DISCRETE FLOW MATCHING FOR EXTREMAL GRAPH SEARCH WITH ZERO-SHOT SIZE TRANSFER

Anonymous authors
Paper under double-blind review

ABSTRACT

Many existing neural systems for mathematical discovery of extremal combinatorial structures alternate between generation and local search, retraining a generative model on the best refined constructions. This model is typically parametrized as an autoregressive transformer over a serialization of the object, which is a questionable choice for graphs: vertex symmetry is not necessarily encoded, and edge decisions cannot be revised within the autoregressive sampling pass. This work instead draws on graph neural networks and discrete flow matching: a permutation-equivariant pair network predicts all edge decisions at once, and iterative denoising lets each be revisited as the rest of the graph changes. The same weights can generate graphs of different sizes. We show that this reuse works: generators trained on smaller graphs construct optimal C_4-free graphs at unseen sizes before local search. Reward fine-tuning further improves their value as search starts, with gains beyond the fine-tuning size and selected pipelines using up to 49% less H200 training time than target-size training. One selected adapted generator produces a 137-edge C_4-free graph on 42 vertices before local search, matching the best-known count without training at that size. To the best of our knowledge, this is the first time a learned approach attains this bound. Experiments on Ramsey construction demonstrate zero-shot size transfer under a second constraint family.

1 INTRODUCTION

Recent advances in AI for mathematics have produced systems that discover explicit constructions and counterexamples across combinatorics and geometry (Mehrabian et al., 2024; Nagda et al., 2026; Mundinger et al., 2025), find new algorithms (Romera-Paredes et al., 2024), and even assist in formal proofs (Trinh et al., 2024; Hubert et al., 2026). This paper is concerned with a specific paradigm within this development: the *generative learning of extremal structures*. Pioneered by Wagner (2021), the idea was turned into a general template by PatternBoost (Charton et al., 2024) and its continuous extension FlowBoost (Bérczi et al., 2026). It alternates a global and a local stage (Algorithm 1), where a generative model draws complete candidate constructions, and local search refines them. The best refined constructions then serve as the training set for the next generator.

Typically, the generative model is an autoregressive transformer operating on a serialization of the combinatorial object. This makes the framework flexible and broadly applicable across discrete constructions. For graphs, two features of this design motivate exploring an alternative. First, symmetry is not necessarily encoded: an unordered vertex collection is forced into a linear order that carries no mathematical meaning, and the model must learn to be insensitive to it. Second, decisions cannot be taken back within the autoregressive sampling pass: an edge emitted early may turn out to be harmful once the rest of the graph has materialized, but the sampler provides no mechanism to revise it before local search. Sharing learned operations over graph structure also offers the prospect of reusing construction behavior at other sizes. This makes size transfer an additional reason to investigate a structure-aware generator: can training on smaller graphs produce useful constructions on larger ones without learning anew at each size?

To address these limitations, we integrate the permutation-equivariant Pairformer of Mehrabian et al. (2024) with discrete flow matching (Gat et al., 2024; Qin et al., 2025) in the PatternBoost loop. The generator updates all potential edges in parallel, allowing earlier decisions to be revised in light of the current whole-graph state.

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.

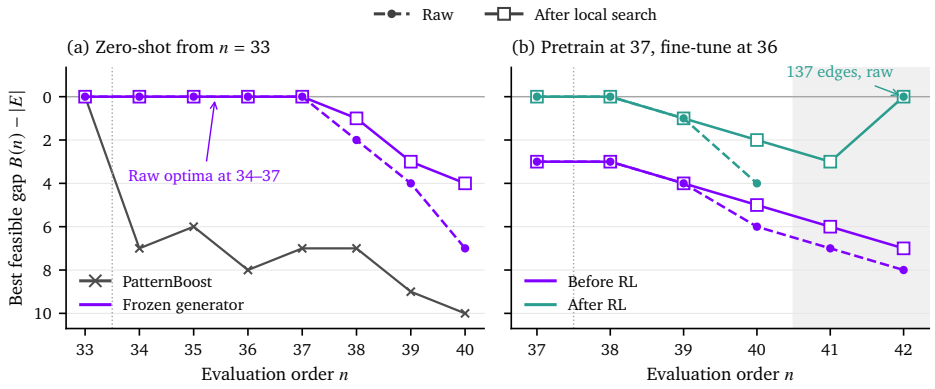


Figure 1: **Zero-shot upward transfer.** Gap to the best-known edge count $B(n)$, with zero at the top. Left: a frozen 33-vertex generator creates optimal graphs up to order 37 before local search. Right: Reward fine-tuning enables discovery up to order 42. Full details are in Appendix G.1.

The learned maps are shared across potential edges, so the same weights can be evaluated at different graph sizes. Our central empirical finding is that learned construction behavior remains useful at unseen larger sizes (Figure 1). We demonstrate the framework on two complementary constraint families. In C_4-free edge maximization, we seek a graph with as many edges as possible, subject to the constraint that no four distinct vertices form a cycle. In Ramsey construction, we seek graphs with neither a clique nor an independent set of specified sizes: a clique has every possible edge, whereas an independent set has none. For the first task, a generator trained only on smaller graphs produces larger graphs with the maximum possible number of edges, without any local search. For the second, we also observe generalization beyond the training regime.

We further introduce reward fine-tuning (Black et al., 2024) to improve this reuse. The generator is rewarded for the quality its proposals reach after local search. These gains extend to graph sizes that receive no fine-tuning (Section 4.2). In particular, a generator pretrained on 37 vertices and fine-tuned on 36 produces a C_4-free graph on 42 vertices with 137 edges before local search, matching the best-known edge count (Afzaly & McKay, 2026) without training at the evaluation size. To the best of our knowledge, this is the first instance of a learned approach attaining that bound. Throughout, we evaluate raw and refined outputs separately to distinguish direct construction from the quality of proposals as search starts.

A structural analysis in Section 3.5 complements these experiments: shared triangle-attention operations can represent the path counts governing C_4 edge moves over a bounded range of orders. This identifies objective-relevant computations that can be shared across orders, while successful learned transfer remains an empirical finding.

Our contributions are:

1. **Graph generation within learned search.** We integrate an equivariant pair network and discrete flow matching into PatternBoost as the proposal mechanism, allowing all edge decisions to respond to the complete tentative graph and to change during generation.
2. **Zero-shot size transfer and reward fine-tuning.** We show that learned graph construction transfers to unseen sizes, including an optimal 33-vertex C_4-free graph generated by a model trained at 25 vertices, without fine-tuning or local search, and valid Ramsey constructions across multiple training seeds. Reward fine-tuning improves mean refined quality across training and evaluation sizes, with gains extending beyond the fine-tuning size and improving on unchanged target-size models in mean refined quality.
3. **Structural motivation for transfer.** We show that shared triangle-attention operations can represent the path counts governing C_4 edge moves over a bounded range of orders, and contrast this with an exact obstruction for Ramsey move costs. These representation results motivate studying transfer further.

2 RELATED WORK

Learning from successful constructions. The cross-entropy method fits a distribution to its best samples (Rubinstein, 1999); neural construction policies bring this idea to mathematical objects (Wagner, 2021). PatternBoost adds local improvement before selection (Charton et al., 2024). We retain this learning loop and change how graphs are generated. Unlike FunSearch and AlphaEvolve, which search over construction programs (Romera-Paredes et al., 2024; Nagda et al., 2026), our model directly specifies a distribution over graphs.

Graph structure and generation. We reuse the Pairformer of Mehrabian et al. (2024), whose Evoformer-derived triangle attention (Jumper et al., 2021) guides AlphaZero edge edits for graphs avoiding both triangles and four-cycles. Their search initializes larger problems with good smaller graphs; we transfer the generator's weights. GraphRNN and GRAN also encode graph structure, but generate autoregressively (You et al., 2018; Liao et al., 2019). DiGress introduces categorical graph denoising (Vignac et al., 2023), DeFoG develops discrete graph flow matching (Qin et al., 2025), and GGFlow applies flow matching, triangle attention, and policy-gradient guidance for molecular properties (Hou et al., 2024). Its policy gradient favors rewarded clean-graph predictions while penalizing divergence from the pretrained model. Our fine-tuning instead clips probability ratios for sampled whole-graph transitions and rewards the construction reached after external local search. We use the generator as the proposal distribution in an evolving extremal search, retraining on selected refined graphs in each generation, and test reuse of its weights at unseen vertex counts.

Reward and search. FlowBoost combines flow proposals, refinement, and reward-guided adaptation for geometric discovery (Bérczi et al., 2026). It reweights endpoints while regularizing toward a frozen teacher. Our adaptation instead applies a clipped policy gradient (Black et al., 2024) to whole-graph transitions, rewarding the construction reached by external local search. GFlowNets learn reward-proportional distributions (Bengio et al., 2021), while we maximize expected refined quality. Local Search GFlowNets learn to backtrack and reconstruct sampling trajectories (Kim et al., 2024); our refiner is a separate graph heuristic. Finally, Erdős Goes Neural and DIFUSCO seek solutions on a supplied graph (Karalias & Loukas, 2020; Sun & Yang, 2023). In our work, the graph itself is the solution.

3 METHOD

We train our GNN generator via the PatternBoost loop. Optional reward fine-tuning optimizes quality after local search. Both use the same whole-graph sampler, with weights shared across graph sizes. We represent a simple undirected graph G on n vertices by its binary edge vector $X = (X_{\{i,j\}})_{i<j} \in \{0,1\}^M$, where $M = \binom{n}{2}$ and $X_{\{i,j\}} = 1$ exactly when $\{i,j\} \in E(G)$. We identify graphs with their edge-vector representations throughout.

3.1 LEARNING GRAPH PROPOSALS

The PatternBoost loop: search and selection. The full loop is detailed in Algorithm 1. In generation g, the generator p_{θ_g} draws a *raw* population $\mathcal{P}_g$. A local search operator S improves each candidate according to cost L, yielding the *refined* population $\widetilde{\mathcal{P}}_g$. The best refined graphs and an elite archive form the corpus $\mathcal{C}_g$ for the next fit. Our contribution concerns the two learned operations: How the generator is trained during Fit and how samples are drawn from p_θ.

Training: Learning by denoising. PatternBoost and its variants learn a graph serialization $s(X) = (s_1, \ldots, s_T)$ autoregressively:

$$p_\theta(s) = \prod_{\ell=1}^{T} p_\theta(s_\ell \mid s_{<\ell}), \qquad \mathcal{L}_{\mathrm{AR}}(\theta) = -\mathbb{E}_{X\sim q_{\mathcal{C}_g}} \sum_{\ell=1}^{T} \log p_\theta(s_\ell(X) \mid s_{<\ell}(X)). \tag{1}$$

Each prediction sees only the preceding tokens, and an arbitrary order is enforced on the edges. We instead formulate graph generation as a continuous-time Markov chain on graphs, with transition rates parameterized through a denoising network. Our Fit operation (line 9 of Algorithm 1) learns

Algorithm 1 Learning graph proposals at a fixed size

1: **Input:** generator p_{θ_0}, cost L, local search S, population size B, selection size k, elite size m, generations G_{gen}
2: $\mathcal{E}_{-1} \leftarrow \emptyset$
3: **for** $g = 0, \ldots, G_{\text{gen}} - 1$ **do**
4: $\quad \mathcal{P}_g \leftarrow (G_g^{(b)})_{b=1}^{B}$ with $G_g^{(b)} \sim p_{\theta_g}$
5: $\quad \widetilde{\mathcal{P}}_g \leftarrow (S(G_g^{(b)}))_{b=1}^{B}$
6: $\quad \mathcal{T}_g \leftarrow$ the k lowest-cost entries of $\widetilde{\mathcal{P}}_g$
7: $\quad \mathcal{E}_g \leftarrow$ the m best distinct graphs in $(\mathcal{E}_{g-1} \cup \widetilde{\mathcal{P}}_g)$
8: $\quad \mathcal{C}_g \leftarrow \mathcal{T}_g \uplus \mathcal{E}_g$
9: $\quad \theta_{g+1} \leftarrow \mathrm{Fit}(\theta_g, \mathcal{C}_g)$
10: **end for**

these rates indirectly by training the network to predict clean edge indicators from a corrupted whole graph.

Our training procedure follows DeFoG (Qin et al., 2025): We draw a batch of target graphs $X \sim q_{\mathcal{C}_g}$ uniformly from the corpus entries and a shared time $t \sim \mathcal{U}[0, 1)$. We independently retain each pair's category with probability t and otherwise draw it uniformly, giving a single corrupted graph Z_t the distribution:

$$q_t(z \mid x) = \prod_e \left[t\mathbf{1}_{\{z_e = x_e\}} + \frac{1-t}{2} \right]. \tag{2}$$

Thus Z_0 is random noise, while Z_t approaches X as $t \to 1$. Every entry is an edge or nonedge, with no corruption marker. Given (Z_t, t), the denoiser $d_{\theta,e}(\cdot \mid Z_t, t)$ predicts clean pair categories by minimizing

$$\mathcal{L}_{\text{DFM}}(\theta) = -\mathbb{E}_{X,t,Z_t} \frac{1}{M} \sum_e \log d_{\theta,e}(X_e \mid Z_t, t). \tag{3}$$

For the interpolation in Equation (2), the trained denoiser defines the transition rate (Gat et al., 2024)

$$r_{\theta,e}(z, c, t) = \frac{d_{\theta,e}(c \mid z, t)}{1-t}, \qquad c \neq z_e, \tag{4}$$

for changing edge indicator e to category c. Thus, training determines the chain's generation dynamics without requiring simulation of full trajectories. Fresh times and corruptions provide new learning inputs even for a small corpus.

Inference: Generating by repeated revision. To draw a proposal (line 4 of Algorithm 1), we approximate this learned continuous-time chain with K parallel update steps, starting from independent uniform binary indicators Z_0. With step size $\Delta t = 1/K$, we use $t_k = k/K$ and $\eta_k = \Delta t/(1 - t_k) = 1/(K - k)$:

$$P_{\theta,k}(z'_e = c \mid z) = (1 - \eta_k)\mathbf{1}_{\{c = z_e\}} + \eta_k d_{\theta,e}(c \mid z, t_k), \qquad k = 0, \ldots, K-1. \tag{5}$$

At each step, independently redraw each edge indicator from the denoiser with probability η_k, otherwise retaining its value. Conditioning subsequent predictions on the updated graph couples edge decisions across steps and permits revision.

3.2 A SIZE-INDEPENDENT PARAMETRIZATION OF THE DENOISER

Our GNN is inspired by triangle-attention Pairformer of Mehrabian et al. (2024). Each potential edge, present or absent, carries a vector $h_{ij} \in \mathbb{R}^d$. The inputs are all vertex pairs and time t. We supply no node features, vertex identifiers, positional encodings, or precomputed graph statistics. Structural context comes entirely from interactions between pairs.

Triangle attention. To update (i, j), attention gathers information from (i, k) using (j, k) as context. These three potential edges form a triangle regardless of which are present. In one head,

$$\alpha_{ijk} = \mathrm{softmax}_k \left(q_{ij}^{\mathsf{T}} k_{ik} / \sqrt{d_h} + b_{jk} \right), \qquad o_{ij} = \sum_k \alpha_{ijk} v_{ik}. \tag{6}$$

Here q, k, v are learned pair-state projections, d_h is the head width, and the learned scalar b_{jk} lets the third pair control attention. Each block applies attention along both endpoints, then a pairwise feedforward update. A shared readout predicts edge or nonedge. Appendix A gives the full equations.

Symmetry, size, and cost. Learned maps act on feature channels and are shared across pairs. Relabeling vertices therefore relabels predictions, making the architecture equivariant to vertex permutations (Appendix A.3). Changing n requires no new weights or output heads, enabling size transfer. Section 4.1 tests its usefulness. Attention sums over a third vertex for every pair, covering all $\binom{n}{3}$ triples of distinct vertices. A network call therefore costs $O(n^3)$ at fixed width and depth. This cubic cost further motivates training on smaller graphs and reusing the weights at larger sizes.

3.3 Reward fine-tuning

After denoising pretraining, we fine-tune the generator to produce graphs that lead to good constructions after local search. Let $p_{\theta,n}$ denote the distribution of sampling trajectories $\tau = (Z_0, \dots, Z_K)$ and R_n the reward for the refined graph. We maximize

$$J_n(\theta) = \mathbb{E}_{\tau \sim p_{\theta,n}}[R_n(S(Z_K))]. \tag{7}$$

We treat each whole-graph transition as a stochastic action and apply the clipped policy update of Black et al. (2024). The whole-graph transition probability is the product of the edge-transition probabilities in Equation (5): $P_{\theta,k}(z' \mid z) = \prod_e P_{\theta,k}(z'_e \mid z)$. With batch-standardized refined reward $\widehat{A}$ and transition ratio $\rho_{\theta,k} = P_{\theta,k}(Z_{k+1} \mid Z_k)/P_{\theta_{\text{old}},k}(Z_{k+1} \mid Z_k)$, we minimize

$$\mathcal{L}_{\text{PG}}(\theta) = -\mathbb{E}_{\tau,k} \min\Big\{\rho_{\theta,k}\widehat{A}, \text{clip}(\rho_{\theta,k}, 1-\epsilon, 1+\epsilon)\widehat{A}\Big\}. \tag{8}$$

Above-average refined rewards increase the likelihood of sampled transitions, without differentiating through local search (Appendix C).

3.4 Graph objectives

We instantiate the cost L and local search S for two constraint families.

C_4-free edge maximization. We seek as many edges as possible without a four-cycle:

$$\max_{G:\,|V(G)|=n} |E(G)| \qquad \text{s.t.} \quad N_{C_4}(G) = 0. \tag{9}$$

Here $|E(G)|$ counts edges and $N_{C_4}(G)$ counts four-cycles: four distinct vertices joined in a closed path of four edges. Extra edges between those vertices do not remove the cycle. Since a graph has at most M edges, we choose a penalty of $M+1$ per four-cycle so that one violation costs more than any possible edge gain:

$$L_{C_4}(G) = (M+1)N_{C_4}(G) - |E(G)|. \tag{10}$$

Local search deletes edges until no four-cycle remains, then greedily adds edges preserving feasibility. For reward fine-tuning, $R_n(G) = |E(G)|$. Thus, infeasible proposals can earn high rewards if refinement turns them into dense feasible graphs.

Ramsey construction. A Ramsey $R(r, s)$ graph contains no *clique* (complete subgraph) of size r and no *independent set* (vertices with no edges between them) of size s. We count these violations as cliques in G and its complement $\overline{G}$:

$$L_{\text{R}}(G) = N_{K_r}(G) + N_{K_s}(\overline{G}). \tag{11}$$

Here N_{K_r} counts r-cliques. A graph is valid exactly when $L_{\text{R}} = 0$. We denote the task on n vertices by $R(r, s; n)$. A solution certifies $R(r, s) \geq n + 1$. Local search uses a bounded tabu procedure to add or delete edges and reduce violations (Appendix C).

3.5 WHOLE-GRAPH CONTEXT AND SIZE TRANSFER

An edge can look harmless until others complete a forbidden pattern. If a graph H lacks edge $e = \{i, j\}$, adding it closes each simple length-three path from i to j into a four-cycle. Writing $P_3(H; i, j)$ for their count,

$$L_{C_4}(H + e) - L_{C_4}(H) = (M + 1)P_3(H; i, j) - 1. \tag{12}$$

If H is feasible, adding e helps precisely when no such path exists. An autoregressive model deciding e before the rest of the path must account for future completions. Our denoiser sees a current value for every edge and can reconsider e as the graph changes. These path counts also provide a concrete example of objective-relevant computations that the architecture can share across graph sizes. We provide the following motivational proposition:

Proposition 1 (Expressivity of C_4 edge-move computations) *For every finite $N \geq 2$, a two-block triangle-attention Pairformer with hidden and feedforward widths that may depend on N has a parameter setting whose scalar logit readout gives the simple length-three path count for every potential edge of every graph on $2 \leq n \leq N$ vertices. The same architecture family can represent the exact C_4 single-edge flip costs throughout this size range.*

The construction composes triangle-attention operations to recover length-three walk counts and subtract walks that revisit an endpoint. The result establishes that one parameter setting can express these computations across the stated size range; whether training on smaller graphs discovers useful shared computations is evaluated empirically in Section 4.1. The proof is in Appendix B, where also the limited applicability to the Ramsey problem is discussed.

4 EXPERIMENTS

We test whether learned graph construction transfers beyond the training size (Section 4.1), whether reward fine-tuning improves that transfer (Section 4.2), and whether reusing a model can outperform training at the target size (Section 4.3). We distinguish raw generation from generation followed by local search throughout. For C_4, refinement always returns a feasible graph. We compare edge counts with $B(n)$, the proved optimum through $n = 40$ and the best-known lower bound at $n = 41, \ldots, 49$ (Afzaly & McKay, 2026). All hyperparameters are collected in Appendix C.

4.1 GENERATING BEYOND THE TRAINING SIZE

We pretrain separate C_4 generators at orders 20, 25, 30 and 33, then freeze their weights and evaluate generation at orders up to 49. We measure both raw constructions and the same proposals after greedy refinement (Figures 1 and 8).

All four training sizes yield zero-shot optima far above their training size. Training at 25 vertices produces an optimal 33-vertex graph with 96 edges *before local search* and an optimal 36-vertex graph with 110 edges after refinement. These are size increases of 32% and 44%, respectively (Figure 8 and Tables 1 and 11). The best constructions pool draws across the evaluated models and seeds (protocol in Table 1). At the same target orders, PatternBoost reports 96 and 102 edges under its target-size training and search protocols.

Transfer also holds across consecutive unseen sizes. A single generator trained on 33 vertices reaches the proved optimum at each of orders 34–37 before local search (Figure 1, left). This checkpoint is selected using only performance at its training size; its weights remain frozen at every target size. Full distributions, transfer from smaller sources and through order 49, and density and extension controls appear in Appendices E and F.

Transfer under Ramsey constraints. We separately train generators for each Ramsey problem: $R(4, 4)$ at 17 vertices, $R(4, 5)$ at 24, $R(3, 7)$ at 21, and $R(4, 6)$ at 30. We evaluate the first two at their training size and the latter two at 22 and 31 vertices, respectively (Table 2). The models generate valid $R(4, 4; 17)$ graphs directly; $R(4, 5; 24)$ constructions are rarer and benefit from refinement, which succeeds on the learned proposals. For upward transfer, all three models trained on $R(3, 7; 21)$ produce valid 22-vertex graphs before local search. For $R(4, 6)$, training at 30 vertices provides

Table 1: **Farthest unseen size reaching the proved C_4 optimum.** Each cell gives target order (increase from training order), before or after local search. The order-20/25/30 rows pool 90,000 draws per target over three training seeds; the order-33 row uses one checkpoint selected at its training size and 10,000 draws per target. These are observed maxima, not success rates. Full distributions: Appendices E and F.

Pretrained at	Raw n (Δn)	Refined n (Δn)
20	27 (+7)	31 (+11)
25	33 (+8)	36 (+11)
30	33 (+3)	36 (+6)
33	37 (+4)	37 (+4)

useful 31-vertex search starts: refinement finds a valid graph in every learned-model evaluation. Matched random-start controls are reported in Appendix H.2. Fixed-size benchmarks and the full range of transfer settings are reported in Appendices H.1 and H.3.

Table 2: **Ramsey construction at trained and unseen sizes.** Arrows give training and evaluation orders. Valid counts include duplicates; mean costs include all draws. Learned rows pool independently trained models and generated samples. Protocols and random-start controls: Appendix H.2.

			Raw		Refined	
Task / transfer	Proposals	Draws	Valid	Mean cost	Valid	Mean cost
Construction at the training size						
$R(4,4)$, $17 \to 17$	Learned	30,000	28,300	0.74	28,698	0.27
$R(4,5)$, $24 \to 24$	Learned	30,000	0	14.30	2	7.95
Transfer to larger graphs						
$R(3,7)$, $21 \to 22$	Learned	90,000	2,620	7.28	5,409	2.23
$R(4,6)$, $30 \to 31$	Learned	60,000	0	31.82	63	7.89

4.2 Fine-tuning improves search across sizes

We pretrain seven models, one at orders 36–42 each. They are independently fine-tuned at every order 33–42 and also evaluated at every order 33–42. This gives $7 \times 10 \times 10 = 700$ source/adaptation/evaluation settings per method. We compare reward fine-tuning with the unchanged source and a fixed-corpus denoising control. Starting from the same pretrained weights, this control continues the denoising objective equation 3 on one selected graph corpus at the fine-tuning order, held unchanged throughout adaptation. Evaluation uses the same sampler, refiner and number of evaluation draws.

Reward fine-tuning improves mean refined quality in 699 of these 700 comparisons with unchanged sources, including 629 of 630 evaluations away from the fine-tuning order. It also outperforms fixed-corpus denoising in all 700 comparisons. Figure 2 illustrates transfer from one fixed source pretrained at 36. Averages across all seven source orders and source-wise results are in Figure 9 and table 15. Evaluations reuse the same adapted models across target sizes.

We also pretrained models at order 25 and fine-tuned and evaluated them at orders 33–42, giving $10 \times 10 = 100$ settings. Adaptations improve mean refined edge counts in all 100 fine-tuning/evaluation settings, including all 45 evaluations above the fine-tuning order. Improvements average 4.99 edges over the unchanged source, ranging from 3.51 to 6.24 (Figure 20 and appendix G.3). Reward fine-tuning also lowers both raw and refined mean costs in all paired Ramsey adaptation settings (Appendix H.4).

Better search starts need not be more feasible. The $36 \to 37 \to 38$ example explains what refinement gains can look like: reward fine-tuning shifts proposals toward denser graphs, often

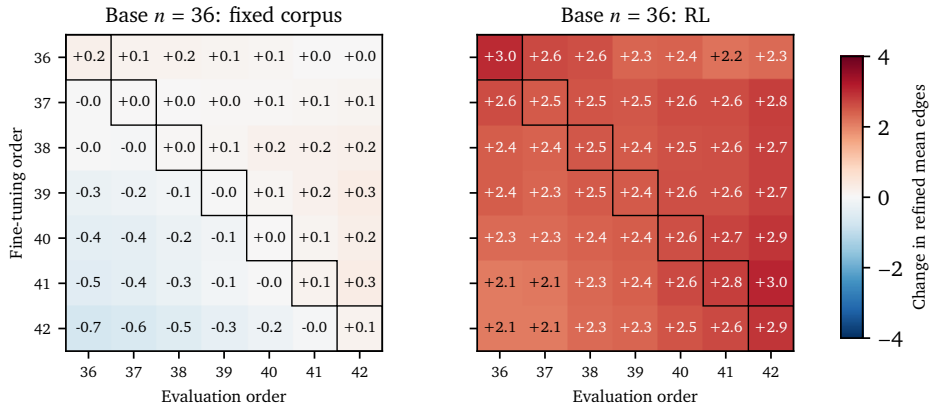


Figure 2: **Reward gains transfer across graph sizes.** One fixed model pretrained at 36 is independently adapted and evaluated at orders 36–42. Cells show changes in mean refined edges relative to this unchanged source, with 10,000 draws per population. Cells above the diagonal test orders above training sizes. Averages and individual source grids: Appendix G.2.

beyond the feasible edge limit. Raw feasibility falls, while the refined quality distribution improves (Figure 3). The learned proposals become more useful inputs to local search.

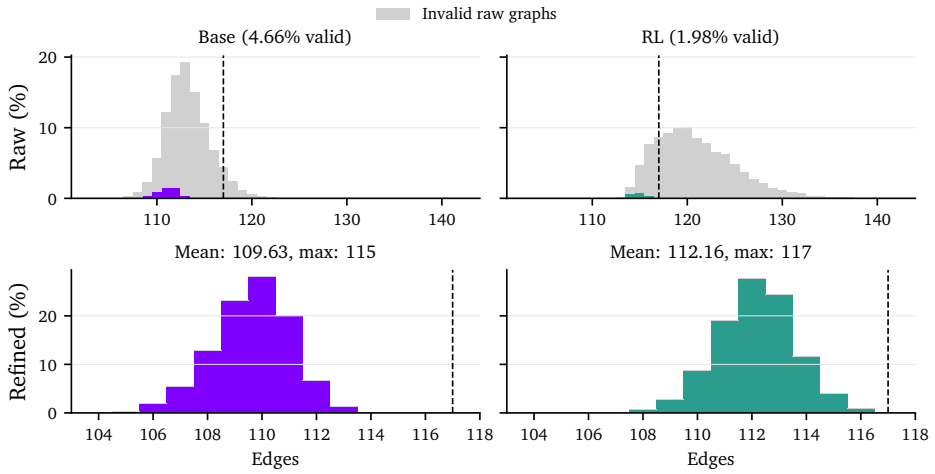


Figure 3: **Fine-tuning learns useful search starts.** Left: A model pretrained at 36. Right: The same checkpoint fine-tuned at 37. Both are evaluated at 38. Colored bars are feasible; grey bars contain four-cycles. Dashed lines mark the optimum of 117 edges. Fine-tuning visibly moves the mean of both raw and refined graphs upwards, at the cost of a lower feasibility rate. Fixed-corpus control: Figure 10.

Rewarding refined quality can also yield a best-known construction *before* local search. A generator pretrained at 37 and fine-tuned at 36 produces a 42-vertex graph with 137 edges, matching the best-known count without training at the evaluation size (Figure 1, right). Together with the mean-quality gains above, this shows that reward fine-tuning improves both the typical and the best-case search starts of the generator.

4.3 Pretrain small, adapt, and reuse

Triangle attention processes vertex triples, so training a new generator becomes more expensive as graph size grows. An order-25 source can instead be trained once and adapted for larger orders. The

selected adapted pipelines have higher mean refined edge counts and lower H200 training times than Direct at every tested order (Tables 3 and 8). The training-time saving is up to 49% at $n = 42$, even when each pipeline is charged the full pretraining cost. Reusing the source across several targets can spread training cost further. These results show that pretraining on smaller graphs combined with focused reward fine-tuning can yield better search starts at larger orders while avoiding the cost of a full training run at each target size.

Table 3: **Model reuse improves construction quality.** Train → FT gives pretraining and fine-tuning orders. Quality uses 10,000 draws; PB Best is source-reported. $\mathrm{RL}_{\mathrm{best}}$ uses the adaptation grid; RL_{25} uses the order-25 source. Times are hours, including H200 timing replications. LS counts per-graph refinements in millions. Costs exclude model selection. Bold marks quality maxima and total-time/search minima. Selection and accounting: Appendix D.

			Raw		**Refined**		Training time (h)			
n	Method	Train → FT	Valid (%)	Best	Mean	Best	Pretrain	FT	Total	LS (M)
33	PB	33	—	—	—	**96**	—	—	—	116.500
	Direct (ours)	33	4.93	**96**	91.61	**96**	9.30	—	9.30	0.620
	$\mathrm{RL}_{\mathrm{best}}$ (ours)	36 → 33	**11.59**	**96**	**94.02**	**96**	8.81	0.89	9.70	0.778
	RL_{25} (ours)	25 → 33	0.00	—	93.64	**96**	5.44	0.81	**6.26**	0.778
36	PB	36	—	—	—	102	—	—	—	38.400
	Direct (ours)	36	**12.49**	108	102.08	108	8.81	—	8.81	0.630
	$\mathrm{RL}_{\mathrm{best}}$ (ours)	36 → 35	6.29	**110**	**105.06**	**110**	8.81	0.85	9.67	0.778
	RL_{25} (ours)	25 → 34	0.00	—	104.73	**110**	5.44	0.83	**6.27**	0.778
39	PB	39	—	—	—	113	—	—	—	11.700
	Direct (ours)	39	**24.00**	116	112.61	117	11.63	—	11.63	0.630
	$\mathrm{RL}_{\mathrm{best}}$ (ours)	36 → 37	0.40	**120**	**115.88**	**121**	8.81	0.94	9.75	0.778
	RL_{25} (ours)	25 → 37	0.00	—	114.71	**121**	5.44	0.95	**6.40**	0.778
40	PB	40	—	—	—	117	—	—	—	11.200
	Direct (ours)	40	**31.64**	121	115.76	121	11.34	—	11.34	0.630
	$\mathrm{RL}_{\mathrm{best}}$ (ours)	38 → 33	0.04	**127**	**118.64**	**127**	10.65	0.89	11.54	0.778
	RL_{25} (ours)	25 → 38	0.00	—	118.58	125	5.44	1.07	**6.52**	0.778
42	PB	42	—	—	—	124	—	—	—	6.800
	Direct (ours)	42	**22.95**	128	123.34	128	12.58	—	12.58	0.630
	$\mathrm{RL}_{\mathrm{best}}$ (ours)	37 → 36	0.02	**137**	**126.98**	**137**	10.70	0.89	11.59	0.778
	RL_{25} (ours)	25 → 37	0.00	—	126.19	134	5.44	0.95	**6.40**	0.778

5 Conclusion and Limitations

Equivariant flow matching reaches C_4 optima at unseen orders before local search, and Ramsey results extend the same zero-shot transfer to a second constraint family. Pretraining small followed by reward fine-tuning improves larger-order refined constructions with 29–49% lower H200 training totals for the selected pipelines, including the full pretraining cost. Reusing the source further amortizes that cost across adaptations and targets. Most notably, our pipeline discovered a 137-edge C_4-free graph on 42 vertices before local search, using a model never trained at that size. To our knowledge this is the first learned construction to attain that count.

Limitations. Our experimental setup does not fully quantify how much of the gains come from the equivariant architecture versus flow-matching training. Our headline transfer results are also rare events: the 137-edge witness appears once in 10,000 draws, and the farthest zero-shot optima are best-of-pool draws rather than typical ones. Rare events are, however, precisely what mathematical discovery is about: a single verified construction settles an existence claim. The relevant question is what the rare events are, and here they are best-known constructions produced before local search by models never trained at the target size, while the typical draws steadily improve as search starts.

Reproducibility statement

The main text specifies the objectives, corruption, sampler and policy update. Full statements and proofs are in Appendices A and B. Appendix C collects every hyperparameter and training setting; Appendices D to G give model selection, evaluation budgets, and evaluation protocols. Best construction witnesses were independently recounted.

AI use statement

We used OpenAI Codex with GPT-5.6 Sol, GPT-5.6 Terra, GPT-6 Astra and GPT-6 Sol for literature research, assistance with writing research code, and text polishing. An author reviewed all AI-assisted outputs. We take responsibility for the final content of this work, including AI-assisted text, code, and claims.

References

Narjess Afzaly and Brendan D. McKay. Extremal Graphs and Turan numbers. Author-maintained extremal graph catalogue, 2026. URL https://users.cecs.anu.edu.au/~bdm/data/extremal.html. Accessed September 15, 2026; edge maxima and completeness of graph lists have separate status flags.

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow Network based Generative Models for Non-Iterative Diverse Candidate Generation. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://arxiv.org/abs/2106.04399.

Gergely Bérczi, Baran Hashemi, and Jonas Klüver. Flow-based Extremal Mathematical Structure Discovery, 2026. URL https://arxiv.org/abs/2601.18005v1. arXiv preprint.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training Diffusion Models with Reinforcement Learning. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2305.13301.

Cristian Bodnar, Fabrizio Frasca, Yuguang Wang, Nina Otter, Guido F. Montufar, Pietro Lió, and Michael Bronstein. Weisfeiler and Lehman Go Topological: Message Passing Simplicial Networks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 1026–1037. PMLR, 2021. URL https://proceedings.mlr.press/v139/bodnar21a.html.

François Charton, Jordan S. Ellenberg, Adam Zsolt Wagner, and Geordie Williamson. PatternBoost: Constructions in Mathematics with a Little Help from AI, 2024. URL https://arxiv.org/abs/2411.00566. arXiv preprint.

Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can Graph Neural Networks Count Substructures? In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://arxiv.org/abs/2002.04025.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete Flow Matching. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/f0d629a734b56a642701bba7bc8bb3ed-Abstract-Conference.html.

Xiaoyang Hou, Tian Zhu, Milong Ren, Dongbo Bu, Xin Gao, Chunming Zhang, and Shiwei Sun. Improving Molecular Graph Generation with Flow Matching and Optimal Transport, 2024. URL https://arxiv.org/abs/2411.05676v1. Version 1. Later titled *GGFlow: A Graph Flow Matching Method with Efficient Optimal Transport*.

Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z. Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom, Ottavia Bertolli, Tom Zahavy, Amol Mandhane, J.-Y. Yung, Iuliya Beloshapka, Borja Ibarz, Vivek Veeriah, Lei

Yu, Oliver Nash, Paul Lezeau, Salvatore Mercuri, Calle Sönne, Bhavik Mehta, Alex Davies, Daniel Zheng, Fabián Pedregosa, Yin Li, Ingrid von Glehn, Mark Rowland, Samuel Albanie, Ameya Velingker, Simon Schmitt, Edward Lockhart, Edward Hughes, Henryk Michalewski, Nicolas Sonnerat, Demis Hassabis, Pushmeet Kohli, and David Silver. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, 651(8106):607–613, 2026. doi: 10.1038/s41586-025-09833-y.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon Köhl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michał Zieliński, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian W. Bodenstein, David Silver, Oriol Vinyals, Andrew Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596:583–589, 2021. doi: 10.1038/s41586-021-03819-2. URL https://www.nature.com/articles/s41586-021-03819-2.

Nikolaos Karalias and Andreas Loukas. Erdos Goes Neural: an Unsupervised Learning Framework for Combinatorial Optimization on Graphs. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://arxiv.org/abs/2006.10643.

Minsu Kim, Taeyoung Yun, Emmanuel Bengio, Dinghuai Zhang, Yoshua Bengio, Sungsoo Ahn, and Jinkyoo Park. Local Search GFlowNets. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.02710v2.

Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Charlie Nash, William L. Hamilton, David Duvenaud, Raquel Urtasun, and Richard S. Zemel. Efficient Graph Generation with Graph Recurrent Attention Networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://arxiv.org/abs/1910.00760.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably Powerful Graph Networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://arxiv.org/abs/1905.11136.

Abbas Mehrabian, Ankit Anand, Hyunjik Kim, Nicolas Sonnerat, Matej Balog, Gheorghe Comanici, Tudor Berariu, Andrew Lee, Anian Ruoss, Anna Bulanova, Daniel Toyama, Sam Blackwell, Bernardino Romera Paredes, Petar Veličković, Laurent Orseau, Joonkyung Lee, Anurag Murty Naredla, Doina Precup, and Adam Zsolt Wagner. Finding Increasingly Large Extremal Graphs with AlphaZero and Tabu Search. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 6985–6993, 2024. doi: 10.24963/ijcai.2024/772. URL https://www.ijcai.org/proceedings/2024/772.

Konrad Mundinger, Max Zimmer, Aldo Kiem, Christoph Spiegel, and Sebastian Pokutta. Neural Discovery in Mathematics: Do Machines Dream of Colored Planes?, 2025. URL https://arxiv.org/abs/2501.18527. arXiv preprint.

Ansh Nagda, Prabhakar Raghavan, and Abhradeep Thakurta. Reinforced Generation of Combinatorial Structures: Ramsey Numbers, 2026. URL https://arxiv.org/abs/2603.09172. arXiv preprint.

Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. DeFoG: Discrete Flow Matching for Graph Generation. In *Proceedings of the International Conference on Machine Learning*, volume 267, 2025. URL https://proceedings.mlr.press/v267/qin25d.html.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. doi: 10.1038/s41586-023-06924-6. URL https://doi.org/10.1038/s41586-023-06924-6. First published online in December 2023.

Reuven Y. Rubinstein. The Cross-Entropy Method for Combinatorial and Continuous Optimization. *Methodology and Computing in Applied Probability*, 1:127–190, 1999. doi: 10.1023/A:1010091220143. URL https://link.springer.com/article/10.1023/A:1010091220143.

Zhiqing Sun and Yiming Yang. DIFUSCO: Graph-based Diffusion Solvers for Combinatorial Optimization. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2302.08224.

Trieu H. Trinh, Yuhuai Wu, Quoc V. Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024. doi: 10.1038/s41586-023-06747-5.

Clément Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. DiGress: Discrete Denoising diffusion for graph generation. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2209.14734.

Adam Zsolt Wagner. Constructions in combinatorics via neural networks, 2021. URL https://arxiv.org/abs/2104.14516. arXiv preprint.

Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models. In *Proceedings of the International Conference on Machine Learning*, 2018. URL https://arxiv.org/abs/1802.08773.

Cai Zhou, Rose Yu, and Yusu Wang. On the Theoretical Expressive Power and the Design Space of Higher-Order Graph Transformers. In *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 2179–2187. PMLR, 2024. URL https://proceedings.mlr.press/v238/zhou24a.html.

A MATHEMATICAL SPECIFICATION OF THE PAIRFORMER

We specify the triangle-attention Pairformer used in the main experiments: a time-conditioned map on dense vertex-pair tensors, used as denoisers in the categorical discrete-flow generator. The equations below describe our implementation, including its diagonal convention and residual ordering. The triangle-attention design follows the Evoformer operations of Jumper et al. (2021); Pairformer-guided extremal search is an earlier application of pair attention (Mehrabian et al., 2024). Our specification concerns the generative backbone, without an AlphaZero policy or value head. Figure 4 shows the pair interactions and the order of operations within one block.

A.1 STATE SPACE AND SHARED OPERATIONS

Fix $n \geq 2$ vertices, $C \geq 2$ edge categories and $M = \binom{n}{2}$ unordered pairs. A graph state is $x \in \{0, \dots, C-1\}^M$; in the experiments $C = 2$, with category zero representing a nonedge. Its dense encoding $X \in \mathbb{R}^{n \times n \times C}$ is $X_{ij} = X_{ji} = e_{x_{\{i,j\}}}$ for $i \neq j$, where e_c is a standard basis vector. The categorical adapter sets $X_{ii} = e_0$; this input convention is erased by the first hidden-state mask. There are no vertex identifiers or positional embeddings.

Write $Z \in \mathbb{R}^{n \times n \times d}$ for a hidden pair state and $Z^{\mathsf{T}}_{ij} = Z_{ji}$ for transposition of its vertex axes. Define the linear operators

$$(\mathcal{M}Z)_{ij} = \mathbf{1}_{\{i \neq j\}} Z_{ij}, \qquad \mathcal{S}Z = \tfrac{1}{2}(Z + Z^{\mathsf{T}}). \tag{13}$$

They commute. The hidden states at block boundaries are symmetric and have zero diagonal, although intermediate states within a block may be asymmetric. Every affine map acts on the channel axis and shares its parameters across all vertex pairs. Different layers and differently named maps have independent parameters unless stated otherwise.

Layer normalization acts separately on each pair vector:

$$\mathrm{LN}(z) = \gamma \odot \frac{z - \mu(z)\mathbf{1}}{\sqrt{v(z) + \epsilon}} + \beta, \quad \mu(z) = \frac{1}{d}\sum_{a=1}^{d} z_a, \quad v(z) = \frac{1}{d}\sum_{a=1}^{d}(z_a - \mu(z))^2, \tag{14}$$

where $\gamma, \beta \in \mathbb{R}^d$ are learned and $\epsilon = 10^{-5}$. Each occurrence corresponding to a different normalization module has its own γ, β. In particular, $\mathrm{LN}(0) = \beta$ need not vanish. We use $\odot$ for componentwise multiplication and $\sigma(u) = (1 + \exp(-u))^{-1}$ componentwise.

All dropout is disabled ($p = 0$) in the experiments and throughout the analysis of Appendix B, so the model is deterministic given (X, t). For an activation ϕ, define a pairwise transition

$$\mathcal{F}_\phi(Z)_{ij} = W_2 \phi(W_1 \,\mathrm{LN}(Z_{ij}) + b_1) + b_2, \tag{15}$$

with $W_1 \in \mathbb{R}^{f \times d}$, $W_2 \in \mathbb{R}^{d \times f}$.

Time conditioning and readout. For even time width $w = 2q$, define

$$\omega_a = 10000^{-a/q}, \quad a = 0, \dots, q-1, \qquad \tau(t) = \big(\cos(t\omega_a)\big)_{a=0}^{q-1} \,\|\, \big(\sin(t\omega_a)\big)_{a=0}^{q-1}. \tag{16}$$

Here $\|$ denotes concatenation. A shared time network computes

$$(s(t), b(t)) = W_{t,2}\,\mathrm{SiLU}(W_{t,1}\tau(t) + b_{t,1}) + b_{t,2}, \qquad \mathrm{SiLU}(u) = u\sigma(u), \tag{17}$$

where $W_{t,1} \in \mathbb{R}^{d \times w}$ and $W_{t,2} \in \mathbb{R}^{2d \times d}$. The Pairformer initializes

$$Z^{(0)} = \mathcal{S}\mathcal{M}[(W_{\mathrm{in}}X + b_{\mathrm{in}}) \odot (\mathbf{1} + s(t)) + b(t)], \tag{18}$$

where the time vectors broadcast over both vertex axes. After L blocks, its output logits are

$$\ell = \mathcal{S}\mathcal{M}\Big[W_{\mathrm{out}}\,\mathrm{LN}_{\mathrm{out}}(Z^{(L)}) + b_{\mathrm{out}}\Big] \in \mathbb{R}^{n \times n \times C}. \tag{19}$$

Only $i < j$ entries are returned to the generative method; no diagonal prediction participates in its loss or sampling.

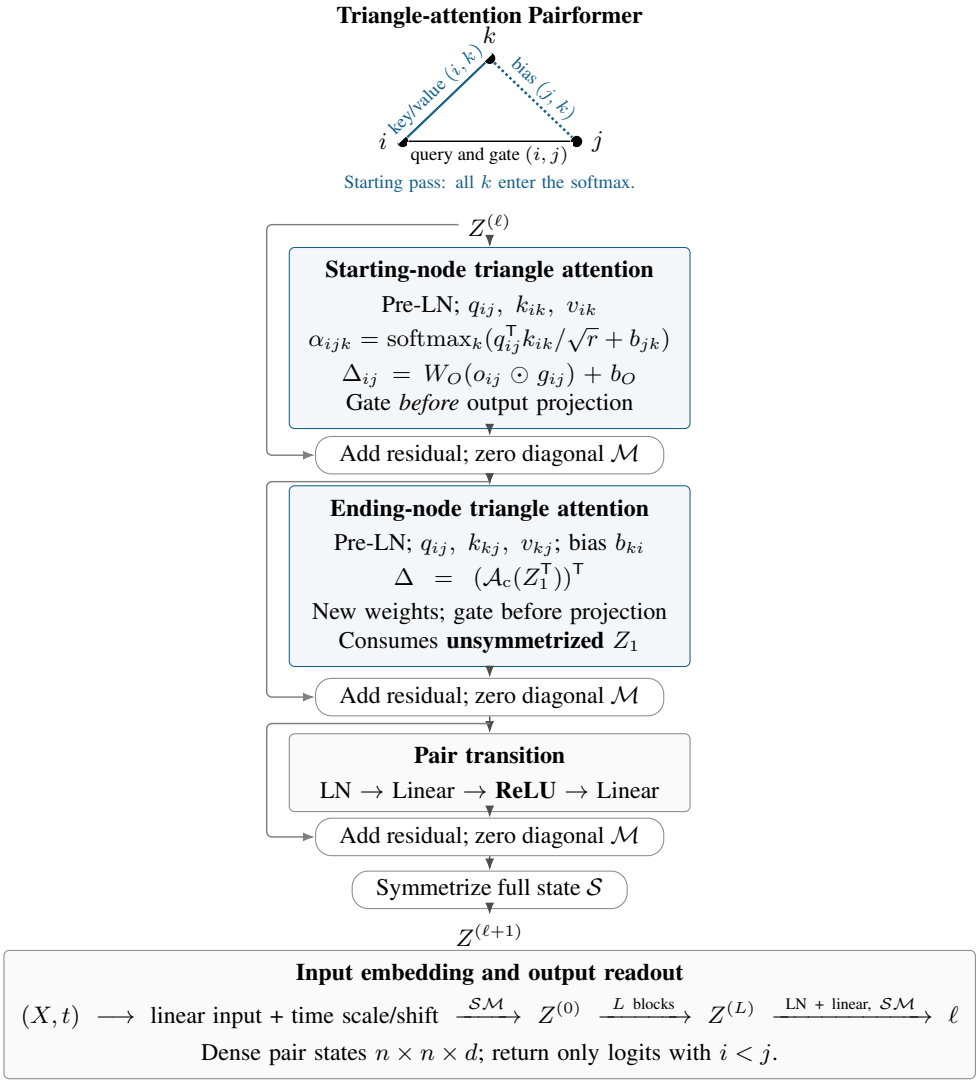


Figure 4: **Pair interactions and block order in the Pairformer.** The upper sketch shows feature roles for a third vertex k; lines denote pair features, which represent both edges and nonedges. The diagram shows one block, repeated L times with independent parameters. Side arrows carry the residual state. Here g_{ij} is a sigmoid gate computed from the normalized target pair, and o_{ij} concatenates the value-weighted head outputs; head superscripts are suppressed. Dropout is disabled ($p = 0$). All attention sums retain diagonal slots, although $\mathcal{M}$ zeros the state diagonal after each residual. $\mathcal{S}$ averages the two vertex orientations at the block exit. Equations in Appendix A specify biases, gates and the update order in full.

A.2 TRIANGLE-ATTENTION PAIRFORMER

Let H be the number of attention heads, with $d = Hr$ and head width r. We first define a starting-node attention operator $\mathcal{A}$ on an arbitrary, possibly asymmetric Z. Set $U = \mathrm{LN}_A(Z)$ and, for each head h, form

$$q_{ij}^h = W_Q^h U_{ij}, \qquad k_{ij}^h = W_K^h U_{ij}, \qquad v_{ij}^h = W_V^h U_{ij}, \qquad b_{jk}^h = (w_B^h)^\mathsf{T} U_{jk}. \tag{20}$$

The query, key and value maps have shape $r \times d$, and the bias vector has length d. These four projections have no additive biases. For fixed (i, j, h), attention is normalized over all n choices of k:

$$a_{ijk}^h = \frac{(q_{ij}^h)^\mathsf{T} k_{ik}^h}{\sqrt{r}} + b_{jk}^h, \qquad \alpha_{ijk}^h = \frac{\exp(a_{ijk}^h)}{\sum_{u=1}^n \exp(a_{iju}^h)}, \tag{21}$$

$$o_{ij}^h = \sum_{k=1}^n \alpha_{ijk}^h v_{ik}^h, \qquad \mathcal{A}(Z)_{ij} = W_O\big[(o_{ij}^1 \| \cdots \| o_{ij}^H) \odot \sigma(W_G U_{ij} + b_G)\big] + b_O. \tag{22}$$

Here $W_O, W_G \in \mathbb{R}^{d \times d}$. The gate acts on the concatenated head outputs *before* the output projection. The learned third-pair term b_{jk}^h makes the interaction involve (i, j), (i, k) and (j, k).

A block has independently parameterized starting-node and ending-node operators $\mathcal{A}_\mathrm{r}$ and $\mathcal{A}_\mathrm{c}$ of this form. Its exact update order is

$$Z_1 = \mathcal{M}\big[Z + \mathcal{A}_\mathrm{r}(Z)\big], \tag{23}$$

$$Z_2 = \mathcal{M}\big[Z_1 + (\mathcal{A}_\mathrm{c}(Z_1^\mathsf{T}))^\mathsf{T}\big], \tag{24}$$

$$\mathrm{Block}_\mathrm{PF}(Z) = \mathcal{S}\mathcal{M}\big[Z_2 + \mathcal{F}_\mathrm{ReLU}(Z_2)\big], \qquad \mathrm{ReLU}(u) = \max(u, 0). \tag{25}$$

The ending-node pass consumes the row-updated state before symmetrization. Expressed in the original axes, its query pair is (i, j), its key/value pair is (k, j), and its bias pair is (k, i), using that pass's normalization and parameters. Replacing it by a second starting-node pass on the initial symmetric state would therefore change the model.

All k, including $k = i$ and $k = j$, occur in the attention sums. Zeroing diagonal hidden states does not remove diagonal keys or values: normalization can give them nonzero features. There is no attention mask excluding them.

A.3 SYMMETRY GUARANTEES

For a permutation π of the vertices, define $(\pi \cdot Z)_{ij} = Z_{\pi^{-1}(i), \pi^{-1}(j)}$. The corresponding action on a graph permutes its unordered pairs and does not permute its edge categories.

Proposition (equivariance and undirected output). For the Pairformer, with dropout disabled and in exact arithmetic,

$$\ell_\theta(\pi \cdot X, t) = \pi \cdot \ell_\theta(X, t). \tag{26}$$

For every input, the output is symmetric and has zero diagonal.

Proof. Shared channelwise affine maps, layer normalization, activations and gates commute with vertex permutation. So do $\mathcal{M}$, $\mathcal{S}$ and vertex-axis transposition. Time conditioning is shared across all pairs. For attention, relabeling simultaneously permutes the query, key and third-pair indices. The score array is therefore permuted along all three vertex axes. Softmax commutes with permutation of its summation axis, and the value-weighted sum is preserved by the same change of index. This proves equivariance of both attention operators, including the transposed column pass. Residual addition and composition preserve the property, so induction over blocks and the readout establishes equation (26). Finally, equation (19) applies $\mathcal{S}\mathcal{M}$, giving the asserted symmetry and zero diagonal. □

Floating-point implementations satisfy the algebraic identities only up to numerical roundoff.

B CONSTRAINT EXPRESSIVITY OF TRIANGLE ATTENTION

We analyze the triangle-attention Pairformer specified in Appendix A.2, with categorical adjacency input, no vertex identifiers, and dropout disabled, restricted to deterministic predictions on a supplied graph. The results give a constructive representation result for the C_4 objective and an explicit obstruction for a Ramsey objective. Substructure counting and higher-order graph expressivity have an established literature (Chen et al., 2020; Maron et al., 2019); our contribution is the specialization to the actual normalized, third-pair-biased attention operator, including its diagonal convention.

B.1 RECOVERING COUNTS FROM NORMALIZED TRIANGLE ATTENTION

Let A be the adjacency matrix of a simple undirected graph and let $d_j = \sum_k A_{jk}$. Suppose a pair state exposes a scalar B_{ik}, with $B_{ii} = 0$, and distinguishes diagonal from off-diagonal slots. Consider two starting-node attention heads with zero query and key maps. Their value vectors are $(B_{ik}, \mathbf{1}_{\{i=k\}})$. One head has zero third-pair bias; the other has bias $(\log 2)A_{jk}$. Denote their outputs by (u^0_{ij}, a^0_{ij}) and (u^1_{ij}, a^1_{ij}), respectively. Directly from softmax,

$$u^0_{ij} = \frac{\sum_k B_{ik}}{n}, \qquad a^0_{ij} = \frac{1}{n}, \tag{27}$$

$$u^1_{ij} = \frac{\sum_k B_{ik} + \sum_k B_{ik}A_{jk}}{n + d_j}, \qquad a^1_{ij} = \frac{2^{A_{ij}}}{n + d_j}. \tag{28}$$

Consequently, the pointwise decoder

$$T(B, A)_{ij} = 2^{A_{ij}} \frac{u^1_{ij}}{a^1_{ij}} - \frac{u^0_{ij}}{a^0_{ij}} = \sum_k B_{ik}A_{jk} = (BA)_{ij} \tag{29}$$

recovers an unnormalized matrix contraction, and also $n = 1/a^0_{ij}$ and $d_j = 2^{A_{ij}}/a^1_{ij} - n$. These identities hold at every order with the same attention parameters: the diagonal value is an anchor exposing the normalizing denominator, and learned layer-normalization offsets can distinguish it from off-diagonal states. Masking diagonal keys out of attention would invalidate this construction.

Theorem (bounded-order representation of the C_4 move cost). For every finite $N \geq 2$, there are finite hidden and feedforward widths and a single parameter setting of a two-block triangle-attention Pairformer whose scalar logit readout, on every simple graph with $2 \leq n \leq N$, equals the number of simple length-three paths between the endpoints of each potential edge. The count excludes the queried edge itself. The same architecture family can represent the exact single-edge flip cost for $L_{C_4}(A) = (\binom{n}{2} + 1)N_{C_4}(A) - |E(A)|$. Parameters are shared across all orders in the stated range.

Proof of the graph identities. In the first block use $B = A$ in equation (29), obtaining $(A^2)_{ij}$ on off-diagonal slots. Retain A_{ij} and mask the diagonal, so the next block receives $B = A^2 - \mathrm{diag}(d)$. A second application yields

$$T(B, A)_{ij} = (A^3)_{ij} - d_i A_{ij}. \tag{30}$$

For $i \neq j$, the non-simple length-three walks from i to j repeat i at their third vertex or j at their second vertex; these occur only when $A_{ij} = 1$, with $d_i + d_j - 1$ of them, the subtraction correcting the doubly counted walk i, j, i, j. Thus

$$P_{ij} = (A^3)_{ij} - A_{ij}(d_i + d_j - 1) = T(B, A)_{ij} - A_{ij}(d_j - 1). \tag{31}$$

All quantities on the right are available from the heads and the retained edge category, and a simple path with these endpoints cannot use the queried edge. Hence P_{ij} counts exactly the four-cycles created by adding an absent edge or destroyed by removing a present edge, so

$$L_{C_4}(A \triangle \{i, j\}) - L_{C_4}(A) = (1 - 2A_{ij})\big[(\tbinom{n}{2} + 1)P_{ij} - 1\big], \tag{32}$$

where $\triangle$ denotes toggling that edge.

Realization by the specified blocks. The rational decoder above is not a ReLU layer, but for fixed N the input domain is finite. Encode each possible off-diagonal pair type by a one-hot vector at every block boundary (initially edge and nonedge; after the first block $(A_{ij}, (A^2)_{ij})$), keep the diagonal state zero, and ignore time by zeroing the time-conditioning output. All off-diagonal codes share a common mean and variance, so a layer-normalization scale and offset makes the normalized type codes together with the zero diagonal linearly independent; bias-free projections then expose A_{ij}, B_{ij} and the diagonal indicator exactly, two heads implement equation (28), a constant gate can be absorbed into the output projection, and the ending-node attention can be zeroed. Layer normalization is injective on the finite set of residual states, so a two-layer ReLU network interpolates the desired block-to-block map; symmetrization preserves the desired states and a linear readout after the final layer normalization returns the encoded scalar. All interpolation tables cover every $2 \le n \le N$ at once. □

Scope. The attention identities are independent of graph order, but the finite interpolation argument permits widths and parameters to depend on N. It does not establish exact counting at the experimental width of 128, learnability from the denoising objective, or generalization beyond N. The readout in the theorem is a logit, not a probability equal to a count; representing the clean objective's move statistic is not the same as representing the Bayes-optimal noisy denoiser.

Proposition (the third-pair bias gives a strict separation). If the third-pair bias is removed from every attention head, no network of the remaining form, at any width or depth, represents all C_4 edge-flip costs: it gives the same predictions by edge type on C_8 and $C_4 \sqcup C_4$, although deleting an edge has cost change $+1$ on the former and -28 on the latter.

Proof. On any k-regular graph of order n, initial pair states have just three types—diagonal, edge, and nonedge—and without the third-pair bias a row or column attention output depends only on its own type and the counts of each key type. All other operations preserve this property, so induction shows that all regular graphs with the same (n, k) receive identical predictions by pair type. Both witnesses have $n = 8, k = 2$. Deleting an edge of C_8 destroys no four-cycle, giving cost change $+1$; deleting an edge of $C_4 \sqcup C_4$ destroys one four-cycle, giving $-(\binom{8}{2} + 1) + 1 = -28$. The positive theorem with $N = 8$ shows the full architecture can distinguish these move costs. □

B.2 An explicit Ramsey obstruction at every depth

Theorem (identical Pairformer predictions, different Ramsey moves). Let G be the 4×4 rook graph and H the Shrikhande graph. For any fixed weights, width, depth, and time, the specified Pairformer assigns one common logit vector to every edge of either graph and another common logit vector to every nonedge of either graph. Nevertheless, deleting an edge has cost change -1 in G and 0 in H for $L_{4,6} = N_{K_4}(A) + N_{K_6}(\overline{A})$. No such Pairformer is an exact edge-flip-cost oracle for this Ramsey objective on all inputs, even if its width and depth are unrestricted.

Proof of identical predictions. Both graphs are strongly regular with parameters $(v, k, \lambda, \mu) = (16, 6, 2, 2)$, and a pair state is determined by whether it is a diagonal, edge, or nonedge, with the same three state vectors in both graphs; suppose this holds at some layer. For an off-diagonal query (i, j), write $a = A_{ij}$ and let c be its common-neighbor count (λ for an edge, μ for a nonedge). The vertices other than i, j split as

(A_{ik}, A_{jk})	$(1,1)$	$(1,0)$	$(0,1)$	$(0,0)$
count	c	$k-a-c$	$k-a-c$	$v-2-2k+2a+c$

with the two remaining terms having types (D, a) and (a, D). These counts depend only on a and the shared parameters, so every head's scores, softmax denominators and weighted value sums are identical for queries of the same type in either graph; the same holds for the ending-node pass, and normalization, gates, residuals, feedforward layers, masking and symmetrization preserve the property. Induction proves the assertion at arbitrary depth. This is the familiar strongly-regular-graph obstruction, here checked directly for the implementation. □

Proof of the different costs. The rook graph G on $\mathbb{Z}_4^2$ joins vertices sharing a coordinate: its eight rows and columns are exactly its eight four-cliques, each edge in exactly one, so deleting

an edge drops N_{K_4} by one. The Shrikhande graph H is the Cayley graph on $\mathbb{Z}_4^2$ with connection set $\{\pm(1,0), \pm(0,1), \pm(1,1)\}$; each neighborhood induces a six-cycle, so it has no four-clique. Both graphs have independence number at most four: for G from its four rows, for H from the Hoffman bound (degree six, least adjacency eigenvalue -2—its spectrum follows from $A^2 = 4I + 2J$ and regularity—giving $\alpha \leq 16 \cdot 2/(6+2) = 4$). Deleting one edge raises the independence number by at most one (it removes an endpoint from any newly independent set), so neither graph acquires an independent six-set. The four-clique count therefore drops by one in G and by zero in H, establishing the claimed cost changes. □

Implication and limits. Together, the results give an objective-specific distinction: the architecture family can represent all C_4 move costs on a bounded size range, the third-pair bias is necessary for this capability, and the full architecture has an exact obstruction for the $R(4,6)$ cost already at order 16. This motivates analyzing the two tasks separately, but does not prove that the obstruction causes the observed Ramsey transfer failures: random corruption and stochastic sampling can break the symmetries of these clean inputs, and the theorem is not an impossibility result for the full generative process.

Verification and prior work. We enumerate small graphs and check the attention identities against simple-path and cycle counts, and check both 16-vertex witnesses and their Ramsey flip costs against the implemented Pairformer. Numerical checks support, but do not replace, the proofs. The rook/Shrikhande witness and the limitations of pair representations are established; see Bodnar et al. (2021) and Zhou et al. (2024) for higher-order attention expressivity.

C HYPERPARAMETERS AND SEARCH DETAILS

This section collects every hyperparameter and search detail used in the paper's experiments. The generator architecture is fixed throughout; task-specific training budgets (fits per task, evaluation draws) follow the protocols in the sections referenced from the main text. The denoising objective and sampler are given in Section 3.1. A denoising target is a selected graph and need not be feasible, as can occur for Ramsey construction.

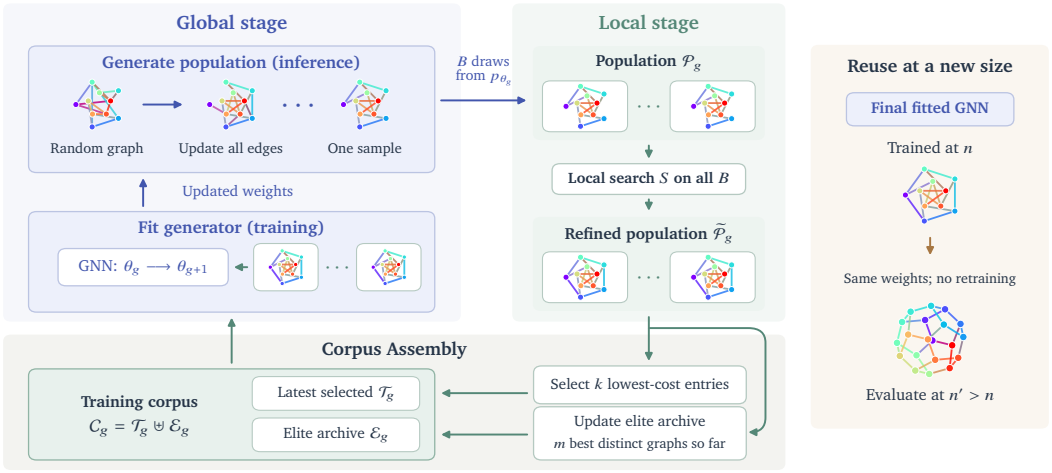


Figure 5: **Learn to generate, improve with search, and reuse at a new size.** Each generation samples a population $\mathcal{P}_g$ of B graphs. The GNN repeatedly updates every potential edge within each sample. Local search refines the whole population. Selected entries $\mathcal{T}_g$ and the updated elite archive $\mathcal{E}_g$ form the training corpus $\mathcal{C}_g$ for the next fit (Algorithm 1). Networks transfer to unseen orders.

Table 4: **Generator architecture (triangle-attention Pairformer).** Shared across all experiments and all graph sizes; the model has no vertex features, identifiers, positional encodings, or precomputed graph statistics (Mehrabian et al., 2024).

Parameter	Value
Attention blocks L	4
Feature width d	128
Attention heads H	8
Head width r	16
Feedforward width f	512
Time-embedding width w	64
Edge categories C	2
Dropout p	0
Trainable parameters	1,237,762

C4 outer loop. The order-33 source checkpoint in Table 9 uses the architecture of Section 3.2 with the pretraining settings of Table 5. C_4 refinement deletes an edge with the largest four-cycle participation, breaking ties randomly, until no four-cycle remains, then greedily extends the graph while preserving feasibility. Deletion reduces the number of four-cycles and terminates at zero, so every refined C_4 output is feasible by construction.

Ramsey refinement. Ramsey refinement uses a bounded tabu search over edge toggles: at most 128 steps, each considering 16 candidate edges with tabu tenure 7, returning the best visited graph.

Table 5: **Denoising pretraining (PatternBoost fit).** Corpus entries are sampled uniformly with multiplicity; selected and elite rows are concatenated without cross-component deduplication.

Parameter	Value
Objective	Categorical flow matching equation 3
Time t	Shared per graph, $t \sim \mathcal{U}[0, 1)$
Corruption	Per pair: retain with prob. t, else uniform equation 2
Optimizer	AdamW
Learning rate	2×10^{-4}
Batch size	256
Updates per fit	5,000
Proposals per generation	10,000
Corpus	Top 1% of refined proposals + 50 elite graphs
Augmentation	None (no construction injection, no relabeling)
Fits per task	12 (C_4, order-33); 30 (C_4, orders 20/25/30); 20 (Ramsey)

Table 6: **Reward fine-tuning.** Reward is the refined quality $R_n(S(Z_K))$ of equation 7, batch-standardized; updates are skipped when all rewards in a batch are equal.

Parameter	Value
Objective	Clipped policy gradient equation 8
Clipping ϵ	0.1
Learning rate	10^{-5}
Updates	2,000
Trajectories	256 fresh proposals per four updates
Batch / microbatch	256 / 32

Table 7: **Sampling, evaluation, and local search.** Sampling and refinement use separately seeded random streams.

Parameter	Value
Flow steps K (evaluation)	64, FP32
Sampling batch size	64
Draws per evaluation cell	10,000
Evaluation sampling hardware	A40 GPUs
C_4 refinement	Greedy: delete max-participation edge until feasible, then extend; batch 256
Ramsey refinement	Tabu: $\leq$128 steps, 16 candidate edges, tenure 7; batch 32

Reward fine-tuning details. Reward fine-tuning uses the clipped policy loss in equation (8); each transition probability is the product of the edge probabilities in equation (5), so transition log probabilities sum over potential edges. Rewards are supplied by local search without being differentiated through or replacing any sampled action. Above-average rewards increase the likelihood of sampled transitions, while clipping limits changes in their ratios; if all rewards in a batch are equal, the update is skipped rather than dividing by zero when standardizing.

Adaptation grid protocol. The 2,000-update full-size grid uses the sources and evaluation protocol in Appendix G. Fixed-corpus controls retain one selected corpus throughout adaptation; reward adaptation uses newly sampled trajectories throughout. Continuing the outer search loop would instead construct and fit a new corpus at every generation.

D CONSTRUCTION COMPARISON: PROTOCOL AND EXTENDED RESULTS

Evaluation scope and controls. The adaptation grid starts from one selected checkpoint at each order 36–42. Each is independently adapted at every order 33–42 for 2,000 updates and evaluated at all ten orders with 10,000 draws under the same sampler and refiner as its unchanged source. Reward fine-tuning uses fresh proposals; the supervised control freezes one selected corpus at the fine-tuning size and continues denoising. Continuing PatternBoost would instead refresh the corpus each generation. The grid reuses models across target sizes. The order-25 zero-shot results pool three training seeds and three sampling seeds, totaling 90,000 draws per evaluation order. The separate order-33 zero-shot checkpoint is selected at its training order from three seeds and evaluated with 10,000 draws per target order.

Model selection and training times. At orders 36–42, the main-table Direct rows use H200 pretraining runs: three seeds per order, 30 rounds of 5,000 updates, batch size 512, learning rate 5×10^{-4} and 20,000 proposals per round. At each of these target orders we select only among its three same-order sources, using their fresh 10,000-draw matrix evaluations. The order-33 Direct row uses the best checkpoint from a separate order-33 training run. Fine-tuned results select the best checkpoint at each evaluation order from the full source-order/adaptation-order matrix, including fixed-corpus fitting and RL. Each checkpoint receives 10,000 evaluation draws at each target order. Selection maximizes refined best edges, breaking ties by refined mean and raw best, with any remaining ties resolved deterministically. Final-population means and validity describe that same selected checkpoint; they are not separate maxima over columns or seed averages. Unadapted matrix checkpoints inform source selection but do not compete in the fine-tuned column.

Order-25 RL rows. The RL ($n_0 = 25$) rows select among ten adaptations of the selected order-25 checkpoint, using the same refined-best/mean/raw-best rule. This source is selected at order 25 from three training seeds by final refined best, then optimum frequency, then mean. It completes 30 fits of 5,000 updates. The ten adaptation orders are 33–42; the 140-candidate matrix selection is unaffected. Every selected row uses one 10,000-draw evaluation, not the pooled maxima of several models. Full grid: Appendix G.3.

Training time. Training and FT report elapsed pretraining and adaptation times; Total is their sum. Order-25 pretraining uses a measured H200 timing replication (5.44 hours). Nine selected adaptations use H200 timing replications: $36 \to 33, 35, 37$, $38 \to 33$, $37 \to 36$, and $25 \to 33, 34, 37, 38$. These retain the source checkpoints, seeds and adaptation settings; quality and local-search counts come from the original runs. The $36 \to 33$, $38 \to 33$ and $25 \to 38$ replications use H200 NVL; the others report H200. Unreplicated rows retain their original training times. Thus the reported totals combine measured stages, including timing replications where available. Runtimes include sampling, refinement, diagnostics, setup and uploads within training, including final populations; separate matrix evaluations are excluded. The pretrained source is reusable across adaptations and target orders, although each table row includes its full pretraining cost. These selected-pipeline comparisons exclude model selection and use different training protocols; they do not establish efficiency at matched compute.

Local-search counts. One local search (LS) means refining one proposed graph, not one edge edit, objective evaluation or batched function call. Counts use the selected training run and, for adapted models, its source pretraining run, fine-tuning run and 10,000-draw target evaluation. Main-table Direct counts one population per fit, its training-run final population and the separate 10,000-draw matrix evaluation. The historical flow baseline counts the training populations and its final population. Fine-tuning counts the initial corpus/probe population, step-zero and milestone diagnostic populations, and all logged reward rollouts. Counts are derived from the completed-run configurations and checked rollout counters. Source-training searches are reusable across targets; each row includes them once. These are full selected-run counts, not counts until first discovery or totals over other model-selection candidates. PB counts retain the definition reported in its Table 3 and are not multiplied by the number of models. Search implementations and graph orders differ, so these counts are workload context, not matched compute.

Table 8: **Quality and training costs at orders 33–42.** Training time refers to the selected model, including source pretraining and fine-tuning (FT) for adapted models; Total is their sum. Dashes in FT indicate no fine-tuning or an unreported time. Direct and RL use 10,000-draw final evaluations. Means and validity describe the final evaluation populations. Raw means average feasible proposals; refined means average all proposals, which are feasible after search. Refined Best is the headline endpoint. PB Best is source-reported; its population statistics and training times are unreported. LS counts local searches in millions for the selected model, including pretraining, fine-tuning and its target evaluation; PB counts are source-reported. Bold marks maxima in raw and refined columns and minima in Total within each order, including all ties at the displayed precision. Times are elapsed training-run hours; the accompanying protocol defines their scope and the selection rule.

			Raw			**Refined**		Training time (h)			
n	Method	Train → FT	Valid (%)	Mean	Best	Mean	Best	Train	FT	Total	LS (M)
33	PB	33	—	—	—	—	**96**	—	—	—	116.500
	Direct (ours)	33	4.93	94.88	**96**	91.61	**96**	9.30	—	9.30	0.620
	RL (ours)	36 → 33	**11.59**	**95.81**	**96**	**94.02**	**96**	8.81	0.89	9.70	0.778
	RL ($n_0 = 25$)	25 → 33	0.00	—	—	93.64	**96**	5.44	0.81	**6.26**	0.778
34	PB	34	—	—	—	—	95	—	—	—	39.900
	RL (ours)	36 → 34	**9.13**	**99.91**	**102**	**97.62**	**102**	8.81	2.49	11.30	0.778
	RL ($n_0 = 25$)	25 → 34	0.00	—	—	97.24	**102**	5.44	0.83	**6.27**	0.778
35	PB	35	—	—	—	—	100	—	—	—	121.300
	RL (ours)	36 → 35	**9.92**	103.46	**106**	**101.22**	**106**	8.81	0.85	9.67	0.778
	RL ($n_0 = 25$)	25 → 34	0.02	**105.00**	105	100.80	**106**	5.44	0.83	**6.27**	0.778
36	PB	36	—	—	—	—	102	—	—	—	38.400
	Direct (ours)	36	**12.49**	103.07	108	102.08	108	8.81	—	8.81	0.630
	RL (ours)	36 → 35	6.29	**107.96**	**110**	**105.06**	**110**	8.81	0.85	9.67	0.778
	RL ($n_0 = 25$)	25 → 34	0.00	—	—	104.73	**110**	5.44	0.83	**6.27**	0.778
37	PB	37	—	—	—	—	106	—	—	—	29.800
	Direct (ours)	37	**22.81**	106.14	110	105.62	110	10.70	—	10.70	0.630
	RL (ours)	36 → 35	2.25	**111.78**	**113**	**108.49**	**113**	8.81	0.85	9.67	0.778
	RL ($n_0 = 25$)	25 → 34	0.00	—	—	107.79	**113**	5.44	0.83	**6.27**	0.778
38	PB	38	—	—	—	—	110	—	—	—	77.100
	Direct (ours)	38	**26.57**	109.29	113	108.82	114	10.72	—	10.72	0.630
	RL (ours)	36 → 36	0.59	**115.97**	**117**	**112.21**	**117**	8.81	2.76	11.57	0.778
	RL ($n_0 = 25$)	25 → 34	0.00	—	—	111.27	**117**	5.44	0.83	**6.27**	0.778
39	PB	39	—	—	—	—	113	—	—	—	11.700
	Direct (ours)	39	**24.00**	113.06	116	112.61	117	11.63	—	11.63	0.630
	RL (ours)	36 → 37	0.40	**118.90**	**120**	**115.88**	**121**	8.81	0.94	9.75	0.778
	RL ($n_0 = 25$)	25 → 37	0.00	—	—	114.71	**121**	5.44	0.95	**6.40**	0.778
40	PB	40	—	—	—	—	117	—	—	—	11.200
	Direct (ours)	40	**31.64**	115.92	121	115.76	121	11.34	—	11.34	0.630
	RL (ours)	38 → 33	0.04	**124.25**	**127**	**118.64**	**127**	10.65	0.89	11.54	0.778
	RL ($n_0 = 25$)	25 → 38	0.00	—	—	118.58	125	5.44	1.07	**6.52**	0.778
41	PB	41	—	—	—	—	102	—	—	—	8.600
	Direct (ours)	41	**27.84**	**119.71**	**124**	119.59	125	12.71	—	12.71	0.630
	RL (ours)	36 → 34	0.00	—	—	**122.77**	**132**	8.81	2.49	11.30	0.778
	RL ($n_0 = 25$)	25 → 37	0.00	—	—	122.30	130	5.44	0.95	**6.40**	0.778
42	PB	42	—	—	—	—	124	—	—	—	6.800
	Direct (ours)	42	**22.95**	123.52	128	123.34	128	12.58	—	12.58	0.630
	RL (ours)	37 → 36	0.02	**134.50**	**137**	**126.98**	**137**	10.70	0.89	11.59	0.778
	RL ($n_0 = 25$)	25 → 37	0.00	—	—	126.19	134	5.44	0.95	**6.40**	0.778

E C4 ZERO-SHOT TRANSFER FROM A SINGLE CHECKPOINT

E.1 CHECKPOINT TRAINED AT $n = 33$

Protocol. We evaluate unchanged $n = 33$ weights at orders $n = 33, \ldots, 40$. Among three completed source-order training seeds, we select the final checkpoint by refined maximum feasible edges, then zero-gap frequency, then mean feasible edges, using only $n = 33$ results. All three reached 96 edges. The architecture and all training, sampling, and refinement settings are those of Appendix C; only edge indexing and diagonal masks change with order, and all learned tensors are loaded strictly and unchanged, with no retraining or target-order tuning.

We retain both endpoints, defining feasibility by zero four-cycles. Mean edges are conditional on feasibility; gap is the reference maximum minus the best feasible sample. The reference table gives $96, 102, 106, 110, 113, 117, 122, 127$ edges at orders 33–40 (Afzaly & McKay, 2026).

Table 9: **One generator, larger graphs.** One fixed model trained on 33 vertices, evaluated with 10,000 draws per size. Raw means and maxima are over feasible proposals only. Bold maxima attain the proved optimum.

		Raw generator			After refinement	
n	$B(n)$	Valid (%)	Mean	Best	Mean	Best
33	96	19.58	92.03	**96**	91.10	**96**
34	102	12.59	95.79	**102**	94.71	**102**
35	106	7.80	99.60	**106**	98.46	**106**
36	110	3.56	103.71	**110**	102.10	**110**
37	113	1.58	107.62	**113**	105.77	**113**
38	117	0.73	111.42	115	109.46	116
39	122	0.08	115.88	118	113.12	119
40	127	0.02	119.00	120	116.81	123

Results. Table 9 reports the raw and refined endpoints. The unchanged generator also reaches the reference maxima at $n = 34, 35, 36, 37$—102, 106, 110 and 113 edges—in respectively 6, 1, 1 and 1 of the 10,000 raw draws. Refinement yields respectively 12, 2, 2 and 1 optimal samples at these orders. At $n = 38, 39, 40$, the best refined gaps are 1, 3 and 4 edges. Raw feasibility falls to 0.02% at $n = 40$, while every refined sample is feasible. The raw feasible means at $n = 39, 40$ use only eight and two samples, respectively. Independent neighbor-set cycle recounts verify the best raw/refined witnesses at every order.

Scope. These results use one selected checkpoint and one sampling seed per order. Raw feasibility falls with graph order, so larger-order performance increasingly depends on refinement.

Figures 6 and 7 show the full raw and refined distributions, including infeasible proposals, for the same populations as Table 9.

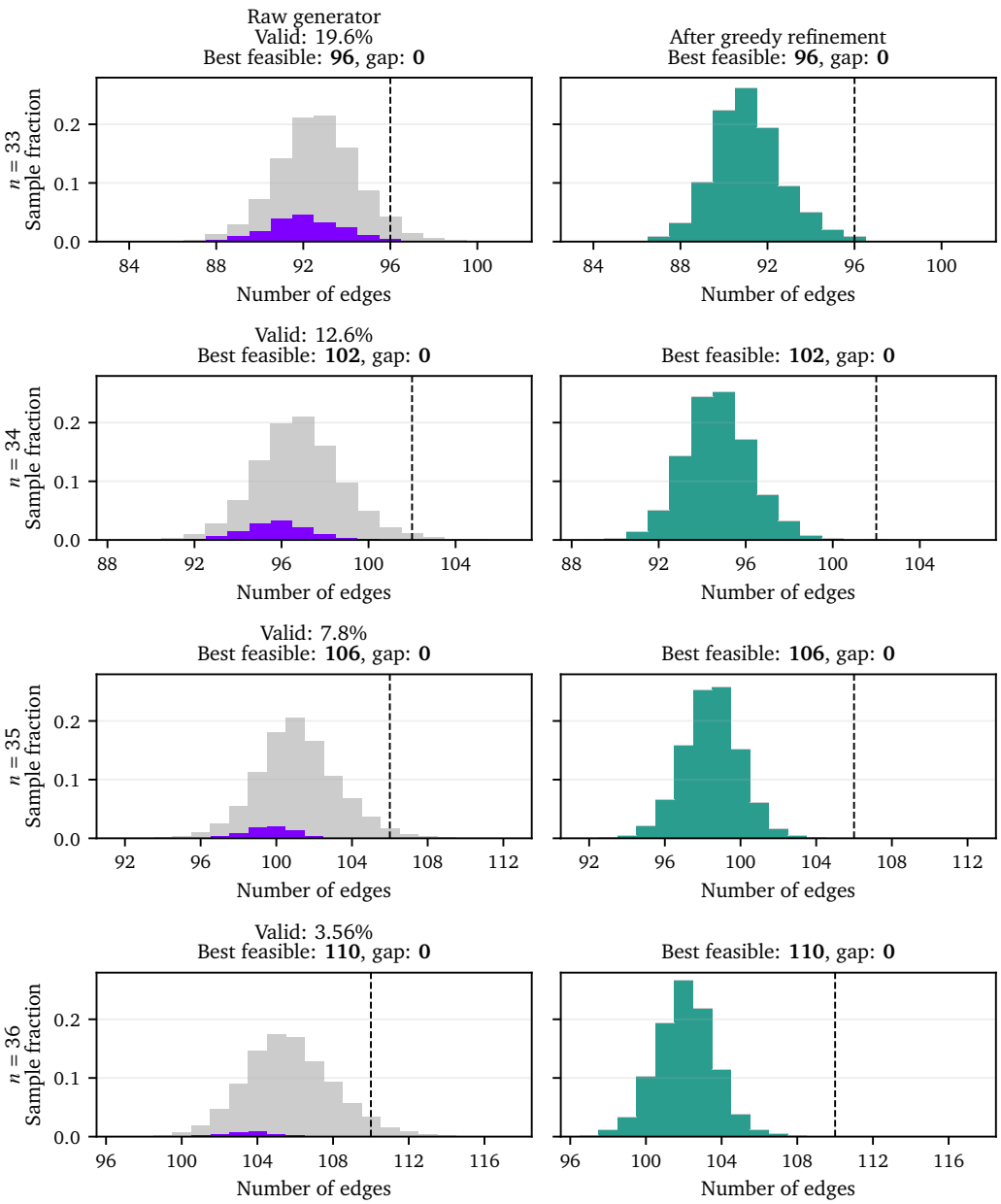


Figure 6: Distributions from unchanged $n = 33$ weights at orders 33–36. Left: raw generator proposals; right: the same proposals after greedy repair and extension. Colored bars denote C4-free graphs, gray bars denote invalid graphs, and dashed lines mark the reference edge maxima. Each row contains 10,000 samples; all bars use fractions of the whole population.

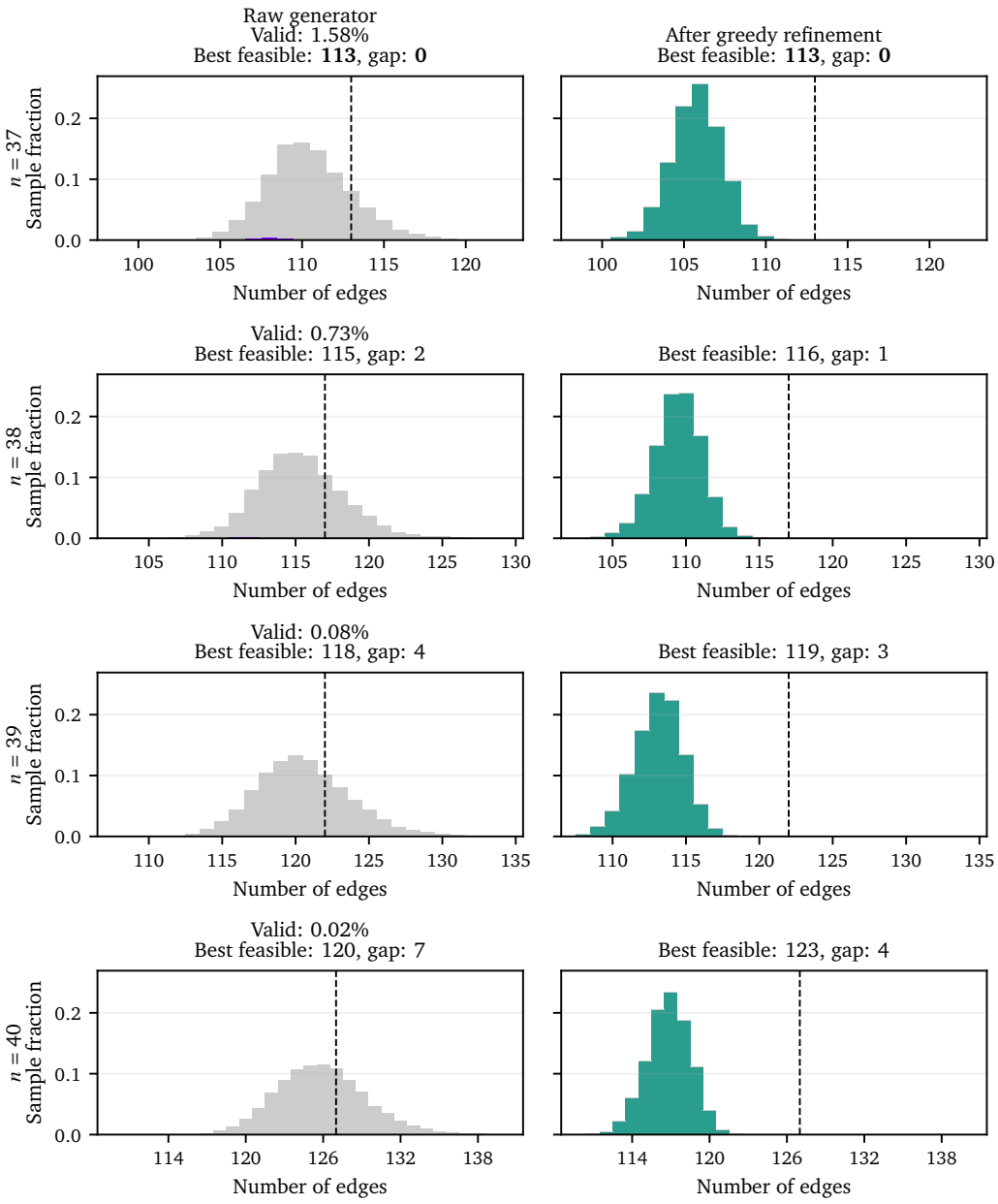


Figure 7: The same fixed-checkpoint evaluation at orders 37–40. Colors, normalization and reference lines follow Figure 6. No weights or sampling settings are tuned at these larger orders.

F C4 ZERO-SHOT TRANSFER ACROSS TRAINING SIZES, THE FRONTIER, AND SEARCH CONTROLS

These evaluations keep the learned weights fixed. We use 10,000 graphs per evaluation, 64 FP32 flow steps, sampling batch 64 and C4 greedy refinement with batch 256, on A40 GPUs. Raw means condition on feasibility; refined populations are entirely feasible.

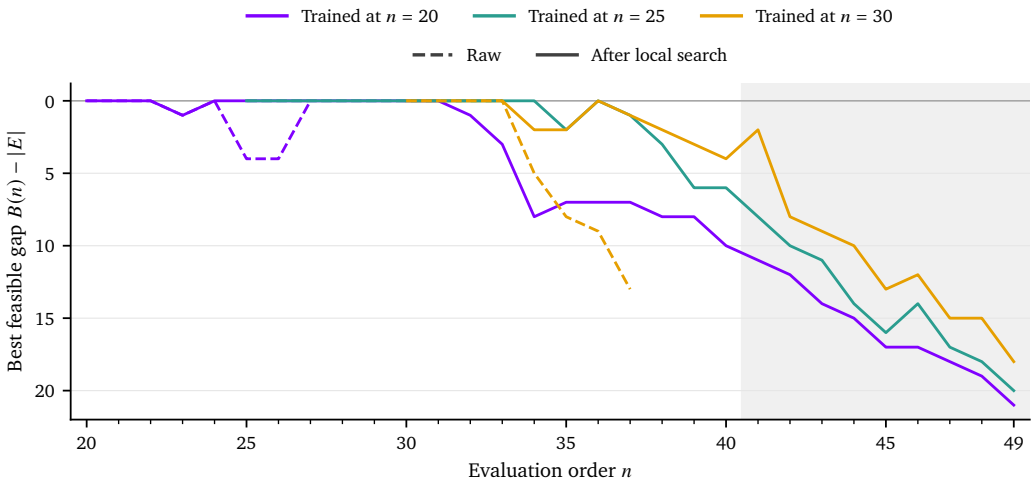


Figure 8: **Transfer from smaller training graphs.** Best feasible edge gaps from Tables 10 to 12, before (dashed) and after (solid) local search. Each point takes the best over 90,000 draws: three training seeds, each with three sampling seeds of 10,000 draws. All checkpoints remain fixed, with no fine-tuning. Zero at the top is best; missing raw points indicate no feasible sample. Purple, teal and orange denote training at orders 20, 25 and 30, respectively. Each curve starts at its training-size reference; larger orders are unseen. Shading marks unproved best-known references at orders 41–49. These curves show pooled maxima; the tables also report feasibility, means and training-seed uncertainty.

F.1 TRANSFER FROM ORDER 20

Table 10 evaluates three independently trained order-20 models at orders 20–49, giving 270 cells: nine same-order references and 261 upward-transfer cells. Each target order pools 90,000 draws (three training seeds, each with three sampling seeds). Refined means average the three training-seed means after pooling sampling seeds; SD is their sample standard deviation; Best is the maximum over all nine evaluations. Raw means at orders 25, 26 and 27 use only 17, three and one feasible draws, respectively.

Table 10: **Transfer from 20 vertices without fine-tuning.** Each row contains 90,000 draws. All refined samples are feasible. Bold best counts attain $B(n)$; * denotes an unproved best-known reference. Order 20 is the training-size reference; all other orders are unseen.

		Raw			Refined	
n	$B(n)$	Valid (%)	Mean	Best	Mean $\pm$ SD	Best
20	46	97.306	45.99	**46**	45.96 ± 0.01	**46**
21	50	76.122	49.38	**50**	49.38 ± 0.12	**50**
22	52	25.204	49.80	**52**	51.10 ± 0.10	**52**
23	56	4.069	52.19	55	53.30 ± 0.14	55
24	59	0.803	55.57	**59**	56.20 ± 0.11	**59**
25	63	0.019	57.47	59	58.86 ± 0.06	**63**
26	67	0.003	61.00	63	61.97 ± 0.09	**67**
27	71	0.001	71.00	**71**	64.98 ± 0.12	**71**
28	76	0.000	—	—	68.25 ± 0.14	**76**
29	80	0.000	—	—	71.39 ± 0.15	**80**
30	85	0.000	—	—	74.67 ± 0.18	**85**
31	90	0.000	—	—	77.93 ± 0.19	**90**
32	92	0.000	—	—	81.28 ± 0.21	91
33	96	0.000	—	—	84.67 ± 0.20	93
34	102	0.000	—	—	88.14 ± 0.18	94
35	106	0.000	—	—	91.64 ± 0.18	99
36	110	0.000	—	—	95.22 ± 0.17	103
37	113	0.000	—	—	98.83 ± 0.17	106
38	117	0.000	—	—	102.52 ± 0.15	109
39	122	0.000	—	—	106.25 ± 0.14	114
40	127	0.000	—	—	110.01 ± 0.15	117
41	132*	0.000	—	—	113.82 ± 0.13	121
42	137*	0.000	—	—	117.66 ± 0.13	125
43	142*	0.000	—	—	121.53 ± 0.13	128
44	148*	0.000	—	—	125.47 ± 0.12	133
45	154*	0.000	—	—	129.42 ± 0.11	137
46	157*	0.000	—	—	133.40 ± 0.11	140
47	163*	0.000	—	—	137.41 ± 0.10	145
48	168*	0.000	—	—	141.47 ± 0.11	149
49	174*	0.000	—	—	145.56 ± 0.10	153

F.2 TRANSFER FROM ORDER 25

Table 11 applies the same protocol and aggregation to three independently trained order-25 models at orders 25–49 (225 cells). Raw means at orders 32 and 33 use only 60 and nine feasible draws, respectively.

Table 11: **Transfer from 25 vertices without fine-tuning.** Each row contains 90,000 draws. All refined samples are feasible. Bold best counts attain $B(n)$; * denotes an unproved best-known reference. Order 25 is the training-size reference; all other orders are unseen.

		Raw			Refined	
n	$B(n)$	Valid (%)	Mean	Best	Mean $\pm$ SD	Best
25	63	38.138	62.74	**63**	60.15 ± 3.45	**63**
26	67	27.540	66.54	**67**	63.43 ± 3.57	**67**
27	71	18.823	70.49	**71**	66.72 ± 3.67	**71**
28	76	12.853	74.95	**76**	70.08 ± 3.78	**76**
29	80	7.118	79.32	**80**	73.41 ± 3.82	**80**
30	85	3.008	83.93	**85**	76.62 ± 3.71	**85**
31	90	0.491	86.83	**90**	79.88 ± 3.61	**90**
32	92	0.067	88.43	**92**	83.16 ± 3.48	**92**
33	96	0.010	91.78	**96**	86.54 ± 3.40	**96**
34	102	0.000	—	—	89.93 ± 3.28	**102**
35	106	0.000	—	—	93.36 ± 3.17	104
36	110	0.000	—	—	96.83 ± 3.06	**110**
37	113	0.000	—	—	100.36 ± 2.95	112
38	117	0.000	—	—	103.89 ± 2.83	114
39	122	0.000	—	—	107.48 ± 2.70	116
40	127	0.000	—	—	111.11 ± 2.56	121
41	132*	0.000	—	—	114.77 ± 2.44	124
42	137*	0.000	—	—	118.47 ± 2.30	127
43	142*	0.000	—	—	122.22 ± 2.20	131
44	148*	0.000	—	—	126.01 ± 2.08	134
45	154*	0.000	—	—	129.85 ± 1.97	138
46	157*	0.000	—	—	133.72 ± 1.88	143
47	163*	0.000	—	—	137.65 ± 1.80	146
48	168*	0.000	—	—	141.61 ± 1.72	150
49	174*	0.000	—	—	145.63 ± 1.65	154

F.3 TRANSFER FROM ORDER 30

Table 12 applies the same protocol and aggregation to three independently trained order-30 models at orders 30–49 (180 cells). Raw means at orders 35, 36 and 37 use only 58, 13 and five feasible draws, respectively.

Table 12: **Transfer from 30 vertices without fine-tuning.** Each row contains 90,000 draws. All refined samples are feasible. Bold best counts attain $B(n)$; * denotes an unproved best-known reference. Order 30 is the training-size reference; all other orders are unseen.

		Raw			Refined	
n	$B(n)$	Valid (%)	Mean	Best	Mean $\pm$ SD	Best
30	85	15.867	84.07	**85**	75.91 ± 5.48	**85**
31	90	11.589	89.09	**90**	79.45 ± 5.74	**90**
32	92	2.477	88.50	**92**	82.50 ± 5.15	**92**
33	96	1.713	91.14	**96**	85.70 ± 4.71	**96**
34	102	0.610	93.70	97	89.08 ± 4.48	100
35	106	0.064	94.48	98	92.46 ± 4.23	104
36	110	0.014	96.15	101	95.98 ± 4.13	**110**
37	113	0.006	96.20	100	99.51 ± 3.99	112
38	117	0.000	—	—	103.16 ± 3.96	115
39	122	0.000	—	—	106.80 ± 3.88	119
40	127	0.000	—	—	110.53 ± 3.85	123
41	132*	0.000	—	—	114.24 ± 3.75	130
42	137*	0.000	—	—	117.99 ± 3.65	129
43	142*	0.000	—	—	121.78 ± 3.54	133
44	148*	0.000	—	—	125.57 ± 3.40	138
45	154*	0.000	—	—	129.41 ± 3.26	141
46	157*	0.000	—	—	133.27 ± 3.13	145
47	163*	0.000	—	—	137.18 ± 2.98	148
48	168*	0.000	—	—	141.13 ± 2.85	153
49	174*	0.000	—	—	145.10 ± 2.71	156

F.4 Frozen-checkpoint evaluation at orders 43–49

Four checkpoints are fixed before evaluating the new orders: the selected order 33 generator; the order 37 source with the best unadapted refined mean at 42; the source 36 model fine-tuned at 41 with the best adapted refined mean at 42; and the source 37 model fine-tuned at 36 that produced the raw 137-edge construction at 42. Thus selection uses only evaluations at 42 or below. Each checkpoint is evaluated at all seven orders with three sampling seeds (120,000 proposals per target order across the panel); none receives training or fine-tuning at 43–49.

The best refined edge counts at 43–49 are 139, 142, 146, 150, 154, 158 and 164. At 49 the four checkpoints reach 158, 159, 164 and 162, with no feasible raw sample among the 120,000 proposals. PatternBoost's published Table 3 reports 128, 132, 136, 140, 144, 147 and 152 edges at 43–49 under its own search budgets (Charton et al., 2024). This is a comparison of attained constructions, not equal compute or discovery time.

Table 13: **Four frozen generators at unseen larger orders.** Each row pools three 10,000-draw evaluations. A single training order denotes an unadapted model; an arrow gives pretraining and fine-tuning orders. Best is the largest feasible edge count among the 30,000 draws, and raw means condition on feasibility. A dash denotes no feasible raw sample.

n	Train $\rightarrow$ FT	Raw valid (%)	Raw mean	Raw best	Ref. mean	Ref. best
43	33	0.000	—	—	127.84	134
43	37	0.937	131.04	134	129.01	135
43	36 $\rightarrow$ 41	0.043	135.08	136	131.90	139
43	37 $\rightarrow$ 36	0.000	—	—	130.74	138
44	33	0.000	—	—	131.53	139
44	37	0.323	135.37	138	132.90	139
44	36 $\rightarrow$ 41	0.003	139.00	139	136.04	141
44	37 $\rightarrow$ 36	0.000	—	—	134.60	142
45	33	0.000	—	—	135.19	143
45	37	0.117	139.77	142	136.77	143
45	36 $\rightarrow$ 41	0.003	146.00	146	140.12	146
45	37 $\rightarrow$ 36	0.000	—	—	138.36	145
46	33	0.000	—	—	138.87	146
46	37	0.027	144.62	147	140.61	147
46	36 $\rightarrow$ 41	0.000	—	—	144.29	150
46	37 $\rightarrow$ 36	0.000	—	—	142.17	150
47	33	0.000	—	—	142.55	151
47	37	0.007	149.50	150	144.40	151
47	36 $\rightarrow$ 41	0.000	—	—	148.40	154
47	37 $\rightarrow$ 36	0.000	—	—	145.90	154
48	33	0.000	—	—	146.28	155
48	37	0.000	—	—	148.19	156
48	36 $\rightarrow$ 41	0.000	—	—	152.57	158
48	37 $\rightarrow$ 36	0.000	—	—	149.61	158
49	33	0.000	—	—	150.00	158
49	37	0.000	—	—	151.96	159
49	36 $\rightarrow$ 41	0.000	—	—	156.70	164
49	37 $\rightarrow$ 36	0.000	—	—	153.32	162

F.5 SEARCH CONTROLS WITHOUT TARGET-ORDER GENERATION

We compare proposals from the selected order 33 generator with two alternative ways to initialize C4 greedy refinement at orders 34–49, with five seeds per control and order (160 runs). Each run refines up to one million proposals, recording cumulative endpoints at 10,000, 100,000 and 1,000,000. The density control draws an edge count from the learned generator's archived raw target-order population, then chooses that many edges uniformly. The extension control starts from the generator's best archived raw order 33 graph, adds uniformly selected edges incident to new vertices to reach the sampled target count, and randomly relabels the graph. Both controls use the learned model's target-order edge-count distribution; extension additionally uses a learned source graph. The comparison tests whether learned target-order structure improves refined results beyond those inputs. Because both controls use information from the learned generator, they are not model-free pipelines. The carried-over edge counts are drawn from the generator's own raw proposals, whose density is favorable for this objective; the comparison therefore biases the controls in their favor, so exceeding them is a conservative statement about the value of learned structure.

The controls use the same C4 greedy refinement as learned proposals, with processing batch 256. At 49, the order 33 generator reaches 158 edges in 30,000 target samples, compared with 151 for density control and 156 for extension, each among five million refined proposals. These budgets are not matched; the controls' refined means are nearly identical at their 10,000- and 1,000,000-proposal endpoints, so the comparisons below hold at matched per-seed budgets. The unchanged generator's refined means exceed the density control at every order and the extension control from order 38 onward. At orders 34–37 the extension control's refined means remain higher (for example 98.2 versus 94.7 at order 34), reflecting its initialization from the generator's own optimal order-33 graph; there the generator's refined best still matches or exceeds the extension control's best, reaching 102, 106, 110 and 113 edges at orders 34–37 (Table 14).

Table 14: **Alternative search starts across orders.** Each row pools five million proposals from five seeds. Raw means condition on feasibility; all refined samples are feasible. Best denotes the largest feasible edge count over that arm's five runs.

n	Control	Raw valid (%)	Raw mean	Raw best	Ref. mean	Ref. best
34	Density	0.000	—	—	84.70	94
34	Extension	67.438	96.36	100	98.18	102
35	Density	0.000	—	—	88.20	96
35	Extension	8.473	97.61	102	100.55	106
36	Density	0.000	—	—	91.74	100
36	Extension	0.255	99.19	104	103.27	109
37	Density	0.000	—	—	95.32	105
37	Extension	0.003	101.72	105	106.16	113
38	Density	0.000	—	—	98.94	108
38	Extension	0.000	—	—	109.16	117
39	Density	0.000	—	—	102.60	111
39	Extension	0.000	—	—	112.23	120
40	Density	0.000	—	—	106.29	115
40	Extension	0.000	—	—	115.37	124
41	Density	0.000	—	—	110.03	119
41	Extension	0.000	—	—	118.58	127
42	Density	0.000	—	—	113.80	123
42	Extension	0.000	—	—	121.86	131
43	Density	0.000	—	—	117.61	126
43	Extension	0.000	—	—	125.21	134
44	Density	0.000	—	—	121.46	130
44	Extension	0.000	—	—	128.61	138
45	Density	0.000	—	—	125.34	134
45	Extension	0.000	—	—	132.08	141
46	Density	0.000	—	—	129.26	139
46	Extension	0.000	—	—	135.62	145
47	Density	0.000	—	—	133.21	143
47	Extension	0.000	—	—	139.21	149
48	Density	0.000	—	—	137.20	147
48	Extension	0.000	—	—	142.85	153
49	Density	0.000	—	—	141.23	151
49	Extension	0.000	—	—	146.55	156

G C4 REWARD ADAPTATION

Models and pairing. Three triangle-Pairformer checkpoints were trained at each source order 36–42. One checkpoint per order was selected for adaptation. Each selected source initializes ten reward-policy fits and ten fixed-corpus fits, one at every fine-tuning order 33–42. Each adaptation has 2,000 updates, learning rate 10^{-5}, and one training seed per setting. The fixed control fits an unchanged initial corpus; reward training scores fresh trajectories after C4 repair. An adapted model is always paired with the *same* initial checkpoint evaluated at the *same* target order. Table 15 summarizes each selected source.

Evaluation. All 21 pretrained checkpoints and 140 adapted checkpoints were evaluated at all ten orders 33–42. This gives 210 unadapted and 700 evaluations per adaptation arm, 1,610 cells in total. Each cell has 10,000 fresh samples, 64 FP32 flow steps, and sampling batch size 64. The same sampled graphs are repaired and greedily extended with C4 search batch size 256. Sampling and refinement use separately seeded random streams, held fixed across settings. Every refined graph is feasible. We retain raw valid fraction, raw and refined maximum feasible edges, and raw and refined mean feasible edges. Means are conditional on feasibility; if a raw population has no feasible graph, its raw maximum and conditional mean are undefined. $B(n)$ is exact through $n = 40$; $n = 41, 42$ use best-known constructions (Afzaly & McKay, 2026).

G.1 CHECKPOINTS IN THE MAIN RESULTS FIGURE

Figure 1 fixes one checkpoint per curve. The left panel uses the selected order-33 checkpoint and its archived 10,000-draw populations at orders 33–40 (Appendix E). PatternBoost values are its published Table 3 results at the respective training orders (Charton et al., 2024). Those search budgets differ from our fixed-checkpoint evaluations. The right panel pairs an order-37 source with its reward adaptation at order 36, using the same initial checkpoint and evaluation protocol at every order 37–42. Order 37 provides the pretraining-size reference. This adaptation is the 137-edge construction setting, selected using order-42 results from the full grid. It is held fixed across evaluation orders. The panel reports an individual model pair, while Figure 9 summarizes gains across all adaptation settings. Both neural panels use 10,000 graphs per order, with the same proposals evaluated before and after refinement. Population statistics and best feasible witnesses are independently verified. The missing raw RL point at 41 means that this population contains no feasible graph. The curve is not joined across it.

Reward adaptation across source orders. Table 15 averages each source's 100 paired fine-tuning/evaluation comparisons. Reward improves the refined mean in 699/700 cells relative to its exact initial checkpoint and 700/700 relative to the matched fixed-corpus fit. The one decrease from the initial model is for source $n = 37$, fine-tuning $n = 33$, evaluation $n = 42$: -0.398 edge. The raw valid fraction falls on average for every source order. These are observations across 70 adaptation settings with one training seed each; 100 evaluations of a source's ten adapted models are not 100 independent training seeds. No seed-based uncertainty interval is inferred from the 700 paired evaluation cells.

How reward changes proposals. Figure 10 fixes an increasing-size pipeline: pretrain at 36, adapt at 37, and evaluate at 38. Reward-adapted proposals more often violate the constraint, yet their refined distribution shifts toward more edges. Raw feasibility alone would miss this gain.

Order-42 construction. The generator pretrained at order 37 is reward-adapted at order 36 for 2,000 updates. Both training stages use no injected constructions. At evaluation order 42, the raw population has two feasible graphs among 10,000 draws, with 132 and 137 edges. The refined population is entirely feasible, has mean edge count 126.976, and contains one 137-edge graph. Both endpoints therefore match the best-known reference value 137, whose optimality remains unproved.

Results by evaluation order. Table 16 and figure 11 summarize refined gains and raw feasibility across evaluation orders.

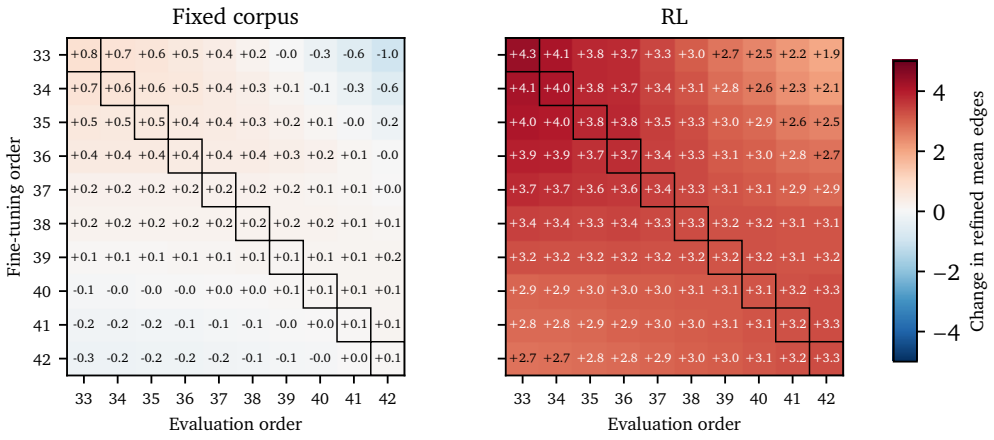


Figure 9: **Reward gains averaged over seven pretraining orders.** Mean refined-edge gains over each unchanged source, averaged over pretraining orders 36–42. Outlined cells evaluate at the fine-tuning order. Each cell combines seven 10,000-draw evaluations, reusing 70 adapted models per method with one adaptation seed each.

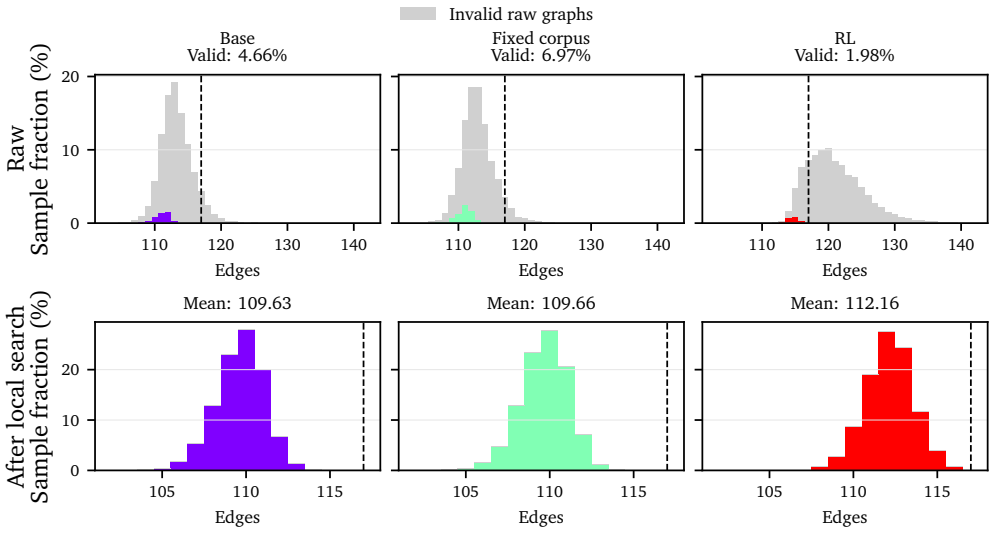


Figure 10: **Fine-tuning learns what local search can use.** Pretrain on 36 vertices, adapt on 37, evaluate on 38. Rows show the same 10,000 samples before and after search. Colored bars are feasible. Gray bars violate the constraint. Fractions use all samples, axes match within rows, and the dashed line marks the optimum. The consecutive sizes were chosen independently of scores.

Table 15: **Source-wise robustness of C4 reward adaptation.** Each row uses one selected source checkpoint at the indicated pretraining order and its ten reward and ten fixed-corpus fits. Gains are differences in refined mean feasible edges, averaged equally over the 100 fine-tuning/evaluation pairs. Raw valid Δ is the change in percentage points relative to the same unchanged source. Win counts use unrounded paired values.

Source n	Reward − base	Reward − fixed	Raw valid Δ (pp)	Wins/base	Wins/fixed
36	+2.42	+2.55	-2.47	100/100	100/100
37	+2.29	+2.48	-11.51	99/100	100/100
38	+2.92	+2.91	-29.16	100/100	100/100
39	+2.97	+2.87	-22.12	100/100	100/100
40	+3.86	+3.55	-39.10	100/100	100/100
41	+3.72	+3.46	-36.59	100/100	100/100
42	+4.10	+3.60	-38.21	100/100	100/100

Table 16: **C4 adaptation by evaluation order.** Base averages the seven selected unchanged sources. Fixed and Reward Δ are paired changes in mean feasible edges after repair over seven source orders and ten fine-tuning orders. Raw valid percentages use the whole 10,000-sample population per cell. The final column counts settings with a positive reward-versus-source gain among 70 settings. Every refined graph is feasible.

	Mean refined edges			Raw valid (%)		
Eval. n	Base	Fixed Δ	Reward Δ	Base	Reward	Reward > base
33	89.20	+0.24	+3.51	50.14	23.99	70/70
34	92.99	+0.23	+3.47	47.48	18.86	70/70
35	96.81	+0.22	+3.37	44.93	13.60	70/70
36	100.65	+0.20	+3.39	40.82	9.67	70/70
37	104.51	+0.19	+3.24	36.73	5.80	70/70
38	108.40	+0.16	+3.15	31.70	3.40	70/70
39	112.31	+0.10	+3.02	27.52	1.77	70/70
40	116.26	+0.04	+2.97	22.76	0.93	70/70
41	120.21	-0.03	+2.86	18.30	0.45	70/70
42	124.19	-0.13	+2.82	14.27	0.20	69/70

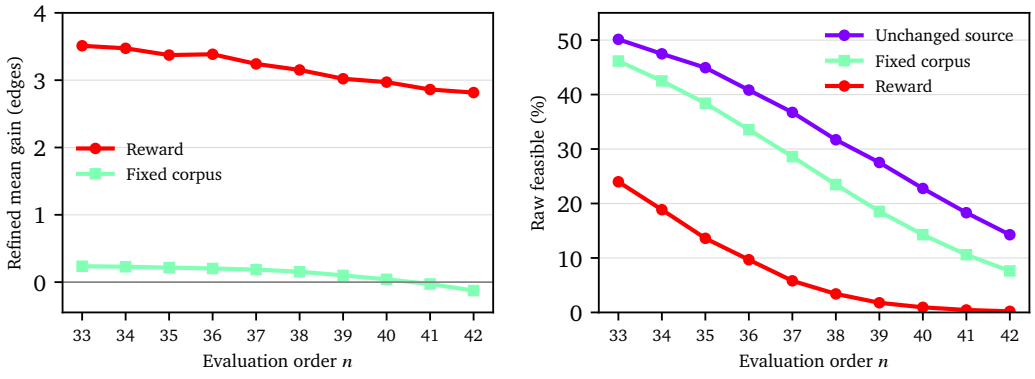


Figure 11: **Refined gains and raw feasibility across evaluation orders.** Left: change in mean refined edges relative to the exact unchanged source, averaged over seven source orders and ten fine-tuning orders. Right: raw valid fraction for the base, fixed-corpus, and reward generators. The base averages seven sources; adapted methods average 70 models. Each population has 10,000 samples. These grid averages do not estimate training-seed uncertainty. The fine-tuning-by-evaluation maps are in Figure 9.

G.2 INDIVIDUAL BASE MODELS

Figures 12 to 18 separate the seven bases averaged in Figure 9. Each figure compares fixed-corpus and reward adaptation against the same unchanged base. The shared color scale makes gains comparable across bases; cell values are rounded to one decimal place. Figure 19 shows each base's absolute quality before and after reward fine-tuning, averaging the ten fine-tuning orders. Under the same architecture and training schedule, the smaller pretraining orders generally produce stronger generators across evaluation sizes. Among the selected checkpoints, this advantage in mean refined edges persists after reward fine-tuning, averaged over all ten fine-tuning orders, despite larger gains for models pretrained on larger graphs.

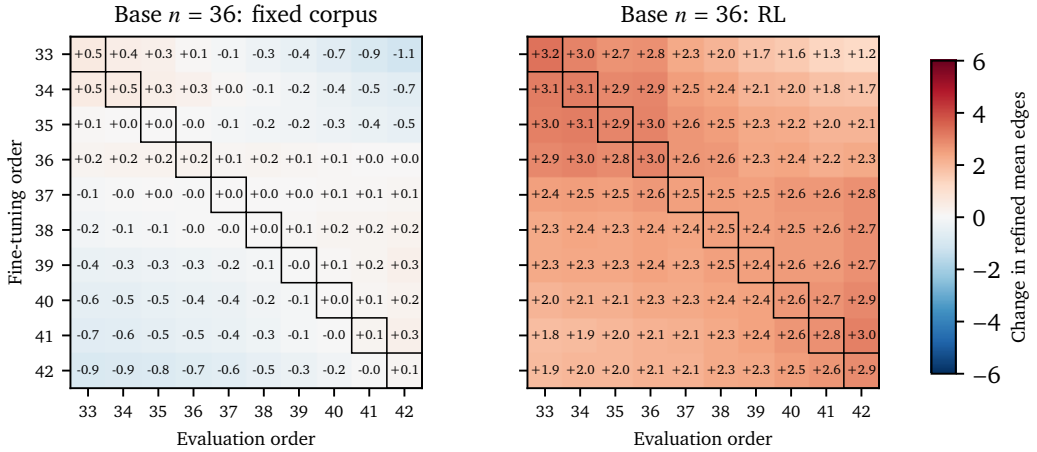


Figure 12: **Adaptation from the base trained at $n = 36$.** Numbers show changes in mean refined edges relative to this exact unchanged checkpoint at each evaluation order, using 10,000 draws per population. Outlined cells evaluate at the fine-tuning order. Both methods use one adaptation seed; the color scale is shared across all seven base-model figures.

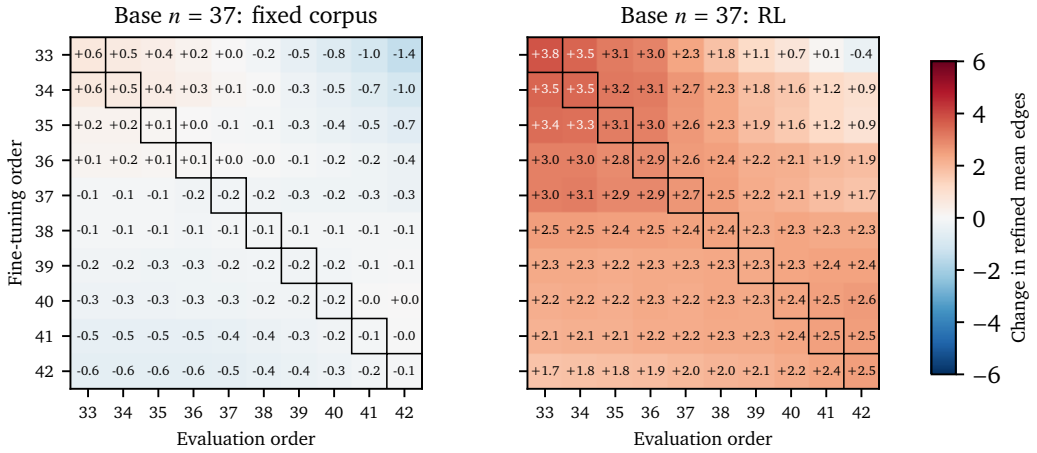


Figure 13: **Adaptation from the base trained at $n = 37$.** Numbers show changes in mean refined edges relative to this exact unchanged checkpoint at each evaluation order, using 10,000 draws per population. Outlined cells evaluate at the fine-tuning order. Both methods use one adaptation seed; the color scale is shared across all seven base-model figures.

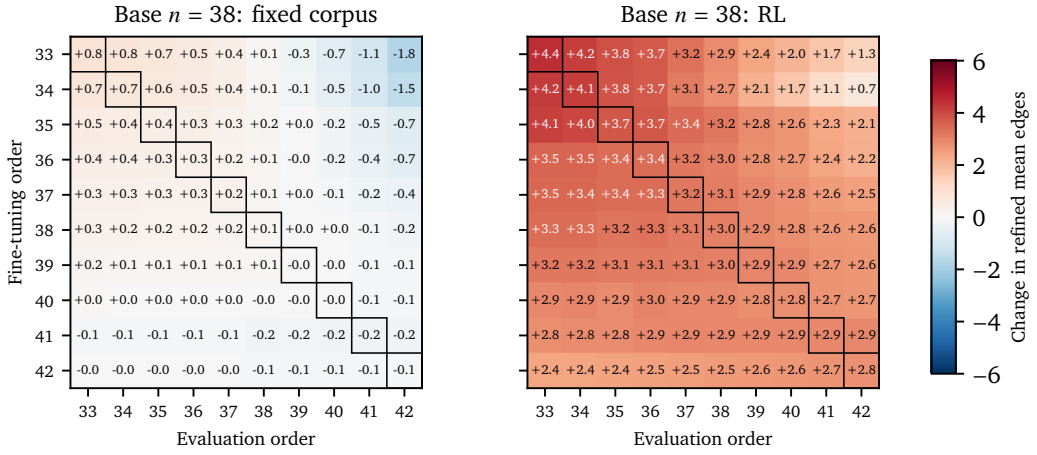


Figure 14: **Adaptation from the base trained at $n = 38$.** Numbers show changes in mean refined edges relative to this exact unchanged checkpoint at each evaluation order, using 10,000 draws per population. Outlined cells evaluate at the fine-tuning order. Both methods use one adaptation seed; the color scale is shared across all seven base-model figures.

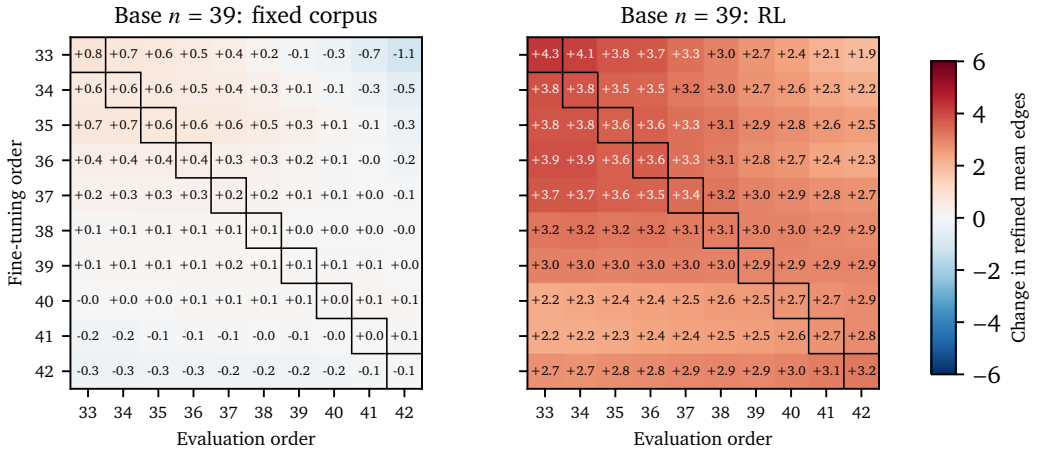


Figure 15: **Adaptation from the base trained at $n = 39$.** Numbers show changes in mean refined edges relative to this exact unchanged checkpoint at each evaluation order, using 10,000 draws per population. Outlined cells evaluate at the fine-tuning order. Both methods use one adaptation seed; the color scale is shared across all seven base-model figures.

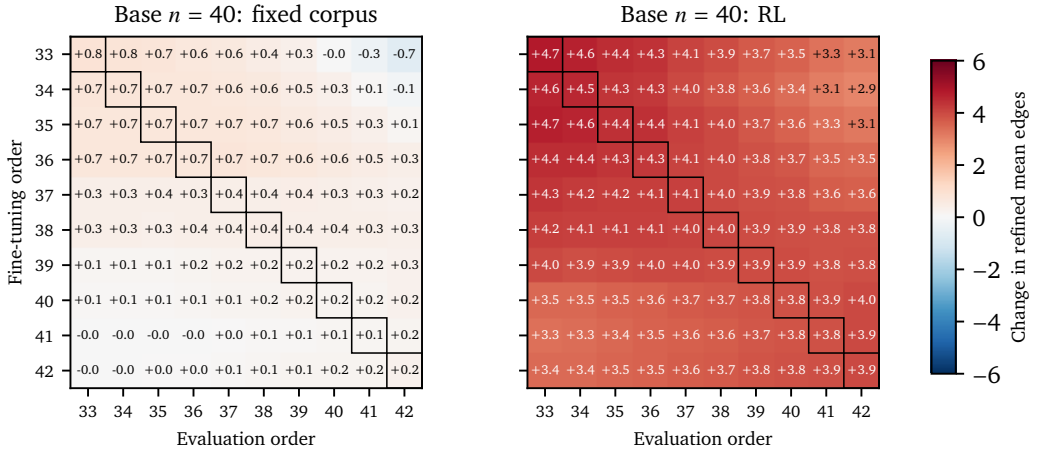


Figure 16: **Adaptation from the base trained at $n = 40$.** Numbers show changes in mean refined edges relative to this exact unchanged checkpoint at each evaluation order, using 10,000 draws per population. Outlined cells evaluate at the fine-tuning order. Both methods use one adaptation seed; the color scale is shared across all seven base-model figures.

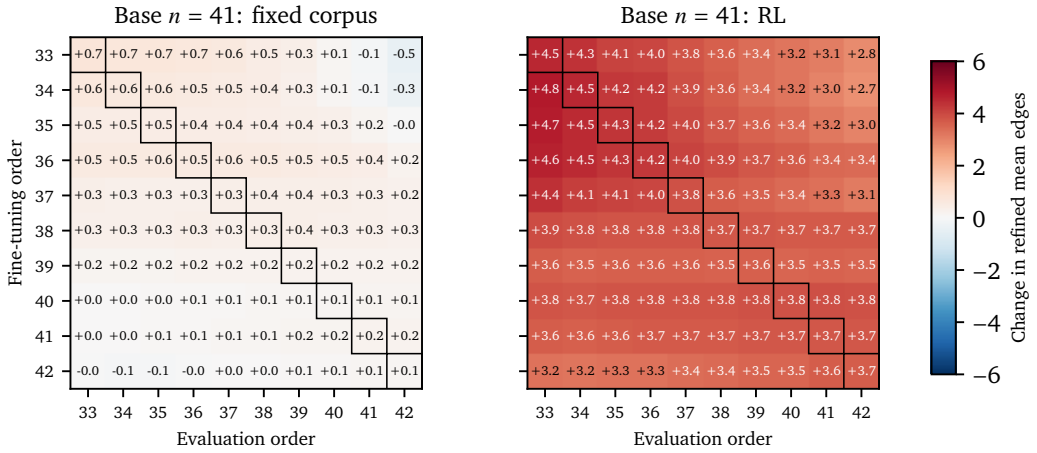


Figure 17: **Adaptation from the base trained at $n = 41$.** Numbers show changes in mean refined edges relative to this exact unchanged checkpoint at each evaluation order, using 10,000 draws per population. Outlined cells evaluate at the fine-tuning order. Both methods use one adaptation seed; the color scale is shared across all seven base-model figures.

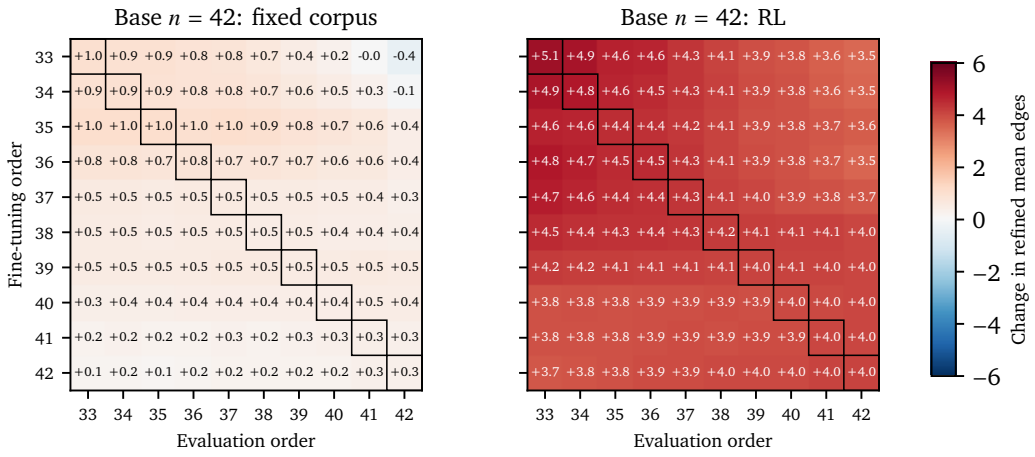


Figure 18: **Adaptation from the base trained at** $n = 42$**.** Numbers show changes in mean refined edges relative to this exact unchanged checkpoint at each evaluation order, using 10,000 draws per population. Outlined cells evaluate at the fine-tuning order. Both methods use one adaptation seed; the color scale is shared across all seven base-model figures.

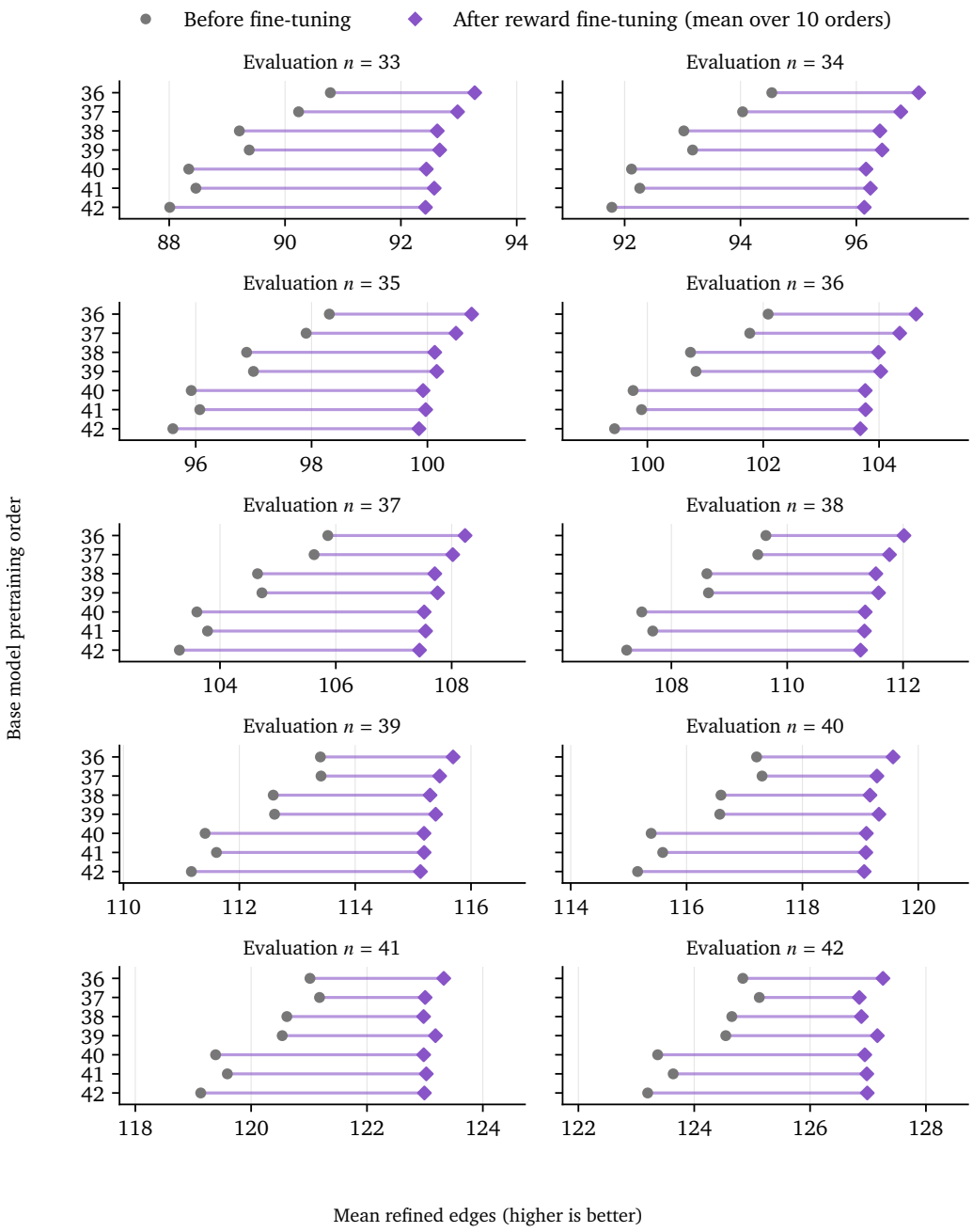


Figure 19: **Starting strength and improvement from reward fine-tuning.** Each panel fixes the evaluation order; each row uses one selected base model. Gray circles show its unchanged mean refined edge count. Purple diamonds average the mean refined edge counts of its ten reward-adapted models, one per fine-tuning order 33–42. Each population has 10,000 draws. Connecting lines show the gain, not uncertainty. All panels use the same horizontal span, so equal line lengths represent equal gains. Endpoints average fine-tuning orders, not independent training seeds or selected best models.

G.3 Reward adaptation from order 25

Source and adaptation protocol. We select training seed 2 from the three order-25 models using only the final training-size population: highest refined best edge count, then highest optimum frequency, then highest refined mean. Seeds 1 and 2 both reach 63 edges; seed 2 has optimum frequency 71.035% versus 68.360%. The selected source completes 30 fits of 5,000 updates. Ten independent reward adaptations start from this same checkpoint, one at each order 33–42. They use the same protocol as the adaptation matrix: 2,000 updates, learning rate 10^{-5}, 256 fresh trajectories per four updates, batch 256, microbatch 32, and clipping parameter 0.1. There is one adaptation seed per order and no fixed-corpus control. Source training and all ten adaptations use H200 GPUs.

Evaluation and uncertainty. Each final adapted checkpoint is evaluated at every order 33–42 with 10,000 samples, 64 FP32 flow steps, sampling batch 64 and C4 greedy refinement with batch 256. The unchanged-source baseline pools three existing 10,000-draw evaluations per order. Thus comparisons share the source and evaluation protocol but not sampling seeds. The 100 cells reuse one selected training seed and ten adapted models; they do not provide 100 independent training replicates or a training-seed uncertainty estimate.

Results. Mean refined quality improves in all 100 cells, including all 90 away from the fine-tuning order and all 45 above it. Gains range from 3.51 to 6.24 edges, averaging 4.99; every adapted mean exceeds each of the three baseline sampling-seed means at its evaluation order. Raw feasibility remains rare: 482 of one million adapted proposals are feasible. The panel includes raw optima at orders 33–36 and 38, but the order-38 hit comes from adaptation at 40 and is not upward transfer beyond adaptation. The $25 \to 33 \to 34$ setting produces one raw optimal 102-edge graph. These isolated constructions supplement the primary finding of consistently improved inputs to local search.

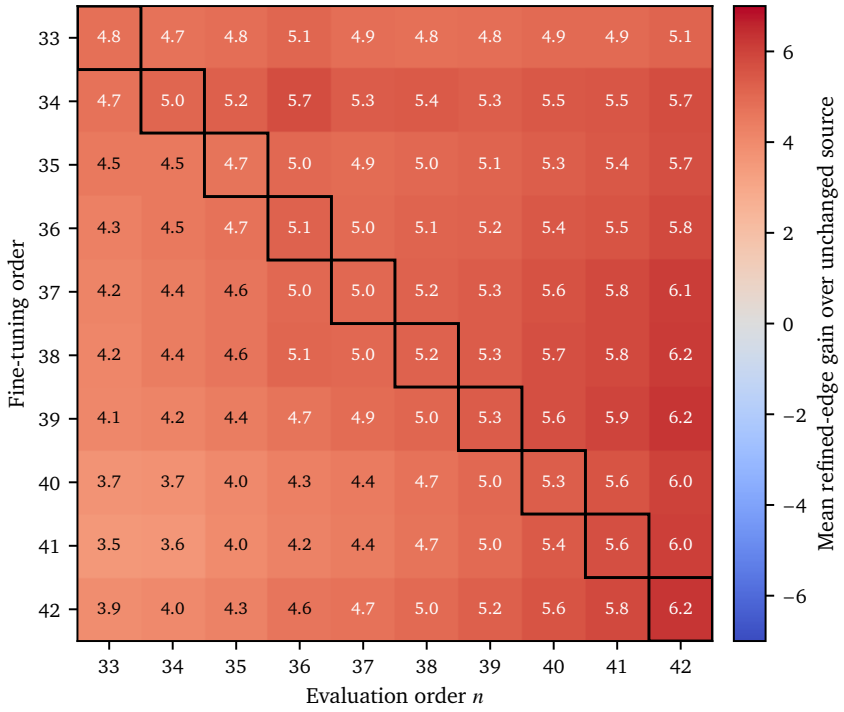


Figure 20: **Reward adaptation from a 25-vertex source.** Cells show mean refined-edge gains over the unchanged source at the same evaluation order. Each adapted cell has 10,000 draws; each baseline mean pools 30,000 draws. Outlined cells evaluate at the fine-tuning order; cells to their right evaluate above both training sizes. All cells are positive. Sampling seeds differ between adapted and baseline evaluations.

Selection and costs in the main comparison. The RL ($n_0 = 25$) rows select among these ten adaptations at each evaluation order by refined best edges, then refined mean, then raw best. All columns describe that selected model; raw maxima from other adapted models are not substituted. The 140-candidate matrix selection is unaffected. Each cost includes the selected source's full pre-training and the selected adaptation run, excluding the other candidates and separate transfer evaluations. Selecting across ten candidates is distinct from a fixed-model comparison. Local-search counts follow Appendix D.

Table 17: **Full order-25 reward-adaptation grid.** Base pools three 10,000-draw evaluations of the same unchanged checkpoint; each FT row uses 10,000 draws from one adapted checkpoint. Raw means and best counts condition on feasibility; all refined graphs are feasible. Δ mean subtracts the Base refined mean at the same order. Dashes indicate no feasible raw sample or no adaptation. These are individual settings, not averages across training seeds.

Eval. n	FT	Valid (%)	Raw mean	Raw best	Ref. mean	Ref. best	Δ mean
33	**Base**	0.030	91.78	96	88.86	96	—
	33	0.000	—	—	93.64	96	4.78
	34	0.010	94.00	94	93.58	96	4.73
	35	0.000	—	—	93.35	96	4.50
	36	0.010	93.00	93	93.16	96	4.30
	37	0.170	93.24	95	93.06	96	4.20
	38	0.160	93.00	95	93.01	96	4.15
	39	0.510	93.22	96	92.95	96	4.09
	40	0.410	93.54	96	92.53	96	3.67
	41	1.420	92.37	96	92.37	96	3.51
	42	0.540	92.87	96	92.79	96	3.93
34	**Base**	0.000	—	—	92.21	102	—
	33	0.010	102.00	102	96.93	102	4.73
	34	0.000	—	—	97.24	102	5.03
	35	0.000	—	—	96.70	102	4.50
	36	0.000	—	—	96.73	102	4.52
	37	0.040	100.25	102	96.59	102	4.38
	38	0.030	101.33	102	96.59	102	4.38
	39	0.290	100.52	102	96.37	102	4.16
	40	0.200	100.25	102	95.96	102	3.75
	41	0.360	99.78	102	95.82	102	3.61
	42	0.190	100.26	102	96.22	102	4.01
35	**Base**	0.000	—	—	95.56	104	—
	33	0.000	—	—	100.38	106	4.82
	34	0.020	105.00	105	100.80	106	5.23
	35	0.000	—	—	100.23	106	4.67
	36	0.000	—	—	100.30	106	4.73
	37	0.060	105.33	106	100.19	106	4.63
	38	0.020	104.00	104	100.20	106	4.64
	39	0.100	106.00	106	99.97	106	4.41
	40	0.030	105.00	106	99.56	106	4.00
	41	0.100	104.60	106	99.53	106	3.96
	42	0.070	104.29	106	99.82	106	4.26
36	**Base**	0.000	—	—	98.99	110	—
	33	0.000	—	—	104.04	110	5.05
	34	0.000	—	—	104.73	110	5.74
	35	0.000	—	—	104.00	110	5.01
	36	0.000	—	—	104.13	110	5.14
	37	0.010	110.00	110	104.00	110	5.01
	38	0.000	—	—	104.06	110	5.07
	39	0.020	109.00	110	103.68	110	4.69
	40	0.000	—	—	103.29	110	4.31
	41	0.010	108.00	108	103.20	110	4.22
	42	0.010	108.00	108	103.59	110	4.61
37	**Base**	0.000	—	—	102.45	112	—

Continued on next page.

Order-25 reward adaptation (continued)

Eval. n	FT	Valid (%)	Raw mean	Raw best	Ref. mean	Ref. best	Δ mean
	33	0.000	—	—	107.31	113	4.86
	34	0.000	—	—	107.79	113	5.35
	35	0.000	—	—	107.37	113	4.92
	36	0.000	—	—	107.42	113	4.98
	37	0.000	—	—	107.46	113	5.02
	38	0.000	—	—	107.42	113	4.98
	39	0.000	—	—	107.32	113	4.87
	40	0.000	—	—	106.89	113	4.45
	41	0.000	—	—	106.85	113	4.41
	42	0.000	—	—	107.19	113	4.75
38	**Base**	0.000	—	—	105.90	114	—
	33	0.000	—	—	110.69	117	4.79
	34	0.000	—	—	111.27	117	5.37
	35	0.000	—	—	110.88	117	4.98
	36	0.000	—	—	111.02	117	5.12
	37	0.000	—	—	111.06	117	5.16
	38	0.000	—	—	111.09	117	5.19
	39	0.000	—	—	110.94	117	5.04
	40	0.010	117.00	117	110.60	117	4.71
	41	0.000	—	—	110.58	117	4.68
	42	0.000	—	—	110.85	117	4.96
39	**Base**	0.000	—	—	109.40	116	—
	33	0.000	—	—	114.24	121	4.84
	34	0.000	—	—	114.70	121	5.30
	35	0.000	—	—	114.53	121	5.13
	36	0.000	—	—	114.58	121	5.18
	37	0.000	—	—	114.71	121	5.31
	38	0.000	—	—	114.71	121	5.31
	39	0.000	—	—	114.71	121	5.31
	40	0.010	121.00	121	114.39	121	4.99
	41	0.000	—	—	114.37	120	4.97
	42	0.000	—	—	114.60	121	5.20
40	**Base**	0.000	—	—	112.92	121	—
	33	0.000	—	—	117.86	125	4.94
	34	0.000	—	—	118.38	125	5.46
	35	0.000	—	—	118.21	125	5.29
	36	0.000	—	—	118.35	125	5.42
	37	0.000	—	—	118.55	125	5.63
	38	0.000	—	—	118.58	125	5.65
	39	0.000	—	—	118.57	125	5.64
	40	0.000	—	—	118.21	125	5.29
	41	0.000	—	—	118.29	125	5.36
	42	0.000	—	—	118.49	125	5.57
41	**Base**	0.000	—	—	116.50	124	—
	33	0.000	—	—	121.44	129	4.94
	34	0.000	—	—	122.02	129	5.52
	35	0.000	—	—	121.93	129	5.43
	36	0.000	—	—	122.04	128	5.54
	37	0.000	—	—	122.30	130	5.80
	38	0.000	—	—	122.30	129	5.80
	39	0.000	—	—	122.41	129	5.91
	40	0.000	—	—	122.07	128	5.57
	41	0.000	—	—	122.13	128	5.64
	42	0.000	—	—	122.32	129	5.83
42	**Base**	0.000	—	—	120.09	127	—
	33	0.000	—	—	125.20	132	5.10
	34	0.000	—	—	125.81	132	5.72
	35	0.000	—	—	125.74	132	5.65

Continued on next page.

Order-25 reward adaptation (continued)

Eval. n	FT	Valid (%)	Raw mean	Raw best	Ref. mean	Ref. best	Δ mean
	36	0.000	—	—	125.93	132	5.84
	37	0.000	—	—	126.19	134	6.10
	38	0.000	—	—	126.25	133	6.16
	39	0.000	—	—	126.33	133	6.24
	40	0.000	—	—	126.05	133	5.96
	41	0.000	—	—	126.13	132	6.04
	42	0.000	—	—	126.33	133	6.24

H RAMSEY CONSTRUCTION

H.1 SMALL RAMSEY CONSTRUCTION BENCHMARKS

Table 18 evaluates four fixed-size tasks over three training seeds each. On $R(4, 4; 17)$, more than 94% of the final raw proposals are valid. Both $R(3, 7)$ sizes also yield valid graphs frequently before search. On the harder $R(4, 5; 24)$, the final raw populations contain no valid graph, while tabu refinement finds two among 30,000 draws.

Table 18: **Learning valid Ramsey constructions.** Fresh final populations after 20 fits, with 10,000 graphs for each of three training seeds. Entries are seed mean ± sample SD. Cost counts forbidden cliques and independent sets, with zero indicating validity. Minima are computed per population before averaging. Mean costs include invalid graphs. These counts include possible duplicate graphs.

	Raw			Tabu-refined		
Instance	Min. cost	Mean cost	Valid (%)	Min. cost	Mean cost	Valid (%)
$R(3, 7; 21)$	0.00 ± 0.00	0.70 ± 0.15	72.830 ± 2.451	0.00 ± 0.00	0.09 ± 0.01	92.180 ± 0.410
$R(3, 7; 22)$	0.00 ± 0.00	1.82 ± 0.34	68.707 ± 0.614	0.00 ± 0.00	0.45 ± 0.02	74.140 ± 1.082
$R(4, 4; 17)$	0.00 ± 0.00	0.74 ± 0.09	94.333 ± 0.571	0.00 ± 0.00	0.27 ± 0.03	95.660 ± 0.477
$R(4, 5; 24)$	2.00 ± 1.00	14.30 ± 0.30	0.000 ± 0.000	0.33 ± 0.58	7.95 ± 0.02	0.007 ± 0.006

H.2 RAMSEY TRANSFER AT UNSEEN SIZES AND RANDOM-START CONTROLS

All three final $R(3, 7; 21)$ checkpoints from Appendix H.1 are evaluated at $n = 22$, and both order-30 $R(4, 6)$ checkpoints at $n = 31$, with 30,000 proposals per checkpoint (three sampling seeds of 10,000 each), 64 FP32 flow steps, sampling batch size 64, and the tabu policy described below. All evaluations use A40 GPUs.

For $R(3, 7)$, each of the three training seeds produces valid raw graphs at the unseen order $n = 22$: 761, 721 and 1,138 among 30,000 draws each, and 1,630, 1,681 and 2,098 valid graphs after refinement. For $R(4, 6)$, refinement finds a valid graph in every evaluation at $n = 31$: 39 and 24 valid refined graphs among 30,000 draws each, with no valid raw draws.

The $R(4, 6; 31)$ controls initialize edges independently, either uniformly or with the mean edge density of the corresponding generator's archived raw order-31 population, and receive the same refinement policy and number of steps per proposal as the learned arm. Neither control finds a valid graph. These counts include possible duplicates; they do not establish isomorphism novelty.

H.3 RAMSEY FIXED-WEIGHT TRANSFER WITH TABU SEARCH

We evaluate all four final checkpoints for $R(4,6)$, trained at orders 30 and 32, and all four for $R(5,5)$, trained at orders 35 and 38. Each source order has two seeds. No checkpoint is selected using target-order results. The generator uses four triangle-attention Pairformer blocks, width 128 and eight heads. Training uses 20 fits of 5,000 AdamW updates, learning rate 2×10^{-4}, batch size 256, and 10,000 proposals per generation. The selected top 1% of refined proposals and 50 elite rows form each fitting corpus, without construction injection or vertex-relabel augmentation. These sources were trained with tabu refinement. Evaluation keeps at most 128 tabu steps, 16 candidate edges, tabu tenure 7 and search batch size 32, returning the best visited graph per input.

Each cell uses 10,000 fresh graphs, uniform binary initial noise, 64 FP32 flow steps, sampling batch size 64, with separate random streams for sampling and refinement. All 48 evaluations use A40 GPUs. We evaluate $R(4,6)$ at orders 30–35 and $R(5,5)$ at orders 35–40, without changing learned weights. Cost counts r-cliques plus independent s-sets; zero is valid. Unlike the C4 feasible-conditioned edge means, Ramsey mean costs include *all* samples. For $R(4,6)$, the four checkpoints jointly produce four valid raw draws at $n = 30$. Tabu refinement yields 5,979 valid samples at $n = 30$ and 41 at $n = 31$, out of 40,000 samples at each order; none is valid at $n = 32, \ldots, 35$. These counts include possible duplicates. For $R(5,5)$, none of the 240,000 raw or corresponding refined graphs is valid at orders 35–40. This is a finite-sample result, not an impossibility claim.

H.4 RAMSEY REWARD ADAPTATION

One selected $R(4,6)$ checkpoint trained at $n = 30$ and one trained at $n = 32$ initialize both the reward-policy and fixed-corpus arms. Each source is adapted separately at every order 30–35, giving 12 paired settings. Both arms have 2,000 updates and one training seed per setting. Reward uses fresh trajectories scored after tabu refinement; the control continues denoising on one fixed corpus. A 1,024-graph final population at the fine-tuning order measures both raw and refined endpoints, using one evaluation seed. The arms are paired by the exact source checkpoint and target order.

Reward gives lower raw and refined mean forbidden-clique cost in all 12 pairs. No refined valid graph is observed at orders 32–35, where mean cost still distinguishes the methods. Table 19 averages the two source-specific values at each target order. These are descriptive means over the two selected sources, not uncertainty estimates over adaptation seeds.

Table 19: **Reward versus fixed-corpus adaptation for $R(4,6)$.** Each value averages a pair of final 1,024-sample evaluations, one from each selected source order. Mean cost counts forbidden cliques and independent sets over all graphs. Validity is the fraction with zero cost after tabu search. Both raw and refined mean-cost comparisons favor reward in every individual paired setting.

	Raw mean cost		Refined mean cost		Refined valid (%)	
Target n	Fixed	Reward	Fixed	Reward	Fixed	Reward
30	16.55	14.40	2.26	1.29	16.65	32.86
31	24.09	18.87	7.68	5.15	0.05	0.88
32	34.83	25.43	16.61	12.59	0.00	0.00
33	51.25	39.95	29.06	23.18	0.00	0.00
34	72.33	60.40	45.67	37.71	0.00	0.00
35	96.82	74.30	66.45	54.98	0.00	0.00