to represent valid coloring strategies.

3.2 COLOR SYMMETRIES

While the equivariance to permutations of the nodes is common in most graph tasks, graph coloring presents another symmetry: the color labels are arbitrary. Indeed, consistently permuting the color labels leaves the validity and the number of monochromatic edges of a coloring unchanged. Figure 2 shows two instances that are exactly the same up to a permutation of the partial colors, and illustrates that their solutions are also the same up to the same permutation. Formally, let a color permutation $\tau \in S_k$ act as $(\tau \cdot c)(v) = \tau(c(v))$ if $c(v) \neq \varepsilon$ and $(\tau \cdot c)(v) = \varepsilon$ otherwise. If c^* is a valid solution for (G, c), then for any $\tau \in S_k$ we have that $\tau \cdot c^*$ is a valid solution for $(G, \tau \cdot c)$.

We therefore seek a *color-equivariant* model in order to exploit the symmetries of the problem. Intuitively, if the model outputs the highest score for color blue on node v_6 for the instance (G, c)

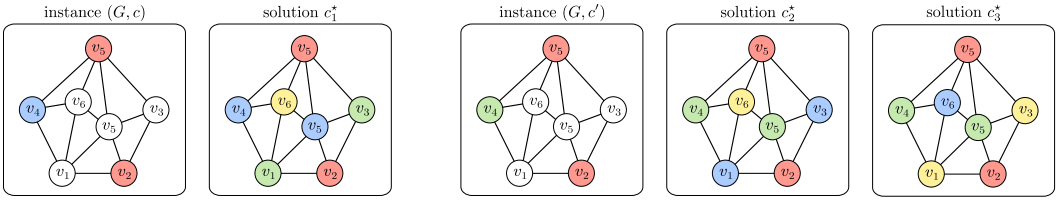

Figure 2: Graph coloring is equivariant to color permutations. The instances (G, c) and (G, c') differ only by exchanging blue and green, so the solutions $c_1^\star$ and $c_2^\star$ are related by the same permutation. Moreover, blue and yellow are unused in c' and thus interchangeable, so $c_2^\star$ and $c_3^\star$ are equivalent.

in Figure 2, then the model should output exactly the same score for color green on node v_6 for the instance (G, c'). Let $p_u(G, c) \in \mathbb{R}^k$ denote the color scores for an uncolored node u. For every color permutation $\tau \in S_k$, we require $p_u(G, \tau \cdot c) = P_\tau p_u(G, c)$. Importantly, color equivariance couples the policy across states that differ only by a permutation of the color labels. If $s' = \tau \cdot s$, then the action scores at s' are determined by those at s, up to the same permutation τ. Therefore, learning the policy on one representative determines its behavior on the entire orbit $\{\tau \cdot s : \tau \in S_k\}$, reducing the size of the effective state space. Moreover, color equivariance does not hinder the model when several colors are symmetric, as the policy will select any one of the tied colors arbitrarily.

For an instance distribution supported on a set $\mathcal{I}$ closed under color permutations, define the corresponding state space as $\mathcal{S}_\mathcal{I} = \bigcup_{(G,c)\in\mathcal{I}} \mathcal{S}_{(G,c)}$. Then, the effective state space of a color-equivariant model is the quotient $\mathcal{S}_\mathcal{I}/S_k$, while an unconstrained model may need to learn the appropriate behavior separately for all states in $\mathcal{S}_\mathcal{I}$, including states belonging to the same equivalence class. In fact, for the purpose of solving a feasible instance exactly, it is sufficient for the policy to learn how to act on extendable states. Indeed, if every choice preserves extendability, the decoding trajectory remains in $\mathcal{S}_\mathcal{I}^+$ until it reaches a proper coloring. However, non-extendable states are still relevant when considering approximate solutions or suboptimal decisions, so we consider both $\mathcal{S}_\mathcal{I}$ and $\mathcal{S}_\mathcal{I}^+$.

Our goal is thus to quantify the reduction in effective state space induced by color equivariance. We begin by bounding how many states related by color permutations form a single equivalence class.

Lemma 1. *Consider a state $(G, c) \in \bigcup_{\tau\in S_k} \mathcal{S}_{\tau\cdot(G,c_0)}$ with exactly r colors appearing in c. Let $R(G, c) = |S_k \cdot (G, c)|$ denote the size of the orbit of (G, c) under color permutations. Then,*

$$\frac{k!}{(k-r)!\,\min(|\mathrm{Aut}(G)|, r!)} \le R(G, c) \le \frac{k!}{(k-r)!}.$$

Intuitively, there are exactly $k!/(k-r)!$ injective ways to relabel the r colors currently used by c, and graph automorphisms can cause some of these relabelings to yield the same state up to isomorphism. For example, for standard 9×9 Sudoku problems, we have $k = 9$. Any uniquely solvable instance contains at least eight distinct fixed numbers, i.e. $r \in \{8, 9\}$. A careful application of the lemma yields $R(G, c) \ge 9!/648 = 560$ (McGuire et al., 2014). Since this is valid for any extendable state, color equivariance reduces the number of distinct states by a factor $|\mathcal{S}_\mathcal{I}^+|/|\mathcal{S}_\mathcal{I}^+/S_k|$ of at least 560.

In general, however, no nontrivial lower bound on r holds uniformly over the state space. For instance, register-allocation and timetabling instances typically begin with no assigned colors, so early decoding states may use only one color. Applying the lemma uniformly over all states would therefore use the worst case $r = 1$, yielding only an overly pessimistic orbit-size bound of order k. The following more nuanced analysis, which exploits the fact that states using few colors are rare, yields a much stronger lower bound on the state space reduction, which is at least exponential in k.

Theorem 1. *Consider a finite k-colorability instance set $\mathcal{I}$ with no starting colors. Let $|\mathrm{Aut}(G)| \le B\ \forall G \in \mathcal{I}$. Let $\mathcal{S}_\mathcal{I}$ be its associated state space and $\mathcal{S}_\mathcal{I}^+$ the space of extendable states. Then:*

1. *If all graphs $G \in \mathcal{I}$ have more than k nodes, then $|\mathcal{S}_\mathcal{I}/S_k| \le B\,\frac{16}{e^k}\,|\mathcal{S}_\mathcal{I}|$.*
2. *If all graphs $G \in \mathcal{I}$ satisfy $0 \le k - \chi(G) \le \kappa$, then $|\mathcal{S}_\mathcal{I}^+/S_k| \le B\,\frac{\kappa!\,2^{k-\kappa}}{k!}\,|\mathcal{S}_\mathcal{I}^+|$.*

Thus, for graph families with bounded B, color equivariance yields an exponential reduction over the full state space, and an even stronger reduction over the extendable states when k is close to the chromatic numbers. This reduction in effective state space suggests a corresponding improvement in sample efficiency, as the model needs to infer appropriate behavior over substantially fewer states.

4 SCONE

In this section, we describe the design and theoretical properties of SCONE, which follows the sequential framework introduced in the previous section. Starting from an initial partial coloring, SCONE constructs a complete coloring one node at a time. At each step, the color predictor produces color scores for every uncolored node. A decoding rule then selects one uncolored node and assigns a color according to that node's scores. The resulting partial coloring becomes the next state.

The main component of SCONE is its color predictor, which we require to be equivariant to both node and color permutations. While message-passing GNNs naturally provide node equivariance, enforcing color equivariance requires that, for every parameter setting, input (G, c), and $\tau \in S_k$, $\mathrm{GNN}(G, \tau \cdot c) = \tau \cdot \mathrm{GNN}(G, c)$. Standard GNN constructions do not enforce this property. For example, representing color as one-hot encodings followed by unconstrained linear or MLP layers assign distinct parameters to each color, so permuting colors does not necessarily permute the output scores accordingly. We therefore design a predictor that is color-equivariant by construction.

4.1 A COLOR-EQUIVARIANT MESSAGE-PASSING ARCHITECTURE

Our color predictor is based on a message-passing mechanism. Each node u will maintain, instead of a single embedding $h_u^\ell \in \mathbb{R}^d$, a collection of k embeddings, one per color, as a matrix $H_u^\ell = (h_{u,1}^\ell, \ldots, h_{u,k}^\ell) \in \mathbb{R}^{k \times d}$. Each slot $h_{u,i}^\ell$ represents the state of node u relative to color i.

We initialize the embeddings of all nodes with color i with a vector e_{on} in the i-th slot and e_{off} in all other slots, while uncolored nodes receive e_{unk} in every slot. Formally, if $c(u) = \varepsilon$, then $H_u^0 = (e_{\mathrm{unk}}, \ldots, e_{\mathrm{unk}})$, whereas if $c(u) \neq \varepsilon$, then $H_{u,c(u)}^0 = e_{\mathrm{on}}$ and $H_{u,i}^0 = e_{\mathrm{off}}$ for all $i \neq c(u)$. The vectors e_{on}, e_{off}, and e_{unk} are learned parameters. This initialization is color-equivariant by construction, as for every $\tau \in S_k$, $H_u^0(G, \tau \cdot c) = P_\tau H_u^0(G, c)$.

As in standard message-passing GNNs, we aggregate the embeddings of neighboring nodes, i.e.,

$$\hat{H}_u^\ell := f_{\mathrm{agg}}(\{\!\{H_v^{\ell-1} : v \in \mathcal{N}(u)\}\!\}) = \bigoplus_{v \in \mathcal{N}(u)} H_v^{\ell-1} \in \mathbb{R}^{k \times d}.$$

Moreover, common element-wise aggregators $\oplus$ such as sum, mean, and max commute with permutations of the color slots, and are therefore S_k-equivariant.

In a traditional GNN one would then process the aggregated messages to produce the new embeddings via the update function f_{upd}, which is typically an MLP. However, this is in general not S_k-equivariant, and therefore we process them with DeepSet-like layers (Zaheer et al., 2017). In particular we can build an S_k-equivariant linear map $L : \mathbb{R}^{k \times d_1} \to \mathbb{R}^{k \times d_2}$ with the form

$$L(H)[i] = W_{\mathrm{self}}\, H[i] + W_{\mathrm{all}} \sum_{j=1}^{k} H[j] + b \quad \text{with} \quad W_{\mathrm{self}}, W_{\mathrm{all}} \in \mathbb{R}^{d_2 \times d_1}, b \in \mathbb{R}^{d_2}.$$

Since the same parameters are used for every slot and the sum over all slots is permutation-invariant, L is S_k-equivariant. Then, we set $\mathrm{MLP}_{S_k}(H) = L_2\big(\sigma\big(L_1(H)\big)\big)$, with σ a point-wise nonlinearity like ReLU or SiLU. Then, our update function is $H_u^\ell := f_{\mathrm{upd}}\big(H_u^{\ell-1}, \hat{H}_u^\ell\big) = \mathrm{MLP}_{S_k}(H_u^{\ell-1} \| \hat{H}_u^\ell)$, where $\|$ denotes concatenation on the d-axis. This is guaranteed to be S_k-equivariant.

Moreover, we apply LayerNorm independently to each color slot, with shared parameters, and add a residual connection, as they have been shown to improve training stability (Luo et al., 2025), as

$$H_u^\ell = \mathrm{LayerNorm}_d\left(\mathrm{MLP}_{S_k}\left(H_u^{\ell-1} \,\middle\|\, \bigoplus_{v \in \mathcal{N}(u)} H_v^{\ell-1}\right)\right) + H_u^{\ell-1}.$$

Finally, after $\mathcal{L}$ message-passing layers, for each node u we produce the color scores from the final layer embeddings $H_u^{\mathcal{L}}$ via a S_k-equivariant readout $\mathrm{MLP}_{S_k} : \mathbb{R}^{k \times d} \to \mathbb{R}^{k \times 1}$.

All operations are color-equivariant, and they compose with the permutation equivariance acting on nodes typical of message-passing architectures. Therefore, we obtain the following result.

Theorem 2. *The color predictor of* SCONE *is equivariant to the joint action of $S_{|V_G|} \times S_k$. Let (G, c) be any graph k-coloring instance, and let $p_u(G, c) \in \mathbb{R}^k$ be the output of the predictor for node u. Then, for every parameter setting, every $\pi \in S_{|V_G|}$, every $\tau \in S_k$, and every $u \in V_G$,*

$$p_{\pi(u)}\big(\pi \cdot G,\ \tau \cdot (\pi \cdot c)\big) = P_\tau\, p_u(G, c).$$

Moreover, the parameterization of our color-equivariant message-passing architecture is independent of k. Since all transformations are shared across color slots, the same parameters apply to any value of k. This permits zero-shot evaluation on instances with palette sizes unseen during training.

4.2 SYMMETRY-AWARE TRAINING

We train SCONE in a supervised fashion, as a node prediction task, from target proper colorings. Given a partial coloring (G, c) and a target completion $c^\star$, a natural approach would be to train each uncolored node u against the one-hot label corresponding to $c^\star(u)$. However, such labels can conflict with color symmetries that are still present in the current state.

Let $U(c) = \{i \in \mathcal{C} : \exists u \in V_G,\ c(u) = i\}$ and $F(c) = \mathcal{C} \setminus U(c)$ denote the used and free colors, and let $\mathrm{Stab}(c) = \{\tau \in S_k : \tau \cdot c = c\}$ be the stabilizer of c. These permutations fix every used color and freely permute the remaining ones. Color equivariance therefore implies $p_u(G, c) = P_\tau p_u(G, c)\ \forall \tau \in \mathrm{Stab}(c)$, so all free colors receive the same score. Hence, if $c^\star(u) \in F(c)$ and $|F(c)| > 1$, a one-hot target for $c^\star(u)$ cannot be matched and produces an irreducible loss.

This ambiguity is intrinsic to the problem rather than a limitation of the architecture. Indeed, for every $\tau \in \mathrm{Stab}(c)$, the coloring $\tau \cdot c^\star$ is an equally valid completion of (G, c). Figure 2 illustrates an example of this phenomenon. There, since only red and green have appeared in c', blue and yellow remain interchangeable, so completions obtained by swapping these two colors are equivalent.

While full symmetry-consistent supervision would also require averaging targets over automorphic nodes, doing so requires computing node orbits at each state, which can be expensive. In practice, we therefore replace the one-hot target by its average over all color-equivalent completions, i.e.,

$$y_u(i) := \frac{1}{|\mathrm{Stab}(c)|} \sum_{\tau \in \mathrm{Stab}(c)} \mathbb{1}[i = \tau(c^\star(u))] = \begin{cases} 1, & \text{if } c^\star(u) \in U(c) \text{ and } i = c^\star(u), \\ 1/|F(c)|, & \text{if } c^\star(u) \in F(c) \text{ and } i \in F(c), \\ 0, & \text{otherwise.} \end{cases}$$

We train the predictor by minimizing the cross-entropy of the softmax of color scores with these symmetry-aware labels y. The loss supervises every uncolored node rather than only a node selected by an expert ordering. The predictor is thus trained to provide a useful color distribution for whichever node the decoder selects next, without requiring a separate node-selection objective.

Ground truth construction For graphs drawn from a given distribution, we can obtain proper colorings using an exact solver, for example via integer linear programs (ILP). While computationally expensive, this is a one-time cost which is amortized by the much cheaper model at inference time.

Moreover, there is an asymmetry in the difficulty of solving given k-coloring instances and *generating* instances with valid solutions. Indeed, we can first draw a certain number of nodes, each with a color from $\mathcal{C}$, and then draw edges according to a distribution of interest, *only* between nodes of different colors. Masking some or all of the assigned colors yields a k-coloring instance together with a proper coloring by construction, avoiding the need to solve the generated instance.

Given an instance (G, c) and a target completion $c^\star$, we generate multiple training states by partially revealing $c^\star$. Specifically, for a subset R of the uncolored nodes, we reveal $c^\star(u)$ for every $u \in R$, obtaining a partial coloring c'. All such instances (G, c') belong to the state space $\mathcal{S}_{(G,c)}$, with small sets R representing states near the beginning of the decoding process, and larger sets representing later stages. For a graph with m uncolored nodes, this construction yields up to 2^m training states.

4.3 INFERENCE STRATEGIES

We train our model to predict a correct color for *every* uncolored node at any stage of the decoding process. Therefore, we could simply color the nodes in their input order. However, we find (see Appendix E) that in practice it is more effective to select nodes using the common *most-constrained-first* principle (Haralick & Elliott, 1980) from constraint satisfaction. In particular, we select an uncolored node with the largest number of distinct colors already assigned to its neighbors. In case of ties, we select the node with the highest maximum softmax probability, as a proxy for confidence. Then, we assign to this node the color with the maximum score, breaking any ties arbitrarily.

Moreover, we can improve over greedy decoding using beam search (Pearl, 1984). We maintain a beam of B partial colorings and, at each step, expand each of them with multiple color assignments,

retaining the B states with highest cumulative score. This allows the model to maintain multiple decoding trajectories and recover from wrong choices made at earlier steps. We further increase diversity by ensembling independent decodings from E starting nodes, yielding up to $B \times E$ candidate colorings. If an instance is not solved, we can repeat the decoding at increasing width, e.g., by doubling B or E. We return the coloring with the fewest monochromatic edges across all attempts.

Finally, we can avoid some model evaluations by directly propagating *forced assignments*. If an uncolored node has only one color that would not create a monochromatic edge, we assign it without running the color predictor. We repeatedly apply this propagation rule until no forced assignment remains, and then resume model-based decoding. When no conflict-free color is available for the selected node, SCONE assigns a conflicting color, so it always produces a complete coloring.

4.4 SCONE IS EXPRESSIVE

In Section 3.1, we argued that sequential coloring is expressive in principle. However, the message-passing-based predictor of SCONE is limited by 1-WL at each step (Morris et al., 2019), so it is interesting to characterize on which graph families the model can implement a policy solving them.

The 1-WL algorithm iteratively updates each node's label from its current label and those of its neighbors. A graph is *amenable* if its 1-WL multiset distinguishes it from every non-isomorphic graph (Arvind et al., 2017). We call a graph family *cover-amenable* if no two non-isomorphic connected components occurring in the family have isomorphic universal covers, and each is amenable.

On cover-amenable families of graphs, stable rooted 1-WL types determine rooted isomorphism types throughout decoding. This yields the following result. The condition holds, e.g., for families of trees or discrete graphs, which include almost all dense Erdos-Renyi graphs (Babai et al., 1980).

Theorem 3. *Let $n, k \in \mathbb{N}$, and let $\mathcal{G}$ be a cover-amenable family of graphs with at most n nodes. There exists a parameter setting of* SCONE *s.t. every k-colorable (G, c_0) with $G \in \mathcal{G}$ can be solved.*

Note that the result, as common in expressivity results, is in the non-uniform regime, i.e. the model works for graphs of size up to n, but might not generalize to larger graphs. However, for some classes of graphs, we can actually show that SCONE can simulate classical algorithms for graph coloring, and therefore obtain *uniform* results. In particular, we can show that SCONE returns a proper coloring for all graphs in some graph families, even for ones of unbounded size.

Theorem 4. *Let $k, \Delta \in \mathbb{N}$. There exists a parameter setting of* SCONE *s.t. every k-colorable cycle, chordal or bipartite graph with degree at most Δ can be properly colored from the empty coloring.*

5 EXPERIMENTAL RESULTS

In this section, we experimentally evaluate the solution quality and efficiency of SCONE. We evaluate our method and baselines on both real-world graph coloring instances and on controlled synthetic data. Our code and the curated graph-coloring benchmark datasets will be provided publicly.

Datasets We place particular emphasis on graph-coloring instances arising from practical applications. We consider Sudoku instances, register-allocation graphs, and timetabling graphs, which exhibit substantially different graph structures, numbers of colors, and partial-coloring patterns. In addition, we use structured synthetic graph families to study controlled settings in which particular symmetries or expressivity limitations become relevant. See Appendix D.1 for additional details.

Metrics We evaluate each method using the percentage of instances solved (Solved%), the percentage of monochromatic edges in the returned colorings (Mono%), and the average inference time per graph. The solve percentage shows how often a method returns a proper coloring, while the percentage of monochromatic edges measures the solution quality when a proper coloring is not found. Unless specified, we report averages and standard deviations over five independent repeats.

Methods We compare SCONE against both classical methods and neural solvers. As classical baselines, we consider greedy, DSATUR, an ILP formulation with timelimit, and ACE, a frontier CSP solver. We also compare against the learning-based ROS (Qiu et al., 2025), PI-SAGE (Schuetz et al., 2022), ANYCSP (Tönshoff et al., 2023), and CONSFORMER (Xu et al., 2025). For SCONE, we fix the number of layers to $\mathcal{L} = 10$ and the embedding dimension to $d = 128$, which provides

Method	sudoku_17			sudoku_satnet			sudoku_rrn			sudoku_top1465		
	Solved%	Mono%	Time (s)	Solved%	Mono%	Time (s)	Solved%	Mono%	Time (s)	Solved%	Mono%	Time (s)
GREEDY	$0.0_{\pm 0.0}$	$1.83_{\pm 0.00}$	$0.00_{\pm 0.00}$	$1.4_{\pm 0.0}$	$1.48_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.0_{\pm 0.0}$	$1.83_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.0_{\pm 0.0}$	$1.93_{\pm 0.00}$	$0.00_{\pm 0.00}$
DSATUR	$0.1_{\pm 0.0}$	$0.91_{\pm 0.00}$	$0.00_{\pm 0.00}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.00_{\pm 0.00}$	$32.0_{\pm 0.0}$	$0.63_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.7_{\pm 0.0}$	$0.88_{\pm 0.00}$	$0.00_{\pm 0.00}$
ILP (1s)	$97.6_{\pm 0.0}$	$0.01_{\pm 0.00}$	$0.72_{\pm 0.00}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.02_{\pm 0.00}$	$99.7_{\pm 0.0}$	$0.00_{\pm 0.00}$	$0.24_{\pm 0.00}$	$97.9_{\pm 0.0}$	$0.01_{\pm 0.00}$	$0.59_{\pm 0.00}$
ACE	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.51_{\pm 0.00}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.51_{\pm 0.00}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.51_{\pm 0.00}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.53_{\pm 0.00}$
ANYCSP	$26.5_{\pm 1.2}$	$0.18_{\pm 0.00}$	$4.37_{\pm 0.03}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.04_{\pm 0.00}$	$91.3_{\pm 0.6}$	$0.02_{\pm 0.00}$	$0.87_{\pm 0.00}$	$82.7_{\pm 0.2}$	$0.04_{\pm 0.00}$	$1.99_{\pm 0.00}$
CONSFORMER	$4.1_{\pm 0.5}$	$0.24_{\pm 0.00}$	$0.22_{\pm 0.00}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.01_{\pm 0.01}$	$68.0_{\pm 0.3}$	$0.08_{\pm 0.00}$	$0.21_{\pm 0.00}$	$20.5_{\pm 1.1}$	$0.20_{\pm 0.00}$	$0.22_{\pm 0.00}$
OURS	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.01_{\pm 0.00}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.00_{\pm 0.00}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.00_{\pm 0.00}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.02_{\pm 0.00}$

Table 1: Solution quality on Sudoku-graph coloring: Solved% (↑), Mono% (↓) and inference times.

a good trade-off between solution quality and inference time. As aggregation $\oplus$, we use a $\sqrt{\mathcal{N}(u)}$-normalized sum, which retains degree information while controlling the scale of the aggregated messages. We train the model using the AdamW optimizer ($\mathrm{lr} = 10^{-3}$). Checkpoints and the choice between beams and ensembling are selected on validation Solved%. See also Appendix D.3.

5.1 SUDOKUS

Sudoku can be naturally formulated as a graph-coloring problem, with one node for each cell, edges connecting cells in the same row, column, or block, and digits represented as colors. Our main dataset is `sudoku_17`, which contains all known non-equivalent Sudoku puzzles with 17 clues, split into $70\%/15\%/15\%$. We also evaluate our model (without retraining it) on three datasets of increasing difficulty, `satnet`, `rrn`, and `top1465`. Since ANYCSP does not natively support fixed colors, we extended it accordingly and retrained it. PI-SAGE and ROS do not support fixed colors. Table 1 shows that SCONE performs strongly both in distribution and under increasingly difficult distribution shifts. ANYCSP is the strongest competing learning method, but its solution quality deteriorates on the harder instances and its iterative search is roughly two orders of magnitude slower. Classical heuristics struggle, while ILP and ACE have near-perfect quality, but are usually slower.

Motivated by these results, we evaluate SCONE on `sudoku_extreme` (Wang et al., 2025), the most challenging public Sudoku benchmark. Table 2 compares it with the state-of-the-art reasoning models LDT (Davis et al., 2026) and TRM (Jolicoeur-Martineau, 2025), using ANYCSP as a reference point. We use the splits introduced by LDT, so we train all the models on 1000 training instances, and evaluate on 4000 test instances. While the baselines leave some puzzles unsolved, SCONE consistently solves every test instance, with only a few requiring as many as 256 beams. Moreover, despite being a general graph-coloring model, SCONE is at least five times faster than LDT.

Method	sudoku_extreme	
	Solved%	Time (s)
ANYCSP	$79.03_{\pm 0.30}$	$2.02_{\pm 0.02}$
TRM	$94.87_{\pm 0.00}$	$0.02_{\pm 0.00}$
LDT	$99.95_{\pm 0.03}$	$0.12_{\pm 0.01}$
OURS	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$0.02_{\pm 0.00}$

Table 2: Solution quality and inference times on `sudoku_extreme`

5.2 REGISTER ALLOCATION GRAPHS

Register allocation can be formulated as graph coloring. Indeed, nodes represent program variables, edges connect two variables that are used simultaneously and thus cannot occupy the same register, and colors represent registers. A proper coloring gives a conflict-free register allocation.

We evaluate on a filtered dataset of 8999 register-interference graphs collected by George & Appel (1996), split $70\%/15\%/15\%$. See Appendix D.1 for additional details. Rather than using the original budget of 21 registers, we set the palette size for each graph to its chromatic number, thereby testing whether each method can recover an *optimal* coloring. As shown in Table 3, ACE, ANYCSP and SCONE solve every test instance, while the other learning-based methods produce lower-quality solutions. DSATUR and greedy also performs strongly, indicating that the structure of these graphs is well suited to constraint-aware node ordering. Among the learning-based methods, SCONE also has the lowest inference time, at approximately 4 ms per graph.

Method	registers		
	Solved%	Mono%	Time (s)
GREEDY	$90.1_{\pm 0.0}$	$0.09_{\pm 0.00}$	$0.00_{\pm 0.00}$
DSATUR	$99.0_{\pm 0.0}$	$0.01_{\pm 0.00}$	$0.00_{\pm 0.00}$
ILP (1s)	$99.7_{\pm 0.0}$	$0.01_{\pm 0.00}$	$0.02_{\pm 0.00}$
ACE	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.41_{\pm 0.00}$
PI-SAGE	$53.8_{\pm 2.0}$	$0.52_{\pm 0.02}$	$2.85_{\pm 0.01}$
ROS	$35.5_{\pm 1.0}$	$0.70_{\pm 0.01}$	$0.39_{\pm 0.00}$
ANYCSP	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.02_{\pm 0.00}$
CONSFORMER	$80.5_{\pm 0.8}$	$0.51_{\pm 0.01}$	$0.13_{\pm 0.01}$
OURS	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.00_{\pm 0.00}$

Table 3: Solution quality on `registers`: Solved% (↑), Mono (↓) and inference times.

Method	toronto (small)			toronto (large)			queens			cycles		
	Solved%	Mono%	Time (s)	Solved%	Mono%	Time (s)	Solved%	Mono%	Time (s)	Solved%	Mono%	Time (s)
GREEDY	$0.0_{\pm 0.0}$	$0.40_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.0_{\pm 0.0}$	$0.35_{\pm 0.00}$	$0.01_{\pm 0.00}$	$0.0_{\pm 0.0}$	$1.44_{\pm 0.00}$	$0.00_{\pm 0.00}$	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{0.00}_{\pm 0.00}$	$0.00_{\pm 0.00}$
DSATUR	$50.0_{\pm 0.0}$	$0.02_{\pm 0.00}$	$0.00_{\pm 0.00}$	$20.0_{\pm 0.0}$	$0.04_{\pm 0.00}$	$0.02_{\pm 0.00}$	$0.0_{\pm 0.0}$	$1.24_{\pm 0.00}$	$0.00_{\pm 0.00}$	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{0.00}_{\pm 0.00}$	$0.00_{\pm 0.00}$
ILP (600s)	$50.0_{\pm 0.0}$	$0.05_{\pm 0.00}$	$328.9_{\pm 0.00}$	$20.0_{\pm 0.0}$	$0.56_{\pm 0.00}$	$548.8_{\pm 0.00}$	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{0.00}_{\pm 0.00}$	$170.8_{\pm 0.00}$	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{0.00}_{\pm 0.00}$	$0.05_{\pm 0.00}$
ACE	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{0.00}_{\pm 0.00}$	$0.71_{\pm 0.01}$	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{0.00}_{\pm 0.00}$	$65.98_{\pm 1.60}$	$66.7_{\pm 0.0}$	–	$206.5_{\pm 0.04}$	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{0.00}_{\pm 0.00}$	$0.45_{\pm 0.00}$
PI-SAGE	$0.0_{\pm 0.0}$	$0.66_{\pm 0.03}$	$7.09_{\pm 0.22}$	$0.0_{\pm 0.0}$	$0.58_{\pm 0.01}$	$14.07_{\pm 0.86}$	$0.0_{\pm 0.0}$	$1.79_{\pm 0.12}$	$5.65_{\pm 0.37}$	$53.1_{\pm 0.0}$	$4.30_{\pm 0.02}$	$3.89_{\pm 0.01}$
ROS	$0.0_{\pm 0.0}$	$0.52_{\pm 0.04}$	$0.66_{\pm 0.05}$	$0.0_{\pm 0.0}$	$0.53_{\pm 0.02}$	$0.82_{\pm 0.06}$	$0.0_{\pm 0.0}$	$1.36_{\pm 0.14}$	$0.69_{\pm 0.05}$	$53.1_{\pm 0.0}$	$3.96_{\pm 0.04}$	$0.78_{\pm 0.00}$
ANYCSP	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{0.00}_{\pm 0.00}$	$0.94_{\pm 0.04}$	$70.0_{\pm 14.8}$	$0.01_{\pm 0.00}$	$25.91_{\pm 1.45}$	$96.6_{\pm 10.0}$	$\mathbf{0.00}_{\pm 0.01}$	$3.96_{\pm 2.41}$	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{0.00}_{\pm 0.00}$	$0.19_{\pm 0.01}$
CONSFORMER	$0.0_{\pm 0.0}$	$2.06_{\pm 0.06}$	$24.83_{\pm 0.10}$	$0.0_{\pm 0.0}$	$1.93_{\pm 0.02}$	$24.77_{\pm 0.14}$	$11.7_{\pm 19.6}$	$1.32_{\pm 0.19}$	$23.17_{\pm 2.93}$	$61.5_{\pm 3.2}$	$0.32_{\pm 0.03}$	$2.12_{\pm 0.01}$
OURS	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{0.00}_{\pm 0.00}$	$0.76_{\pm 0.00}$	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{0.00}_{\pm 0.00}$	$4.88_{\pm 0.82}$	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{0.00}_{\pm 0.00}$	$0.54_{\pm 0.14}$	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{0.00}_{\pm 0.00}$	$0.04_{\pm 0.00}$

Table 4: Solution quality on isolated instances: Solved% (↑), Mono% (↓) and inference time (↓).

5.3 GENERALIZATION FROM SYNTHETIC TRAINING

Many important coloring benchmarks contain only a few isolated instances and therefore provide no natural training distribution. To evaluate transfer in this setting, we train a single model exclusively on synthetic data, as described in Sec. 4.2, and apply it without finetuning to multiple instance families. The Toronto timetabling graphs (Carter et al., 1996) represent exams as nodes, conflicts due to common students as edges, and timeslots as colors. We aggregate 4 smaller instances and 5 larger instances. We also consider queen graphs (Vasquez, 2004), for $p \in \{8, 9, 10\}$, where nodes represent squares of a $p \times p$ chessboard and edges connect squares sharing a row, column or diagonal.

Table 4, using 20 repeats, shows that the synthetically trained model transfers reliably across these families, with SCONE being the only learned method to solve every instance. On the smaller Toronto instances, SCONE matches the 100% solve rate of ANYCSP while being slightly faster. The gap widens on the larger instances, where SCONE solves every graph, while ANYCSP fails on some and takes five times longer. On queen graphs, SCONE and ANYCSP both solve all instances, with the former being more consistent and faster. Overall, ACE also achieves near-perfect solution quality.

We test the same model on structured synthetic families that expose potential failure modes, reporting cycle graphs here and full results in Table 9. With greedy decoding alone, SCONE solves every graph with fast inference times. ANYCSP also succeeds but is slower, while other models struggle.

5.4 ABLATION STUDY

We study the effect of equivariance by comparing SCONE and a variant without color equivariance, with and without data augmentation by color permutations. First, we train on increasing subsets of `sudoku_17`, see Fig. 3 (a). With only $N = 100$ training instances, SCONE solves 49% of the test set, compared with just 2% for the non-equivariant model. Data augmentation narrows the gap in the low-data regime, but even at the largest subset it remains substantial, highlighting the greater sample efficiency of architectural equivariance. Second, we generate datasets *targeting* increasing chromatic numbers χ. While the non-equivariant variants are competitive for small k, their performance degrades as k increases, see Fig. 3 (b). Data augmentation provides no improvement, and the gap to SCONE widens with k, indicating that color equivariance becomes increasingly important as k grows, as suggested by Thm. 1.

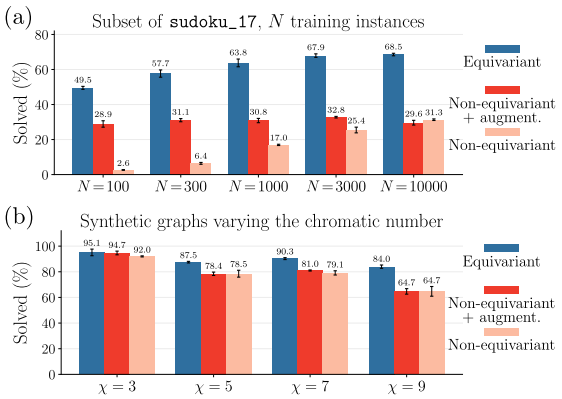

Figure 3: Ablation on Sudoku and synthetic graphs.

6 CONCLUSIONS

We introduced SCONE, a graph-coloring method combining color equivariance with sequential decoding, as supported by our theory. Across real-world and synthetic benchmarks, it achieves near-perfect solution quality with low inference times. As future work, these positive results motivate extending SCONE to a broader range of practical coloring-based applications in operations research, as well as studying which symmetries can be exploited in other combinatorial problems.

AI USE STATEMENT

In this work, we used generative AI tools to: assist in the writing of proofs, provide feedback on research methodology or experiments, propose or refine hypotheses, implement methods.

We have not used generative AI tools to: generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, design the research methodology or experiments, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results, create or modify scientific figures or images, draft parts of a research paper.

Additionally, we used generative AI tools to: create or edit software code, edit the paper to improve readability, identify relevant literature, propose a title or keywords for a research paper.

We have reviewed all AI-assisted work. We have implemented the core parts of the model ourselves, and verified all the code for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This paper presents work whose goal is to advance the field of Machine Learning. There are potential and indirect societal consequences of our work, none of which we feel must be specifically highlighted here.

REPRODUCIBILITY STATEMENT

For all our theoretical results, clear explanations of any assumptions and a complete proof of the claims are included in Appendix C. Descriptions and references for datasets and baselines are reported in Sections D.1 and D.2 respectively. A description of our model's implementation and of the hyperparameters we used is reported in Section D.3. Section D.4 describes our computing infrastructure to ensure reproducible inference times. The code, checkpoints and datasets to reproduce our results are available as supplementary material, and will be made publicly available on GitHub upon acceptance.

REFERENCES

Kenneth Appel and Wolfgang Haken. Every planar map is four colorable. part i: Discharging. *Illinois Journal of Mathematics*, 21(3):429–490, 1977.

Stefan Arnborg and Andrzej Proskurowski. Linear time algorithms for np-hard problems restricted to partial k-trees. *Discrete applied mathematics*, 23(1):11–24, 1989.

Vikraman Arvind, Johannes Köbler, Gaurav Rattan, and Oleg Verbitsky. Graph isomorphism, color refinement, and compactness. *computational complexity*, 26(3):627–685, 2017.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

László Babai, Paul Erdos, and Stanley M Selkow. Random graph isomorphism. *SIaM Journal on computing*, 9(3):628–635, 1980.

Thomas Barrett, William Clements, Jakob Foerster, and Alex Lvovsky. Exploratory combinatorial optimization with reinforcement learning. In *AAAI*, 2020.

Richard Beigel and David Eppstein. 3-coloring in time o (1.3289 n). *Journal of Algorithms*, 54(2): 168–204, 2005.

Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2): 405–421, 2021.

Daniel Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4): 251–256, 1979.

Rowland Leonard Brooks. On colouring the nodes of a network. *Mathematical Proceedings of the Cambridge Philosophical Society*, 37(2):194–197, 1941.

Quentin Cappart, Didier Chételat, Elias B Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks. *Journal of Machine Learning Research*, 24(130):1–61, 2023.

Michael W Carter, Gilbert Laporte, and Sau Yan Lee. Examination timetabling: Algorithmic strategies and applications. *Journal of the operational research society*, 47(3):373–383, 1996.

Gregory J Chaitin, Marc A Auslander, Ashok K Chandra, John Cocke, Martin E Hopkins, and Peter W Markstein. Register allocation via coloring. *Computer languages*, 6(1):47–57, 1981.

James Crawford, Matthew Ginsberg, Eugene Luks, and Amitabha Roy. Symmetry-breaking predicates for search problems. *KR*, 96(1996):148–159, 1996.

Liam Davis, Leopold Haller, Alberto Alfarano, and Mark Santolucito. Lattice deduction transformers. *arXiv preprint arXiv:2605.08605*, 2026.

Darko Drakulic, Sofia Michel, Florian Mai, Arnaud Sors, and Jean-Marc Andreoli. Bq-nco: Bisimulation quotienting for efficient neural combinatorial optimization. In *NeurIPS*, 2023.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Michael R Garey, David S Johnson, and Larry Stockmeyer. Some simplified np-complete problems. In *ACM STOC*, pp. 47–63, 1974.

Lal George and Andrew W Appel. Iterated register coalescing. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 18(3):300–324, 1996.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.

Ronald L Graham, Donald Ervin Knuth, and Oren Patashnik. *Concrete mathematics: a foundation for computer science*. Addison-Wesley Reading, MA, 1994.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024. URL `https://www.gurobi.com`.

Robert M. Haralick and Gordon L. Elliott. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14(3):263–313, 1980.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Alain Hertz and Dominique de Werra. Using tabu search techniques for graph coloring. *Computing*, 39(4):345–351, 1987.

Agnes M Herzberg and M Ram Murty. Sudoku squares and chromatic polynomials. *Notices of the AMS*, 54(6):708–717, 2007.

Marijn J Heule, Anthony Karahalios, and Willem-Jan van Hoeve. From cliques to colorings and back again. In *International Conference on Principles and Practice of Constraint Programming*, 2022.

Jiayi Huang, Mostofa Patwary, and Gregory Diamos. Coloring big graphs with alphagozero. *arXiv preprint arXiv:1902.10162*, 2019.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv preprint arXiv:2510.04871*, 2025.

Nikolaos Karalias, Akbar Rafiey, Yifei Xu, Zhishang Luo, Behrooz Tahmasebi, Connie Jiang, and Stefanie Jegelka. Geometric algorithms for neural combinatorial optimization with constraints. In *NeurIPS*, 2025.

Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *NeurIPS*, 2017.

Minsu Kim, Junyoung Park, and Jinkyoo Park. Sym-nco: Leveraging symmetricity for neural combinatorial optimization. In *NeurIPS*, 2022.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Andreas Krebs and Oleg Verbitsky. Universal covers, color refinement, and two-variable counting logic: Lower bounds for the depth. In *ACM/IEEE Symposium on Logic in Computer Science*, 2015.

Thien Le, Tianyu Zhao, and Melanie Weber. Contrastive neural algorithmic reasoning for graph coloring. *arXiv preprint arXiv:2606.03923*, 2026.

Christophe Lecoutre. Ace, a generic constraint solver. *arXiv preprint arXiv:2302.05405*, 2023.

Henrique Lemos, Marcelo Prates, Pedro Avelar, and Luis Lamb. Graph colouring meets deep learning: Effective graph neural network models for combinatorial problems. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 879–885. IEEE, 2019.

Wei Li, Ruxuan Li, Yuzhe Ma, Siu On Chan, David Pan, and Bei Yu. Rethinking graph neural networks for the graph coloring problem. *arXiv preprint arXiv:2208.06975*, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Can classic gnns be strong baselines for graph-level tasks? simple architectures meet excellence. In *ICML*, 2025.

Enrico Malaguti and Paolo Toth. A survey on vertex coloring problems. *International transactions in operational research*, 17(1):1–34, 2010.

Gary McGuire, Bastian Tugemann, and Gilles Civario. There is no 16-clue sudoku: Solving the sudoku minimum number of clues problem via hitting set enumeration. *Experimental Mathematics*, 23(2):190–217, 2014.

Isabel Méndez-Díaz and Paula Zabala. A branch-and-cut algorithm for graph coloring. *Discrete Applied Mathematics*, 154(5):826–847, 2006.

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 *AAAI*, 2019.

Rasmus Palm, Ulrich Paquet, and Ole Winther. Recurrent relational networks. In *NeurIPS*, volume 31, 2018.

Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.

Jean-François Puget. An efficient way of breaking value symmetries. In *AAAI*, 2006.

Yeqing Qiu, Ye XUE, Akang Wang, Yiheng Wang, Qingjiang Shi, and Zhi-Quan Luo. ROS: A GNN-based relax-optimize-and-sample framework for max-k-cut problems. In *ICML*, 2025.

Rong Qu, Edmund K Burke, Barry McCollum, Liam TG Merlot, and Sau Y Lee. A survey of search methodologies and automated system development for examination timetabling. *Journal of scheduling*, 12(1):55–89, 2009.

Neil Robertson, Daniel Sanders, Paul Seymour, and Robin Thomas. The four-colour theorem. *journal of combinatorial theory, Series B*, 70(1):2–44, 1997.

Joseph J Rotman. *An introduction to the theory of groups*. Springer Science & Business Media, 2012.

Martin JA Schuetz, J Kyle Brubaker, Zhihuai Zhu, and Helmut G Katzgraber. Graph coloring with physics-inspired graph neural networks. *Physical Review Research*, 4(4):043131, 2022.

Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. Learning a SAT solver from single-bit supervision. In *ICLR*, 2019.

Robert E Tarjan and Mihalis Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on computing*, 13(3):566–579, 1984.

Jan Toenshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Graph neural networks for maximum constraint satisfaction. *Frontiers in artificial intelligence*, 3:580607, 2021.

Jan Tönshoff, Berke Kisin, Jakob Lindner, and Martin Grohe. One model, any csp: graph neural networks as fast global search heuristics for constraint satisfaction. In *IJCAI*, 2023.

Knut Vanderbush and Melanie Weber. Neural algorithmic reasoning for approximate k-coloring with recursive warm starts. *Machine Learning: Science and Technology*, 7(3):035054, 2026.

Michel Vasquez. New results on the queens_ n 2 graph coloring problem. *Journal of Heuristics*, 10 (4):407–413, 2004.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *NeurIPS*, 2015.

Toby Walsh. Symmetry breaking constraints: Recent results. In *AAAI*, 2012.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model. *arXiv preprint arXiv:2506.21734*, 2025.

Po-Wei Wang, Priya Donti, Bryan Wilder, and Zico Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *ICML*, 2019.

George Watkins, Giovanni Montana, and Juergen Branke. Generating a graph colouring heuristic with deep q-learning and graph neural networks. In *International conference on learning and intelligent optimization*, pp. 491–505. Springer, 2023.

Dominic JA Welsh and Martin B Powell. An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 10(1):85–86, 1967.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

Yudong Xu, Wenhao Li, Scott Sanner, and Elias Boutros Khalil. Self-supervised transformers as iterative solution improvers for constraint satisfaction. In *ICML*, 2025.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander Smola. Deep sets. In *NeurIPS*, 2017.

APPENDIX

A LIMITATIONS AND FUTURE WORK

While SCONE achieves state-of-the-art performance, there are some limitations, which suggest promising directions for future work. For example, SCONE constructs a coloring sequentially and therefore requires up to $O(n)$ forward passes for a graph with n nodes, which limits the applicability to larger graphs. A promising direction is to color multiple compatible nodes speculatively, retaining their assignments when they are consistent and revising them otherwise. Moreover, SCONE is an approximate solver. It always returns a complete coloring but cannot guarantee that it is proper or certify that no proper coloring exists for the given palette. Combining the model with backtracking or an exact solver could provide such guarantees while retaining its ability to identify promising assignments.

B ADDITIONAL RELATED WORK

Classical graph coloring. The k-colorability problem is NP-complete for every fixed $k \geq 3$ (Garey et al., 1974), while it can be solved in linear time for $k = 2$. Exact algorithms can however be faster than enumerating all k^n assignments. For example, 3-colorability can be decided in $O(1.33^n)$ time using reductions to constraint satisfaction (Beigel & Eppstein, 2005). Moreover, several graph classes admit polynomial-time algorithms. Chordal graphs can be optimally colored in linear time (Tarjan & Yannakakis, 1984). Graphs of bounded treewidth can also be colored efficiently by dynamic programming over a tree decomposition (Arnborg & Proskurowski, 1989). Brooks' theorem shows that every connected graph of maximum degree Δ can be colored with Δ colors unless it is a complete graph or an odd cycle (Brooks, 1941). The four-color theorem guarantees that every planar graph admits a coloring with at most four colors (Appel & Haken, 1977), and the later proof of Robertson et al. (1997) also yields a quadratic-time coloring algorithm.

Neural combinatorial optimization on graphs. Other works have addressed the graph coloring problem with machine learning. Watkins et al. (2023) propose a constructive heuristic that uses a GNN-based deep Q-network to select the next node, while a fixed rule assigns it the lowest permissible color. Huang et al. (2019) target very large graphs, and combine Monte Carlo tree search with a model that predicts a distribution over valid colors for the current node. Lemos et al. (2019) train a GNN to predict whether a graph is k-colorable, representing both nodes and candidate colors within the message-passing graph, and colorings are recovered indirectly by clustering the learned node embeddings. Fianlly, (Li et al., 2022) also identify node symmetry as a limitation of simultaneous GNN-based coloring and address it through randomly initialized node features. Their model is also color-equivariant, but uses a substantially more restricted class of linear equivariant updates. In contrast, SCONE employs expressive nonlinear equivariant layers and breaks node symmetries sequentially, as previous color assignments become part of the state.

Neural combinatorial optimization has mostly focused on routing problems (Vinyals et al., 2015; Bello et al., 2016; Drakulic et al., 2023), but it has also been applied to several graph problems closely related to coloring, such as Max-Cut (Karalias et al., 2025). Khalil et al. (2017) learn sequential construction policies for Max-Cut, minimum vertex cover and TSP using reinforcement learning, and Barrett et al. (2020) learn a local-search policy that may revise previous decisions at inference time. See also Cappart et al. (2023) for an overview of the field.

Symmetry in combinatorial reasoning. Symmetry has long been exploited in constraint programming and SAT to reduce redundant search. For example, Lex-leader constraints select one representative from each orbit (Crawford et al., 1996), while specialized methods handle interchangeable values (Puget, 2006; Walsh, 2012). Symmetry-aware architectures have also been used for neural combinatorial optimization, notably in NEUROSAT (Selsam et al., 2019), which respects the natural permutations of SAT formulas by encoding them as graphs and using a message-passing mechanism. Methods for more general constraint-satisfaction problems based on message-passing paradigms like RUN-CSP (Toenshoff et al., 2021) and ANYCSP (Tönshoff et al., 2023) also inherit equivariance to the natural permutations of the constraint graphs. Finally, Kim et al. (2022) introduce a regularization-based training scheme that exploits problem and solution symmetries in neural routing problems, including TSP and CVRP.

C PROOFS

C.1 SECTION 3

Theorem 5 (Expressivity of sequential coloring). *Fix $n, k \in \mathbb{N}$, and let $s_\theta(G, c, u) \in \mathbb{R}^k$ denote the color scores assigned to an uncolored node u. Assume that the predictor is universal over rooted isomorphism types of size at most n. Consider any sequential decoder that repeatedly selects an arbitrary uncolored node u, chooses $i \in \arg\max_{j \in \mathcal{C}} [s_\theta(G, c, u)]_j$, and assigns color i to u, breaking ties arbitrarily. Then there exists a single parameter setting such that, for every extendable partial k-coloring (G, c_0) with $|V_G| \le n$, every decoding trajectory yields a proper coloring extending c_0.*

Proof. For an uncolored node u, let $c^{u \leftarrow i}$ denote the partial coloring obtained by assigning color i to u. Define the target score function as $s(G, c, u)_i = 1$ if $(G, c^{u \leftarrow i})$ is extendable and 0 otherwise.

This function depends only on the rooted partial-coloring isomorphism type of (G, c, u). In fact, it is also color-equivariant. There are finitely many rooted partial-coloring isomorphism types with at most n nodes. By the assumed universality, a single parameter setting θ realizes this score function on all of them.

Now consider any extendable state (G, c) and any uncolored node u. Let $c^\star$ be a proper completion of c. Assigning $i = c^\star(u)$ to u preserves extendability, so the maximum score at u is 1. Consequently, every maximizing color preserves extendability.

The selected node and all score ties may therefore be resolved arbitrarily. Every decoding step preserves extendability and colors one additional node. By induction, decoding reaches a complete extendable state, which is a proper coloring extending c_0. □

Lemma 1. *Consider a state $(G, c) \in \bigcup_{\tau \in S_k} \mathcal{S}_{\tau \cdot (G, c_0)}$ with exactly r colors appearing in c. Let $R(G, c) = |S_k \cdot (G, c)|$ denote the size of the orbit of (G, c) under color permutations. Then,*

$$\frac{k!}{(k-r)!\,\min(|\mathrm{Aut}(G)|, r!)} \le R(G, c) \le \frac{k!}{(k-r)!}.$$

Proof. Let $C_1, \ldots, C_r$ be the used colors, and let $\Pi_c \le S_r$ be the group of permutations of the used colors that can be realized by automorphisms of G, that is,

$$\Pi_c = \left\{ \tau \in S_r : \exists\, \varphi \in \mathrm{Aut}(G) \text{ s.t. } c(v) = C_i \Rightarrow c(\varphi(v)) = C_{\tau(i)} \ \ \forall v \in V_G,\ \forall i = 1, \ldots, r \right\}.$$

A permutation $\tau \in S_k$ stabilizes the state (G, c) if the partial coloring obtained by relabeling the colors according to τ is isomorphic to (G, c). Hence, on the r used colors, τ must induce a permutation belonging to Π_c, while the remaining $k - r$ unused colors may be permuted arbitrarily. Therefore, $|\mathrm{Stab}_{S_k}(G, c)| = |\Pi_c|(k-r)!$. By the orbit-stabilizer theorem (Rotman, 2012, Chapter 3), the orbit $S_k \cdot (G, c)$ of (G, c) has cardinality

$$R(G, c) = \frac{|S_k|}{|\mathrm{Stab}_{S_k}(G, c)|} = \frac{k!}{(k-r)!\,|\Pi_c|}.$$

Since $1 \le |\Pi_c| \le \min(|\mathrm{Aut}(G)|, r!)$, we have the claim. □

In the main paper, we claim that, for Sudoku puzzles, a careful application of the lemma yields a reduction in the state space size by a factor of $R(G, c) = \frac{9!}{|\Pi_c|} \ge \frac{9!}{648} = 560$. This requires first the following lemma.

Lemma 2 (Automorphisms of a uniquely completable Sudoku). *Let G, c be a 9×9 Sudoku puzzle with unique completion $c^\star$, and let Π_c be as defined in Lemma 1. Then*

$$|\Pi_c| \le |\Pi_{c^\star}| \le 648.$$

Proof. First, c must contain 8 or 9 distinct digits. Otherwise, two digits would be absent from c, and swapping those digits in $c^\star$ would produce a second completion of c. Therefore, if r is the number of digits occurring in c, then $r \in \{8, 9\}$.

Let $\tau \in \Pi_c$. By the definition of Π_c, there exists $\phi \in \mathrm{Aut}(G)$ such that $c(\phi(v)) = \tau(c(v))$ for every colored vertex v. Note that even if $r = 8$, the permutation τ has a unique extension in S_9 by fixing the remaining digit.

Define a completed coloring c' by $c'(\phi(v)) = \tau(c^\star(v))$. The automorphism ϕ preserves the Sudoku constraint graph, and τ only relabels the digits, so c' is a valid completed Sudoku grid.

Let V_c be the set of colored nodes and note that $\phi(V_c) = V_c$. Now let $x \in V_c$. Since $\phi(V_c) = V_c$, there is some $v \in V_c$ such that $x = \phi(v)$. Because $c^\star$ extends c, we have

$$c'(x) = \tau(c^\star(v)) = \tau(c(v)) = c(\phi(v)) = c(x).$$

Hence c' extends c. Since c has the unique completion $c^\star$, it follows that $c' = c^\star$. Therefore, $c^\star(\phi(v)) = \tau(c^\star(v))$ for every $v \in V_G$. Hence $\tau \in \Pi_{c^\star}$ and $|\Pi_c| \leq |\Pi_{c^\star}|$.

Let $\Gamma_{c^\star}$ denote the full automorphism group of the completed grid, whose elements are the pairs $(\phi, \sigma) \in \mathrm{Aut}(G) \times S_9$ satisfying $c^\star(\phi(v)) = \sigma(c^\star(v))$ for every v. The projection of $\Gamma_{c^\star}$ onto its digit-permutation component has image $\Pi_{c^\star}$. Therefore, $|\Pi_{c^\star}| \leq |\Gamma_{c^\star}|$. Finally, McGuire et al. (2014) argue that $|\Gamma_{c^\star}| \leq 648$, which proves the result. □

Theorem 1 (Exponential reduction in the state space)**.** *Consider a finite k-colorability instance set $\mathcal{I}$ with no starting colors. Let $|\mathrm{Aut}(G)| \leq B \; \forall G \in \mathcal{I}$. Let $\mathcal{S}_\mathcal{I}$ be its associated state space and $\mathcal{S}^+_\mathcal{I}$ the space of extendable states. Then:*

1. *If all graphs $G \in \mathcal{I}$ have more than k nodes, then $|\mathcal{S}_\mathcal{I}/S_k| \leq B \, \dfrac{16}{e^k} \, |\mathcal{S}_\mathcal{I}|$.*
2. *If all graphs $G \in \mathcal{I}$ satisfy $0 \leq k - \chi(G) \leq \kappa$, then $|\mathcal{S}^+_\mathcal{I}/S_k| \leq B \, \dfrac{\kappa! \, 2^{k-\kappa}}{k!} \, |\mathcal{S}^+_\mathcal{I}|$.*

Proof. Let $\tilde{\mathcal{S}}_G$ be the state space before quotienting by graph isomorphisms, i.e. where each node has an unique identifier. We then have that $\mathcal{S}_G = \tilde{\mathcal{S}}_G/\mathrm{Aut}(G)$ by definition, and since $\mathrm{Aut}(G)$ commutes with the color permutations S_k, we have that $\mathcal{S}_G/S_k = \tilde{\mathcal{S}}_G/(\mathrm{Aut}(G) \times S_k)$. Moreover, $|\tilde{\mathcal{S}}_G| \leq |\mathrm{Aut}(G)| \, |\tilde{\mathcal{S}}_G/\mathrm{Aut}(G)|$ as every $\mathrm{Aut}(G)$-orbit can only have size at most $|\mathrm{Aut}(G)|$, and $\tilde{\mathcal{S}}_G/(\mathrm{Aut}(G) \times S_k) \leq \tilde{\mathcal{S}}_G/S_k$ as quotienting by $\mathrm{Aut}(G)$ can only decrease the number of elements. Therefore, we have that

$$\frac{|\mathcal{S}_G|}{|\mathcal{S}_G/S_k|} = \frac{|\tilde{\mathcal{S}}_G/\mathrm{Aut}(G)|}{|\tilde{\mathcal{S}}_G/(\mathrm{Aut}(G) \times S_k)|} \geq \frac{|\tilde{\mathcal{S}}_G|}{|\tilde{\mathcal{S}}_G/S_k| \, |\mathrm{Aut}(G)|} \geq \frac{1}{B} \frac{|\tilde{\mathcal{S}}_G|}{|\tilde{\mathcal{S}}_G/S_k|}.$$

The same derivation follows for the extendable versions, yielding $\frac{|\mathcal{S}^+_G|}{|\mathcal{S}^+_G/S_k|} \geq \frac{1}{B} \frac{|\tilde{\mathcal{S}}^+_G|}{|\tilde{\mathcal{S}}^+_G/S_k|}$. Since the $1/B$ factor does not depend on G, we can get the same factor uniformly over $\mathcal{I}$. Therefore, we will work with the states with identified nodes, which ensures that coloring different nodes leads to different states.

We first prove the reduction for $\tilde{\mathcal{S}}_\mathcal{I}$. Let's consider an graph G with n nodes. In the resulting state space $\tilde{\mathcal{S}}_G$, each of those nodes can be either uncolored or have one of the k available colors, leading to a total of $(k+1)^n$ possible states. The quotient space $\tilde{\mathcal{S}}_G/S_k$ ignores the identities of the colors. Therefore, to count the number of states using r colors, we count the number of partitions of n elements (the nodes) into r equivalent non-empty classes (the colors) and one distinct and possibly empty class (the uncolored nodes). To count these, we use the following construction. We add a dummy node, and we say that whichever class this node is assigned to, is the class of uncolored nodes. We can then count the number of partitions of $n+1$ elements into $r+1$ equivalent and non-empty classes, which is the Stirling number of second kind $\left\{ {n+1 \atop r+1} \right\}$ (Graham et al., 1994). We then have that $|\tilde{\mathcal{S}}_G/S_k| = \sum_{r=0}^{k} \left\{ {n+1 \atop r+1} \right\}$. Therefore, we obtain the exact formula $R := |\tilde{\mathcal{S}}_G| \, / \, |\tilde{\mathcal{S}}_G/S_k| = \frac{(k+1)^n}{\sum_{r=0}^{k} \left\{ {n+1 \atop r+1} \right\}}$.

Then, a standard bound yields that $\left\{ {n+1 \atop r+1} \right\} \leq \frac{(r+1)^n}{r!}$, which then implies a reduction of factor

$$R \geq \left(\sum_{r=0}^{k} \frac{1}{r!} \left(\frac{r+1}{k+1} \right)^n \right)^{-1} \geq \left(\sum_{r=0}^{k} \frac{1}{r!} \left(1 - \frac{k-r}{k+1} \right)^{k+1} \right)^{-1} \geq \left(\sum_{r=0}^{k} \frac{e^{r-k}}{r!} \right)^{-1} = e^k \left(\sum_{r=0}^{k} \frac{e^r}{r!} \right)^{-1} \geq \frac{e^k}{e^e}.$$

Taking the bound over all $G \in \mathcal{I}$ yields the claim.

We then prove the reduction for $\tilde{\mathcal{S}}^+_{\mathcal{I}}$. Let's consider a graph G with $\chi(G) = q$. Let E_r be the number of extendable quotient states (i.e., states in $\tilde{\mathcal{S}}^+_G/S_k$) that use exactly r colors. If $r < q$, then at least $q - r$ colors absent from the current state occur on uncolored vertices. Choosing one vertex carrying each such color produces at least $q - r$ distinct $(r + 1)$-color successor states. On the other hand, each $r + 1$-color state has an r-color predecessor of this type only by uncoloring the unique vertex of a singleton color class. Hence it has at most $r + 1$ predecessors. This implies the inequality $(q - r)E_r \leq (r + 1)E_{r+1}$. Importantly, if we didn't work on $\tilde{\mathcal{S}}$ and didn't have the unique node identifiers, several such successors may collapse to the same quotient state under graph automorphisms, so the lower bound $(q - r)$ on the number of distinct successors could fail. We therefore have

$$\frac{E_r}{\binom{q}{r}} \leq \frac{E_{r+1}}{\binom{q}{r+1}} \quad \forall\, 0 \leq r < q\,,$$

i.e., $a_r := E_r/\binom{q}{r}$ is nondecreasing in r.

We claim that, for every $R \leq q$,

$$\frac{\sum_{r=0}^{R} E_r}{\sum_{r=0}^{k} E_r} \leq \frac{\sum_{r=0}^{R} E_r}{\sum_{r=0}^{q} E_r} \leq 2^{-q} \sum_{r=0}^{R} \binom{q}{r}.$$

To prove it, set $A := \sum_{r=0}^{R} a_r \binom{q}{r}$, $B := \sum_{r=R+1}^{q} a_r \binom{q}{r}$, $C := \sum_{r=0}^{R} \binom{q}{r}$, and $D := \sum_{r=R+1}^{q} \binom{q}{r}$. Then it suffices to show

$$\frac{A}{A + B} \leq \frac{C}{C + D},$$

or equivalently, $AD \leq BC$.

Since $a_r \leq a_R$ for every $r \leq R$, we have

$$A = \sum_{r=0}^{R} a_r \binom{q}{r} \leq a_R \sum_{r=0}^{R} \binom{q}{r} = a_R C.$$

Similarly, since $a_r \geq a_R$ for every $r > R$,

$$B = \sum_{r=R+1}^{q} a_r \binom{q}{r} \geq a_R \sum_{r=R+1}^{q} \binom{q}{r} = a_R D.$$

Multiplying the first inequality by D and the second by C we obtain $AD \leq a_R CD \leq BC$. We then have that

$$\frac{\sum_{r=0}^{R} a_r \binom{q}{r}}{\sum_{r=0}^{q} a_r \binom{q}{r}} \leq \frac{\sum_{r=0}^{R} \binom{q}{r}}{\sum_{r=0}^{q} \binom{q}{r}}.$$

Finally, using $\sum_{r=0}^{q} \binom{q}{r} = 2^q$, we obtain

$$\frac{\sum_{r=0}^{R} E_r}{\sum_{r=0}^{k} E_r} \leq \frac{\sum_{r=0}^{R} E_r}{\sum_{r=0}^{q} E_r} \leq 2^{-q} \sum_{r=0}^{R} \binom{q}{r}.$$

Therefore the distribution with density $f(r) = \frac{E_r}{\sum_{j=0}^{k} E_j}$ stochastically dominates $\mathrm{Bin}(q, 1/2)$, which stochastically dominates $\mathrm{Bin}(k - \kappa, 1/2)$, for $\kappa \in \{0, \ldots, k\}$. Since $0 \leq k - \chi(G) \leq \kappa \leq k$, this is well defined. Since $\frac{k!}{(k-r)!}$ is nondecreasing in r, we then have that

$$\frac{|\tilde{\mathcal{S}}^+_G|}{|\tilde{\mathcal{S}}^+_G/S_k|} = \frac{\sum_{r=0}^{k} E_r \frac{k!}{(k-r)!}}{\sum_{r=0}^{k} E_r} = \sum_{r=0}^{k} \frac{E_r}{\sum_{j=0}^{k} E_j} \frac{k!}{(k-r)!} \geq \sum_{r=0}^{q} \frac{1}{2^q} \binom{q}{r} \frac{k!}{(k-r)!} \geq \sum_{r=0}^{k-\kappa} \frac{1}{2^{k-\kappa}} \binom{k-\kappa}{r} \frac{k!}{(k-r)!}.$$

If we take only the $r = k - \kappa$ term, we obtain a $\frac{k!}{2^{k-\kappa}\kappa!}$ bound. Taking the bound over all instances then yields the claim.

□

C.2 SECTION 4

We prove that the color predictor of SCONE is equivariant to node and color permutations. Importantly, this guarantee applies only to the predicted color scores. We do not require tie-breaking among colors or the complete decoding procedure to be equivariant.

Theorem 2. *The color predictor of* SCONE *is equivariant to the joint action of $S_{|V_G|} \times S_k$. Let (G, c) be any graph k-coloring instance, and let $p_u(G, c) \in \mathbb{R}^k$ be the output of the predictor for node u. Then, for every parameter setting, every $\pi \in S_{|V_G|}$, every $\tau \in S_k$, and every $u \in V_G$,*

$$p_{\pi(u)}\big(\pi \cdot G,\ \tau \cdot (\pi \cdot c)\big) = P_\tau\, p_u(G, c).$$

Proof. Let $(G', c') = (\pi \cdot G, \tau \cdot (\pi \cdot c))$. We show by induction over the message-passing layers that

$$H^\ell_{\pi(u)}(G', c') = P_\tau H^\ell_u(G, c) \qquad \forall u \in V_G.$$

For $\ell = 0$, if $c(u) = \varepsilon$, then $c'(\pi(u)) = \varepsilon$, and both sides contain e_{unk} in every color slot. Otherwise, $c'(\pi(u)) = \tau(c(u))$. Hence the slot containing e_{on} is permuted from $c(u)$ to $\tau(c(u))$, while every other slot contains e_{off}. Therefore, we have that $H^0_{\pi(u)}(G', c') = P_\tau H^0_u(G, c)$.

Suppose that the claim holds at layer $\ell - 1$. Since node relabeling preserves neighborhoods,

$$\mathcal{N}_{G'}(\pi(u)) = \{\pi(v) : v \in \mathcal{N}_G(u)\}.$$

Using the induction hypothesis and the equivariance of the element-wise aggregator, we obtain

$$\begin{aligned}
\hat{H}^\ell_{\pi(u)}(G', c') &= \bigoplus_{w \in \mathcal{N}_{G'}(\pi(u))} H^{\ell-1}_w(G', c') \\
&= \bigoplus_{v \in \mathcal{N}_G(u)} P_\tau H^{\ell-1}_v(G, c) \\
&= P_\tau \hat{H}^\ell_u(G, c).
\end{aligned}$$

We next verify that every linear map L used in MLP_{S_k} is color-equivariant. For every slot i,

$$\begin{aligned}
L(P_\tau H)[i] &= W_{\mathrm{self}} H[\tau^{-1}(i)] + W_{\mathrm{all}} \sum_{j=1}^{k} H[\tau^{-1}(j)] + b \\
&= W_{\mathrm{self}} H[\tau^{-1}(i)] + W_{\mathrm{all}} \sum_{j=1}^{k} H[j] + b \\
&= [P_\tau L(H)]_i.
\end{aligned}$$

Point-wise nonlinearities also commute with permutations of the color slots. Therefore, their composition MLP_{S_k} is S_k-equivariant. Moreover, concatenation along the feature axis satisfies

$$P_\tau(A \| B) = (P_\tau A) \| (P_\tau B).$$

LayerNorm is applied independently to each color slot with shared parameters, so it also commutes with P_τ. The same holds for the residual connection. Consequently,

$$H^\ell_{\pi(u)}(G', c') = P_\tau H^\ell_u(G, c),$$

completing the induction.

Finally, the readout MLP_{S_k} is color-equivariant by the same argument. If a softmax is applied to obtain probabilities, it also commutes with color permutations. We therefore have

$$p_{\pi(u)}\big(\pi \cdot G, \tau \cdot (\pi \cdot c)\big) = P_\tau p_u(G, c)$$

for every node and parameter setting, proving equivariance to the joint action of $S_{|V_G|} \times S_k$. □

Theorem 4. *Let $k, \Delta \in \mathbb{N}$. There exists a parameter setting of* SCONE *s.t. every k-colorable cycle, chordal or bipartite graph with degree at most Δ can be properly colored from the empty coloring.*

Proof. Let c be the current proper partial coloring, and let ε denote the label of an uncolored node. For every uncolored node v, define

$$F_c(v) = \{c(u) : u \in \mathcal{N}_G(v),\ c(u) \neq \varepsilon\},$$

$$T_c(v) = \{c(w) : \exists u,\ \{v,u\},\{u,w\} \in E_G,\ c(u), c(w) \neq \varepsilon\} \setminus F_c(v).$$

Thus, $F_c(v)$ contains the colors appearing on colored neighbors of v, while $T_c(v)$ contains the available colors observed at distance two through a colored neighbor.

The most-constrained-first decoder of SCONE selects an uncolored node maximizing $|F_c(v)|$.

The color predictor realizing the claim gives maximum score to the colors in $T_c(v)$ when this set is nonempty, and to the colors in $[k] \setminus F_c(v)$ otherwise. We first show that this is realizable by SCONE. The set $F_c(v)$ is determined by the one-hop neighborhood of v. In the first message-passing layer, each colored node can additionally record the colors appearing in its neighborhood. Aggregating these representations in a second layer determines $T_c(v)$ and whether it is empty. Since k and Δ are fixed, only finitely many relevant aggregate inputs can occur. With sufficient width, the color-equivariant MLPs can assign distinct codes to these inputs. The codes can be chosen to remain distinct after LayerNorm, while the residual term can be incorporated into the finite lookup performed by the following layer. Any additional layers can implement the identity by setting their residual update to zero. Hence the rule can be realized by a single parameter setting of the full architecture SCONE. We then show that it succeeds on each graph family.

First, let G be bipartite. We treat each connected component separately, and isolated nodes are immediate. Let's then consider a component with bipartition (A, B), and assume without loss of generality that its first selected node lies in A and receives color a. The connectivity of the component guarantees that while there are uncolored nodes, at least one has $|F_c(v)| \geq 1$. Moreover, all nodes other components have $|F_c(v)| = 0$, so the decoder will color the current component before moving to another one. The first selected node in B receives some color $b \neq a$.

We show inductively that every colored node in A has color a and every colored node in B has color b. The base case is given by the first two selected nodes. Let's then suppose that the inductive hypothesis holds. Suppose that the decoder selects the next node to be $v \in A$. Then it must have a colored neighbor $u \in B$, so $F_c(v) = \{b\}$. Moreover, u has a previously colored neighbor in A. Thus v has a colored node of A at distance two. Since every node at distance two from v lies in A, we have $T_c(v) = \{a\}$, and the decoder assigns color a to v. The argument for $v \in B$ is symmetric. Hence, inductively, the component is colored properly using only a and b.

Every even cycle is bipartite, so the preceding argument also applies to it. Now consider an odd cycle. After the first two adjacent nodes receive distinct colors a and b, the same two-hop rule extends an alternating coloring along the cycle. Until only one node remains uncolored, the colored nodes form a path whose colors alternate between a and b. The path between the two neighbors of the final node has an odd number of edges, so these neighbors have different colors. Thus both a and b belong to $F_c(v)$ for the final node and $T_c(v) = \emptyset$. Since an odd cycle is k-colorable only if $k \geq 3$, some color in $[k] \setminus \{a, b\}$ remains available and completes a proper coloring.

Finally, let G be chordal. Let $x_1, \ldots, x_n$ denote the order in which the decoder colors the nodes. We show inductively that every prefix $x_1, \ldots, x_t$ is a valid prefix of maximum cardinality search (MCS), as introduced in Tarjan & Yannakakis (1984). The first choice is valid because all nodes initially have no colored neighbors.

Suppose that $x_1, \ldots, x_{t-1}$ is an MCS prefix and that the current partial coloring is proper. Continue this prefix arbitrarily according to MCS, and number the resulting ordering by $\alpha(x_j) = n - j + 1$. On a chordal graph, every ordering produced by MCS is a perfect elimination ordering (Tarjan & Yannakakis, 1984). With this numbering convention, the higher-numbered neighbors of every node form a clique. Therefore, the already-colored neighbors of every uncolored node form a clique.

Since the partial coloring is proper, these neighbors have pairwise distinct colors. Therefore, for every uncolored node v, its saturation $|F_c(v)|$ equals its number of already-colored neighbors. Maximizing saturation is therefore exactly the MCS selection rule. The node selected by the decoder is a valid next MCS choice, while its confidence-based secondary criterion only breaks ties between valid choices.

The already-colored neighbors of the selected node form a clique of size at most $\omega(G) - 1$. Chordal graphs are perfect, so $\omega(G) = \chi(G) \leq k$. Hence at most $k - 1$ colors are forbidden. The set $[k] \setminus F_c(v)$ is therefore nonempty, and every color preferred by the predictor is available. Assigning one of these colors preserves properness and completes the induction.

Thus, the same parameter setting properly colors every k-colorable bipartite, chordal, or cycle graph of maximum degree at most Δ, independently of its number of nodes. □

C.3 Proof of Theorem 3

Theorem 5 shows that sequential coloring succeeds whenever the predictor can realize arbitrary scoring rules on the relevant rooted isomorphism types. We now show that the sum-aggregating version of SCONE has this property on cover-separated families of amenable graphs, and results easily extend to any injective aggregation $\oplus$. Throughout this section, the depth and width of the network may depend on n and k.

Recall that the 1-dimensional Weisfeiler-Leman algorithm (1-WL) iteratively refines node labels according to the labels of neighboring nodes (Arvind et al., 2017; Xu et al., 2019; Morris et al., 2019). Given a partially colored graph (G, c), we initialize

$$\mathrm{WL}^0(G, c, v) = c(v)$$

for every $v \in V_G$, where ε is treated as an additional label. For $\ell \geq 1$, we define

$$\mathrm{WL}^\ell(G, c, v) = \mathrm{HASH}\Big(\mathrm{WL}^{\ell-1}(G, c, v), \{\!\{\mathrm{WL}^{\ell-1}(G, c, w) : w \in \mathcal{N}_G(v)\}\!\}\Big),$$

where HASH is injective and shared across graphs. We denote the resulting multiset of labels by

$$\mathrm{WL}^\ell(G, c) = \{\!\{\mathrm{WL}^\ell(G, c, v) : v \in V_G\}\!\}.$$

Two partially colored graphs are distinguished by 1-WL if these multisets differ at some iteration. We say that (G, c) is *identified* by 1-WL if it is distinguished from every nonisomorphic partially colored graph.

Since the partition induced by the 1-WL labels is refined at every iteration, it eventually stabilizes. We refer to the resulting label of a node as its *stable* 1*-WL label*. Two rooted partially colored graphs (G, c, u) and (G', c', u') have the same *stable rooted* 1*-WL type* if u and u' receive the same stable 1-WL label when color refinement is run with a common labeling scheme. An uncolored graph G is *amenable* if (G, c_ε) is identified by 1-WL, where c_ε denotes the empty coloring.

We first assess that the color-equivariant MLPs used by SCONE can realize arbitrary equivariant functions on finite sets.

Lemma 3. *Let $\mathcal{X} \subseteq \mathbb{R}^{k\times d}$ be finite and closed under row permutations, and let $F : \mathcal{X} \to \mathbb{R}^{k\times p}$ be S_k-equivariant. Then F can be realized on $\mathcal{X}$ by a finite-width ReLU network whose linear layers have the form $L(H)[i] = W_{\mathrm{self}} H[i] + W_{\mathrm{all}} \sum_{j=1}^{k} H[j] + b$.*

Proof. We first use the fact that a ReLU MLP can realize any function on a finite set. Indeed, for distinct vectors $a_1, \ldots, a_m$, choose w such that the scalars $w^\top a_1, \ldots, w^\top a_m$ are distinct. A piecewise-linear function can interpolate arbitrary prescribed values at these points, and every such function is representable by a one-hidden-layer ReLU network.

Let $\mathcal{A} = \{a_1, \ldots, a_m\}$ be the set of all rows occurring in matrices in $\mathcal{X}$. Using the preceding fact, construct a row-wise ReLU MLP ψ such that $\psi(a_\ell) = e_\ell$. For $H \in \mathcal{X}$ and $i \in [k]$, define

$$z(H, i) = \psi(H[i]) \Big\| \sum_{j=1}^{k} \psi(H[j]).$$

The second component records the multiplicity of every row type in H.

If $z(H, i) = z(H', i')$, then $H[i] = H'[i']$ and H, H' have the same multiset of rows. Hence there exists $\tau \in S_k$ such that $H' = P_\tau H$ and $i' = \tau(i)$. Equivariance gives $F(H')[i'] = F(H)[i]$. Therefore, the assignment $\rho(z(H, i)) = F(H)[i]$ is well defined on the finite set of pointed signatures.

Another row-wise ReLU MLP realizes ρ exactly. Then, we have that

$$N(H)[i] = \rho\left(\psi(H[i]) \Big\| \sum_{j=1}^{k} \psi(H[j])\right)$$

satisfies $N(H) = F(H)$ on $\mathcal{X}$. Both ψ and ρ are implemented with $W_{\text{all}} = 0$, while the intermediate concatenation is one linear layer of the required form. Thus N has the stated S_k-equivariant architecture. □

Lemma 4. *For fixed n and k, there exist a depth, a width, and a parameter setting of* SCONE *such that the score predictor can realize any S_k-equivariant scoring rule on the finite set of stable rooted 1-WL types occurring in partially colored graphs with at most n nodes.*

Proof. For fixed n and k, only finitely many 1-WL labels and neighborhood multisets can occur. The initialization of SCONE can be chosen to give distinct, S_k-equivariant representations to the initial node labels.

Suppose that the embeddings after layer $\ell-1$ distinguish all $(\ell-1)$-round 1-WL labels. Since every degree is at most $n-1$, the embeddings can be chosen so that their sums uniquely determine all multisets of at most $n-1$ neighboring labels. There are only finitely many such multisets. Lemma 3 therefore allows the equivariant update MLP to map each pair consisting of a node embedding and its neighborhood aggregate to a distinct equivariant code for the corresponding ℓ-round 1-WL label.

LayerNorm and the residual connections do not affect this finite-set construction. With sufficiently large width, choose the update codes orbit by orbit so that every color slot has zero mean and the same nonzero variance. The codes can be chosen generically so that adding the residual embeddings and summing the resulting node embeddings introduces no unintended collisions among the finitely many inputs under consideration. The update MLP realizes these normalized codes by Lemma 3. Shared per-slot LayerNorm preserves them up to its learnable affine transformation, which can absorb the normalization constant, and the residual term produces the chosen next-layer embeddings.

Induction therefore shows that sufficiently many message-passing layers distinguish all stable rooted 1-WL types occurring on graphs with at most n nodes. Finally, an equivariant readout of sufficient width realizes any prescribed S_k-equivariant scoring rule on this finite set of types. □

We next record the closure of identification under refinements of the initial coloring. Throughout the remainder of the proof, partially colored graphs are regarded as vertex-colored graphs, with ε as an additional color, and all automorphisms and fractional automorphisms are required to preserve these colors. The definitions and implications from amenability to compactness and the Godsil property extend to vertex-colored graphs (Arvind et al., 2017). In particular, every partially colored graph identified by 1-WL is compact and Godsil.

Lemma 5. *Let (G, c) be a partially colored graph identified by 1-WL, and let c' be a refinement of c. Then (G, c') is also identified by 1-WL.*

Proof. Suppose that (G, c') and (G', d') are not distinguished by 1-WL. Since c' refines c, there exists a map ρ such that $c = \rho \circ c'$. Applying the same coarsening to d', let $d = \rho \circ d'$. The partially colored graphs (G, c) and (G', d) are also indistinguishable by 1-WL. Since (G, c) is identified by 1-WL, there exists an isomorphism φ between them.

We use φ to transport d' to a refinement $\widetilde{d'}$ of c on G, defined by $\widetilde{d'}(u) = d'(\varphi(u))$. The 1-WL equivalence of (G, c') and $(G, \widetilde{d'})$ gives a fractional isomorphism X (Arvind et al., 2017). Since both colorings refine c, the matrix X is also a fractional automorphism of (G, c). Moreover, (G, c) is compact because it is identified by 1-WL. We may therefore write $X = \sum_t \alpha_t P_{\pi_t}$, where $\alpha_t \geq 0$, $\sum_t \alpha_t = 1$, and each π_t is an automorphism of (G, c).

Since X is a fractional isomorphism between the two refinements, $X_{uv} = 0$ whenever $c'(u) \neq \widetilde{d'}(v)$. All coefficients in the decomposition of X are nonnegative. Therefore, every π_t with $\alpha_t > 0$ satisfies $c'(u) = \widetilde{d'}(\pi_t(u))$ for every $u \in V_G$. Such a π_t is an isomorphism between (G, c') and $(G, \widetilde{d'})$. Composing it with φ gives an isomorphism between (G, c') and (G', d'). Hence (G, c') is identified by 1-WL. □

We now use the assumption that distinct underlying graphs in the family have distinct universal covers (Krebs & Verbitsky, 2015). Importantly, this assumption concerns the uncolored graphs rather than the initial instances or their subsequent partial colorings.

Lemma 6 (Rooted types in a cover-separated family)**.** *Let $\mathcal{K}$ be a collection of connected amenable graphs such that no two nonisomorphic graphs in $\mathcal{K}$ have isomorphic universal covers. Let $G, G' \in \mathcal{K}$, let c, c' be arbitrary partial colorings of G, G', and let $u \in V_G$ and $u' \in V_{G'}$. If u and u' have the same stable 1-WL label, then (G, c, u) and (G', c', u') are isomorphic as rooted partially colored graphs.*

Proof. The colorings c and c' refine the empty colorings of the amenable graphs G and G'. Hence Lemma 5 shows that (G, c) and (G', c') are identified by 1-WL.

Equality of the stable labels of u and u' implies that their rooted color-labeled universal covers are isomorphic (Krebs & Verbitsky, 2015). Forgetting the color labels gives isomorphic universal covers of the underlying graphs G and G'. The assumption on $\mathcal{K}$ therefore implies that G and G' are isomorphic. In particular, $|V_G| = |V_{G'}|$.

The isomorphic rooted universal covers induce corresponding stable cells $V_1, \dots, V_m$ and $V_1', \dots, V_m'$ with the same equitable-quotient parameters. Let d_{pq} denote the number of neighbors in V_q of a node in V_p. We have

$$|V_p| d_{pq} = |V_q| d_{qp} \qquad \text{and} \qquad |V_p'| d_{pq} = |V_q'| d_{qp}.$$

Since the graphs are connected, their equitable quotients are connected. These equations therefore determine the cell sizes up to a common positive factor. Hence there exists $\lambda > 0$ such that $|V_p'| = \lambda |V_p|$ for every p. Summing over the cells and using $|V_G| = |V_{G'}|$ gives $\lambda = 1$.

Thus (G, c) and (G', c') have the same stable 1-WL label multiset and are not distinguished by 1-WL. Since they are identified by 1-WL, there is a color-preserving isomorphism φ from (G, c) to (G', c').

The nodes $\varphi(u)$ and u' have the same stable 1-WL label in (G', c'). Moreover, every graph identified by 1-WL is Godsil (Arvind et al., 2017), so each stable cell is an orbit of a subgroup of color-preserving automorphisms. There is therefore an automorphism α of (G', c') such that $\alpha(\varphi(u)) = u'$. The composition $\alpha \circ \varphi$ is an isomorphism from (G, c) to (G', c') that maps u to u'. □

We can now prove the main expressivity result.

Theorem 3. *Let $n, k \in \mathbb{N}$, and let $\mathcal{G}$ be a cover-amenable family of graphs with at most n nodes. There exists a parameter setting of* SCONE *s.t. every k-colorable (G, c_0) with $G \in \mathcal{G}$ can be solved.*

Proof. For an uncolored node u, let K_u denote its connected component, let c_u denote the restriction of c to K_u, and let $c_u^{u \leftarrow i}$ be obtained by assigning color i to u. Define the target score function by

$$s(G, c, u)_i = \begin{cases} 1, & \text{if } (K_u, c_u^{u \leftarrow i}) \text{ is extendable,} \\ 0, & \text{otherwise.} \end{cases}$$

By Lemma 6, the stable rooted 1-WL type of u determines the rooted partially colored component (K_u, c_u, u) up to isomorphism. Consequently, the target score function is well defined on stable rooted 1-WL types. It is also S_k-equivariant, since permuting the color labels maps proper completions to proper completions with the same permutation applied. Lemma 4 therefore gives a single parameter setting of SCONE that realizes this scoring rule.

Consider an extendable state (G, c) and any uncolored node u. A proper completion $c^\star$ of c assigns some color $i = c^\star(u)$ to u. Its restriction to K_u shows that $(K_u, c_u^{u \leftarrow i})$ is extendable. Hence the maximum score at u is 1, and every maximizing color preserves the extendability of the component containing u. The other components are unchanged, so the complete state also remains extendable.

The decoder may therefore select any uncolored node and break every color tie arbitrarily. Each step preserves extendability and colors one additional node. By induction, decoding reaches a complete extendable state, which is a proper coloring extending c_0. □

A sufficient condition for Theorem 3 is that every connected component be 1-WL-discrete (Arvind et al., 2017). Such components are amenable, and two of them with isomorphic universal covers are necessarily isomorphic. This condition holds with high probability in the dense Erdős–Rényi model (Babai et al., 1980). Moreover, the proof gives the stronger conclusion that every decoding trajectory solves the instance, independently of node selection and tie-breaking.

D EXPERIMENTAL SETUP

In this section, we discuss additional details about the experimental setup, including dataset preparation, details on baselines, on our implementation of SCONE, and on the computing infrastructure.

D.1 DATASETS

Sudoku instances. We evaluate on five collections of 9×9 Sudoku puzzles. The `sudoku_satnet` and `sudoku_rrn` datasets are taken from SATNet (Wang et al., 2019) and Recurrent Relational Networks (Palm et al., 2018), respectively. We additionally use `sudoku_17`, a collection of non-equivalent puzzles with only 17 clues,[1] and `sudoku_top1465`, a collection of 1,465 particularly difficult puzzles.[2] Finally, we use `sudoku_extreme`, using the splits introduced by Davis et al. (2026) (1000 training instances and 4000 test instances), which is the most challenging public Sudoku benchmark. We independently divide `sudoku_17` and `sudoku_satnet` into $70\%/15\%/15\%$ training, validation, and test splits. In contrast, `sudoku_rrn` and `sudoku_top1465` are used only for testing. The optimal solutions used for training are either provided in the original data, or are obtained with an exact ILP solver.

Register-allocation instances. We curate this benchmark from Appel and George's collection of 27,922 register-interference graphs (George & Appel, 1996) generated by Standard ML of New Jersey while compiling itself.[3] We parse the data into undirected graphs, remove self-loops, and remap vertex identifiers to a contiguous range. The original format specifies $K = 21$ available registers and assumes an implicit clique (which we do not materialize) of K precolored register nodes. We also discard the move edges, which encode preferences for assigning two nodes the same register rather than interference constraints. We retain graphs with $|V_G| \geq 4$ and filter out duplicated graphs. Finally, we compute a greedy clique lower bound and retain instances whose bound lies between 3 and 21, filtering graphs that are essentially trivially colourable or graphs whose clique lower bound already exceeds the original $K = 21$ available registers budget. The resulting 8999 instances contain no initially assigned colors and are all 21-colorable (or better). The optimal solutions used for training are obtained with an exact ILP solver.

Toronto examination timetabling. We use version I of the Carter–Laporte–Lee Toronto benchmark [4], see Carter et al. (1996). Each file records the exams and their enrollments, and the exams taken by each student. We construct an undirected conflict graph in which vertices represent exams and two exams are adjacent if some student takes both. Then, the exams taken by each student induce a clique. We adopt the uncapacitated minimum-timeslot formulation, ignoring room capacities and proximity costs, so a proper coloring represents a feasible timetable. Duplicate entries in each student's exam list are removed before constructing edges, while isolated exams and the original exam numbering are retained. We set k to the best-known feasible number of colors for each instance as reported in Qu et al. (2009, Table 11), and these values are upper bounds on the chromatic number rather than certified optima. To reduce the size of the graphs without affecting the validity of the instance, we apply the standard k-core reduction, iteratively removing vertices of degree less than k, as these can be colored trivially. We partition the thirteen Toronto instances according to their original size before preprocessing. The four smallest instances, `hec-s-92`, `sta-f-83`, `yor-f-83`, and `ute-s-92`, form the validation set and are used only to select the decoding configuration. The remaining instances form the test set, which we divide into the

[1] `https://web.archive.org/web/20131019184812if_/http://school.maths.uwa.edu.au/~gordon/sudokumin.php`

[2] `http://magictour.free.fr/sudoku.htm`

[3] `https://www.cs.princeton.edu/~appel/graphdata/`

[4] `https://people.cs.nott.ac.uk/pszrq/data.htm`

smaller instances ear-f-83, tre-s-92, lse-f-91, and kfu-s-93, and the large instances rye-s-93, car-f-92, uta-s-92, car-s-91, and pur-s-93. No Toronto instance is used for training, as we apply the same synthetically trained model throughout.

Queen graphs. For each p, queens_p is the queen graph of a $p \times p$ chessboard, with one node per square and edges connecting squares in the same row, column, or diagonal (Vasquez, 2004). A proper coloring therefore partitions the board into sets of mutually non-attacking queens. We use queens_7 exclusively for validation and reserve queens_8, queens_9, and queens_10 for testing. None of these instances is used for training, and we apply the same synthetically trained model throughout.

Structured graphs. Alongside the application benchmarks, we evaluate on synthetic families with chromatic numbers known exactly by construction. These instances are straightforward for classical algorithms but isolate potential failure modes of learned colorers. The cycle graphs contain hundreds of nodes, with equal numbers of even and odd cycles requiring two and three colors, respectively, and thus test long-range parity reasoning. The grid graphs are 4-regular bipartite lattices with hundreds of nodes. The complete graphs are K_n for $32 \leq n \leq 64$ and admit no slack, since every vertex must receive a distinct color.

Synthetic training data. In Section 5.3, the results for SCONE are obtained using a single generalist checkpoint trained exclusively on synthetic graphs, and no benchmark instance is used for training. We generate graphs with planted proper colorings by partitioning the nodes into color classes and adding edges only between classes. The dataset spans diverse graph sizes, color budgets, densities, and degree distributions, combining random, preferential-attachment, and clique-augmented constructions. In particular, it contains 10000 graphs, split into $8000/1000/1000$ training, validation, and test instances. We draw n log-uniformly from $[20, 200]$ and k uniformly from $[3, \min(30, n)]$. Graph families are sampled as uniform-edge, preferential-attachment, and clique-augmented constructions with equal probabilities. We retain connected candidates with an edge between every pair of planted color classes for which the generator's deterministic DSATUR requires at least $k + 1$ colors. This provides valid supervision while preventing the training set from being dominated by instances that a standard constructive heuristic solves immediately. Importantly, the planted proper coloring is generated jointly with each graph, providing ground-truth supervision without requiring an additional graph-coloring solve, which would be computationally expensive.

D.2 BASELINES

Greedy and DSATUR. We adapt both heuristics to the fixed-budget setting considered in this work. Greedy (Welsh & Powell, 1967) processes nodes in a fixed order, while DSATUR (Brélaz, 1979) repeatedly selects an uncolored node with the largest number of distinct colors among its colored neighbors, breaking ties by residual degree. In their classical formulations, both methods assign the first available color and introduce a new color whenever all existing colors are blocked; they therefore always return a proper coloring, with the number of colors used serving as their output. In our setting, by contrast, the palette size k is fixed. We restrict both methods to these k colors and, whenever every color already appears in the selected node's neighborhood, assign a color that conflicts with the fewest colored neighbors. Thus, the methods always return a complete coloring, which may be improper and can be evaluated using the same solved-instance and monochromatic-edge metrics as the learning-based methods. On feasible bipartite, chordal, and complete graphs, where DSATUR is exact, the palette cap does not bind and this modification leaves its classical behavior unchanged.

ILP. We use a standard integer linear programming (ILP) formulation with binary variables assigning one of the k colors to each node and auxiliary variables indicating monochromatic edges. The objective minimizes the total number of monochromatic edges, with initial colors enforced as fixed assignments. The ILP uses color-symmetry-breaking constraints to ensure an efficient representation. We solve the resulting problem using Gurobi (Gurobi Optimization, LLC, 2024).

ACE. ACE (Lecoutre, 2023) is a frontier general-purpose constraint solver for finite-domain constraint-satisfaction problems (CSP). We represent each node by an integer variable with domain $\{1, \ldots, k\}$, restrict precolored nodes to singleton domains, and impose a binary inequality

constraint for every edge. We run ACE with its default search configuration and the same dataset-specific time limits as the ILP baseline. Since ACE searches directly for a feasible coloring, it either returns a proper coloring or terminates without a solution within the time limit. Therefore, either its Mono% is 0 or is not defined. To provide ACE with the most favorable comparison, we report only its internal solve time, excluding JVM startup and instance-loading overhead, which would add an additional $0.5s$ per instance.

PI-SAGE and ROS. Both methods optimize a separate GNN from scratch on each test instance using a relaxed coloring objective, and then round its predictions to obtain a discrete coloring. Thus, neither method uses a pretrained checkpoint or requires a training split. For PI-SAGE (Schuetz et al., 2022), we use two mean-aggregating SAGECONV layers with hidden dimension 32 and minimize the soft Potts objective $\sum_{u,v\in E}\langle p_u, p_v\rangle$. We optimize with learning rate 10^{-4} for at most $100{,}000$ epochs, with tolerance 10^{-4} and early-stopping patience $1{,}000$, and retain the best argmax coloring encountered during optimization. For ROS (Qiu et al., 2025), we use the authors' two-layer GRAPHCONV architecture with GRAPHNORM and hidden dimension 100, and optimize the QUBO objective $\mathrm{tr}(XWX^\top)$ with learning rate 10^{-2} for at most $10{,}000$ epochs, using tolerance 10^{-2} and patience 100. We obtain discrete solutions by drawing 100 multinomial samples and retaining the best coloring. All hyperparameters follow the authors' published defaults. Our implementations faithfully port PI-SAGE from DGL to PyG and reproduce the original ROS GCN.

ANYCSP. ANYCSP (Tönshoff et al., 2023) is a pretrained anytime solver for constraint-satisfaction problems. We train its graph-coloring model using the authors' provided code and retain the default `num_boost`=8, which replicates each instance within a batch to perform eight parallel search restarts. Since the original checkpoint was not released, we compare our retrained model with the runtimes reported by the authors as a sanity check. They report 5 to 17 seconds for queen graphs of sizes 8 to 10, while our model solves the same sizes in 0.4 to 11 seconds. Although runtimes across different hardware are not directly comparable, these results suggest that our checkpoint is comparable to, and potentially stronger than, the original. Training takes more than 24 hours on an H200 GPU. We run ANYCSP with a time limit of 5 seconds on Sudokus to limit the runtime, and with a time limit of 30s per instance otherwise, albeit sometimes it exceeds this timelimit. Note that this limit is only reached on some large `toronto` instances and on `queens_10_10`, and this time limit is comfortably higher than the time used by SCONE. The original method does not support fixed colors. We represent each pinned node using a singleton domain, making clue violations impossible, rather than introducing unary constraints that the solver could violate and whose penalties would be mixed with edge violations in its objective. Since the released model was not trained on instances with pinned nodes, we retrain it on `sudoku_17` and on `sudoku_extreme` using the authors' training procedure.

CONSFORMER. CONSFORMER (Xu et al., 2025) is a self-supervised transformer that iteratively refines a complete assignment. Since no pretrained checkpoints are provided, we train all models ourselves. We use a candidate pool of one, corresponding to the standard model in the authors' main experiments; we do not report the optional multi-start extension, for which additional candidates trade against refinement iterations under a fixed time budget. Following the authors' ablations, we use masked relative positional encodings rather than the repository default of no relative encoding. The remaining settings follow the published configurations: seven layers, three attention heads, hidden dimension 128, the ABSE loss, and 5000 training epochs. For Sudoku, we use `ape_dim`=2 and threshold 0.5; for general coloring, we use `ape_dim`=1 and threshold 0.7.

We formulate Sudoku as graph coloring rather than using CONSFORMER's native all-different constraints. Each puzzle is represented as an 81-node, 20-regular constraint graph, with edges encoding pairwise inequalities and clues fixed as one-hot assignments. We retain the two-dimensional Sudoku positional encoding, so the model receives the grid geometry while using the same constraint formulation and evaluation metrics as the other coloring datasets. The native all-different formulation yields only marginally better results. We train the model on `sudoku_satnet` as in the original paper, as training on `sudoku_17` yields significantly worse results both in terms of training loss and of validation solution quality.

LDT. The Lattice Deduction Transformer (LDT) (Davis et al., 2026) combines a recurrent Transformer with stochastic search over a lattice of candidate assignments. It cannot be applied directly to general graph-coloring instances and is therefore included only in the Sudoku experiments. Since no checkpoint is available, we train the model using the authors' unmodified implementation. We fol-

low the configuration from their Table 1, using 4000 training steps while retaining all other defaults. At inference, we use the default 64 parallel chains and at most 1000 search rounds. We verified that the training and test instances exactly match the split used for SCONE.

TRM. The Tiny Recursive Model (TRM) (Jolicoeur-Martineau, 2025) is a recurrent Transformer-based refinement model. It cannot be applied directly to general graph-coloring instances and is therefore included only in the Sudoku experiments. Since no checkpoint is available, we train the official model on the same 1000 puzzles used by SCONE and LDT, following the authors' `sudoku_extreme_1k_aug_1k` configuration with 50000 training steps and 1000 augmentations per puzzle. The model and optimization procedure are unchanged. At inference, the number of refinement steps is controlled by `n_supervision`. We report 128 steps, the largest setting within the matched runtime budget and eight times the 16 steps used during training, thereby giving TRM a favorable test-time compute allowance.

D.3 SETUP FOR SCONE

Architecture and training. We implement SCONE in PyTorch Geometric (Fey & Lenssen, 2019). Following Section 4, each node maintains a $k \times d$ representation with one slot per color, and all updates are equivariant to permutations of these slots. Unless otherwise stated, we use $\mathcal{L} = 10$ message-passing layers, hidden dimension $d = 128$, and the degree-normalized aggregation $\sum_{v \in \mathcal{N}(u)} H_v / \sqrt{|\mathcal{N}(u)|}$, as it retains degree information while being stable across scales. See Section E for a comparison with different depths and widths. We use LayerNorm (Ba et al., 2016) and residual connections (He et al., 2016), as they have been shown to make training more stable (Luo et al., 2025).

We generate training states by sampling a masking fraction uniformly for each graph and hiding the corresponding fraction of non-fixed colors from a target solution. Clues provided with the instance are retained. We train against the permutation-aware targets of Section 4.2 using AdamW (Kingma & Ba, 2014; Loshchilov & Hutter, 2017) with learning rate 10^{-3}, weight decay 10^{-5}, batch size 64, and a cosine schedule with five warm-up epochs. We train for 50 epochs on `sudoku_17` and `registers`, which takes less than 30 minutes, for 300 epochs on the synthetic data, which takes roughly 6 hours, and for 2000 epochs on `sudoku_extreme`, given its small size, which takes roughly 2.5 hours. We select the checkpoint that maximizes the fraction of solved instances (Solved%), and in case of ties the ones that minimizes the fraction of monochromatic edges (Mono%) on *validation* instances.

Inference strategies. For beam search, each state scores all k color slots for the selected node. After removing children equivalent under S_k color permutations, we retain the global top B states across the full candidate set. There is no per-parent quota, so a state may contribute multiple children or none. Beam states are ranked by the cumulative sum of assignment log-probabilities, while forced assignments contribute no score. Ensembling uses E independent replicas, with replica r initialized from the r-th highest-degree uncolored node. The final output is the coloring with the fewest monochromatic edges across all beams or replicas.

Adaptive test-time scaling. Our decoder has two width axes: a beam size B within each decoding replica and a number E of independent replicas, for total width $W = B\,E$. These axes are not interchangeable. Additional replicas never compete with previous ones, whereas increasing the beam size changes which partial colorings are retained and is therefore not monotone. We use the *validation split only* to determine whether additional width should be allocated to beams, replicas, or a mixture of both, see Section E for results.

At test time, each graph (or batch of graphs) is first decoded at the smallest width. If the returned coloring is proper, decoding stops. Otherwise, the width is doubled according to the selected allocation rule and the graph is decoded again, up to the fixed cap. Since correctness can be verified directly by counting monochromatic edges, this stopping rule requires neither ground-truth colorings nor validation labels. This adaptive scaling puts SCONE in the same broad category as ANYCSP, CONSFORMER, LDT, and TRM, which all employ anytime-style inference procedures that terminate either upon reaching a search limit or upon solving the instance at hand. We retain the best coloring found across all attempted widths because a wider beam may discard a trajectory completed by a narrower one. The reported decoding cost is the total time consumed across all rungs

entered by each graph, averaged over the benchmark. In the experiments, we set a maximum width of 512 to avoid out-of-memory errors, but this value is reached only sporadically.

Architectures for ablations. All ablations use the same training configuration as the full model. The *non-equivariant* variant embeds a one-hot encoding of the current color and uses MLPs as update functions, while retaining the same message-passing depth and aggregation. We evaluate every variant using greedy decoding, i.e. setting $B = 1, E = 1$, with forced-assignment propagation disabled, so that additional search does not mask differences between the models.

D.4 Computing infrastructure

The experiments are run on a cluster equipped with AMD EPYC 9374F CPUs and NVIDIA H200 GPUs (143 GB). The code is based on PyTorch 2.11 and PyTorch-Geometric 2.8, with CUDA 12.8.

E Hyperparameter studies

In this section, we discuss the sensitivity of SCONE to its hyperparameters. In particular, we analyze the importance of the node selection strategy, the effect of using E ensembles or B beams for beam search on different datasets, as well as the tradeoff between solution quality and inference times as the embedding dimension d, the number of GNN layers $\mathcal{L}$ and the decoding width $W = B\,E$. Importantly, all the following experiments are run exclusively on the *validation* splits, so that the selection of the best model configurations are not influenced by test set results.

We start by analyzing the effect of the node selection strategy. The simplest strategy is to just color the nodes in their natural order, i.e. the order in which the nodes are provided in input. By contrast, as described in Section 4.3, our decoder instead follows the common most- constrained-first (MCF) heuristic (Haralick & Elliott, 1980) from constraint satisfaction.

We randomly permute the input node order in every run and average over five repeats for Sudoku and registers. For `queens_7` and Toronto, which contain fewer validation graphs, we use 20 repeats. Since at width 1 `queens_7` cannot be solved, we use $B = 8$ beams. Figure 4 compares the validation Solved% for the two strategies, showing that most-constrained-first performs consistently better than the natural order, confirming the validity of SCONE's design.

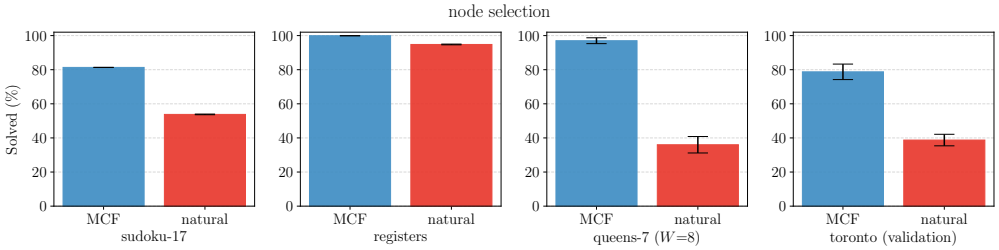

Figure 4: Validation Solved% for coloring nodes in their natural order and in MCF order.

We next study how a fixed decoding width is allocated between an ensemble of E independent runs and beam width B. Figure 5 shows that, at total width $W = B\,E = 8$, Sudoku consistently benefits more from beam search than from ensembling, while Toronto exhibits the opposite trend. Results on queens are noisier but also favor beams, while register allocation is nearly saturated across all mixtures, even at smaller widths. Based exclusively on these validation results, we use beams for Sudoku, queens, and registers, and ensembling for Toronto throughout the paper.

Finally, we study how the embedding dimension d, the number of message-passing layers $\mathcal{L}$, and the decode width W affect solution quality and inference time. We focus on Sudoku and Toronto, which exhibit the clearest trends in the preceding hyperparameter studies. For both datasets, we retrain the model with $d \in \{96, 128, 192\}$ while fixing $\mathcal{L} = 10$, and with $\mathcal{L} \in \{5, 10, 15\}$ while fixing $d = 128$. These models are decoded at width $W = 1$. We then vary $W \in \{1, 4, 16\}$ while fixing $d = 128$ and $\mathcal{L} = 10$, using beams for Sudoku and ensembles for Toronto. As in the main experiments, Sudoku is trained in-distribution, while Toronto is trained on synthetic graphs. All results are reported exclusively on the validation splits.

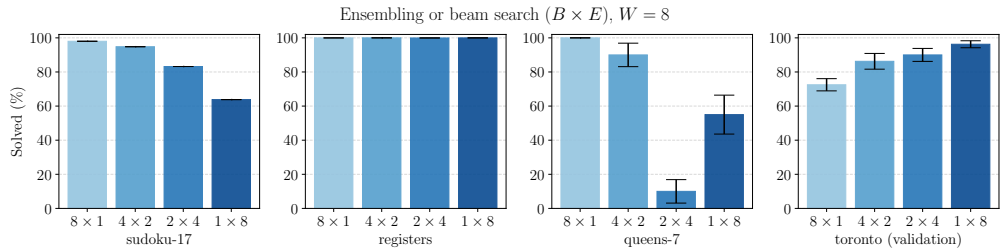

Figure 5: Validation Solved% for different ratios between independent runs E and beam width B.

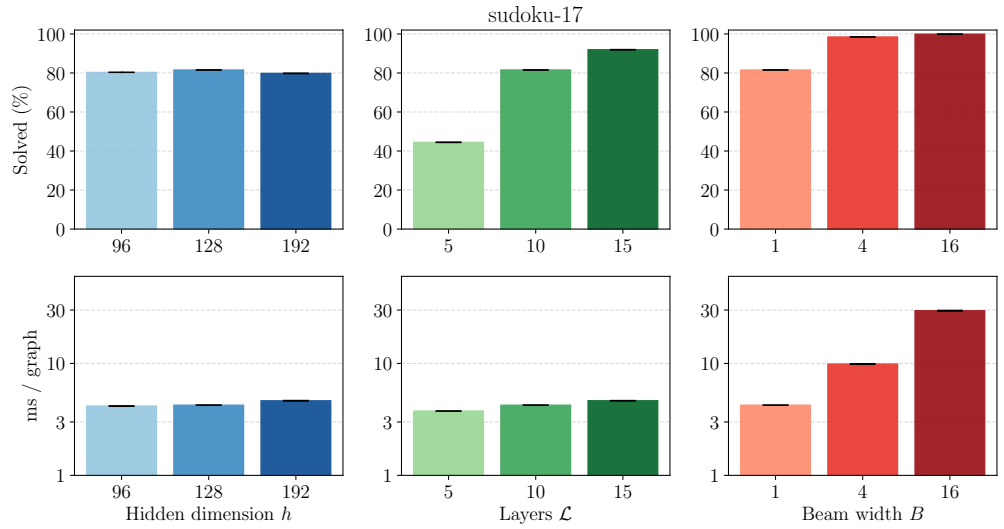

Figure 6: Validation Solved% and average inference times on `sudoku_17` for different values of embedding dimension d, message-passing layers $\mathcal{L}$ and beam width B.

Figures 6 and 7 show that both solution quality and inference time vary only marginally with d, although $d = 128$ achieves the best quality. Sudoku benefits from greater depth, while Toronto performs best at $\mathcal{L} = 10$. As expected, inference time grows approximately linearly with $\mathcal{L}$, with a particularly noticeable effect on Toronto. Increasing the decode width consistently improves solution quality in these experiments, while also increasing inference time approximately linearly. Based on these results, we fix $\mathcal{L} = 10$ and $d = 128$ throughout, as they provide a good trade-off between solution quality and inference time across datasets. Tuning these values separately for each dataset could yield slightly better results but would require additional training.

F Additional Experimental Results

In this section, we report some additional experimental results that were omitted from the main text for space reasons.

Transfer across Sudokus. First, we evaluate SCONE across Sudoku benchmarks in a transfer-learning setting. In the main results, the model trained on `sudoku_17` was evaluated in-distribution and transferred to `sudoku_satnet`, `sudoku_rrn`, and `sudoku_top1465`, while a separate model was trained and evaluated on `sudoku_extreme`. Here, we evaluate both checkpoints on all five datasets. We additionally consider a checkpoint trained on the training split of `sudoku_satnet`, which was used, e.g., in Xu et al. (2025).

Table 5 reports the results for the three checkpoints, decoded with up to $B = 512$ beams. The model trained on `sudoku_satnet` transfers least effectively, failing to solve every instance on several benchmarks and requiring on average wider beams, which then increases inference time.

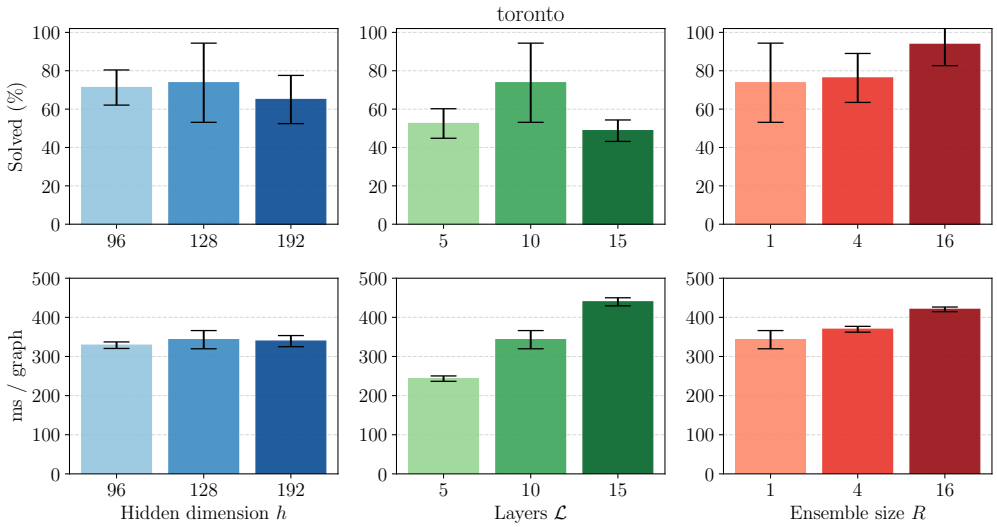

Figure 7: Validation Solved% and average inference times on `toronto` for different values of embedding dimension d, message-passing layers $\mathcal{L}$ and ensemble size E.

As expected, each checkpoint performs best on its training distribution, where it solves every instance with the lowest inference time. Under transfer, the models trained on `sudoku_17` and `sudoku_extreme` perform similarly, although the former is slightly stronger, possibly because it was trained on more instances.

Method	sudoku_17		sudoku_satnet		sudoku_rrn		sudoku_top1465		sudoku_extreme	
	Solved%	Time (s)	Solved%	Time (s)	Solved%	Time (s)	Solved%	Time (s)	Solved%	Time (s)
OURS (sudoku_17)	**100.00**$_{\pm 0.00}$	0.009$_{\pm 0.000}$	**100.00**$_{\pm 0.00}$	0.000$_{\pm 0.000}$	**100.00**$_{\pm 0.00}$	0.003$_{\pm 0.000}$	**100.00**$_{\pm 0.00}$	0.017$_{\pm 0.000}$	**100.00**$_{\pm 0.00}$	0.054$_{\pm 0.000}$
OURS (sudoku_extreme)	99.94$_{\pm 0.01}$	0.043$_{\pm 0.000}$	**100.00**$_{\pm 0.00}$	0.000$_{\pm 0.000}$	**100.00**$_{\pm 0.00}$	0.005$_{\pm 0.000}$	**100.00**$_{\pm 0.00}$	0.022$_{\pm 0.000}$	**100.00**$_{\pm 0.00}$	0.020$_{\pm 0.000}$
OURS (sudoku_satnet)	99.85$_{\pm 0.00}$	0.057$_{\pm 0.000}$	**100.00**$_{\pm 0.00}$	0.000$_{\pm 0.000}$	99.98$_{\pm 0.00}$	0.007$_{\pm 0.000}$	**100.00**$_{\pm 0.00}$	0.023$_{\pm 0.001}$	99.97$_{\pm 0.00}$	0.059$_{\pm 0.000}$

Table 5: Solution quality on Sudoku benchmarks with multiple checkpoints: Solved% (↑) and inference time per graph (↓).

Per-instance results. We then report the per-instance results for the Toronto timetabling and queens benchmarks. Each of these instances corresponds to a single graph, so pooling instances into one column, although necessary due to space constraints, averages over problems of substantially different difficulty. Tables 6, 7, and 8 decompose the three pooled columns of Table 4 into their constituent graphs. Since each column now represents one graph, Solved% denotes the fraction of 20 repeated runs that find a proper coloring, rather than the fraction of graphs solved.

Consistent with the pooled results, SCONE and ANYCSP are closely matched on the smaller Toronto instances, with each method holding a slight advantage on some graphs and falling behind on others. On the larger Toronto instances, SCONE consistently achieves both better solution quality and lower inference times. On queens, the two methods are essentially tied for sizes 8 and 9, while SCONE is faster for size 10.

Method	ear-f-83 n=159, k=22			tre-s-92 n=204, k=20			lse-f-91 n=132, k=17			kfu-s-93 n=196, k=19		
	Solved%	Mono%	Time (s)	Solved%	Mono%	Time (s)	Solved%	Mono%	Time (s)	Solved%	Mono%	Time (s)
GREEDY	0.0$_{\pm 0.0}$	0.38$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.0$_{\pm 0.0}$	0.49$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.0$_{\pm 0.0}$	0.38$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.0$_{\pm 0.0}$	0.35$_{\pm 0.00}$	0.00$_{\pm 0.00}$
DSATUR	0.0$_{\pm 0.0}$	0.02$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.0$_{\pm 0.0}$	0.07$_{\pm 0.00}$	0.00$_{\pm 0.00}$	**100.0**$_{\pm 0.0}$	**0.00**$_{\pm 0.00}$	0.00$_{\pm 0.00}$	**100.0**$_{\pm 0.0}$	**0.00**$_{\pm 0.00}$	0.00$_{\pm 0.00}$
ILP	0.0$_{\pm 0.0}$	0.04$_{\pm 0.00}$	600.00$_{\pm 0.00}$	0.0$_{\pm 0.0}$	0.15$_{\pm 0.00}$	600.00$_{\pm 0.00}$	**100.0**$_{\pm 0.0}$	**0.00**$_{\pm 0.00}$	51.10$_{\pm 0.00}$	**100.0**$_{\pm 0.0}$	**0.00**$_{\pm 0.00}$	64.60$_{\pm 0.00}$
ACE	**100.0**$_{\pm 0.0}$	**0.00**$_{\pm 0.00}$	0.75$_{\pm 0.03}$	**100.0**$_{\pm 0.0}$	**0.00**$_{\pm 0.00}$	0.76$_{\pm 0.01}$	**100.0**$_{\pm 0.0}$	**0.00**$_{\pm 0.00}$	0.60$_{\pm 0.01}$	**100.0**$_{\pm 0.0}$	**0.00**$_{\pm 0.00}$	0.69$_{\pm 0.03}$
PI-SAGE	0.0$_{\pm 0.0}$	0.64$_{\pm 0.10}$	6.89$_{\pm 0.19}$	0.0$_{\pm 0.0}$	0.82$_{\pm 0.06}$	7.52$_{\pm 0.46}$	0.0$_{\pm 0.0}$	0.62$_{\pm 0.04}$	6.90$_{\pm 1.02}$	0.0$_{\pm 0.0}$	0.55$_{\pm 0.03}$	7.06$_{\pm 0.39}$
ROS	0.0$_{\pm 0.0}$	0.58$_{\pm 0.06}$	0.63$_{\pm 0.10}$	0.0$_{\pm 0.0}$	0.62$_{\pm 0.06}$	0.75$_{\pm 0.12}$	0.0$_{\pm 0.0}$	0.47$_{\pm 0.08}$	0.61$_{\pm 0.05}$	0.0$_{\pm 0.0}$	0.41$_{\pm 0.06}$	0.65$_{\pm 0.07}$
ANYCSP	**100.0**$_{\pm 0.0}$	**0.00**$_{\pm 0.00}$	1.12$_{\pm 0.05}$	**100.0**$_{\pm 0.0}$	**0.00**$_{\pm 0.00}$	1.45$_{\pm 0.09}$	**100.0**$_{\pm 0.0}$	**0.00**$_{\pm 0.00}$	0.43$_{\pm 0.10}$	**100.0**$_{\pm 0.0}$	**0.00**$_{\pm 0.00}$	0.75$_{\pm 0.02}$
CONSFORMER	0.0$_{\pm 0.0}$	1.71$_{\pm 0.09}$	24.59$_{\pm 0.09}$	0.0$_{\pm 0.0}$	2.40$_{\pm 0.08}$	24.86$_{\pm 0.12}$	0.0$_{\pm 0.0}$	2.02$_{\pm 0.16}$	24.91$_{\pm 0.12}$	0.0$_{\pm 0.0}$	1.98$_{\pm 0.08}$	24.97$_{\pm 0.10}$
OURS	**100.0**$_{\pm 0.0}$	**0.00**$_{\pm 0.00}$	0.67$_{\pm 0.01}$	**100.0**$_{\pm 0.0}$	**0.00**$_{\pm 0.00}$	0.90$_{\pm 0.01}$	**100.0**$_{\pm 0.0}$	**0.00**$_{\pm 0.00}$	0.64$_{\pm 0.00}$	**100.0**$_{\pm 0.0}$	**0.00**$_{\pm 0.00}$	0.84$_{\pm 0.00}$

Table 6: Per-instance breakdown of the results on `toronto(small)` instances. Solved% (↑), Mono% (↓) and inference time per graph (↓).

Method	rye-s-93 n=236, k=21			car-f-92 n=413, k=27			uta-s-92 n=482, k=30			car-s-91 n=556, k=28			pur-s-93 n=1374, k=35		
	Solved%	Mono%	Time (s)	Solved%	Mono%	Time (s)	Solved%	Mono%	Time (s)	Solved%	Mono%	Time (s)	Solved%	Mono%	Time (s)
GREEDY	$0.0_{\pm 0.0}$	$0.29_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.0_{\pm 0.0}$	$0.49_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.0_{\pm 0.0}$	$0.24_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.0_{\pm 0.0}$	$0.50_{\pm 0.00}$	$0.01_{\pm 0.00}$	$0.0_{\pm 0.0}$	$0.21_{\pm 0.00}$	$0.02_{\pm 0.00}$
DSATUR	$0.0_{\pm 0.0}$	$0.08_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.0_{\pm 0.0}$	$0.10_{\pm 0.00}$	$0.01_{\pm 0.00}$	$0.0_{\pm 0.0}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.01_{\pm 0.00}$	$0.0_{\pm 0.0}$	$0.03_{\pm 0.00}$	$0.02_{\pm 0.00}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.07_{\pm 0.00}$
ILP	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$395.10_{\pm 0.00}$	$0.0_{\pm 0.0}$	$0.43_{\pm 0.00}$	$600.00_{\pm 0.00}$	$0.0_{\pm 0.0}$	$0.54_{\pm 0.00}$	$600.00_{\pm 0.00}$	$0.0_{\pm 0.0}$	$0.82_{\pm 0.00}$	$600.00_{\pm 0.00}$	$0.0_{\pm 0.0}$	$0.93_{\pm 0.00}$	$600.00_{\pm 0.00}$
ACE	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.70_{\pm 0.02}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$306.01_{\pm 7.03}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$2.06_{\pm 0.13}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$11.24_{\pm 0.69}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$9.87_{\pm 1.22}$
PI-SAGE	$0.0_{\pm 0.0}$	$0.45_{\pm 0.07}$	$7.47_{\pm 0.27}$	$0.0_{\pm 0.0}$	$0.70_{\pm 0.04}$	$10.53_{\pm 1.09}$	$0.0_{\pm 0.0}$	$0.58_{\pm 0.02}$	$10.45_{\pm 0.66}$	$0.0_{\pm 0.0}$	$0.76_{\pm 0.02}$	$12.12_{\pm 1.08}$	$0.0_{\pm 0.0}$	$0.40_{\pm 0.01}$	$29.78_{\pm 4.35}$
ROS	$0.0_{\pm 0.0}$	$0.37_{\pm 0.04}$	$0.65_{\pm 0.04}$	$0.0_{\pm 0.0}$	$0.68_{\pm 0.04}$	$0.84_{\pm 0.12}$	$0.0_{\pm 0.0}$	$0.52_{\pm 0.04}$	$0.75_{\pm 0.08}$	$0.0_{\pm 0.0}$	$0.72_{\pm 0.04}$	$0.83_{\pm 0.09}$	$0.0_{\pm 0.0}$	$0.38_{\pm 0.01}$	$1.01_{\pm 0.21}$
ANYCSP	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$1.22_{\pm 0.09}$	$60.0_{\pm 49.0}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$29.11_{\pm 5.07}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$18.56_{\pm 3.27}$	$10.0_{\pm 30.0}$	$0.02_{\pm 0.01}$	$35.90_{\pm 1.97}$	$80.0_{\pm 40.0}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$44.76_{\pm 4.78}$
CONSFORMER	$0.0_{\pm 0.0}$	$1.77_{\pm 0.08}$	$24.82_{\pm 0.09}$	$0.0_{\pm 0.0}$	$1.98_{\pm 0.05}$	$24.94_{\pm 0.28}$	$0.0_{\pm 0.0}$	$1.71_{\pm 0.03}$	$24.56_{\pm 0.12}$	$0.0_{\pm 0.0}$	$2.11_{\pm 0.05}$	$24.75_{\pm 0.10}$	–	–	–
OURS	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$1.38_{\pm 0.63}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$6.34_{\pm 3.19}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$3.14_{\pm 1.48}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$3.42_{\pm 1.52}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$10.10_{\pm 0.05}$

Table 7: Per-instance breakdown of the results on `toronto(large)` instances. Solved% (↑), Mono% (↓) and inference time per graph (↓).

Method	queens_8_8 n=64, k=9			queens_9_9 n=81, k=10			queens_10_10 n=100, k=11		
	Solved%	Mono%	Time (s)	Solved%	Mono%	Time (s)	Solved%	Mono%	Time (s)
GREEDY	$0.0_{\pm 0.0}$	$1.24_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.0_{\pm 0.0}$	$1.52_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.0_{\pm 0.0}$	$1.56_{\pm 0.00}$	$0.00_{\pm 0.00}$
DSATUR	$0.0_{\pm 0.0}$	$1.24_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.0_{\pm 0.0}$	$1.04_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.0_{\pm 0.0}$	$1.43_{\pm 0.00}$	$0.00_{\pm 0.00}$
ILP	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$9.99_{\pm 0.00}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$2.94_{\pm 0.00}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$499.53_{\pm 0.00}$
ACE	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$2.05_{\pm 0.02}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$20.06_{\pm 0.15}$	$0.0_{\pm 0.0}$	–	$597.54_{\pm 0.07}$
PI-SAGE	$0.0_{\pm 0.0}$	$1.88_{\pm 0.33}$	$5.43_{\pm 0.66}$	$0.0_{\pm 0.0}$	$1.83_{\pm 0.29}$	$5.39_{\pm 0.46}$	$0.0_{\pm 0.0}$	$1.67_{\pm 0.19}$	$6.13_{\pm 0.78}$
ROS	$0.0_{\pm 0.0}$	$1.45_{\pm 0.32}$	$0.69_{\pm 0.14}$	$0.0_{\pm 0.0}$	$1.37_{\pm 0.20}$	$0.67_{\pm 0.01}$	$0.0_{\pm 0.0}$	$1.26_{\pm 0.21}$	$0.71_{\pm 0.04}$
ANYCSP	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.42_{\pm 0.23}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.63_{\pm 0.39}$	$90.0_{\pm 30.0}$	$0.01_{\pm 0.02}$	$10.82_{\pm 7.22}$
CONSFORMER	$30.0_{\pm 47.0}$	$0.10_{\pm 0.08}$	$20.40_{\pm 7.14}$	$5.0_{\pm 22.4}$	$0.13_{\pm 0.06}$	$24.21_{\pm 2.67}$	$0.0_{\pm 0.0}$	$2.78_{\pm 0.42}$	$24.90_{\pm 0.17}$
OURS	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.38_{\pm 0.01}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.39_{\pm 0.02}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.84_{\pm 0.41}$

Table 8: Per-instance breakdown of the results on `queens(8-10)` instances. Solved% (↑), Mono% (↓) and inference time per graph (↓).

Synthetic structured graphs. Finally, we report results on synthetic structured graphs to examine potential failure modes of the different methods. In addition to the cycle benchmarks presented in the main paper, we consider complete graphs with up to 64 nodes, regular bipartite grid graphs, a separate collection containing only even cycles, and the Mycielski graphs M_5, M_6, and M_7. The latter are triangle-free despite having chromatic numbers 5, 6, and 7, respectively. As a stress test, SCONE uses width 1 with forced assignments disabled. Its performance therefore reflects the learned policy itself, without beam search, ensembling, or constraint propagation. SCONE and the classical heuristics solve every instance. ANYCSP also solves all four families and is particularly fast on the sparse Mycielski graphs. However, it takes roughly an order of magnitude longer than SCONE on grids and even cycles. These connected bipartite graphs have an essentially unique 2-coloring up to color permutations and require consistency to be propagated over long distances, which may explain the additional search. CONSFORMER cannot be evaluated on the complete graphs because their palettes contain up to 64 colors, whereas its architecture does not support palette sizes beyond the maximum of 30 seen during training. Moreover, it can only solve roughly half of the grid graphs, and no even-size cycles or Mycielski graphs. Finally, the relaxation-based methods frequently fail on grids and Mycielski graphs, and none solves any even cycle.

Method	complete			grid			even_cycles			myciel(5-7)		
	Solved%	Mono%	Time (s)	Solved%	Mono%	Time (s)	Solved%	Mono%	Time (s)	Solved%	Mono%	Time (s)
GREEDY	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.00_{\pm 0.00}$	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.00_{\pm 0.00}$	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.00_{\pm 0.00}$	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.00_{\pm 0.00}$
DSATUR	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.00_{\pm 0.00}$	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.00_{\pm 0.00}$	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.00_{\pm 0.00}$	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.00_{\pm 0.00}$
ILP	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.07_{\pm 0.00}$	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.01_{\pm 0.00}$	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.01_{\pm 0.00}$	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.01_{\pm 0.00}$
ACE	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.54_{\pm 0.00}$	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.47_{\pm 0.00}$	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.45_{\pm 0.00}$	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.50_{\pm 0.01}$
PI-SAGE	$98.75_{\pm 1.53}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$4.24_{\pm 0.01}$	$2.18_{\pm 1.88}$	$7.18_{\pm 0.26}$	$4.71_{\pm 0.06}$	$0.00_{\pm 0.00}$	$8.96_{\pm 0.23}$	$3.77_{\pm 0.01}$	$40.00_{\pm 38.87}$	$0.12_{\pm 0.11}$	$5.14_{\pm 0.30}$
ROS	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.30_{\pm 0.00}$	$10.31_{\pm 2.90}$	$4.37_{\pm 0.24}$	$1.03_{\pm 0.00}$	$0.00_{\pm 0.00}$	$8.18_{\pm 0.04}$	$0.81_{\pm 0.00}$	$60.00_{\pm 13.33}$	$0.03_{\pm 0.02}$	$0.73_{\pm 0.07}$
ANYCSP	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.55_{\pm 0.02}$	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.20_{\pm 0.01}$	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.36_{\pm 0.02}$	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.08_{\pm 0.00}$
CONSFORMER	–	–	–	$50.7_{\pm 4.7}$	$0.95_{\pm 0.15}$	$2.16_{\pm 0.00}$	$19.1_{\pm 4.1}$	$0.68_{\pm 0.04}$	$2.19_{\pm 0.01}$	$\mathbf{100.0}_{\pm \mathbf{0.0}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.08_{\pm 0.16}$
OURS	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.02_{\pm 0.00}$	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.03_{\pm 0.00}$	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.03_{\pm 0.00}$	$\mathbf{100.00}_{\pm \mathbf{0.00}}$	$\mathbf{0.00}_{\pm \mathbf{0.00}}$	$0.36_{\pm 0.00}$

Table 9: Solution quality on additional structured graphs: Solved% (↑), Mono% (↓) and inference time per graph (↓).

Additional details on the equivariance ablations The ablations in the main paper compare SCONE with a sequential model that does not enforce color equivariance, both with and without color-permutation augmentation. Here, we provide additional details on the experimental protocol and capacity controls. All variants use the training configuration described in Sec. D.3 and the same most-constrained-first decoding procedure, ensuring that the color representation is the only

difference between the primary models. Each configuration is trained three times, we then report the mean and standard deviation of the evaluations across training runs. The augmented and unaugmented models share the same data pipeline. Data augmentation samples a fresh uniformly random color permutation on every fetch. Augmentation does not change the number of graph presentations or optimization updates.

For the `sudoku_17` experiment in Fig. 3, we hold the training budget fixed at 10^6 graph presentations for every subset size N, setting the number of epochs to $10^6/N$. We evaluate the validation objective at 100 equally spaced points during training, so all subset sizes receive the same number of checkpoint-selection opportunities. All models are evaluated on the same held-out test set. For the experiments varying the target chromatic number, we generate graphs with a planted χ-coloring and $n \in [128, 512]$ vertices. The planted coloring certifies that $\chi(G) \leq \chi$, but not equality. Since DSATUR fails to find a χ-coloring for every retained graph, the datasets comprise challenging, predominantly tight instances as the palette size k grows. For $\chi \in \{3, 5, 7, 9\}$, we generate 2000 graphs and split them into $80\%/10\%/10\%$ training, validation, and test instances. We use that same synthetic generator as described in Section D.1, fixing the number of colors to be exactly χ. For each dataset, the model is trained with $k = \chi$.

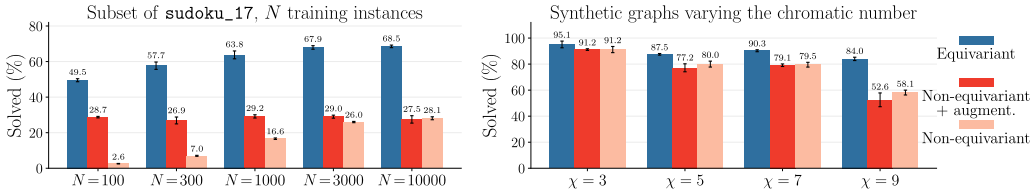

Figure 8: Ablation study on Sudoku and synthetic graphs at equal hidden width ($d = 128$).

Note that at equal hidden width ($d = 128$), the non-equivariant model has roughly half the number of trainable parameters than SCONE. For the experiments reported in the main paper in Figure 3, we therefore set the non-equivariant model at width $d = 256$, giving the baseline approximately twice as many parameters as SCONE. For completeness, Figure 8 gives the full results when all models use hidden width $d = 128$, complementing the results in Figure 3. Nevertheless, the equal-width and parameter-favorable comparisons yield the same qualitative trends.