Learning Minimally Rigid Graphs with High Realization Counts

Oleksandr Slyvka[1] , **Jan Rubeš**[2] , **Rodrigo Alves**[1] and **Jan Legerský**[1]
[1]Faculty of Information Technology, Czech Technical University in Prague, Czechia
[2]ETH Zürich, Switzerland
{slyvkole, rodrigo.alves, jan.legersky}@fit.cvut.cz, jrubes@ethz.ch

Abstract

For minimally rigid graphs, the same edge-length data can admit multiple realizations (up to translations and rotations). Finding graphs with exceptionally many realizations is an extremal problem in rigidity theory, but exhaustive search quickly becomes infeasible due to the super-exponential growth of the number of candidate graphs and the high cost of realization-count evaluation. We propose a reinforcement-learning approach that constructs minimally rigid graphs via 0- and 1-extensions, also known as Henneberg moves. We optimize realization-count invariants using the Deep Cross-Entropy Method with a policy parameterized by a Graph Isomorphism Network encoder and a permutation-equivariant extension-level action head. Empirically, our method matches the known optima for planar realization counts and *improves the best known bounds* for spherical realization counts, yielding new record graphs.

1 Introduction

Consider a fixed team of n robots moving collaboratively in the plane. Each robot is equipped with only a few sensors (typically far fewer than $n-1$) that measure and enforce fixed distances to *some* of the others, thereby imposing a set of pairwise distance constraints on the formation. From these distances alone, can we guarantee that the formation cannot *continuously* deform into a different shape while keeping all measured distances unchanged, so that the robots can reliably synchronize their motion? The sensing pattern is captured by a *measurement graph* whose vertices represent robots and whose edges indicate pairs for which a distance is available. Mathematically, this is formalized by notions from Rigidity Theory [Sitharam *et al.*, 2018]: a planar realization of the measurement graph is called *rigid* if it admits no nontrivial continuous deformation that preserves all edge lengths; otherwise, it is called *flexible*; see Figure 1.

However, even if a realization is rigid, the same edge-length data may still admit (finitely) many distinct realizations *up to rigid rotations and translations*; see Figure 2.

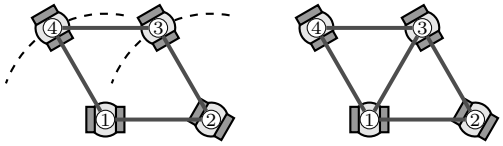

Figure 1: The realization on the left is flexible: when the robots 3 and 4 slide simultaneously along the indicated circles, the measured distances indicated by edges are preserved. The right realization is rigid since the only possible motions preserving the measured distances are rotations or translations of the whole formation. The orientation of individual robots does not play any role.

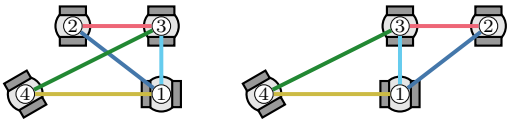

Figure 2: Two configurations of four robots with the same measurement graph and identical measured edge lengths (indicated by colors). Equivalently, they are two non-congruent realizations of the same edge-length-labeled graph. Two additional realizations can be obtained by reflecting each configuration.

Determining the number of these realizations has attracted a lot of attention: from (asymptotic) bounds (e.g. [Borcea and Streinu, 2004; Bartzos *et al.*, 2023; Clarke *et al.*, 2026]), via exact numbers for special cases [Jackson and Owen, 2019] and algorithms for computation [Capco *et al.*, 2018; Gallet *et al.*, 2020; Grasegger *et al.*, 2024; Dewar *et al.*, 2025], to computational search for graphs maximizing it [Grasegger *et al.*, 2020; Capco, 2024; Grasegger, 2025]. For the robots, this means that distance measurements alone need not single out a unique placement of the team; instead, they may determine a *finite set* of feasible formations that all satisfy the same sensing constraints. This multiplicity is not only a source of ambiguity, but can also be seen as a form of *built-in variability*: the robots may be positioned in different geometric arrangements while preserving the *same set of measured distances*. For instance, in bar-and-joint linkages (robots as joints, distances as bars), this non-uniqueness has been used to design multistable compliant structures [Zhang *et al.*, 2021]. Moreover, graph rigidity properties have been applied in various fields beyond multi-robot control [Zelazo *et al.*, 2015; Krick *et al.*, 2009], including sensor network localization [Zhu *et*

Extended version: DOI:10.48550/arXiv.2605.12427
Repository: https://github.com/Esestree/RL-min-rigid-graphs

al., 2010], molecular biology [Gáspár and Csermely, 2012; Billinge *et al.*, 2016], and architecture [Vilnay *et al.*, 2023].

In this work, we study the problem of finding *minimally rigid* graphs with high values of selected rigidity invariants, such as the finite number of planar realizations up to rotations and translations. On the one hand, this is an extremal optimization problem over a rapidly growing combinatorial domain (e.g., $\sim 10^{12}$ minimally rigid graphs for $n = 15$; [Larsson, 2020]), making exhaustive enumeration and evaluation for large n infeasible. On the other hand, improving empirical bounds and understanding the structure of extreme graphs can inform the design of measurement graphs in applications and shed light on the combinatorial and algebraic mechanisms that govern rigidity phenomena. Prior work has relied either on exhaustive search for small n [Capco, 2024], or *extensive* computations combined with expert knowledge [Grasegger *et al.*, 2020; Grasegger, 2025].

In contrast to previous work, we employ reinforcement learning (RL) to search for minimally rigid graphs with exceptionally many realizations. Concretely, constructing a minimally rigid graph G_n on n vertices can be formulated as a sequential decision-making problem using so called 0- and 1-*extensions* (introduced by [Henneberg, 1903]) that add a new vertex each time. At step k, the current intermediate rigid graph G_k defines a *state* $s_k \in \mathcal{S}$; each admissible extension of G_k that yields a rigid graph G_{k+1} corresponds to an *action* $a_k \in \mathcal{A}$; and the value of a chosen rigidity invariant (e.g., the number of realizations of the terminal graph G_n) serves as the *reward*. The objective is to learn a policy π that, given a state s_k, selects actions a_k that transform G_k into G_{k+1}, and repeats this process until reaching a final graph G_n with high reward. We address this problem via the Deep Cross-Entropy Method (Deep CEM), which iteratively updates a stochastic policy over graph extensions by biasing sampling toward elite construction trajectories. To represent intermediate graphs, we employ a Graph Isomorphism Network (GIN) encoder, and we design an extension-level, permutation-equivariant policy module so that action probabilities transform consistently under vertex relabeling. Our main contributions can be summarized as follows:

- We formulate the construction of minimally rigid graphs with large realization counts as a sequential decision-making problem and, to our best knowledge, we are the first to propose a RL framework with permutation-equivariant action representations for its solution.
- We demonstrate through extensive empirical study that our approach matches the known optima in the planar case and improves the best known lower bounds on the sphere for larger n, including **new record graphs** for spherical realization counts.
- Of particular relevance, our method *is orders of magnitude faster* than previous attempts and yields extremal graph instances that can potentially provide evidence toward a better understanding of structural phenomena in rigidity theory, which we further discuss (see Results in Section 4 and the extended version).
- We study the transfer-learning properties of our method and ablate our encoder (in the extended version).

2 Preliminaries and Problem Definition

Basic Notation. We consider only simple undirected graphs. Let $G = (V, E)$ be a graph with $n = |V|$ vertices and $|E|$ edges. We denote the edge formed by vertices u and v by $uv \in E$. Since the edges are undirected, $uv \in E$ is the same as $vu \in E$. We denote by K_n the clique on n vertices.

2.1 Minimally Rigid Graphs

We briefly recall the definition of *minimally rigid graphs* for completeness and to fix notation. To this end, we first introduce flexible and rigid realizations, and then define generic rigidity and minimal rigidity. For a comprehensive treatment, we refer to [Sitharam *et al.*, 2018, Part IV].

Definition 1 (Flexible and Rigid Realizations)**.** *A* planar realization *of a graph $G = (V, E)$ is a map $p\colon V \to \mathbb{R}^2$. It is called* flexible *if it has a nontrivial* flex*, that is, a continuous path of planar realizations p_t, $t \in [0, 1]$, with $p_0 = p$ such that for all $t \in (0, 1]$, $||p(u)-p(v)|| = ||p_t(u)-p_t(v)||$ holds for every edge $uv \in E$, but $||p(u)-p(v)|| \neq ||p_t(u)-p_t(v)||$ for some vertices $u, v \in V$. Otherwise, the realization is called* rigid*.*

In particular, for flexible realizations, the requirement that the flex is nontrivial rules out translations and rotations of the entire framework and ensures that the deformation changes the shape of the realization.

Definition 2 (Generic and Minimal Rigidity)**.** *A planar realization is* generic *if the coordinates of the vertices are algebraically independent over the rational numbers. A graph G is* rigid *if any (equivalently, every) generic planar realization is rigid. Otherwise, the graph is called* flexible*. A graph is* minimally rigid *if it is rigid and the deletion of any edge yields a graph that is flexible.*

2.2 Rigidity Invariants

We study three rigidity invariants for minimally rigid graphs. Given a minimally rigid graph G, we consider: (i) the *number of (complex) planar realizations* of G; (ii) the *number of (complex) spherical realizations* of G; and (iii) the *number of NAC-colorings* of G.

Given a generic planar realization p of a minimally rigid graph $G = (V, E)$, we aim to count the realizations of G that have the same edge lengths as p, modulo rotations and translations. Note that this number is finite, but it might vary between different choices of p (see for instance [Jackson and Owen, 2019]) as this corresponds to the number of solutions of the following polynomial system over the real numbers:

$$\begin{aligned} &(x_u - x_v)^2 + (y_u - y_v)^2 = ||p(u) - p(v)||^2 \quad \forall uv \in E\,, \\ &x_{\bar{u}} = y_{\bar{u}} = y_{\bar{v}} = 0\,, \quad x_{\bar{v}} = ||p(\bar{u}) - p(\bar{v})||\,, \end{aligned} \tag{1}$$

where (x_u, y_u) denotes the coordinates of vertex $u \in V$ and some edge $\bar{u}\bar{v} \in E$ is pinned down to factor out the rotations and translations. However, the number of solutions over the complex numbers is the same for all generic realizations p of the graph G [Jackson and Owen, 2019, Theorem 3.6] and upper bounds the number of (real) realizations.

Definition 3 (The number of planar realizations)**.** *Let G be a minimally rigid graph. The* number of (complex) planar

realizations *of G, denoted by* $\mathrm{Plane}_{\#}(G)$*, is the number of complex solutions of* (1) *for any generic realization p of G.*

Similarly as we have defined planar realizations, *spherical* realizations and their rigidity can be defined by replacing $\mathbb{R}^2$ by the 2-dimensional sphere. The generic rigidity behavior in the plane and the sphere is the same (see for instance [Dewar and Grasegger, 2024, Theorem 2.5]), namely, a graph is minimally rigid in the plane if and only if it is minimally rigid on the sphere. Hence, one may consider rigid graphs without specifying whether rigidity is studied in the plane or on the sphere. A polynomial system of equations similar to (1) corresponding to spherical realizations can be formulated using inner products on the left hand side. The number of complex solutions of the system is then called the *number of spherical realizations* of a minimally rigid graph G and is denoted by $\mathrm{Sphere}_{\#}(G)$. See [Dewar and Grasegger, 2024, Section 4] for a rigorous definition.

Finally, we consider a third rigidity-related invariant ($\mathrm{NAC}_{\#}$, also defined for flexible graphs), defined via certain two-color edge assignments as follows.

Definition 4 (NAC-colorings)**.** *A surjective edge coloring of a graph G by red and blue is called a* NAC-coloring *if for every cycle of G, either all edges have the same color, or there are at least two blue and two red edges. Let $\mathrm{NAC}_{\#}(G)$ denote the number of NAC-colorings of G divided by two (which corresponds to swapping the colors).*

NAC-colorings are interesting from the rigidity point of view, since they yield flexible realizations. More precisely, a connected graph has a planar realization with a flex if and only if it has a NAC-coloring [Grasegger *et al.*, 2019]. Interestingly, rigid graphs can have (non-generic) realizations with a flex. Different NAC-colorings give different flexes (see [Clinch *et al.*, 2026, Section 2.2]), hence it is natural to ask for minimally rigid graphs with many NAC-colorings.

2.3 Optimization Problem

Fix $n \geq 2$ and let $\mathcal{M}_n$ be the set of minimally rigid graphs on n vertices. For a chosen rigidity invariant $\mathcal{R} : \mathcal{M}_n \to \mathbb{R}_{\geq 0}$ (e.g., $\mathcal{R} = \mathrm{Plane}_{\#}$, $\mathcal{R} = \mathrm{Sphere}_{\#}$, or $\mathcal{R} = \mathrm{NAC}_{\#}$), we consider the following optimization problem:

$$G^\star \in \arg\max_{G \in \mathcal{M}_n} \mathcal{R}(G). \tag{2}$$

Since $\mathcal{M}_n$ grows super-exponentially (see the extended version) and evaluating $\mathcal{R}(G)$ can be expensive, directly solving (2) is infeasible beyond small n. Thus, we pursue an approximate solution via RL, by learning a policy that constructs high-reward graphs through sequential extensions.

3 Methodology

3.1 Deep CEM for Minimally Rigid Graphs

In our task of generating minimally rigid graphs with high realization counts, we propose a framework based on the Deep CEM. Classical CEM [Rubinstein and Kroese, 2004] iteratively samples candidates from a simple parametric distribution (e.g., a Gaussian) and updates the distribution to concentrate probability mass around the highest-performing (elite) solutions. Deep CEM extends this idea by replacing the simple parametric distribution with a *stochastic policy network* π_θ, which can represent complex and structured distributions (e.g., over graph extensions). A *stochastic policy network* with parameters θ can be seen as a mapping

$$\pi_\theta : \mathcal{S} \to \Delta(\mathcal{A}),$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, and $\Delta(\mathcal{A})$ denotes the set of probability distributions over $\mathcal{A}$. For each state $s \in \mathcal{S}$, $\pi_\theta(\cdot \mid s)$ is a probability distribution over actions in that state, satisfying $\sum_{a \in \mathcal{A}} \pi_\theta(a \mid s) = 1$. Thus, the value $\pi_\theta(a \mid s)$ gives the probability of selecting action a in state s.

In our setting, a state s corresponds to an intermediate graph G_k in the construction process. Rolling out the policy π_{θ_t} (at step t) from the base graph K_2 (the clique on two vertices) many times produces construction sequences

$$K_2 = G_2 \to G_3 \to \cdots \to G_n,$$

where each transition $G_k \to G_{k+1}$ is obtained by sampling an extension from $\pi_{\theta_t}(\cdot \mid G_k)$. Extensions are particularly natural choices to define the action space $\mathcal{A}$, because a graph G is minimally rigid if and *only* if it can be obtained from K_2 by a finite sequence of 0- and 1-extensions (see e.g. [Sitharam *et al.*, 2018, Theorems 19.2 and 19.11]), whose definition we recall now. Let $G = (V, E)$ be a graph, and let $z \notin V$ be a new vertex. If $u, v \in V$ are distinct vertices, then the graph

$$(V \cup \{z\},\ E \cup \{uz, vz\})$$

is called a 0-*extension* of G. If $u, v, w \in V$ are distinct vertices and $vw \in E$, then the graph

$$(V \cup \{z\},\ E \setminus \{vw\} \cup \{uz, vz, wz\})$$

is called a 1-*extension* of G; for illustration, see Figure 3. Thus, for a transition from G_k to G_{k+1}, we define the action space $\mathcal{A}$ as the set of all 0-extensions and 1-extensions applicable to G_k.

Having fixed the reward function $\mathcal{R}\colon \mathcal{M}_n \to \mathbb{R}_{\geq 0}$ for a given rigidity invariant (e.g., $\mathcal{R} = \mathrm{Plane}_{\#}$), we now describe how Deep CEM is instantiated in our setting; see Algorithm 1. For a given number of vertices n, and number of generations T, we maintain a population of m constructions and iteratively refine a stochastic policy π_θ over 0- and 1-extensions. At generation t, we start from the current survivor set S_{t-1} and build a population P_t. Each new candidate is obtained by rolling out the policy from the base graph K_2: at step k we sample an extension $a_k \sim \pi_{\theta_t}(\cdot \mid G_k)$, apply it to obtain G_{k+1}, and repeat until we reach G_n. Because the action space consists only of 0- and 1-extensions, every sampled G_n is minimally rigid by construction.

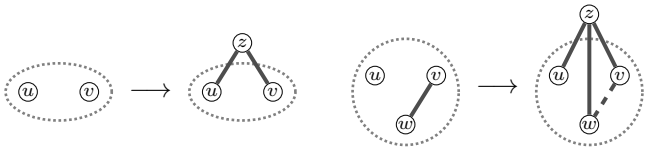

Figure 3: Illustration of the 0-extension (left) and 1-extension (right). In the 1-extension, the edge vw is removed (dashed).

Algorithm 1 Deep CEM for Minimally Rigid Graphs

Require: number of vertices n; population size m; number of generations T; survivor fraction ρ_{surv}; elite fraction ρ_{elite}; reward function $\mathcal{R}\colon \mathcal{M}_n \to \mathbb{R}_{\geq 0}$

1: $S_0 \leftarrow \emptyset$; $\theta_1 \leftarrow$ initialize policy parameters
2: **for** $t = 1$ to T **do** ▷ for details, see the ext. version
3: $P_t \leftarrow S_{t-1}$ ▷ current population of constructions
4: **while** $|P_t| < m$ **do**
5: $G_2 \leftarrow K_2$
6: **for** $k = 2$ to $n-1$ **do**
7: sample extension $a_k \sim \pi_{\theta_t}(\cdot \mid G_k)$
8: $G_{k+1} \leftarrow \mathrm{Apply}(G_k, a_k)$
9: **end for**
10: add G_n to P_t
11: **end while**
12: $\Omega_t \leftarrow$ top $\lfloor m\rho_{\text{elite}} \rfloor$ graphs in P_t by $\mathcal{R}(G)$
13: build dataset $\mathcal{D} = \bigcup_{G\in\Omega}\{(G_k, a_k)\}_{k=2}^{n-1}$
14: $\theta_{t+1} \leftarrow \mathrm{Update}(\theta_t; \mathcal{D}, \mathcal{L})$
15: $S_t \leftarrow$ top $\lfloor m\rho_{\text{surv}} \rfloor$ graphs in P_t by $\mathcal{R}(G)$
16: **end for**
17: **return** $\hat{G}^\star \leftarrow \arg\max_{G\in S_t} \mathcal{R}(G)$

We then evaluate the reward $\mathcal{R}(G)$ for all $G \in P_t$ and select an elite set Ω_t comprising the top $\lfloor m\rho_{\text{elite}} \rfloor$ graphs. From each $G \in \Omega_t$ we extract the sequence of state–action pairs $\{(G_k, a_k)\}_{k=2}^{n-1}$ and aggregate them into a dataset $\mathcal{D}_t$. Next, we update the policy parameters from θ_t to θ_{t+1} by minimizing for few epochs $\mathcal{L}(\theta_t)$ (Eq. 4) on $\mathcal{D}_t$ using Adam. The survivor set S_t for the next generation is then taken as the top $\lfloor m\rho_{\text{surv}} \rfloor$ graphs in P_t. After T generations, the algorithm outputs

$$\hat{G}^\star \leftarrow \arg\max_{G\in S_T} \mathcal{R}(G),$$

an n-vertex minimally rigid graph with a large value of $\mathcal{R}(G)$.

For computing $\mathcal{R} \in \{\mathrm{Plane}_\#, \mathrm{Sphere}_\#\}$, we use the package LNUMBER [Capco, 2024] which is based on [Capco *et al.*, 2018] and [Gallet *et al.*, 2020] respectively. For $\mathrm{NAC}_\#$, we use the algorithm [Laštovička and Legerský, 2024] implemented in the package PYRIGI [Adrian-Himmelmann *et al.*, 2026].

Remark. During graph generation, we must score thousands of candidates to form the elite set Ω_t in each generation t (see Algorithm 1, line 12). Since exact evaluation of $\mathrm{Plane}_\#$ and $\mathrm{Sphere}_\#$ is costly, we use the efficiently computable upper bound m-Bézout(G) [Bartzos *et al.*, 2020] as a surrogate during sampling. In particular, m-Bézout(G) upper-bounds the realization-count objectives, i.e.,

$$\mathrm{Plane}_\#(G) \leq \mathrm{Sphere}_\#(G) \leq \text{m-Bézout}(G)$$

(see [Dewar and Grasegger, 2024] for the first inequality). In practice, we first score all graphs in a generation with m-Bézout(G) and then compute the true reward ($\mathrm{Plane}_\#$ or $\mathrm{Sphere}_\#$) only for the top 25% candidates. This two-stage screening yields a significant speedup in our experiments while maintaining good agreement with the exact reward on the selected candidates. For details, we refer to the extended version.

3.2 Stochastic Policy Network Architecture

In this section, we describe the architecture of our stochastic policy network, which is illustrated in Figure 4.

Graph Encoder. The input to the policy at step k is an intermediate (minimally rigid) graph $G_k = (V_k, E_k)$. Since the choice of extensions must depend only on the underlying unlabeled combinatorial structure, we require the policy to be permutation-equivariant: relabeling the vertices of an intermediate graph G_k with permutation σ permutes the policy's output in the corresponding way:

$$\pi_\theta(\cdot \mid \sigma(G_k)) = \sigma\big(\pi_\theta(\cdot \mid G_k)\big) \tag{3}$$

Moreover, the rigidity invariants we study are themselves invariant under vertex relabeling.

Thus, a natural graph neural network encoder for this setting is the Graph Isomorphism Network (GIN) [Xu *et al.*, 2019]. GIN not only discriminates between different graph structures but also maps similar structures to similar embeddings, capturing dependencies between graph components. At a high level (we refer to the original work for details), a GIN layer aggregates features from a vertex's neighbors, combines them with the vertex's own features, and applies a learned transformation. Formally, the hidden representation $h_v^{(l)}$ of vertex v at layer $l \in \{1, 2, \ldots, L_{\mathcal{G}}\}$ is given by

$$h_v^{(l)} = \mathrm{MLP}^{(l)}\Big(\big(1+\epsilon^{(l)}\big)\, h_v^{(l-1)} \;+\; \sum_{u\in\mathcal{N}(v)} h_u^{(l-1)}\Big),$$

where $\mathcal{N}(v)$ denotes the neighbors of v, $\epsilon^{(l)}$ is a learnable scalar and $\mathrm{MLP}^{(l)}$ denotes a multilayer perceptron. The initial vertex features $h_v^{(0)}$ encode basic local structural information around each vertex and serve as the input to the first GIN layer. Note that GIN may fail to capture certain substructures and to distinguish between specific classes of graphs (e.g., regular graphs, certain trees or other graphs that share the same degree multiset) when all vertices share the same feature values, because they were shown to be at most as powerful as the 1-Weisfeiler–Lehman (1-WL) graph isomorphism test [Huang and Villar, 2021].

To better expose rigidity-relevant structure to the encoder, we therefore augment vertices with explicit local structural features. In particular, prior work on rigidity invariants [Grasegger, 2025] observed recurring patterns in graphs with high $\mathrm{Plane}_\#$ or $\mathrm{Sphere}_\#$, including characteristic degree multisets, constraints on triangle counts, or degree- and Hamiltonicity-driven adjacency rules. These observations indicate that degree-related structural properties are highly relevant for characterizing graphs with favorable rigidity invariants. Thus, for each vertex $v \in V_k$, we build a Local Degree Profile (LDP) [Errica *et al.*, 2019] defined as

$$\mathrm{LDP}(v) = \Big(\deg(v),\; \min_{u\in\mathcal{N}(v)} \deg(u),\; \max_{u\in\mathcal{N}(v)} \deg(u),\; \operatorname*{mean}_{u\in\mathcal{N}(v)} \deg(u),\; \operatorname*{std}_{u\in\mathcal{N}(v)} \deg(u)\Big),$$

where $\deg(v)$ is the degree of vertex v. In summary, we define the initial vertex features as

$$h_v^{(0)} = [\mathrm{LDP}(v)\,;\, \lambda_k\,;\, \kappa_v],$$

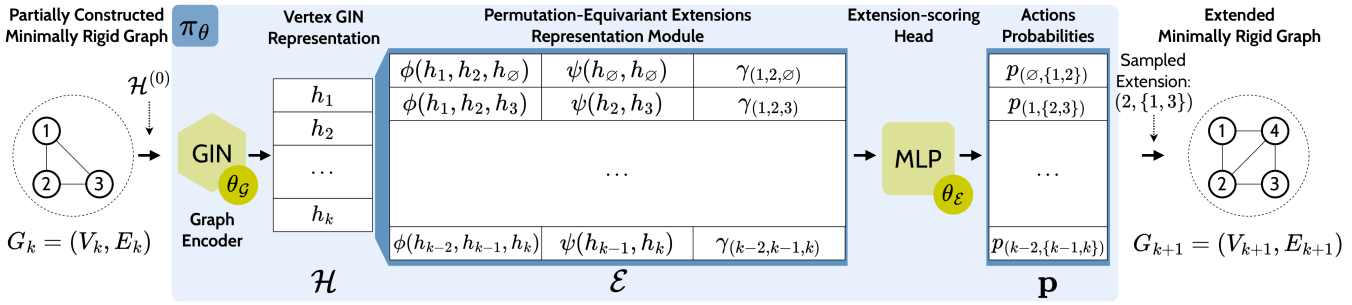

Figure 4: Policy network architecture. Given an intermediate graph G_k, a GIN encoder produces vertex embeddings $\mathcal{H}$. For each candidate 0- or 1-extension (defined by a vertex tuple), we construct an extension-level representation in $\mathcal{E}$ by combining the relevant vertex embeddings with extension-specific indicators (e.g., validity and move type). A shared extension-scoring head outputs a logit per candidate extension, and a softmax normalizes these logits to obtain action probabilities. In the figure, the learnable parameters of the policy π_θ are $\theta=(\theta_G,\theta_\mathcal{E})$, where θ_G parameterizes the GIN encoder and $\theta_\mathcal{E}$ parameterizes the extension-scoring head. The policy is permutation-equivariant: any relabeling of vertices induces the corresponding permutation of extension probabilities.

where $\lambda_k \in \mathbb{R}^d$ is a learnable embedding of the current construction step k (making the policy step-aware), and κ_v is the clustering coefficient of v, defined for unweighted graphs as the fraction of potential triangles through v that are present:

$$\kappa_v = \frac{2\tau(v)}{\deg(v)\big(\deg(v)-1\big)},$$

with $\tau(v)$ denoting the number of triangles incident to v. Finally, we define the initial vertex features as $\mathcal{H}^{(0)} = (h_v^{(0)})_{v\in V_k}$ and denote by $h_v = h_v^{(L_\mathcal{G})}$ the encoded representation of vertex v, i.e., the output of the final GIN layer.

Permutation-Equivariant Extensions Representation Module. We first pass the initial vertex features through our encoder (GIN layers) and obtain vertex representations $\mathcal{H}=(h_v)_{v\in V_k}$. Our goal is then, based on $\mathcal{H}$, to construct an architecture that assigns probabilities to each extension. Note that the extensions are defined in terms of specific vertex tuples (e.g., a 1-extension on (u,v,w)); therefore, under a relabeling of vertices, the *set* of valid extensions is permuted accordingly. Consequently, for isomorphic graphs G and G', the desired behavior is that the policy on G' is a *permutation* of the policy on G (see (3)).

Thus, we must construct extension-level representations $\mathcal{E}$ from vertex embeddings $\mathcal{H}$ so that action scores transform equivariantly under graph isomorphisms. Observe that, in an intermediate graph G_k, a candidate 1-extension acting on vertices u,v,w with removal of edge vw can be encoded by $(u,\{v,w\})$. For all distinct $u,v,w\in V_k$, we consider

$$e_{(u,\{v,w\})} = \big[\phi(h_u,h_v,h_w)\,;\,\psi(h_v,h_w)\,;\,\gamma_{(u,v,w)}\big] \in \mathcal{E},$$

where $\phi(\cdot)$ and $\psi(\cdot)$ are permutation-invariant functions (see [Zaheer *et al.*, 2017]), and $\gamma_{(u,v,w)}$ is a feature vector encoding additional properties of the extension. If $vw\notin E_k$, we say that $(u,\{v,w\})$ is invalid, otherwise $e_{(u,\{v,w\})}$ represents a 1-extension. Note that 0-extensions can be modeled as a special case of the above construction by introducing a dummy vertex $\emptyset$ with embedding $h_\emptyset=\vec{0}$. A 0-extension acting on vertices u and v is then represented as

$$e_{(\emptyset,\{u,v\})} = \big[\phi(h_u,h_v,h_\emptyset)\,;\,\psi(h_\emptyset,h_\emptyset)\,;\,\gamma_{(u,v,\emptyset)}\big] \in \mathcal{E},$$

so that both 0- and 1-extensions share the same extension representation space $\mathcal{E}$. Furthermore, $\gamma_{(u,v,w)}\in\{0,1\}^5$ collects five extension-specific features

$$\gamma_{(u,v,w)} = \big[\mathbf{1}_{\text{invalid}},\mathbf{1}_{E_0},\mathbf{1}_{E_{1a}},\mathbf{1}_{E_{1b}},\mathbf{1}_{E_{1c}}\big].$$

where each component is a binary indicator: $\mathbf{1}_{\text{invalid}}$ denotes whether $(u,\{v,w\})$ is invalid, $\mathbf{1}_{E_0}$ indicates a 0-extension, and $\mathbf{1}_{E_{1a}}$, $\mathbf{1}_{E_{1b}}$, $\mathbf{1}_{E_{1c}}$ indicate the 1-extension subclasses identified in [Grasegger, 2025], which differ only in the number of edges connecting the three vertices. Exactly one of the features $\mathbf{1}_{\text{invalid}},\mathbf{1}_{E_0},\mathbf{1}_{E_{1a}},\mathbf{1}_{E_{1b}},\mathbf{1}_{E_{1c}}$ is equal to 1.

Remark. We encode extension validity via the binary feature $\mathbf{1}_{\text{invalid}}$ instead of removing invalid moves. This keeps a uniform extension representation and lets a shared extension-scoring MLP process all candidates, while learning to assign (near-) zero probability to those with $\mathbf{1}_{\text{invalid}}=1$. This soft treatment stabilizes training (invalid moves still receive gradients) and preserves permutation structure. In the rare case of an invalid extension is sampled, we simply discard it and resample, so the construction remains minimally rigid.

Extension-scoring Head. Finally, we feed each $e_i\in\mathcal{E}$ through an $L_\mathcal{E}$-layer MLP, applied element-wise with shared parameters, so that the final layer produces a scalar logit z_i for each extension while preserving equivariance. Collecting these into a vector $z\in\mathbb{R}^{|\mathcal{E}|}$ and applying a softmax,

$$p_{(u,\{v,w\})} = \frac{\exp(z_{(u,\{v,w\})})}{\sum_{a\in V_k\cup\{\emptyset\};b,c\in V_k}\exp(z_{(a,\{b,c\})})},$$

yields a probability distribution $\mathbf{p}$ over candidate extensions, where $p_{(u,\{v,w\})}$ is the probability of selecting the extension indexed by $(u,\{v,w\})$ with $u\in V_k\cup\{\emptyset\}$ and $v,w\in V_k$.

3.3 Loss Function

A central challenge in reinforcement learning is achieving a suitable trade-off between *exploration* and *exploitation*. In our setting, this is particularly critical: we search for *extremely rare* graph configurations, and insufficient exploration can cause the policy to converge prematurely to

suboptimal structures. To explicitly address this requirement, we adopted Entropy regularization. This regularization technique augments the training objective (negative log-likelihood of elite actions) with an entropy term that penalizes overly confident action distributions:

$$\mathcal{L}(\theta) = -\frac{1}{|\mathcal{D}|}\sum_{(G_j,a_j)\in\mathcal{D}} \log \pi_\theta(a_j \mid G_j) - \eta\,\frac{1}{|\mathcal{D}|}\sum_{(G_j,a_j)\in\mathcal{D}} H\big(\pi_\theta(\cdot \mid G_j)\big), \quad (4)$$

where $H\big(\pi_\theta(\cdot \mid s)\big) = -\sum_{a\in\mathcal{A}} \pi_\theta(a \mid s)\log\pi_\theta(a \mid s)$ is the entropy of the action distribution at state s, and $\eta \in \mathbb{R}_{\geq 0}$ controls the strength of entropy regularization. As training progresses, the model acquires more information about which actions are beneficial. If entropy is kept too high, the policy may remain overly stochastic and fail to reliably construct high-quality graphs. To balance this, we gradually decrease the entropy coefficient over generations:

$$\eta_t = \frac{\eta_0}{1+\alpha\ln(1+t\,e^{-\beta})}, \quad (5)$$

where η_0, α, and β are hyperparameters, and $t \leq T$ indexes the generation. This schedule encourages broad exploration in the early stages and progressively shifts the policy toward exploitation of the most promising graph constructions. Although other methods can be similarly efficient in practice, entropy regularization has some unique properties. First, because it modifies the objective, it encourages the network to build good constructions, while being as uncertain in action choices as possible, which helps in creating diverse generations of good constructions. Second, the network learns to concentrate its 'confidence budget' on terminal actions, optimizing the final steps to maximize rewards even when preceding actions were chosen randomly.

4 Experiments

We evaluate our method on $\mathrm{Plane}_{\#}$, $\mathrm{Sphere}_{\#}$, and $\mathrm{NAC}_{\#}$. We focus on the range $10 \leq n \leq 18$, matching the intervals most extensively studied in prior computational work and for which reference optima or best-known bounds are available. For small n, exhaustive enumeration provides ground-truth optima, whereas for larger n we compare against the strongest previously reported values. For further details on reproducibility (including hardware specifications) and hyperparameter choices, we refer to the extended version and the accompanying repository [Slyvka *et al.*, 2026].

Baselines and Metrics. We compare our method in terms of (i) the best realization counts reported in the literature and (ii) runtime. For the number of realizations, we compare against [Clinch *et al.*, 2026; Grasegger, 2025], which includes also the results obtained previously by [Grasegger *et al.*, 2020; Capco, 2024]. The timing for the results from [Grasegger, 2025] were obtained as follows: for the results known to be optimal, we measured the exhaustive search using [Capco, 2024] except for the largest number of vertices, where the timing was extrapolated using the numbers of non-isomorphic minimally rigid graphs (from [Larsson, 2020]). Since [Grasegger, 2025] claims that *several millions* of graphs with $n \geq 15$ were checked to get the indicated bounds on $\mathrm{Plane}_{\#}$, without providing any timing, we estimate the time of computing $\mathrm{Plane}_{\#}$ for *one million* of graphs with 15 vertices (using the average time to compute $\mathrm{Plane}_{\#}(G)$ for 15-vertex graphs G we have encountered), similarly with half milion 14-vertex graphs for $\mathrm{Sphere}_{\#}$. Note this is a (very) conservative estimate; in reality it may be (much) higher. For our model, to initialize parameters for the computation of $\mathrm{Plane}_{\#}$ on $n = 18$ vertices, we used the weights of the model trained for $n-1$ vertices, and similarly for $\mathrm{Sphere}_{\#}$ for $n \in \{16, 18\}$; hence, we include this pretraining in the timing.

Results. The results we obtained are summarized in Table 1. Besides the timings, we indicate also for how many graphs the rigidity invariants were evaluated. Remarkably, the number of graphs needed for training is by several orders of magnitude lower than the number of minimally rigid graphs. For instance, for $n = 15$, we evaluate $\mathrm{Plane}_{\#}$ and $\mathrm{Sphere}_{\#}$ only for less than 10^5 out of the almost 10^{12} graphs. Empirically, the number of minimally rigid graphs on $n+1$ vertices is expected to be at least $30\times$ larger than on n. However, we observe that the number of graph evaluations required by our method to find constructions with high rigidity invariants increases only mildly as n grows. Thus, the longer computation times are caused by the (costly) reward evaluation for larger graphs, especially for $\mathrm{Sphere}_{\#}$, rather than by the number of evaluated graphs (see N in Table 1).

For $\mathrm{Plane}_{\#}$, we found exactly the same (i.e., isomorphic) graphs and therefore also the same values as [Grasegger, 2025]. However, for larger n, our method finds them orders of magnitude faster than exhaustive search. Note that the implementation [Capco, 2024] is already optimized, for instance, it skips graphs with a vertex of degree two as it is known these cannot attain the maximum.

In contrast to the planar case, for $15 \leq n \leq 18$ we find graphs with larger $\mathrm{Sphere}_{\#}$ than [Grasegger, 2025], *establishing new best-known lower bounds for the maximum value of this invariant*. The 14-vertex graph we found is isomorphic to the one reported in [Grasegger, 2025]. It is known that the maximum can be attained by several graphs. We found two different graphs on 15 with the indicated number of spherical realizations, whereas for $n \geq 16$ we found a single new graph attaining our best value. Previously, multiple graphs attaining to the maximum have been reported only for $n \leq 13$, i.e., coming from exhaustive search. The previously reported best graphs are Hamiltonian and have chromatic number three, minimum degree three and maximum degree four. While for those the search might have been biased towards having these properties, the graphs we have found possess them as well without imposing them. Previously, the graphs contained at least two 3-cycles, while one of our 15-vertex graphs and those with 16, 17 and 18 vertices have even stronger property: every vertex is in a 3-cycle.

Regarding $\mathrm{NAC}_{\#}$, [Clinch *et al.*, 2026] does not explicitly provide graphs with many NAC-colorings on 13 to 17 ver-

	# vertices n	10	11	12	13	14	15	16	17	18
	# min. rigid graphs	110 K	2 M	44 M	1 G	30 G	932 G	-	-	-
$\text{Plane}_{\#}$	[Grasegger, 2025]	880*	2 288*	6 180*	15 536*	42 780*	112 752	312 636	877 960	2 414 388
	Time	1 s	28 s	1177 s	17 h	510 h	36 h	-	-	-
	Ours	880*	2 288*	6 180*	15 536*	42 780*	112 752	312 636	877 960	2 414 388
	Time (Hit / Total)	42 s / –	66 s / –	387 s / –	137 s / –	3208 s / –	3 h / 6 h	8 h / 21 h	33 h / 50 h	83 h / 105 h
	N (Hit / Total)	2 K / –	2 K / –	9 K / –	3 K / –	29 K / –	41 K / 76 K	38 K / 87 K	44 K / 62 K	68 K / 72 K
$\text{Sphere}_{\#}$	[Grasegger, 2025]	1 536*	4 352*	12 288*	34 816*	98 304	274 432	815 104	2 195 456	-
	Time	7 s	470 s	10 h	227 h	158 h	-	-	-	-
	Ours	1 536*	4 352*	12 288*	34 816*	98 304	**278 528**	**819 200**	**2 228 224**	**6 127 616**
	Time (Hit / Total)	10 s / –	89 s / –	1209 s / –	3 h / –	24 h / 41 h	41 h / 139 h	401 h / 446 h	634 h / 890 h	923 h / 1026 h
	N (Hit / Total)	1 K / –	3 K / –	13 K / –	34 K / –	70 K / 111 K	24 K / 86 K	123 K / 130 K	12 K / 17 K	17 K / 18 K
$\text{NAC}_{\#}$	[Clinch *et al.*, 2026]	307*	639*	1 461*	2 923	7 063	14 127	35 133	70 267	180 607
	Time	9 s	259 s	3 h	-	-	-	-	-	-
	Ours	307*	639*	1 461*	**3 125**	**7 521**	**15 963**	**37 496**	**88 257**	**199 719**
	Time (Hit / Total)	139 s / –	303 s / –	915 s / –	772 s / 2445 s	2 h / 2 h	2 h / 3 h	4 h / 5 h	5 h / 5 h	8 h / 13 h
	N (Hit / Total)	20 K / –	49 K / –	110 K / –	83 K / 195 K	258 K / 258 K	202 K / 265 K	208 K / 240 K	124 K / 132 K	121 K / 134 K

Table 1: Comparison of the achieved values for the considered rigidity invariants. The numbers marked by star ($\star$) are the best possible, otherwise, the numbers indicate the best found value. **The bold values are new bounds, i.e., they were not previously discovered.** The gray timings are lower bound estimates (see the text for details). We provide two timings for our method: the time at which the algorithm first identified the best-performing graph, and the total duration of the experiment, including the search time spent after the best graph was found. We indicate also the number of non-isomorphic graphs (N) for which the rigidity invariants were evaluated.

tices. Hence, we constructed them using the same method as the one with 18 they provide. Our method gives better graphs for all $13 \leq n \leq 18$. As expected, none of the graphs contains a 3-cycle since its edges have to be of the same color. Interestingly, our graphs on 13, 15 and 16 vertices can be constructed from the complete bipartite graph on $3 + 3$ vertices using only 0-extensions, and we have observed such graphs also on 14 and 17 vertices that have $\text{NAC}_{\#}$ close to the found maximum. The fact that these graphs can be constructed in similar manner could be potentially used to obtain an infinite family of graphs with many NAC-colorings. The graphs with 14, 17 and 18 have non-trivial automorphism groups and minimum degree three (i.e., the last step is a 1-extension). Instantiating [Clinch *et al.*, 2026, Lemma 5.6] with the 18-vertex graph, we get a new asymptotic lower bound 2.144^n on the maximum number of NAC-colorings.

We provide the graphs for the improved bounds in the extended version and our repository [Slyvka *et al.*, 2026].

5 Related Works

A fundamental work by [Wagner, 2021] introduced a general **reinforcement-learning framework for combinatorial constructions**, modeling them as sequential decision processes and showing that neural policies can effectively guide search toward high-quality mathematical objects. Building on this line of research, [Angileri *et al.*, 2025] analyzed Wagner's framework on Brouwer's conjecture. Like us, they experimented with a Deep CEM-style scheme, but they target conjecture-driven graph search rather than rigidity optimization and do not require permutation-equivariant, extension-level action representation like ours. Furthermore, works such as [Charton *et al.*, 2024] and [Mehrabian *et al.*, 2024] propose effective methods for searching large extremal graphs (e.g., for Erdős-type conjectures), using sophisticated architectures, including transformers, AlphaZero integrated with Pairformer modules, and metaheuristics like incremental Tabu search. However, these approaches are not well suited to our setting: they typically require generating and evaluating very large numbers of candidate graphs, which would be prohibitively here because computing rigidity invariants is the main computational bottleneck. More recently, [Georgiev *et al.*, 2025] demonstrated that combining evolutionary methods with large language models can enable mathematical discovery across a wide range of domains, albeit at the cost of relying on potentially expensive language-model infrastructure. In contrast, our approach is specialized to rigidity-theory invariants and provides an efficient and accurate search procedure without dependence on external models or sources.

To our best knowledge, no comparable machine learning-based approach has yet been applied to **optimizing rigidity invariants**. In a related vein, while we focus on complex realizations, [Bartzos *et al.*, 2021] studied the number of real realizations using a heuristic-based search. For *complex realization counts*, prior work has relied either on exhaustive search for small n [Capco, 2024] or on large-scale computations guided by expert insight [Grasegger *et al.*, 2020; Grasegger, 2025], similarly for $\text{NAC}_{\#}$ [Clinch *et al.*, 2026].

6 Conclusion

We introduced a learning approach to the extremal problem of finding minimally rigid graphs with large rigidity invariants. Empirically, we (i) recover the known optimal values for planar realization counts and (ii) establish improved best-known lower bounds for spherical realization counts (for $15 \leq n \leq 18$) and for the number of NAC-colorings (for $13 \leq n \leq 18$), while evaluating only a tiny fraction of all candidate graphs. Future work will focus on the structural features of the discovered extremal graphs and leverage them to derive new theoretical insights in rigidity theory and the extension of our framework to other invariants and larger n.

Ethical Statement

The research is primarily foundational and is unlikely to have any significant ethical impact.

Acknowledgments

This work was supported by the Student Summer Research Program 2025 of FIT CTU in Prague. J. L. was supported by the Czech Science Foundation (GAČR), project No. 22-04381L.

References

[Adrian-Himmelmann *et al.*, 2026] Matthias Adrian-Himmelmann, Matteo Gallet, Georg Grasegger, and Jan Legerský. PyRigi – a general-purpose Python package for the rigidity and flexibility of bar-and-joint frameworks. *ACM Transactions on Mathematical Software*, 2026. To appear.

[Angileri *et al.*, 2025] Flora Angileri, Giulia Lombardi, Andrea Fois, Renato Faraone, Carlo Metta, Michele Salvi, Luigi Amedeo Bianchi, Marco Fantozzi, Silvia Giulia Galfrè, Daniele Pavesi, Maurizio Parton, and Francesco Morandin. Analyzing RL components for Wagner's framework via Brouwer's conjecture. *Machine Learning*, 114:242, 2025.

[Bartzos *et al.*, 2020] Evangelos Bartzos, Ioannis Z. Emiris, and Josef Schicho. On the multihomogeneous Bézout bound on the number of embeddings of minimally rigid graphs. *Applied Algebra and Engineering Communications*, 31(2):325–357, 2020.

[Bartzos *et al.*, 2021] Evangelos Bartzos, Ioannis Z. Emiris, Jan Legerský, and Elias Tsigaridas. On the maximal number of real embeddings of minimally rigid graphs in $\mathbb{R}^2$, $\mathbb{R}^3$, and $\mathbb{S}^2$. *Journal of Symbolic Computation*, 102:189–208, 2021.

[Bartzos *et al.*, 2023] Evangelos Bartzos, Ioannis Z. Emiris, and Charalambos Tzamos. An asymptotic upper bound for graph embeddings. *Discrete Applied Mathematics*, 327:157–177, 2023.

[Billinge *et al.*, 2016] Simon J.L. Billinge, Phillip M. Duxbury, Douglas S. Gonçalves, Carlile Lavor, and Antonio Mucherino. Assigned and unassigned distance geometry: applications to biological molecules and nanostructures. *4OR*, 14(4):337–376, 2016.

[Borcea and Streinu, 2004] Ciprian S. Borcea and Ileana Streinu. The Number of Embeddings of Minimally Rigid Graphs. *Discrete & Computational Geometry*, 31:287–303, 2004.

[Capco *et al.*, 2018] José Capco, Matteo Gallet, Georg Grasegger, Christoph Koutschan, Niels Lubbes, and Josef Schicho. The Number of Realizations of a Laman Graph. *SIAM Journal on Applied Algebra and Geometry*, 2(1):94–125, 2018.

[Capco, 2024] Jose Capco. Toolkit for computing the Laman number, 2024. https://github.com/jcapco/lnumber, DOI:10.5281/zenodo.8301012.

[Charton *et al.*, 2024] François Charton, Jordan S. Ellenberg, Adam Zsolt Wagner, and Geordie Williamson. PatternBoost: Constructions in Mathematics with a Little Help from AI, 2024. DOI:10.48550/arXiv.2411.00566.

[Clarke *et al.*, 2026] Oliver Clarke, Sean Dewar, Daniel Green Tripp, James Maxwell, Anthony Nixon, Yue Ren, and Ben Smith. A tropical approach to rigidity: Counting realisations of frameworks. *Journal of the London Mathematical Society*, 113(2):e70438, 2026.

[Clinch *et al.*, 2026] Katie Clinch, Dániel Garamvölgyi, John Haslegrave, Tony Huynh, Jan Legerský, and Anthony Nixon. Stable cuts, NAC-colourings and flexible realisations of graphs. *Journal of Graph Theory*, 2026. To appear.

[Dewar and Grasegger, 2024] Sean Dewar and Georg Grasegger. The number of realisations of a rigid graph in Euclidean and spherical geometries. *Algebraic Combinatorics*, 7(6):1615–1645, 2024.

[Dewar *et al.*, 2025] Sean Dewar, Georg Grasegger, Josef Schicho, Ayush Kumar Tewari, and Audie Warren. Computing the number of realisations of a rigid graph, 2025. DOI:10.48550/arXiv.2510.02988.

[Errica *et al.*, 2019] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A simple yet effective baseline for non-attributed graph classification. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[Gallet *et al.*, 2020] Matteo Gallet, Georg Grasegger, and Josef Schicho. Counting realizations of Laman graphs on the sphere. *The Electronic Journal of Combinatorics*, 27(2):P2.5, 2020.

[Gáspár and Csermely, 2012] Merse E. Gáspár and Peter Csermely. Rigidity and flexibility of biological networks. *Briefings in Functional Genomics*, 11(6):443–456, 2012.

[Georgiev *et al.*, 2025] Bogdan Georgiev, Javier Gómez-Serrano, Terence Tao, and Adam Zsolt Wagner. Mathematical exploration and discovery at scale, 2025. DOI:10.48550/arXiv.2511.02864.

[Grasegger *et al.*, 2019] Georg Grasegger, Jan Legerský, and Josef Schicho. Graphs with Flexible Labelings. *Discrete & Computational Geometry*, 62(2):461–480, 2019.

[Grasegger *et al.*, 2020] Georg Grasegger, Christoph Koutschan, and Elias Tsigaridas. Lower bounds on the number of realizations of rigid graphs. *Experimental Mathematics*, 29(2):125–136, 2020.

[Grasegger *et al.*, 2024] Georg Grasegger, Boulos El Hilany, and Niels Lubbes. Coupler curves of moving graphs and counting realizations of rigid graphs. *Mathematics of Computation*, 93:459–504, 2024.

[Grasegger, 2025] Georg Grasegger. Explorations on the number of realizations of minimally rigid graphs, 2025. DOI:10.48550/arXiv.2502.04736.

[Henneberg, 1903] Lebrecht Henneberg. Die graphische Statik der starren Körper. In *Encyklopädie der mathematischen Wissenschaften mit Einschluss ihrer Anwen-*

dungen, volume IV, pages 345–434. B.G. Teubner Verlag, 1903.

[Huang and Villar, 2021] Ningyuan Teresa Huang and Soledad Villar. A Short Tutorial on The Weisfeiler–Lehman Test And Its Variants. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8533–8537. IEEE, 2021.

[Jackson and Owen, 2019] Bill Jackson and John C. Owen. Equivalent realisations of a rigid graph. *Discrete Applied Mathematics*, 256:42–58, 2019.

[Krick *et al.*, 2009] Laura Krick, Mireille E. Broucke, and Bruce A. Francis. Stabilisation of infinitesimally rigid formations of multi-robot networks. *International Journal of Control*, 82(3):423–439, 2009.

[Larsson, 2020] Martin Larsson. Nauty Laman plugin, 2020. https://github.com/martinkjlarsson/nauty-laman-plugin.

[Laštovička and Legerský, 2024] Petr Laštovička and Jan Legerský. Flexible realizations existence: NP-completeness on sparse graphs and algorithms, 2024. DOI:10.48550/arXiv.2412.13721.

[Mehrabian *et al.*, 2024] Abbas Mehrabian, Ankit Anand, et al. Finding Increasingly Large Extremal Graphs with AlphaZero and Tabu Search. In Kate Larson, editor, *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 6985–6993. International Joint Conferences on Artificial Intelligence Organization, 2024.

[Rubinstein and Kroese, 2004] Reuven Y. Rubinstein and Dirk P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*. Springer, New York, 2004.

[Sitharam *et al.*, 2018] Meera Sitharam, Audrey St. John, and Jessica Sidman (Eds.). *Handbook of Geometric Constraint Systems Principles*. CRC Press, 2018.

[Slyvka *et al.*, 2026] Oleksandr Slyvka, Jan Rubeš, Rodrigo Alves, and Jan Legerský. Our project repository. https://github.com/Esestree/RL-min-rigid-graphs, 2026. Code and hyperparameter configurations.

[Vilnay *et al.*, 2023] Oren Vilnay, Leon Chernin, and Margi Vilnay. *Tensegrity Structures Design Methods*. CRC Press / Taylor & Francis, 2023.

[Wagner, 2021] Adam Zsolt Wagner. Constructions in Combinatorics via Neural Networks, 2021. DOI:10.48550/arXiv.2104.14516.

[Xu *et al.*, 2019] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[Zaheer *et al.*, 2017] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R. Salakhutdinov, and Alexander J. Smola. Deep Sets. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[Zelazo *et al.*, 2015] Daniel Zelazo, Antonio Franchi, Heinrich H. Bülthoff, and Paolo Robuffo Giordano. Decentralized rigidity maintenance control with range measurements for multi-robot systems. *The International Journal of Robotics Research*, 34(1):105–128, 2015.

[Zhang *et al.*, 2021] Ran Zhang, Thomas Auzinger, and Bernd Bickel. Computational design of planar multistable compliant structures. *ACM Transactions on Graphics*, 40(5), 2021.

[Zhu *et al.*, 2010] Zhisu Zhu, Anthony Man-Cho So, and Yinyu Ye. Universal rigidity and edge sparsification for sensor network localization. *SIAM Journal on Optimization, 20(6)*, pages 3059–3081, 2010.