STITCH2-FWL: STITCHING GLOBAL EXPRESSIVITY FROM LOCAL REFINEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Dense pair refinement incurs cubic work even on structurally simple components. We introduce STITCH2-FWL, which uses block–cut and SPQR decompositions to confine complete computation to rigid R-skeletons and to reduce an s-cycle to $4s^2 - 3s$ interactions per round. When the induced colored R-configurations are separable and Schurian, the resulting local correspondences agree at their interfaces and compose into a graph isomorphism. Within this class Stitch matches dense 2-FWL, and neither the total graph size nor the decomposition depth is bounded. The class contains every graph without R-skeletons and, under an explicit enumeration premise, every graph whose R-skeletons have at most thirteen vertices. All graphs recorded in our ZINC, QM9, ZINC-full, and PCQM4Mv2 audit meet this size criterion. In exact experiments Stitch preserves every dense distinction across 1,008 long-cycle cases, lowers cumulative interactions by 60.98% at the largest tested subdivision level, and reproduces all 400 BREC decisions. We also report prediction and runtime results. A counterexample outside the guaranteed class motivates the one-hop Stitch-Plus variant, which repairs it; whether that variant is generally equivalent remains open.

1 INTRODUCTION

Which higher-order interactions are necessary to preserve global graph distinctions? Dense 2-FWL updates an ordered-pair state against every vertex as a common pivot, which costs n^3 explicit interactions per round. This expressivity motivates pair-based graph neural networks (Maron et al., 2019), yet the same uniform computation is applied to bridges, long cycles, and rigid components alike.

STITCH2-FWL allocates computation by structural role. Block–cut and SPQR decompositions expose the one- and two-vertex interfaces between pieces, and Stitch keeps complete computation inside rigid R-skeletons while restricting S-cycle pivots to neighbors of the root endpoints (Figure 1). A standalone s-cycle then needs $4s^2 - 3s$ interactions in place of s^3. The pieces still communicate through shared pair states; no decomposition identity is added to the initial colors.

The difficulty lies in composition. Comparing each piece in isolation says nothing about how attached branches relate to one another. We prove that Stitch colors yield local maps agreeing at shared interfaces whenever the induced colored R-configurations are separable and Schurian. Inside that class Stitch and dense 2-FWL distinguish exactly the non-isomorphic pairs. The conditions constrain only local rigid structure; graph size, cycle length, and decomposition depth may all be unbounded. Every no-R graph qualifies, and under a stated enumeration premise so does every graph whose R-skeletons have at most thirteen vertices. The latter sufficient condition covers every graph in several large molecular collections in our decomposition logs.

Our contributions are an exact quadratic cycle selector, a proof that its local comparisons compose under explicit conditions, and an evaluation that separates structural coverage, discrete distinctions, prediction, and runtime. On 1,008 controlled long-cycle pairs the cumulative savings survive the extra rounds, and on BREC all 400 decisions match dense refinement. A neural comparison reports ZINC scores close to the baseline and task-dependent decreases on TU. We also exhibit a cross-interface failure outside the guaranteed class, together with a one-hop Plus extension that repairs it. The Stitch theorem does not depend on the still-open general equivalence of Plus.

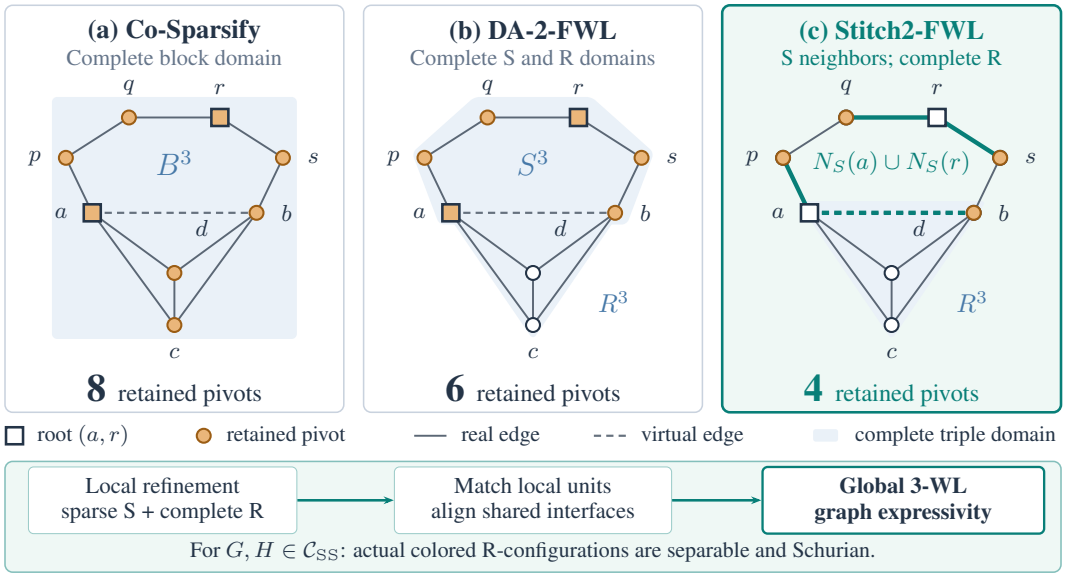

Figure 1: **Allocate computation by structural role.** All panels use the same graph and root (a, r) (squares). The S-cycle a, p, q, r, s, b, a and R-skeleton on $\{a, b, c, d\}$ share virtual edge ab. Orange marks retained pivots: eight block vertices for Co-Sparsify, six S vertices for DA, and $\{b, p, q, s\}$ for Stitch. Blue shading denotes complete triple domains; teal edges identify the S-neighbor rule. The lower strip states the class-conditioned global guarantee.

2 Decomposition and Interfaces

We work with finite simple undirected unlabelled graphs, allowing disconnected and empty inputs, and represent states as ordered pairs. A *block* is either a maximal biconnected subgraph on at least three vertices or a bridge together with its endpoints. The block–cut tree records which blocks meet which cut vertices; isolated vertices are recorded separately (Holm et al., 2018; Hopcroft & Tarjan, 1973).

Every non-bridge block admits a reduced SPQR tree (Fink & Rutter, 2023; Gutwenger & Mutzel, 2001; Di Battista & Tamassia, 1996), whose skeletons are cycles (S), bonds with two poles and at least three parallel edge instances (P), or simple triconnected graphs on at least four vertices (R). We suppress Q leaves and assign each original edge to a single skeleton as a real edge. Adjacent skeletons share a two-vertex interface, represented by a pair of virtual edges. Reconstruction removes these virtual edges and identifies their endpoints, so a virtual edge is never an input edge. In a reduced tree no edge joins two S-nodes or two P-nodes.

The skeletons containing a fixed vertex form a connected subtree, so a branch meets the rest of the graph only at its one- or two-vertex interface. Several branches may share a pole pair through a single P-node, and their instances stay distinct. Figure 2 shows the two levels; Appendix A collects the full conventions and separation proofs.

3 Stitch: Decomposition-Guided Pair Refinement

Expressivity convention. We call the common-pivot joint refinement on ordered pairs *2-FWL* and follow the graph-learning convention in which its graph-distinguishing power equals that of 3-WL (Maron et al., 2019). Some graph-isomorphism references give the same pair procedure the name 2-WL; when we cite those results we rely on their update rule rather than on the dimension in the name (Dell et al., 2018; Kiefer & Neuen, 2022). Throughout, we compare graph signatures: graph-level equivalence does not require the sparse and dense pair-color partitions to coincide.

The selector. A triple (a, t, b) denotes the root (a, b) together with its pivot t. Let $\mathcal{B}_{\mathrm{br}}$, $\mathcal{S}$, and $\mathcal{R}$ collect the bridge blocks, S-skeletons, and R-skeletons of the input. On any input graph, define the

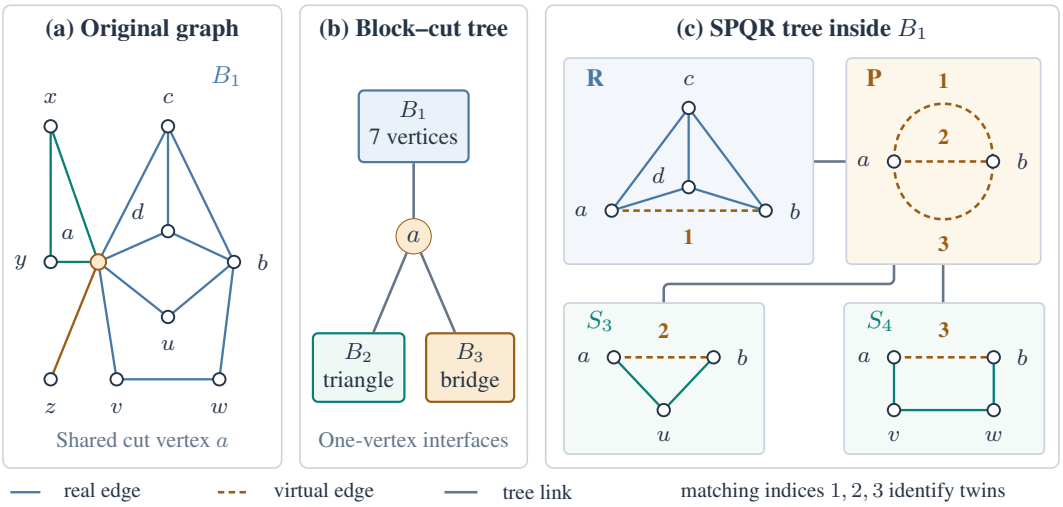

Figure 2: **Two nested decomposition levels.** The blue block B_1, green triangle, and orange bridge meet at cut vertex a. The block–cut tree records these one-vertex interfaces. Expanding B_1 gives an R, two S-cycles, and a P-bond with three distinct virtual ab edges. Gray links are tree incidences; within skeletons, solid edges are real and dashed edges are virtual. Matching indices identify twin virtual records, which are removed when reconstructing B_1. The original graph has no edge ab.

auxiliary selector

$$\widehat{T}_G = \{(v,v,v) : v \in V(G)\} \cup \bigcup_{B \in \mathcal{B}_{\mathrm{br}}} V(B)^3 \cup \bigcup_{R \in \mathcal{R}} V(R)^3 \cup \bigcup_{S \in \mathcal{S}} \{(a,t,b) \in V(S)^3 : at \in E(S) \ \vee \ tb \in E(S)\}. \tag{1}$$

Here $E(S)$ contains real and virtual skeleton edges alike. All unions are set-theoretic, so a triple generated by two skeletons is processed once; P-skeletons contribute no cube of their own. The active pairs and their pivot sets are

$$\widehat{P}_G = \{(a,b) : \exists t \ (a,t,b) \in \widehat{T}_G\}, \qquad \widehat{N}_G(a,b) = \{t : (a,t,b) \in \widehat{T}_G\}. \tag{2}$$

Every internal pair of a bridge, S, or R is active, and so is every diagonal. Whenever (a,t,b) is selected, its two source pairs (a,t) and (t,b) are active as well, which makes the update well-defined.

Endpoint pivots are not automatically selected at a nonadjacent S-root; a diagonal root instead receives its two S-neighbors and the self-triple. Appendix H works out these cases.

Refinement and output. Initialize $\widehat{c}_0(a,b)$ from equality and adjacency in the *original* graph, then apply an injective common encoding,

$$\widehat{c}_{i+1}(a,b) = \mathrm{Enc}\left(\widehat{c}_i(a,b), \{\{(\widehat{c}_i(a,t), \widehat{c}_i(t,b)) : t \in \widehat{N}_G(a,b)\}\}\right). \tag{3}$$

Both source colors of a message are read at the same pivot. When two graphs are compared, their colors are encoded jointly on the disjoint union of the state sets, with pivots confined to their own graph. Stability here means that the partition no longer splits; it does not mean that the numerical color identifiers stop changing. Write $\widehat{c}_G$ for the stable color and

$$\widehat{\mathrm{Sig}}(G) = \{\{\widehat{c}_G(a,b) : (a,b) \in \widehat{P}_G\}\}. \tag{4}$$

Dense 2-FWL is the same update on all of $V(G)^2$, with $N_G(a,b) = V(G)$. We write $G \equiv_{\mathrm{D}} H$ when the two stable graph signatures agree.

Colored R-configurations and the guaranteed class. For each R-skeleton, form the complete ordered-pair coloring

$$\kappa_R(a,b) = (\widehat{c}_G(a,b), \sigma_R(a,b)), \qquad a,b \in V(R), \tag{5}$$

in which σ_R records whether the pair is diagonal, a real skeleton edge, a virtual skeleton edge, or a skeleton nonedge. Let $A_R(G)$ denote this colored object and $\mathcal{X}_R(G) = \mathrm{CC}(A_R(G))$ its coherent closure under dense pair refinement, initialized with the usual endpoint, reverse-color, and equality information. This local closure states the qualification condition; it is never fed back into (3). Its stable relation classes have constant joint intersection counts.

Definition 1 (The class $\mathcal{C}_{\mathrm{SS}}$)**.** A graph belongs to $\mathcal{C}_{\mathrm{SS}}$ if the configuration $\mathcal{X}_R(G)$ is separable and Schurian for every R-skeleton R. Separability requires every specified algebraic isomorphism of the configuration to be induced by a vertex bijection. Schurianity requires its basic relations to be exactly the ordered-pair orbits of its color-preserving automorphism group.

The two conditions play complementary roles: separability realizes a relation correspondence, and Schurianity aligns a prescribed same-colored ordered entry. Appendix D gives the formal definitions.

STITCH2-FWL is the auxiliary refinement (3) restricted to $\mathcal{C}_{\mathrm{SS}}$, with signature $\mathrm{Sig}_{\mathrm{St}} = \widehat{\mathrm{Sig}}$. The auxiliary computation itself is defined on all finite graphs, independently of the qualification condition, and this separation keeps the definition noncircular: $G \mapsto \widehat{c}_G \mapsto A_R(G) \mapsto \mathcal{X}_R(G) \mapsto$ qualification. The sufficient conditions of Section 4 certify membership; computing the general condition may be expensive, and a failed sufficient test leaves membership undecided.

Proposition 2 (Well-definedness and interaction cost)**.** *The auxiliary refinement terminates and is isomorphism-invariant. Writing $n = |V(G)|$, $b = |\mathcal{B}_{\mathrm{br}}|$, $s_S = |V(S)|$, and $r_R = |V(R)|$, its numbers of states and interactions satisfy*

$$|\widehat{P}_G| \le n + 2b + \sum_S s_S(s_S - 1) + \sum_R r_R(r_R - 1), \tag{6}$$

$$|\widehat{T}_G| \le n + 6b + \sum_S (4s_S^2 - 4s_S) + \sum_R (r_R^3 - r_R). \tag{7}$$

For a standalone cycle C_s, $|\widehat{P}| = s^2$ and $|\widehat{T}| = 4s^2 - 3s$.

Appendix H gives the proof and the execution details. The bounds count states and explicit interactions per round. Total cost depends in addition on the decomposition, the encoding, the stabilization depth, and any class-membership checks; a large R-skeleton still contributes cubically, so the bounds do not imply a universally subcubic implementation or a proportional reduction in GPU memory.

When $r_R \le r_0$, the rigid contribution obeys $\sum_R r_R^3 \le r_0^2 \sum_R r_R$ and is therefore linear in the number of R-bag vertex occurrences.

Neural realization and unfoldings. A neural layer learns each pair update from its previous state and the aggregated same-pivot messages; the experiments use the DA backbone with the Stitch domain (Appendix Q). The discrete comparison assumes atomic initialization, while the experimental adjacency-power features are specified separately. Appendix M states the finite-collection realization result. The proof relies on two forms of unfolding: named unfoldings answer queries within one graph, and anonymous colored unfoldings characterize root colors (Appendices B–C).

4 FROM LOCAL REFINEMENT TO GLOBAL EXPRESSIVITY

Theorem 3 (Main expressivity theorem)**.** *For finite simple undirected unlabelled graphs $G, H \in \mathcal{C}_{\mathrm{SS}}$,*

$$G \equiv_{\mathrm{St}} H \quad \Longleftrightarrow \quad G \cong H \quad \Longleftrightarrow \quad G \equiv_{\mathrm{D}} H. \tag{8}$$

Consequently, within $\mathcal{C}_{\mathrm{SS}}$, STITCH2-FWL *preserves global 3-WL graph expressivity in the convention of Section 3.*

The theorem compares two members of $\mathcal{C}_{\mathrm{SS}}$. It does not identify a qualifying graph against arbitrary outside-class opponents, nor does it equate the sparse and dense pair partitions.

Why local maps compose. A certificate message identifies a local unit through the deletion process of Appendix E. Two corresponding messages have equal entry colors and equal ordered source colors, as required by (17).

Lemma 4 (Local decoding and alignment)**.** *For $G, H \in \mathcal{C}_{\mathrm{SS}}$, corresponding stable-color certificate messages at an entry pair select local R-configurations with the same dense comparison type and the same stable ordered-entry color. There is a bijection between their vertices preserving all input colors in* (5) *and the prescribed ordered entry.*

The proof proceeds from a same-colored diagonal root. Sparse messages first recover the local colored comparisons through deletion certificates with at most two boundary vertices (Appendix E). For an R-skeleton, separability realizes the resulting relation correspondence as a vertex bijection, and Schurianity adjusts that map to fix the ordered parent interface. For an S-skeleton, a directed cycle scan deletes the predecessor's message and follows the remaining direction, which yields a map of the whole cycle.

At a vertex, an incidence forest groups the incident blocks by their rooted local types. At a two-vertex interface, the construction removes the already matched source instance and assigns the remaining branches with their multiplicities. Every new bag meets the matched prefix only at its parent interface, where the maps agree, and every new vertex receives its cut-vertex responsibility exactly once while each P-family is dispatched once. The finite task tree then guarantees termination and covers the component (Appendix F). A legal next step therefore follows from the current invariant; no future branch isomorphism is assumed in advance.

Lemma 5 (Single-step stitching)**.** *Let $G, H \in \mathcal{C}_{\mathrm{SS}}$ and start from equal diagonal root colors. In a state of the responsibility procedure in Appendix F, suppose a connected prefix of fully matched decomposition bags has been matched by color-preserving local bijections that agree on their interfaces. Assume the remaining tasks satisfy its responsibility invariant, including disjoint interiors, once-only branch assignments, and the source and parent-interface records. Then one pending task can be matched and committed while preserving these properties. Every new vertex receives its block–cut responsibility once, and every P-interface family is dispatched once.*

The reverse direction must account for the information the selector itself encodes. Using the color-defined decomposition of Kiefer & Neuen (2022), Appendix G shows that dense stable colors determine the predicates that select each sparse message. Hence on arbitrary finite inputs,

$$G \equiv_{\mathrm{D}} H \quad \Longrightarrow \quad \widehat{\mathrm{Sig}}(G) = \widehat{\mathrm{Sig}}(H). \tag{9}$$

Proof of Theorem 3. Equal Stitch signatures contain equal diagonal-color multisets. Lemmas 4 and 5, proved in Appendices E–F, construct an isomorphism of components rooted at corresponding diagonal colors. Subtracting each matched component's diagonal-color multiset and repeating, the component maps assemble into a graph isomorphism, which also covers isolated vertices and the empty graph. Isomorphism implies dense indistinguishability, and (9) closes the equivalence. □

Corollary 6 (No rigid skeletons)**.** *All graphs without R-skeletons lie in $\mathcal{C}_{\mathrm{SS}}$. For any two such graphs,* STITCH2-FWL *and dense refinement distinguish exactly the non-isomorphic pairs. This includes all finite graphs of treewidth at most two.*

Small and large rigid skeletons. Let E be the computational premise that every coherent configuration on at most thirteen points is separable and Schurian, supported by the published enumeration (Ziv-Av, 2018, Table 1 and Sections 2.2–2.3). Under E every graph whose R-skeletons have at most thirteen vertices lies in $\mathcal{C}_{\mathrm{SS}}$. The general theorem depends on neither E nor any bound on R-size: a local closure with singleton diagonal classes, for instance, is discrete and qualifies at any size. Appendix O collects further sufficient regimes and explains why exceeding thirteen vertices does not by itself call for Plus.

5 STRUCTURAL COVERAGE

We audit the size criterion against existing decomposition logs, using the per-graph maximum

$$q(G) = \max_{R \in \mathcal{R}(G)} |V(R)|, \qquad \max \varnothing = 0.$$

A graph passes only when *every* R has order at most thirteen. Under E this is sufficient for membership in $\mathcal{C}_{\mathrm{SS}}$, while no-R graphs qualify unconditionally; a graph above the threshold may still meet the general configuration conditions.

Table 1: **Graph-level structural coverage.** The $q \leq 13$ class includes no-R graphs and assumes E for the others. Rounded 100.00% entries need not mean all graphs qualify; Appendix K gives exact counts and source records.

Log	Graphs	No-R	$q \leq 13$	Max. R
ZINC	12,000	99.58%	100.00%	7
QM9	130,831	90.40%	100.00%	8
ZINC-full	249,456	99.51%	100.00%	9
PCQM4Mv2	3,452,151	99.51%	100.00%	12
Peptides-func	15,535	94.94%	100.00%	8
molhiv	41,127	95.87%	99.98%	22
molpcba	437,929	98.85%	100.00%	14
ENZYMES	600	0.67%	73.33%	57

Every recorded graph in ZINC, QM9, ZINC-full, PCQM4Mv2, and Peptides-func (Irwin et al., 2012; Ramakrishnan et al., 2014; Hu et al., 2020; Dwivedi et al., 2022) passes; molpcba (Wu et al., 2018) fails on one graph of R order fourteen, molhiv on eight of 41,127, and QM9 alone contributes 12,562 R-skeleton graphs, so coverage reaches well beyond the no-R case. ENZYMES (Borgwardt et al., 2005; Dobson & Doig, 2003) is the only collection with a substantial shortfall: 4 of its 600 graphs have no R, 440 pass the size test, and the remaining 160 are uncertified by size rather than demonstrated failures. These counts describe logged unlabelled topologies and certify neither omitted dataset versions nor arbitrary feature initializations nor outside-class comparisons. Appendix K documents the 25 reported collections together with their overlap and the parsing checks.

6 Experiments

We evaluate learned prediction, reported runtime, and exact graph distinctions as three separate lines of evidence. For inputs outside $\mathcal{C}_{\mathrm{SS}}$, "Stitch" denotes the auxiliary algorithm with the same selector, without any assumption of qualification. [1]

6.1 Prediction under the DA protocol

Stitch and DA-PPGN share the dataset-specific backbone, features, optimization, splits, and training protocol; only the retained interaction domain differs (Anonymous Authors, 2026). ZINC uses normalized adjacency powers through $q = 22$ and ten layers of width 80, while TU uses $q = 18$ and four layers of width 64. We evaluate two ZINC regression settings and four TU classification datasets (Dwivedi et al., 2023; Morris et al., 2020a). Appendix Q lists the settings, the per-seed results, and the reproducibility limits.

Table 2: **Prediction under the DA protocol.** Mean $\pm$ SD; DA values are from Anonymous Authors (2026). TU reports ten-fold accuracy (%); Stitch ZINC-full uses five seeds. ZINC-full is DA's ZINC-250k setting.

Dataset	Metric	DA (reported)	Stitch
ZINC-12k	Test MAE $\downarrow$	0.050 ± 0.002	0.05032 ± 0.00139
ZINC-full	Test MAE $\downarrow$	0.018 ± 0.002	0.01906 ± 0.00248
NCI1	Accuracy (%) $\uparrow$	85.18 ± 0.97	84.40 ± 1.28
NCI109	Accuracy (%) $\uparrow$	84.08 ± 1.53	83.01 ± 1.13
FRANKENSTEIN	Accuracy (%) $\uparrow$	77.27 ± 1.34	76.94 ± 1.37
ENZYMES	Accuracy (%) $\uparrow$	60.83 ± 5.64	53.00 ± 4.76

[1] Code available at `https://anonymous.4open.science/r/Stitch/`.

Stitch's ZINC MAEs are close to the reported DA values, whereas its TU mean accuracies fall below them by 0.78, 1.07, 0.33, and 7.83 percentage points on NCI1, NCI109, FRANKENSTEIN, and ENZYMES. The effect is task-dependent, and these descriptive comparisons do not establish statistical non-inferiority.

6.2 Reported runtime

Table 3 separates refinement time from the total, which also includes preprocessing. Stitch reports a lower refinement time than Co-Sp on every listed dataset and stays close to DA at the displayed precision. Preprocessing reverses the picture: the total is lower on the four TU datasets but higher on ZINC and QM9. Because the aggregation unit and the measurement protocol of these timings are still to be documented, we read them as preliminary records rather than as per-epoch training times or a reproducible hardware speedup (Appendix R).

Table 3: **Reported runtime, in seconds.** Total includes preprocessing. Values retain the supplied precision; measurement scope and full component timings are discussed in Appendix R.

	Refinement (s)			Total (s)		
Dataset	Co-Sp	DA	Stitch	Co-Sp	DA	Stitch
FRANKENSTEIN	0.20	0.04	0.04	0.30	0.28	0.28
NCI1	0.52	0.08	0.07	0.76	0.48	0.46
NCI109	0.52	0.08	0.07	0.78	0.48	0.46
ENZYMES	0.92	0.24	0.20	1.43	0.89	0.82
ZINC-12k	0.22	0.05	0.04	0.33	0.34	0.32
ZINC-full	0.22	0.05	0.05	0.33	0.35	0.35
QM9	0.11	0.03	0.03	0.18	0.27	0.27

6.3 Exact refinement: savings and retained distinctions

We compare Dense, Co-Sparsify (Co-Sp), DA, and Stitch under equality/true-adjacency initialization, a common color encoding within each comparison, and exact same-pivot multisets. Each method runs until its joint partition stabilizes, and interaction totals count both inputs including the final no-split update; they therefore measure selected work, not runtime. Appendix P gives the complete protocol and results.

Long cycles. Among all connected graphs on four to eight vertices we select 144 three-connected core pairs that 1-WL cannot distinguish but Dense can, chosen independently of any Stitch outcome. Replacing each edge by a path with $q \in \{0, 1, 2, 4, 8, 16, 32\}$ internal vertices yields 1,008 cases, and every method distinguishes all of them. At $q = 32$ Stitch uses 88.44% fewer interactions per round than DA. It also needs 20–22 rounds instead of 6–7, yet still cuts cumulative interactions by 60.98%, and the cumulative savings hold in each of the 720 cases with $q \geq 2$. Figure 3 and Table 10 give the full curves and counts.

Small graphs and BREC. All four methods distinguish the 332 non-isomorphic, 1-WL-indistinguishable pairs of the four-to-eight-vertex enumeration. On all 400 official BREC pairs (Wang & Zhang, 2024), Stitch matches Dense exactly: both distinguish 270 pairs and leave 130 indistinguishable, for 100.00% decision agreement and a 67.50% distinguishing rate (not a neural accuracy). Agreement persists on the 96 BREC pairs whose two inputs each admit strictly fewer Stitch interactions; both methods distinguish 72 of them. The aggregate additional reduction over DA on BREC is only 0.15%, which confirms that the large savings come from long S-skeletons. All 12,800 relabelling and 12,800 reliability-control groups pass for every method; Appendix P breaks them down by stratum.

7 A boundary example and Stitch-Plus

Complete computation inside R-skeletons does not suffice on arbitrary inputs. Subdividing every edge of the 16-vertex rook and Shrikhande graphs once produces two 64-vertex graphs that auxiliary

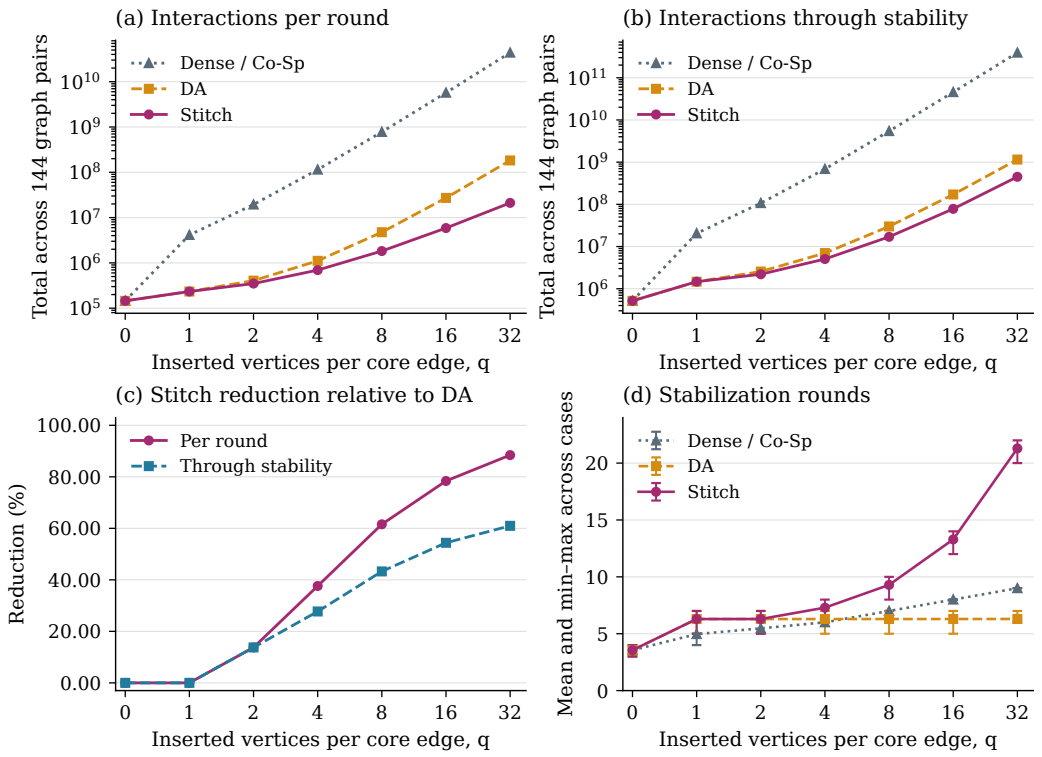

Figure 3: **The cycle reduction remains beneficial through stabilization.** Each point aggregates both graphs in all 144 core pairs at the indicated q. Panels (a–b) show exact interaction totals on logarithmic vertical scales; (c) shows reductions relative to DA; (d) shows mean stabilization rounds with the minimum–maximum case range, not a confidence interval. Horizontal axes are linear from zero to one and logarithmic thereafter. All four methods distinguish all 1,008 expanded pairs. The cumulative curve includes each method's own stabilization depth, rather than imposing an equal number of rounds.

Table 4: **Exact graph-level decisions.** Entries in the first two columns count distinguished pairs; agreement includes both positive and negative Dense decisions. The strict-reduction subset is selected solely by both inputs having fewer Stitch than DA interactions. The counts are deterministic refinement results, not neural accuracies.

Method	Small graphs distinguished	BREC distinguished	BREC agreement	Strict subset agreement
Dense 2-FWL	332/332	270/400	400/400	96/96
Co-Sp	332/332	270/400	400/400	96/96
DA	332/332	270/400	400/400	96/96
Stitch	332/332	270/400	400/400	96/96

Stitch cannot distinguish although dense 2-FWL can (Proposition 22). Dense refinement compares subdivision vertices near different interfaces and detects a four-clique witness that is present in the rook graph and absent from the Shrikhande graph. The induced R-configurations then fail separability, so the pair falls outside $\mathcal{C}_{\mathrm{SS}}$. Appendices I–J supply the exact counts and the proof.

STITCH2-FWL PLUS recovers this loss by expanding each R-skeleton one decomposition hop. Its window H_R contains the neighboring R-skeletons in full together with the two entry buffers of each neighboring S-skeleton, thereby allowing adjacency through a P-node. Plus retains $V(H_R)^3$ and adds endpoint pivots to nonadjacent S-roots. The expansion is applied once and never crosses a block (Appendix L). On the example H_R covers the whole graph, so Plus coincides with dense refinement and restores the distinction. This repairs the exhibited failure; general equivalence for overlapping one-hop windows remains open.

8 RELATED WORK AND LIMITATIONS

WL analyses connect message passing and higher-order states to graph distinctions (Xu et al., 2019; Morris et al., 2019; Maron et al., 2019; Morris et al., 2023). RFGNN studies redundancy-free path propagation (Chen et al., 2022), and enhanced subgraph learning studies connectivity, spectral, and distance encodings for 2-FWL GNNs (Chen et al., 2025). Local WL, subgraph architectures, and counting methods offer further ways to control computation (Morris et al., 2020b; Feng et al., 2023; Bevilacqua et al., 2022; Paolino et al., 2024). Stitch differs in specifying which common-pivot interactions to retain and when their graph distinctions survive.

Its closest sparsification predecessors are Co-Sparsify and DA (Chen et al., 2026; Anonymous Authors, 2026), while PlanE already composes block–cut and SPQR representations for planar graphs (Dimitrov et al., 2023). Both predecessors already route computation through separator pairs, so the added ingredient is a two-way comparison: the sparse refinement is not only smaller than the dense one but also reconstructible from it, which is what makes the two agree on graph signatures. Stitch adds a quadratic S selector and a preservation argument built on colored local configurations and interface alignment; Appendix N gives the detailed comparison.

The guarantee requires both inputs to lie in $\mathcal{C}_{\mathrm{SS}}$, and the thirteen-vertex certificate depends on E. Large R-skeletons keep their cubic computation, and certifying general membership can be expensive. Neither structural coverage nor interaction counts guarantee prediction accuracy or a proportional hardware saving, the preliminary timing records still need a complete protocol, and one-hop Plus equivalence remains unresolved. The theorem is stated for finite simple unlabelled graphs, so attributed, directed, and weighted inputs would each need a separate statement, and the coverage audit concerns underlying topologies rather than feature labelings of them.

9 CONCLUSION

Stitch replaces uniform higher-order computation with a structure-dependent interaction rule. Its cycle reduction preserves dense graph distinctions on an explicit class that admits unbounded graph size and decomposition depth and covers many recorded molecular inputs. What makes the reduction safe is not the decomposition but the alignment it induces at interfaces: sparse messages recover local colored comparisons, and separability with Schurianity turns them into maps that agree wherever two pieces meet, so the local maps compose into a global isomorphism. The reverse direction follows the same way, since dense colors determine the message predicates.

A graph qualifies whenever each of its R-skeletons is small or its local closure is discrete, and a single size check certifies this for every graph in the molecular collections we audited. Exact refinement, prediction, and timing results show both the value of the allocation and its limits: savings concentrate on long S-skeletons, prediction stays close on ZINC while varying with the task on TU, and the timings remain preliminary. Two questions stay open: certifying general $\mathcal{C}_{\mathrm{SS}}$ membership without enumerating configurations, and deciding whether the one-hop Plus variant, which repairs the exhibited boundary failure, matches dense refinement on overlapping windows. Because the composition argument uses interfaces of size at most two, the scheme may extend to other bounded-adhesion decompositions and to attributed or directed inputs.

AI USE DISCLOSURE

We used generative AI tools to assist in drafting sections of this paper. The authors take responsibility for the final content of the paper, including the accuracy and originality of all text and claims produced with AI assistance.

REFERENCES

Anonymous Authors. Decomposition-adaptive 2-FWL: Efficient graph representation learning via cut vertices and separation pairs. Manuscript under review, 2026.

László Babai. Graph isomorphism in quasipolynomial time. *arXiv preprint arXiv:1512.03547*, 2016.

P. Barceló, Egor V. Kostylev, Mikaël Monet, Jorge Pérez, Juan L. Reutter, and J. Silva. Logical Expressiveness of Graph Neural Networks. *International Conference on Learning Representations*, 2019.

Maciej Besta, Florian Scheidl, Lukas Gianinazzi, Grzegorz Kwasniewski, S. Klaiman, Jürgen Müller, and Torsten Hoefler. Demystifying Higher-Order Graph Neural Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(3):2544–2565, 2026.

Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M. Bronstein, and Haggai Maron. Equivariant Subgraph Aggregation Networks. In *International Conference on Learning Representations*, 2022.

Mitchell Black, Zhengchao Wan, Gal Mishne, Amir Nayyeri, and Yusu Wang. Comparing graph transformers via positional encodings. In *International Conference on Machine Learning*, pp. 4103–4139, 2024.

Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alex J. Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl 1):i47–i56, 2005. doi: 10.1093/bioinformatics/bti1007.

Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M. Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45:657–668, 2023. doi: 10.1109/TPAMI.2022.3154319.

R. Chen, F. Mo, P. L. Ip, S. Zhang, D. Wu, Y. Li, and L. H. U. Connectivity-guided sparsification of 2-FWL GNNs: Preserving full expressivity with improved efficiency. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

Rongqin Chen, Shenghui Zhang, Leong Hou U, and Ye Li. Redundancy-free message passing for graph neural networks. In *Advances in Neural Information Processing Systems*, volume 35, pp. 4316–4327, 2022.

Rongqin Chen, Yan Li, Dan Wu, Fan Mo, Shenghui Zhang, Pak Lon Ip, Hoi Cheong Iam, Ye Li, and Leong Hou U. Enhanced subgraph learning in 2-FWL GNNs via local connectivity, spectral, and distance encodings. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pp. 227–238, 2025. doi: 10.1145/3711896.3736933.

Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can Graph Neural Networks Count Substructures? In *Advances in Neural Information Processing Systems*, 2020.

A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, 34(2):786–797, 1991. doi: 10.1021/jm00106a046.

Holger Dell, Martin Grohe, and Gaurav Rattan. Lovász meets Weisfeiler and Leman. In *45th International Colloquium on Automata, Languages, and Programming*, volume 107, pp. 40:1–40:14, 2018.

Giuseppe Di Battista and Roberto Tamassia. On-line maintenance of triconnected components with SPQR-trees. *Algorithmica*, 15(4):302–318, 1996. doi: 10.1007/BF01961541.

Radoslav Dimitrov, Zeyang Zhao, Ralph Abboud, and İsmail İlkan Ceylan. PlanE: Representation Learning over Planar Graphs. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Paul D. Dobson and Andrew J. Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of Molecular Biology*, 330(4):771–783, 2003. doi: 10.1016/S0022-2836(03)00628-4.

Vijay Prakash Dwivedi, Ladislav Rampášek, Mikhail Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. In *Advances in Neural Information Processing Systems*, volume 35, pp. 22326–22340, 2022.

Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24 (43):1–48, 2023.

Sergei Evdokimov and Ilia Ponomarenko. Separability number and schurity number of coherent configurations. *The Electronic Journal of Combinatorics*, 7(1), 2000. Research Paper 31.

Jiarui Feng, Lecheng Kong, Hao Liu, Dacheng Tao, Fuhai Li, Muhan Zhang, and Yixin Chen. Extending the Design Space of Graph Neural Networks by Rethinking Folklore Weisfeiler-Lehman. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Simon D. Fink and Ignaz Rutter. Maintaining triconnected components under node expansion. *arXiv preprint arXiv:2301.03972*, 2023. Version 1, Section 3 and Definition 3.

Fabrizio Frasca, Beatrice Bevilacqua, Michael M. Bronstein, and Haggai Maron. Understanding and Extending Subgraph GNNs by Rethinking Their Symmetries. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

F. Fuhlbrück, J. Köbler, and O. Verbitsky. Identifiability of graphs with small color classes by the Weisfeiler–Leman algorithm. *SIAM Journal on Discrete Mathematics*, 35(3):1792–1853, 2021. doi: 10.1137/20M1327550.

Martin Fürer. On the power of combinatorial and spectral invariants. *Linear Algebra and its Applications*, 432(9):2373–2380, 2010.

Floris Geerts and Juan L. Reutter. Expressiveness and approximation properties of graph neural networks. In *International Conference on Learning Representations*, 2022.

J. Gilmer, S. Schoenholz, Patrick F. Riley, O. Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. In *International Conference on Machine Learning*, pp. 1263–1272, 2017.

Martin Grohe. The Logic of Graph Neural Networks. In *Logic in Computer Science*, pp. 1–17, 2021.

Carsten Gutwenger and Petra Mutzel. A linear time implementation of SPQR-trees. In *International Symposium on Graph Drawing*, volume 2265 of *Lecture Notes in Computer Science*, pp. 77–90. Springer, 2001. doi: 10.1007/3-540-45848-4_7.

Walter Holberg. The decomposition of graphs into k-connected components. *Discrete Mathematics*, 109(1-3):133–145, 1992.

Jacob Holm, Giuseppe F. Italiano, Adam Karczmarz, Jakub Łącki, and Eva Rotenberg. Decremental SPQR-trees for planar graphs. In *26th Annual European Symposium on Algorithms*, volume 112 of *Leibniz International Proceedings in Informatics*, pp. 46:1–46:16, 2018. doi: 10.4230/LIPIcs.ESA.2018.46.

John Hopcroft and Robert Tarjan. Algorithm 447: Efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6):372–378, 1973. doi: 10.1145/362248.362272.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems*, volume 33, pp. 22118–22133, 2020.

Yinan Huang, Xingang Peng, Jianzhu Ma, and Muhan Zhang. Boosting the Cycle Counting Power of Graph Neural Networks with I^2-GNNs. In *International Conference on Learning Representations*, 2023.

John J. Irwin, Teague Sterling, Michael M. Mysinger, Erin S. Bolstad, and Ryan G. Coleman. ZINC: A free tool to discover chemistry for biology. *Journal of Chemical Information and Modeling*, 52 (7):1757–1768, 2012. doi: 10.1021/ci3001277.

Charilaos I. Kanatsoulis and Alejandro Ribeiro. Counting Graph Substructures with Graph Neural Networks. In *International Conference on Learning Representations*, 2024.

Sandra Kiefer and Daniel Neuen. The power of the Weisfeiler–Leman algorithm to decompose graphs. *SIAM Journal on Discrete Mathematics*, 36(1):252–298, 2022. doi: 10.1137/20M1314987.

Sandra Kiefer, Ilia Ponomarenko, and Pascal Schweitzer. The Weisfeiler–Leman dimension of planar graphs is at most 3. *Journal of the ACM*, 66(6), 2019. doi: 10.1145/3333003. Article 44.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Liheng Ma, Chen Lin, Derek Lim, Adriana Romero-Soriano, Puneet K. Dokania, Mark Coates, Philip Torr, and Ser-Nam Lim. Graph inductive biases in transformers without message passing. In *International Conference on Machine Learning*, pp. 23321–23337, 2023.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In *Advances in Neural Information Processing Systems*, 2019.

Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60:94–112, 2014. doi: 10.1016/j.jsc.2013.09.003.

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019. doi: 10.1609/aaai.v33i01.33014602.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. TUDataset: A collection of benchmark datasets for learning with graphs, 2020a.

Christopher Morris, Gaurav Rattan, and Petra Mutzel. Weisfeiler and Leman Go Sparse: Towards Scalable Higher-Order Graph Embeddings. In *Advances in Neural Information Processing Systems*, volume 33, 2020b.

Christopher Morris, Gaurav Rattan, Sandra Kiefer, and Siamak Ravanbakhsh. SpeqNets: Sparsity-aware Permutation-equivariant Graph Networks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16017–16042, 2022.

Christopher Morris, Yaron Lipman, Haggai Maron, Bastian Rieck, Nils M. Kriege, Martin Grohe, Matthias Fey, and Karsten Borgwardt. Weisfeiler and Leman Go Machine Learning: The Story So Far. *Journal of Machine Learning Research*, 24(333):1–59, 2023.

Raffaele Paolino, Sohir Maskey, Pascal Welke, and Gitta Kutyniok. Weisfeiler and Leman Go Loopy: A New Hierarchy for Graph Representational Learning. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Pál András Papp and Roger Wattenhofer. A theoretical comparison of graph neural network extensions. In *International Conference on Machine Learning*, pp. 17323–17345, 2022.

Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1:140022, 2014. doi: 10.1038/sdata.2014.22.

Yanbo Wang and Muhan Zhang. An empirical study of realized GNN expressiveness. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, 2024.

Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: A benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018. doi: 10.1039/C7SC02664A.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*, 2019.

Zuoyu Yan, Junru Zhou, Liangcai Gao, Zhi Tang, and Muhan Zhang. An Efficient Subgraph GNN with Provable Substructure Counting Power. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3702–3713, 2024. doi: 10.1145/3637528.3671731.

DingYi Zeng, Wanlong Liu, Wenyu Chen, Li Zhou, Malu Zhang, and Hong Qu. Substructure Aware Graph Neural Networks. In *AAAI Conference on Artificial Intelligence*, volume 37, pp. 11129–11137. Association for the Advancement of Artificial Intelligence (AAAI), 2023.

Bingxue Zhang, Changjun Fan, Shixuan Liu, Kuihua Huang, Xiang Zhao, Jincai Huang, and Zhong Liu. The Expressive Power of Graph Neural Networks: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 37(3):1455–1474, 2023a.

Bohang Zhang, Guhao Feng, Yiheng Du, Di He, and Liwei Wang. A Complete Expressiveness Hierarchy for Subgraph GNNs via Subgraph Weisfeiler-Lehman Tests. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 41019–41077, 2023b.

Bohang Zhang, Shengjie Luo, Liwei Wang, and Di He. Rethinking the Expressive Power of GNNs via Graph Biconnectivity. In *International Conference on Learning Representations*, 2023c.

Muhan Zhang and Pan Li. Nested Graph Neural Networks. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Junru Zhou, Jiarui Feng, Xiyuan Wang, and Muhan Zhang. Distance-Restricted Folklore Weisfeiler-Leman GNNs with Provable Cycle Counting Power. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Wenhao Zhu, Tianyu Wen, Guojie Song, Liangji Wang, and Bo Zheng. On Structural Expressive Power of Graph Transformers. In *Knowledge Discovery and Data Mining*, pp. 3628–3637. ACM, 2023.

M. Ziv-Av. Enumeration of coherent configurations of order at most 15. *Acta Universitatis Matthiae Belii, series Mathematics*, 26:65–75, 2018.

A Decomposition objects and the responsibility structure

We specify the decomposition data that the selector uses and that named unfoldings recover. Table 5 also fixes the notation used throughout the proofs.

Table 5: Notation and the level of information it represents.

Symbol	Meaning
$\mathfrak{D}(G)$	Named block–cut and reduced SPQR data of a single graph
$\widehat{T}, \widehat{P}, \widehat{N}(a,b)$	Selected ordered triples, active pairs, and root-specific pivots
$\widehat{c}_i, c = \widehat{c}$	Round-i and common stable auxiliary Stitch colors
d	Common stable dense 2-FWL color
$U_G(q), \widehat{U}_G(q), A_G(q)$	Named, named colored, and anonymous colored unfoldings
$\mathrm{End}, E^+, G^\sharp$	Nonreflexive endpoint relation, its reflexive closure, and its simple auxiliary graph
X_U, B_U, r_U, τ	Certificate class, auxiliary surviving boundary, registration round, deletion time
q_i^U, F_i, Q_i	Local dense colors and their two simultaneous decoding functions
$A_R, \mathcal{X}_R = \mathrm{CC}(A_R)$	Actual colored R object and its coherent closure
$\mathcal{I}_x$	Port–unit incidence forest at a fixed vertex x
D, I, f	Committed vertex region, a task's parent interface, and committed vertex map

Names in $\mathfrak{D}(G)$ and U_G are identities within a single graph; they are never colors to be compared across inputs. The symbols B_U and I denote different boundaries: B_U comes from an auxiliary deletion proof, I from the actual traversal.

A.1 Blocks, cut vertices, and reconstruction

In a connected graph, a cut vertex is a vertex whose deletion disconnects the remaining graph, and a bridge is an edge whose deletion disconnects the graph. Our non-bridge blocks are the maximal subgraphs that stay connected after deletion of any one vertex. Bridge blocks are included so that every true edge has an owner. The block–cut incidence graph has one node per block and one node per cut vertex, with incidence meaning membership.

Lemma 7 (Block–cut structure)**.** *For a connected graph containing an edge, this incidence graph is a tree. Distinct blocks meet in at most one vertex, which is a cut vertex when shared. Every true edge belongs to exactly one block. The branches at a cut node x, with x removed from their vertex sets, are exactly the connected components of $G - x$.*

Proof. Induct on the number of vertices. If the graph has no cut vertex, then one of order at least three is itself a single non-bridge block, and the two-vertex case is a single bridge block. Otherwise choose a cut vertex x and let $C_1, \dots, C_k$ be the components of $G - x$, writing $G_j = G[C_j \cup \{x\}]$. A biconnected subgraph cannot meet two different C_j, since deleting x would disconnect it, and a path leaving G_j must return through x, so it creates no additional bypass for an edge inside G_j. Both non-bridge and bridge blocks are therefore exactly those obtained in the smaller graphs G_j.

The vertex x is not a cut vertex of any G_j and, by induction, lies in exactly one block there. Joining the incidence trees of the G_j by a new cut node x incident to those blocks yields a tree with the claimed intersections and edge partition. Removing x from this tree leaves one branch per C_j; inside a branch the blocks stay connected after deleting x and meet only through their other cut vertices, so each branch recovers exactly one component. □

To reconstruct the graph, take copies of the blocks and identify the copies of each cut vertex specified by an incidence. The lemma ensures that these are precisely the repeated vertices and that every true edge is retained once, and the identification relation, hence the result, does not depend on the gluing order. For a disconnected graph, apply the construction componentwise and treat isolated vertices as singleton components.

In Figure 2 the blue block is a K_4 with the edge ab removed, together with two further a–b paths a, u, b and a, v, w, b; the other two blocks are a triangle and a bridge, both attached at a. Inside the blue block the P-bond carries three distinct virtual instances on the same poles, each paired with the

virtual ab of one adjacent S/R. Deleting all paired virtual edges reconstructs the block without adding ab to the original graph.

A.2 THE PRECISE SPQR CONVENTION

For a non-bridge block B the data consist of a tree T_B together with a skeleton K_μ at every node μ. Skeleton vertices retain their identities in B, and edges are *instances*: parallel edges on the same poles may be distinct records. The representation has the following properties.

1. The skeleton vertex sets cover $V(B)$, and their real-edge sets partition $E(B)$.
2. Each virtual edge has exactly one twin in an adjacent skeleton. The twins have the same poles $\{a, b\}$, and adjacent skeletons intersect exactly in those two vertices. The poles separate two nonempty regions of the original block.
3. For every original vertex v, its carrier $\mathrm{Car}_B(v) = \{\mu : v \in V(K_\mu)\}$ is connected.
4. Each skeleton is an S-cycle of length at least three, a P-bond with two poles and at least three edge instances, or a simple triconnected R of order at least four.
5. There are no S–S or P–P tree edges.

Q leaves are suppressed. In a simple input a P has at most one real edge, and its number of tree neighbors equals its number of virtual edges rather than its total number of edges. Real or virtual status is local to a skeleton: a pole pair can be a virtual edge of an R and at the same time an original edge owned by an adjacent P. R-connectivity refers to the full skeleton, virtual edges included.

Existence and uniqueness of the reduced representation are the standard SPQR structure theorem (Fink & Rutter, 2023, Section 3, Definition 3). Uniqueness holds up to an isomorphism that preserves original vertex and real-edge identities and virtual-edge pairing; it prescribes neither numerical node names nor a root. Appendix C gives the explicit finite split-and-reduce procedure behind our named recovery argument.

Lemma 8 (Carriers, interfaces, and gluing)**.** *Vertex carriers, and every nonempty intersection of vertex carriers, are subtrees. Removing a tree edge with poles $I = \{a, b\}$ separates the unions of its two sides into regions that intersect exactly in I, with no true edge between their interiors. Gluing all paired virtual edges recovers B. Three vertices whose carriers intersect pairwise belong to a common skeleton. A pair nonadjacent in an S-skeleton belongs to no other skeleton.*

Proof. Connectedness follows from the carrier axiom, and the intersection of subtrees is connected whenever it is nonempty. A vertex lying on both sides of a deleted tree edge must occur at its two ends and is therefore a pole, and conversely both poles occur on both sides. If a true edge joined the two interiors, its unique owner would lie on one side and would place its other endpoint there as well, a contradiction. Along each twin pair we identify only the corresponding same-named endpoints and remove the two virtual edges. Connected carriers then identify all copies of each vertex, real-edge ownership keeps each original edge once, and the identifications commute, so the block is recovered.

Subtrees of a tree satisfy the Helly property: root the tree, choose among the nearest-to-root vertices of the subtrees one of maximum depth, and use pairwise intersection to see that it lies in every subtree. Applying this to three carriers yields the common skeleton. Finally, if a nonadjacent pair of an S belonged to another skeleton, the path to it would force that pair into the first adhesion, which is an S virtual edge, contradicting nonadjacency. □

An internal nonadjacent pair of an S separates the block: deleting it leaves the two open cycle arcs, and every off-cycle branch attaches along a single cycle edge. By Lemma 8 such a pair is not a tree adhesion, and the selector and the traversal must not interchange these two kinds of separator. At a *tree* adhesion with q incident S/R units and original-edge indicator ε, the case $q = 2$, $\varepsilon = 0$ gives a direct adjacency. Otherwise a single P-bond joins the q units and owns the original pole edge when $\varepsilon = 1$. Both statements follow by grouping the branches at the same poles: no S/R has two edges on that pair, and adjacent P-bonds would already have been reduced. In the direct case the two units cannot both be S.

A.3 Building the finite responsibility tree

The unfoldings introduced below may be infinite, so the construction of a graph isomorphism instead uses a finite tree of responsibilities. Fix a root vertex x of the component and create an auxiliary root CUT bag $\{x\}$, even when x is not an actual cut vertex, attaching its incident blocks. A bridge becomes a two-vertex BRIDGE bag owning its edge. On entering a non-bridge block through a vertex, choose an S/R skeleton containing that vertex and root the SPQR tree there; such a skeleton exists even when the vertex also occurs in a P, because every P is adjacent to S/R units. Retain the S, P, and R bags, their real edges, and their twin records.

The entry vertex's block–cut responsibility belongs to the parent. For every other vertex v that has blocks outside the current block, attach a single CUT bag $\{v\}$ at the node of its carrier nearest the chosen skeleton root; that node is unique because the carrier is connected. This CUT bag handles exactly the other incident blocks, each recursively rooted at v. Inside a unit, the CUT tasks of new vertices are processed before the non-parent SPQR outlets, and a P-bond dispatches its non-source branches once, so entering one child does not reopen its siblings.

Lemma 9 (Responsibility separation and coverage)**.** *This construction is a finite tree of bags with connected vertex carriers. Every non-root task has a parent interface of size one or two. For a task q, let V_q be the union of the bags below it and I_q its parent interface. Its interior $V_q \setminus I_q$ meets no bag outside that subtree. Sibling interiors are disjoint and have no true edges between them. Every vertex is covered and every true edge has exactly one owner.*

Proof. Replacing a block node of the block–cut tree by its SPQR tree, and attaching each cut incidence at one node of its carrier, preserves the tree property. Carriers are connected within a block and are joined across blocks through the single CUT bag, so they remain connected in the combined tree. A vertex occurring both below a task and outside it must therefore lie on the parent edge, hence in I_q. If a true edge crossed interiors, the bag owning it would force an interior endpoint to occur outside its subtree; the same argument excludes intersections between sibling interiors. Coverage and edge ownership then follow from the two decomposition levels. □

When the committed bags form a connected prefix, the nearest unprocessed bag meets it only in its parent interface. The identity of the first introducing bag is bookkeeping within one graph, not a color, and the cross-graph existence of the next map is proved later through local types and source subtraction. The responsibility lemma supplies separation and termination, not that existence statement.

A.4 Tree decompositions, torsos, and the no-R subclass

A tree decomposition (T, β) assigns a vertex bag $\beta(t)$ to each tree node, covers every vertex and edge, and requires the bags containing any fixed vertex to form a connected subtree. Its adhesion is the largest intersection of two adjacent bags, and its width is $\max_t |\beta(t)| - 1$; the treewidth $\mathrm{tw}(G)$ is the minimum width over all such decompositions. The torso of a vertex set X is the simple graph on X obtained from $G[X]$ by making the neighbors in X of each component of $G - X$ a clique, so a torso edge may represent a connection through an outside region without being an original edge. Appendix G interprets the color-defined tree decomposition through these definitions and then normalizes it to our reduced skeleton convention.

When no R occurs, triangulate each S-cycle into bags of size at most three. Every cycle edge, virtual ones included, lies in such a bag, so the resulting decompositions glue along the P pole bags and cut bags into a tree decomposition of width at most two. Conversely, each R is a minor of its block: realize every virtual edge by a path on its outside branch and contract those paths. An R has minimum degree at least three, whereas every graph of treewidth at most two is 2-degenerate and treewidth is minor-monotone, so a graph of treewidth at most two cannot contain an R-skeleton. These standard tree-decomposition facts also follow directly by removing leaf bags and contracting edges in a tree decomposition.

B UNFOLDINGS, COLORS, AND LEGAL MESSAGE MATCHING

The construction uses two forms of unfolding: named unfoldings support queries within a single graph, and anonymous colored unfoldings support comparisons across graphs. Every statement in this appendix concerns the original auxiliary selector, whose colors receive no vertex identifiers or decomposition roles.

B.1 NAMED AND ANONYMOUS OCCURRENCE TREES

For an active root $q = (a, b)$, the *named unfolding* $U_G(q)$ alternates pair occurrences with message occurrences. A pair occurrence labelled (u, v) carries one message node for every $t \in \widehat{N}(u, v)$, and that message has ordered slots $L = (u, t)$ and $R = (t, v)$. Every non-self slot creates a fresh pair occurrence and is expanded recursively; if $t = v$ the left slot is marked SELF_L instead of creating the identical parent pair, and if $t = u$ the right slot is marked SELF_R. The message itself is always retained, including when both slots are SELF.

Only the direct repetition of the same *named ordered pair* is suppressed; repeated pairs in other branches or at greater depth remain distinct tree occurrences, and SELF is a slot attribute rather than a back edge. The unfolding of even a single graph edge can be infinite. Pair depth counts pair-to-message-to-pair steps, and a depth-h truncation expands only pair occurrences of depth less than h; a truncated leaf does not assert that its pivot set is empty.

The *named colored unfolding* $\widehat{U}_G(q)$ records the existing color sequence $(\widehat{c}_0(p), \widehat{c}_1(p), \ldots)$ at every occurrence of pair p and adds no name-dependent colors. To compare two finite graphs one may instead record their common stable color c at every pair; the old-color field then recovers all earlier colors, and stability makes equality of these stable labels the same as equality of the full sequences. When integer color identifiers are used operationally, the encoding table is retained, and an unknown hash value is not presumed invertible.

The *anonymous colored unfolding* $A_G(q)$ erases vertex, pair, and pivot names together with all cross-occurrence equalities induced by those names. It keeps the occurrence tree, the pair/message node types, the color labels, the L/R slots, the SELF marks, and one message occurrence per pivot. Messages at a pair carry no name-based order, so an isomorphism of these trees may permute messages but cannot combine the left source of one message with the right source of another; it need not map repeated occurrences of the same original vertex to a common original vertex.

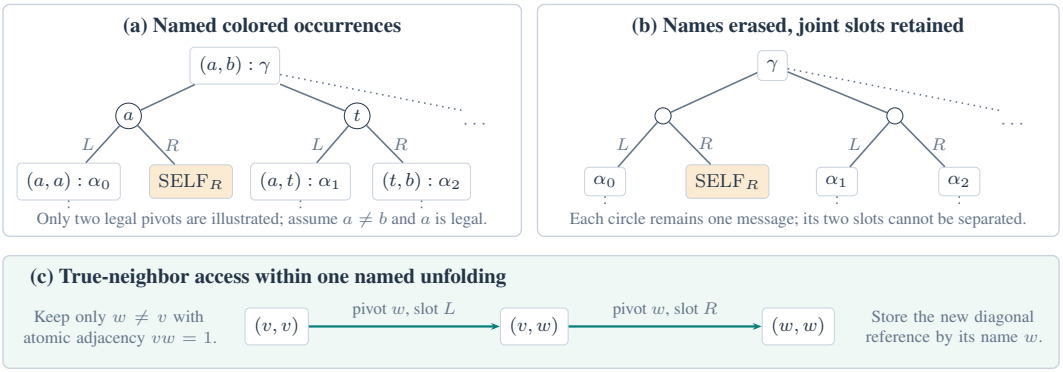

Figure 4: **What an unfolding preserves and what it forgets.** Panels (a,b) show two message occurrences at a non-diagonal root with a legal endpoint pivot a and a third vertex t. The direct repetition (a, b) becomes a SELF slot; other repeated occurrences remain separate. Anonymization erases graph names but retains colors, ordered slots, and one record per pivot. Dots indicate further messages or descendants. Panel (c) gives the named access path used to enumerate true neighbors and save their diagonal references. This is a query within one graph, not a claim that an anonymous tree isomorphism already defines a graph isomorphism.

B.2 Exact correspondence with refinement

Lemma 10 (SELF and message restoration)**.** *Given parent and source colors at the same round, the atomic equality bits determine the SELF pattern. The parent color fills a SELF slot; the non-SELF child colors fill the other slots. This restores the original joint multiset exactly, including multiplicities.*

Proof. At a non-diagonal root, the left slot is SELF exactly when the right source is diagonal, and the right slot is SELF exactly when the left source is diagonal. At a diagonal root, both slots are SELF exactly when the sources are diagonal and otherwise neither is SELF. Every message record contributes once, so restoration neither removes a pivot nor merges two records. □

For a finite round r, take the depth-r truncation and label a pair at depth j by $\widehat{c}_{r-j}$. Write $A_{G,r}(q)$ for its anonymous version and $A_{G,\mathrm{st}}(q)$ for the complete stable-labelled tree.

Lemma 11 (Color–unfolding correspondence)**.** *For active roots in two graphs compared with a common encoding,*

$$\widehat{c}_r^{\,G}(q) = \widehat{c}_r^{\,H}(q') \iff A_{G,r}(q) \cong A_{H,r}(q').$$

At a common stable round,

$$c_G(q) = c_H(q') \iff A_{G,\mathrm{st}}(q) \cong A_{H,\mathrm{st}}(q') \iff A_G(q) \cong A_H(q').$$

Proof. For finite r, induct on r, the case $r = 0$ being a single colored root. Root equality at round $r > 0$ gives equal previous root colors and equal multisets of ordered previous source-color pairs. Matching messages within each joint-type bucket, Lemma 10 preserves the SELF marks and the induction hypothesis matches the remaining subtree of each non-SELF child. Distinct occurrences have disjoint descendant trees, so these maps combine, and the converse follows from the root label. The assertion concerns colored unfoldings; it does not reconstruct colors from an uncolored shape.

At stability, equal root colors must have equal stable joint multisets, since otherwise the next round would split them. For each pair of same-colored named roots, which are finitely many, fix a bucketwise message bijection. Reusing these choices recursively at occurrences extends, rather than replaces, the preceding finite-depth correspondence, and their union is a root-preserving isomorphism of the complete, possibly infinite trees. Stable-label equality and full-sequence equality coincide, which gives the final equivalence. □

Tree depth is not interchangeable with refinement round: only finite-depth labels with decreasing rounds follow from equality at a finite nonstable round. The correspondence above is therefore not by itself a graph-isomorphism theorem.

B.3 Left-first expansion and exact subtraction

Lemma 12 (Bucket matching with fixed contributions)**.** *Let $c(a, b) = c(a', b')$ and define $\mu(t) = (c(a, t), c(t, b))$. The pivot sets then have equal cardinalities in every joint-type bucket. Any injective partial pivot matching that preserves μ extends to a full such matching, and for subsets S, S' of the two pivot sets a type-preserving bijection $S \to S'$ exists precisely when their cardinalities agree in every bucket.*

Proof. Bucket equality follows from stability and the injective update. A partial matching removes equally many elements from each corresponding bucket, so choosing arbitrary bijections on the finite remainders and taking their union completes it. Restricting the same argument to S, S' proves sufficiency, and necessity follows from type preservation. □

The left graph may therefore choose a named occurrence and enumerate all its messages, while the right graph follows with same-type messages; the two child roots remain same-colored and the process continues. This is a correspondence of occurrences, and its choices are not yet committed vertex images. Endpoint pivots are fixed by diagonal source bits *when they are legal*; in particular, a nonadjacent S-root does not automatically admit either endpoint pivot.

A restricted message row is justified in either of two ways: membership may be a common predicate of the root color and the joint source type, or the removed sets may already be known to contribute

equal joint multisets, in which case subtraction leaves equal bucket counts. The removed set is a union of actual pivot instances, so overlapping sources cannot be subtracted twice and removing one source instance does not remove its whole color class.

For example, after matching a pivot t at (a, b), the predecessor b may be held fixed in the row (a, t) whenever b is a legal pivot there on both sides. The new contribution $(c(a, b), c(b, t))$ is then equal by the previous match and transpose; this is the predecessor subtraction used in an S scan. For general sparse selectors, return-pivot legality must be checked rather than inferred from the first triple alone. For a whole source unit or block, equality of the removed contribution follows from the whole-unit maps and the incidence-forest argument proved later.

An implementation may store the same named pair once and reference it from many occurrences, but that storage compression neither identifies message slots nor erases multiplicities nor merges distinct branch responsibilities. Termination of the graph construction is proved on the responsibility tree of Lemma 9, not on the infinite unfolding.

C SIMULATING BLOCK–CUT AND SPQR RECOVERY ON A NAMED UNFOLDING

The input is a complete named colored unfolding of an active root, with access to all messages at every queried occurrence. *Complete* here means an on-demand query interface, not a pre-materialized infinite tree. The allowed operations are equality tests on names within this graph, reads of atomic colors and message slots, and storage of visited flags, occurrence references, edge stacks, and virtual-edge records; no second copy of the original adjacency list or a precomputed decomposition is supplied. The resulting DFS order is local to one graph, and cross-graph synchronization is established separately from anonymous colors.

C.1 AN EXACT NEIGHBOR INTERFACE

Lemma 13 (Diagonal access and true neighbors)**.** *From any active (a, b), a diagonal occurrence (a, a) is reachable within two pair steps. At a diagonal occurrence $d_v = (v, v)$,*

$$\mathrm{Nbr}(d_v) = \{w \neq v : w \in \widehat{N}(v, v),\ \mathrm{adj}_G(v, w) = 1\} = N_G(v),$$

where the adjacency bit is read from the atomic color of the left source (v, w). Every returned neighbor yields a non-SELF path $(v, v) \to (v, w) \to (w, w)$.

Proof. At a diagonal root there is nothing to do. If a is a legal pivot at (a, b), its left child is (a, a); otherwise the active pair can only come from a nonadjacent S-root, since bridge and R cubes admit endpoint pivots. An S-neighbor t of a yields (a, t), at which the pivot a yields (a, a). The first two pair levels can be searched by names without knowing that S in advance.

Every returned w is a true neighbor by the adjacency test. Conversely, a true edge vw owned by a bridge, S, or R admits the triples (v, w, v) and (v, w, w). If the edge is owned by a P, an adjacent S/R shares the same pole pair as a virtual edge and admits those triples too; such an adjacent unit exists because the P has at least three edge instances and at most one real edge. The first triple provides (v, w) at d_v and the second provides its right child (w, w), and both steps change the ordered pair, so neither is suppressed as SELF. □

The diagonal pivot set can also contain virtual neighbors and R-nonedges, so omitting the atomic adjacency filter would be incorrect. Running DFS through this interface, with visited status keyed by original vertex identity rather than occurrence identity, leaves every visited vertex with a saved diagonal reference. The search stays in the root component, visits each vertex once, and cannot miss a vertex, because the first edge of a path from visited to unvisited vertices would have been enumerated; it therefore terminates and covers exactly that component. For a component of order n, a finite unfolding prefix of pair depth $2n + 2$ suffices for these queries. This is an access-depth bound, not a refinement-round or materialized-tree-size bound.

C.2 BLOCK–CUT DFS, INCLUDING ITS OUTPUT CRITERION

Represent each true edge by its unordered endpoint pair, and maintain discovery numbers dfn, low values low, parent edges, a DFS stack, and a stack of traversed but unassigned edges. On first entering v, assign the next discovery number and set $\mathrm{low}(v) = \mathrm{dfn}(v)$. For each $w \in \mathrm{Nbr}(d_v)$:

1. Skip the parent edge. If w is new, push vw, recurse on w, and set $\mathrm{low}(v) \leftarrow \min(\mathrm{low}(v), \mathrm{low}(w))$. When $\mathrm{low}(w) \geq \mathrm{dfn}(v)$, pop through vw inclusive and output the popped edges as one block.
2. If w is an already visited strict ancestor, push vw and lower $\mathrm{low}(v)$ to $\mathrm{dfn}(w)$ if the latter is smaller. Ignore the reverse encounter of a previously processed edge.

A non-root v is a cut vertex exactly when some DFS child w has $\mathrm{low}(w) \geq \mathrm{dfn}(v)$, and the root is a cut vertex exactly when it has at least two DFS children. An output of a single edge is a bridge block, and an isolated vertex is recorded without an edge-stack operation.

Lemma 14 (Correct block recovery)**.** *The procedure outputs each block exactly once, with its vertices and true edges, and hence recovers the named block–cut tree.*

Proof. Fix the neighbor enumeration order. Lemma 13 makes every operation identical to the corresponding operation of DFS on the actual graph. In undirected DFS a non-tree edge joins an ancestor to a descendant, since an edge between two different child subtrees would have discovered the later subtree from the earlier one. Induction on completed calls therefore shows that $\mathrm{low}(w)$ is the smallest discovery number reachable from the subtree at w by descending tree edges and then at most one back edge. The inequality $\mathrm{low}(w) \geq \mathrm{dfn}(v)$ says exactly that this subtree has no bypass of v to a strict ancestor, and strict inequality makes vw a bridge; this yields the cut-vertex criteria above.

For the exact block boundaries, take a true block B and its first discovered vertex r_B. By Lemma 7 any excursion outside B returns through the same cut vertex and is completed before the traversal resumes within B. Restricted to a non-bridge B, the DFS root r_B has a single child because $B - r_B$ is connected, and every internal child subtree has a bypass to a strict ancestor of its parent, since otherwise that parent would cut B. The edges of B are therefore not popped in separate internal segments but all at once, when the unique root child returns to r_B, while the outside excursions have already output their own blocks and contribute no edges to this segment. A bridge is popped alone, and each true edge is pushed once and popped once, so no block is duplicated or omitted. Taking endpoint sets and connecting each cut vertex to its containing blocks gives exactly the incidence tree. □

C.3 SPQR WORKING PIECES AND EDGE INSTANCES

For a recovered non-bridge block, initialize a working piece with exactly its recovered true-edge records. Working pieces may later contain parallel edges but never loops; a real-edge record keeps its original identity, and a newly created virtual edge has a distinct instance identifier, known endpoints, and one twin. Two virtual records with the same endpoints are not merged.

To query adjacency in a piece K, filter $\mathrm{Nbr}(d_v)$ to the real-edge records still assigned to K and add the incident virtual records of K. A deleted-vertex query additionally omits that vertex and its incident records. A virtual step uses the saved reference of its other endpoint and is not asserted to be a true-edge step of the unfolding. Connectivity tests may ignore parallel multiplicity, but edge partitions, P tests, and twin bookkeeping retain it.

A finite split-and-reduce algorithm. For each current piece K, apply the following cases in order.

1. If K has two vertices, output a P-bond; if it is a simple cycle, output an S.
2. If at least two edges have the same poles u, v, place all edges on those poles in E_L and the remaining edges in E_R; split as below.
3. Otherwise K is simple. For each vertex u, run the cut-vertex test on the working view $K - u$. If it has a cut vertex v, choose a component C of $K - \{u, v\}$. Let E_L contain the edges with an endpoint in C and let E_R contain all other edges, including a direct uv edge if present. Split. If no such pair exists, output an R.

4. A split creates fresh twins e_L, e_R on u, v and replaces K by
$$K_L = (V(E_L) \cup \{u, v\}, E_L \cup \{e_L\}), \quad K_R = (V(E_R) \cup \{u, v\}, E_R \cup \{e_R\}).$$
Each old edge record, including an old virtual edge and its external twin link, moves to exactly one child. Connect the two new pieces by their new twin link and recurse.
5. After all pieces terminate, repeatedly merge adjacent S pieces or adjacent P pieces. Remove their shared twins and identify corresponding poles, preserving every other edge record and twin.

This finite procedure serves information access and correctness; no linear-time complexity claim is attached to it.

Lemma 15 (Split invariants and termination)**.** *The procedure terminates and outputs the reduced SPQR data of the recovered block.*

Proof. Maintain that each piece is connected, loopless, has at least three edges, and stays connected after deleting any one vertex, allowing a two-vertex parallel-edge piece. For a parallel-edge split, E_L has at least two edges and the complement contains an internally nonempty u–v path by the invariant; for a separator split, every component of $K - \{u, v\}$ is adjacent to both poles, since otherwise one pole would be a cut vertex. Both sides therefore have at least two edges and supply a u–v path.

Adding the new virtual edge keeps either side biconnected: after deletion of an internal vertex an original path leaving the side can be replaced by the new pole edge, and after deletion of a pole an excursion can return only through the other pole and can be removed from the path. The invariant thus holds for both children. If the parent has $q = l + r$ edges, then $l, r \geq 2$ and each child has fewer than q edges; more strongly, the sum of $|E| - 2$ over current leaves remains $m - 2$, where m is the initial edge count of the block. Every leaf has positive weight, so there are at most $m - 2$ terminal pieces and $m - 3$ splits. An unsplit terminal simple piece has no cut vertex and no two-vertex separator; the only three-vertex biconnected simple graph is a triangle, already handled as S, so the remaining case is a triconnected R of order at least four.

At every split the twin links form a tree, each true edge has one owner, each virtual edge has one twin, adjacent pieces intersect precisely in the twin poles, and vertex carriers remain connected. To verify this, replace one tree node by two adjacent nodes, transfer each old incident link with its old virtual edge, and note that only the new poles occur in both children; joining the children reverses exactly this operation. Joining all pieces therefore recovers the original block with all true-edge identities.

Merging adjacent S pieces replaces two pole-to-pole paths by their union, a simple cycle, and merging adjacent P pieces gives another bond with at least four edges; each merge reduces the number of tree nodes and preserves the invariants. In the final reduced tree each side of a twin link has a vertex beyond its poles. For an adjacent S/R this is immediate, and for an adjacent P at most one edge is real, so besides the current virtual edge there is another virtual edge leading to a non-P neighbor and hence to such a vertex. Carrier separation and true-edge ownership imply that the final poles are actual two-vertex separators of the original block, so the result satisfies all SPQR axioms of Appendix A.2 and uniqueness identifies it with the reduced SPQR tree. □

Theorem 16 (Named decomposition simulation)**.** *A complete named colored unfolding of an active pair suffices to recover the full block–cut tree and reduced SPQR data of its connected component using finite queries and state. The data include vertex identities, real-edge ownership, skeleton membership, and all virtual-edge instances and twins. A collection of such roots covering the components recovers the whole graph's decomposition data.*

Proof. Use diagonal access and the neighbor interface, then Lemmas 14 and 15. All true-edge information comes from atomic colors in queried messages, new virtual information comes only from the verified split records, and all traversals and split trees are finite. □

The theorem is a single-graph information-access result, valid for the auxiliary computation without a $\mathcal{C}_{\mathrm{SS}}$ condition, and it makes neither DFS discovery times nor the chosen split order nor vertex names into anonymous color predicates. In the forward expressivity proof it supplies each side's own map and responsibility queries, while the separate local-decoding and incidence-forest lemmas prove that suitable cross-graph choices exist. Those lemmas do not assume a pre-existing isomorphism of the two complete named maps.

D COLORED CONFIGURATIONS AND INTERFACE ALIGNMENT

We record the definitions needed to interpret $\mathcal{C}_{\mathrm{SS}}$ and to apply its two conditions. They concern complete ordered-pair colorings, diagonal colors included, not merely uncolored rigid skeletons.

D.1 COHERENT CLOSURE AND ALGEBRAIC CORRESPONDENCE

A *coherent configuration* on a finite set V is a partition $\mathcal{R}$ of V^2 into nonempty basic relations such that the diagonal is a union of relations, the transpose of every relation is again a relation, and for all $r, s, t \in \mathcal{R}$ the number

$$p_{rs}^{t} = |\{z : (x,z) \in r,\ (z,y) \in s\}| \qquad ((x,y) \in t)$$

is independent of the chosen $(x, y) \in t$. The diagonal relations define the vertex *fibers*. An algebraic isomorphism is a bijection of the basic relations preserving these intersection numbers, hence also their diagonal and transpose structure, while a combinatorial isomorphism is a vertex bijection inducing that specified correspondence of relations. The configuration is *separable* when every algebraic isomorphism from it to a coherent configuration is induced in this way, and *Schurian* when its basic relations are exactly the ordered-pair orbits of the permutations preserving every basic relation. These are the ordinary, or 1-, versions of the properties (Evdokimov & Ponomarenko, 2000), distinct from their higher-dimensional extensions.

Given a complete colored object A, initialize dense refinement with its input pair color, equality, endpoint colors, and transpose color. Its stable classes form the coherent closure $\mathrm{CC}(A)$: the diagonal and transpose conditions persist, and stability supplies the constants p_{rs}^{t} through joint-message counts. Every automorphism of A preserves each refinement round, and every automorphism of the closure preserves its coarser input colors, so the two color-preserving automorphism groups coincide. This is the colored-graph setting of Fuhlbrück et al. (2021, Section 2.4, Theorem 2.5).

Lemma 17 (A stable local pair color determines the local signature)**.** *In a complete pair domain, equality of one stable pair color under common refinement implies equality of the full stable pair-color histograms. The common stable relation names then induce an algebraic isomorphism of the coherent closures.*

Proof. The messages of (a, b) determine the multiset of first-source colors $\lambda(a, z)$ over all z, and for each such color its stable message table determines the multiset of second-source colors $\lambda(w, z)$ over all w. Summing these column histograms, with the multiplicity of each z taken from the first multiset, counts every ordered pair (w, z) exactly once and thereby reconstructs the whole histogram from $\lambda(a, b)$. Common relation colors and their stable message multiplicities then give equal intersection numbers, as required. □

Lemma 18 (Realization followed by entry alignment)**.** *Suppose actual colored units A, A' have the same local dense stable entry color at (a, b) and (a', b'). If $\mathrm{CC}(A)$ is separable and $\mathrm{CC}(A')$ is Schurian, there is a single color-preserving bijection f of the whole units satisfying $f(a) = a'$ and $f(b) = b'$.*

Proof. Lemma 17 gives the algebraic correspondence specified by the common stable colors, and separability realizes it by a vertex bijection f_0. The pairs $(f_0(a), f_0(b))$ and (a', b') lie in the same target basic relation, so Schurianity supplies a relation-preserving automorphism h mapping the first ordered pair to the second. Then $f = h \circ f_0$ has the required entry and preserves the stable relations, hence the input colors; a one-vertex entry is represented by $a = b$. No temporary third certificate vertex needs a preselected image. □

For an R the input colors are precisely $(\widehat{c}_G, \sigma_R)$ from (5), which already retain the information about the outside graph that survives in the original sparse colors; no labels encoding future subtree isomorphisms are added. Because both input graphs lie in $\mathcal{C}_{\mathrm{SS}}$, the required source and target conditions hold at every selected R.

D.2 Qualification and its scope

The auxiliary selector and its stable colors are computed before qualification, so forming A_R and then its coherent closure is a finite, noncircular procedure. The condition is isomorphism-invariant and independent of the comparison opponent: every message stays inside its own input graph, so joint encoding may rename a graph's colors and postpone the common stopping round, but it cannot change the eventual internal color partition. The actual local configurations used for qualification are therefore unchanged up to relation names. The condition is decidable on finite inputs, though not claimed efficient: enumerate vertex permutations to test whether the pair orbits equal the basic relations, and for separability enumerate configurations and algebraic correspondences of the same order and test realization. The order is fixed by the intersection numbers, since for a diagonal relation δ and any point in its fiber $|V| = \sum_r p^{\delta}_{rr^{\intercal}}$, which makes the enumeration finite. A failed certificate search is not evidence of nonqualification unless the search was exhaustive.

No-R graphs qualify vacuously. For the small-R subclass the extra premise E asserts that every coherent configuration on at most thirteen points is separable and Schurian. The enumeration of Ziv-Av (2018, Table 1 and Sections 2.2–2.3) reports evidence for this premise, including comparisons of algebraically isomorphic configurations and their algebraic automorphisms. Its stated program-correctness caveat is why we keep E explicit rather than presenting the enumeration as an unconditional proof; the theorem on $\mathcal{C}_{\mathrm{SS}}$ itself does not use E.

The class condition is sufficient and not asserted necessary, and the main equivalence compares two graphs inside the class. It neither identifies a qualifying graph against every possible nonqualifying opponent nor asserts that every same-colored non-diagonal root extends to an isomorphism.

E Recovering local comparison from sparse colors

This appendix proves the local-decoding claim used in Lemma 4. Theorem 16 justifies the named queries, while the cross-graph assertions below use colors and message counts rather than names. Throughout, c denotes the stable auxiliary Stitch coloring computed on the original graph with a common palette across the compared graphs, held fixed while the local dense refinements are decoded. Their joint messages are $M(a,b) = \{\{(c(a,t), c(t,b)) : t \in \widehat{N}(a,b)\}\}$. The selector is invariant under reversal of the root coordinates, so induction on the rounds gives a common transpose operation on colors, and a stable color determines its atomic color, its message multiset, and all earlier colors. Figure 5 separates the auxiliary boundary used for decoding from the actual entry that the final local map must preserve.

E.1 The endpoint relation and complete units

For $a \neq b$, define $\mathrm{End}(a,b)$ to mean that (a,b) is active and $a \in \widehat{N}(a,b)$; it is false for inactive roots and on the diagonal. The first update reads it by testing whether some message has a diagonal first source. By the selector it is symmetric and consists exactly of the bridge edges, the S skeleton edges, and all distinct pairs within an R. Let $G^{\sharp}$ be the simple graph with this edge relation and set $E^{+}(a,b) = [a = b] \vee \mathrm{End}(a,b)$. Every E^{+} root admits both endpoint pivots, so its stable color determines both endpoint diagonal colors; in particular, $\widehat{N}(v,v) = N_{G^{\sharp}}[v]$.

The graph $G^{\sharp}$ has the same bridge and S/R vertex bags as G: every R is completed to a clique and virtual-edge endpoints are joined. A valid reduced decomposition is obtained by retaining the S bags, completing the R torsos, adding the corresponding real edge at each P, and inserting a P between directly adjacent S/R bags where necessary. This preserves the generating S and R domains and hence the selector; the construction appears only in the proof, while the original graph adjacency remains in c.

Call an R-bag or a triangular S-bag a *complete unit* U; its full cube is contained in $\widehat{T}$. The vertex bags, together with the bridge, P, and cut-vertex bags, form the tree representation of Lemma 9, with connected carriers and intersections of size at most two. The Helly property of subtrees implies that three vertices pairwise joined in $G^{\sharp}$ occur together in one bag, and a nonadjacent pair in an S cannot occur in a different skeleton, since the carrier path would force a nonedge of that S to be its exit interface. Consequently every triangle in $G^{\sharp}$ lies in a unique complete unit, and the complete units are exactly the maximal cliques of order at least three. For an endpoint root (a,b), the nonendpoint

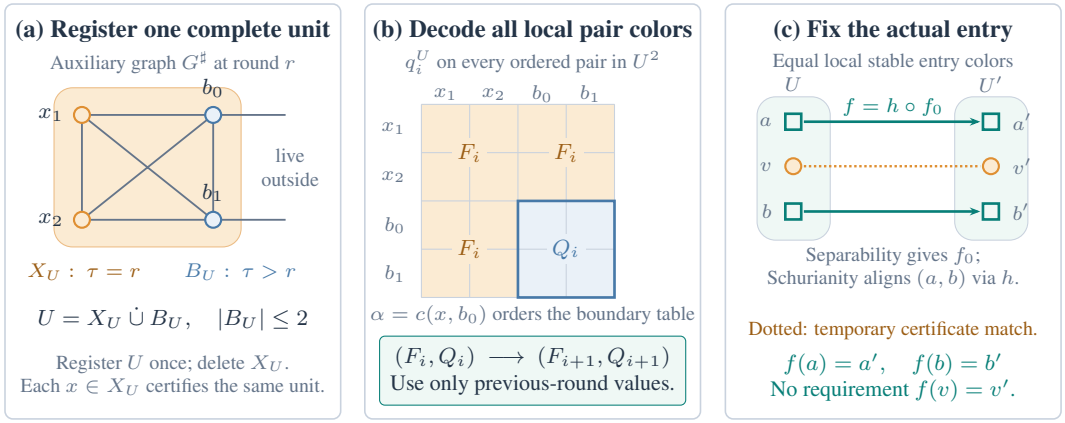

Figure 5: **From a deletion certificate to an entry-preserving local map.** (a) An illustrative complete unit in $G^\sharp$ is registered while all its vertices are alive. The certificate class X_U is deleted and at most two vertices remain in B_U. (b) F_i decodes pairs with an endpoint in X_U; Q_i decodes the remaining boundary square, here of size four. Their simultaneous recurrence uses only the previous round. (c) For qualifying R-configurations, separability supplies a whole-unit map and Schurianity aligns its actual ordered entry. The displayed vertices are selected members of the units. The dotted certificate correspondence is used for comparison and is not imposed on the final map. The actual entry need not equal the auxiliary boundary.

pivots whose two sources both satisfy End split into disjoint cliques, one for each incident complete unit with a, b removed, while the remaining nonendpoint pivots come from long S-skeletons, one of source pattern $(1, 0)$ and one of pattern $(0, 1)$ per S.

To recover this grouping from anonymous colors, we next build deletion certificates that identify each complete unit through its messages.

E.2 A COLOR-READABLE DELETION WITH AT MOST TWO BOUNDARY VERTICES

For a set L selected by diagonal colors and an E^+ root (u, v), the intersection size

$$J_L(u, v) = |N_{G^\sharp}[u] \cap N_{G^\sharp}[v] \cap L| = \sum_{z \in \widehat{N}(u,v)} \mathbf{1}[E^+(u, z) \wedge E^+(z, v) \wedge z \in L] \tag{10}$$

is color-readable: every auxiliary common neighbor is a legal pivot by the preceding description of complete units, and the diagonal case follows from the closed-neighborhood identity.

Start with $L_0 = V(G)$. In round h, write $D_h(v) = N_{G^\sharp}[v] \cap L_h$ and $X_h(v) = \{w \in D_h(v) : D_h(w) = D_h(v)\}$. Whenever $D_h(v)$ is a clique of order at least three and

$$|D_h(v) \setminus X_h(v)| \le 2, \tag{11}$$

register $U = D_h(v)$ once and delete the whole class $X_h(v)$. Also delete every vertex of current degree at most two. All deletions are simultaneous; no edge is added and no path is compressed.

Lemma 19 (Registration certificates)**.** *The process deletes every vertex and registers each complete unit exactly once. If r_U is the registration round and $\tau(v)$ the deletion round, then*

$$X_U = \{x \in U : \tau(x) = r_U\}, \qquad B_U = \{x \in U : \tau(x) > r_U\}, \qquad |B_U| \le 2. \tag{12}$$

For each $x \in X_U$, its diagonal color determines its certificate status, r_U, $|U|$, $|X_U|$, and $|B_U|$.

Proof. For two adjacent live vertices, equality of their closed neighborhoods is equivalent to $|D_h(v)| = |D_h(w)| = J_{L_h}(v, w)$, and $D_h(v)$ is a clique precisely when $J_{L_h}(v, w) = |D_h(v)|$ for every $w \in D_h(v)$. Equation (10) makes these predicates, and the size of $X_h(v)$, readable from

diagonal messages, so inductively every live set and every deletion mark is a common diagonal-color predicate.

An unregistered complete unit keeps all its vertices until its first deletion. If that deletion uses the degree rule, the unit is a triangle and the same closed neighborhood satisfies (11); otherwise the clique neighborhood of the deleted vertex contains the unit and must equal it by maximality. After registration at most two vertices of that unit remain alive. Conversely, a registered clique lies in a unique complete unit, which cannot already have been registered, since only two vertices would remain, and which otherwise still has its whole vertex set alive. No false or repeated unit is therefore registered. Boundary vertices cannot disappear in the registration round, because an additional live outside neighbor prevents the relevant clique condition while membership in the same clique class would place the vertex in X_U.

For termination, restrict the tree representation to any nonempty live set and remove leaf bags contained in their neighbor. A remaining leaf complete unit has private vertices with the same clique neighborhood and at most two boundary points, and a remaining S, bridge, P, or cut bag has a private vertex of degree at most two; the one-bag case satisfies the same alternatives. Every nonempty live graph therefore admits a deletion. □

E.3 Exact internal messages and the boundary table

Fix a registered unit $U = X \dot\cup B$ with registration round r and $|B| \le 2$. For a pair with an endpoint in X, the earlier-deleted endpoint identifies the certificate metadata $(r, |U|, |X|, |B|)$, and if the deletion times tie then both endpoints lie in X for this same unit. Endpoint diagonal colors are available because every internal pair is an E^+ root, so whenever at least one endpoint of $(u, w) \in U^2$ lies in X,

$$\begin{aligned}&\{\{(c(u,z), c(z,w)) : z \in U\}\}\\ &\quad = \{\{(c(u,z), c(z,w)) : {}^{z\in\widehat N(u,w),\ \tau(z)\ge r,}_{E^+(u,z),\ E^+(z,w)}\}\}.\end{aligned} \tag{13}$$

Indeed, every internal pivot is selected, while the live closed neighborhood of the endpoint in X is exactly U. Both E^+ tests are retained, since a deletion-round test alone need not remove outside long-S contributions, and for a message passing these tests the source colors determine the diagonal color of the pivot and hence its deletion time. The cases $\tau(z) = r$ and $\tau(z) > r$ then separate X from B.

Skeleton roles are readable as well for a fixed complete unit of size k. For distinct $a, b \in U$, an outside pivot exists exactly when ab is a virtual skeleton edge: the forward direction follows from connected carriers and the first two-vertex interface leaving U, while for the converse the far side of a virtual edge contains another S/R unit, directly or through a P, which supplies an outside pivot. Virtuality is therefore equivalent to $|\widehat N(a, b)| > k$, and in its absence original adjacency separates real edges from nonedges.

Let q_i^U denote dense refinement on the actual colored local object of U; the same construction applies to a triangle with its skeleton roles. For $x \in X$ and $s \in B$, put $\alpha = c(x, s)$ and order B temporarily as $b_0 = s$ and, when present, its unique other vertex b_1. In the B part of (13) the pivot s is recognized by the diagonal second source, and at most one pivot remains, so α determines

$$V(\alpha)_j = c(x, b_j), \qquad P(\alpha)_{j\ell} = c(b_j, b_\ell). \tag{14}$$

For every $\delta = c(z, s)$ with $z \in X$ the same order is used, since the second endpoint remains s; no permanent labels are added to the boundary.

Lemma 20 (Simultaneous local decoding)**.** *For every finite i, common functions F_i and Q_i recover*

$$F_i(c(u,w)) = q_i^U(u,w) \quad \text{if } \{u,w\} \cap X \ne \varnothing, \qquad Q_i(\alpha)_{j\ell} = q_i^U(b_j, b_\ell). \tag{15}$$

The functions use the certificate metadata determined by their input colors.

Proof. The initial local color is a common function of c, the known size k, and the just-recovered skeleton roles, endpoint colors, and transpose color, which gives F_0 and, through (14), Q_0.

For a root with both endpoints in X, every source in (13) has an endpoint in X, so applying F_i yields its entire local update. For (x, s) the X-pivot sources still use F_i, while a boundary pivot b_j contributes $(F_i(V(\alpha)_j), Q_i(\alpha)_{j0})$, and reversal handles (s, x); this determines F_{i+1}.

To update a boundary pair, obtain $H_\alpha(\delta) = |\{z \in X : c(z,s) = \delta\}|$ from (13). For the root (b_j, b_ℓ) each such color contributes, with this multiplicity,

$$\big(F_i(V(\delta)_j^{\mathsf{T}}), F_i(V(\delta)_\ell)\big). \tag{16}$$

Adding the boundary-pivot messages $(Q_i(\alpha)_{jh}, Q_i(\alpha)_{h\ell})$ for all $0 \le h < |B|$ then determines Q_{i+1} together with the old entry. Both F_{i+1} and Q_{i+1} use only F_i and Q_i, which completes the simultaneous induction; when B is empty only F_i is needed. □

E.4 From a certificate message to a map of the whole unit

Choose an entry (a, b) in a complete unit U and a certificate $v \in X_U$. If a message at (a', b') satisfies

$$c(a,b) = c(a',b'), \quad c(a,v) = c(a',v'), \quad c(v,b) = c(v',b'), \tag{17}$$

then v' is a certificate of a unit U' of the same size containing both a' and b', where membership is tested by $a \in U(v)$ if and only if $E^+(a,v)$ and $\tau(a) \ge \tau(v)$. When one root endpoint lies in X_U, Lemma 20 applies directly; when both lie in B_U, its table reads the diagonal entry or the uniquely ordered off-diagonal entry. It follows that $q_i^U(a,b) = q_i^{U'}(a',b')$ for every i.

Lemma 17 turns this entry-color equality into a full local comparison and the specified algebraic correspondence, and Lemma 18 then gives the single whole-unit map $h \circ f_0$ preserving all input colors and the prescribed entry. No constraint fixes the image of the temporary certificate v.

For a triangular S the same entry-preserving conclusion follows directly: once two distinct endpoints are fixed, the unique third vertex and all incident relations are determined by the joint message. A diagonal entry is handled by first matching one off-diagonal source.

Each unit has exactly $m(U) = |X_U|$ certificates. At an entry, assign weight $1/m(U)$ to every certificate message containing that entry, classified by its recovered rooted local type; all certificates of the same unit have the same type and total weight one. The entry color therefore determines the actual unit-type multiplicities without mistaking the number of certificates for the number of branches. This completes the proof of Lemma 4 and supplies the counts required for global stitching.

F Interface grouping and the global construction

Theorem 16 supplies each graph's named decomposition map, and the task tree together with its separation properties comes from Lemma 9. We now derive compatible choices between the two named maps from their common colors. Figure 6 illustrates the whole-unit maps, the source subtraction, and the single-step commitment used in this construction.

F.1 Long cycles admit a whole-bag map

Consider an S containing the ordered skeleton edge (a, b). A triangle is covered by the complete-unit argument of Appendix E, so let the S be longer and write its complementary path as $b = t_0, t_1, \ldots, t_{s-2}, t_{s-1} = a$. The construction below recovers an entry-preserving whole-cycle map from the joint message that selects the first pivot t_1. At a non-diagonal endpoint root (a, b), a long S-skeleton contributes exactly one nonendpoint pivot t whose source End bits are $(0, 1)$, and complete units contribute none of this type. The root (a, t) then belongs to a unique S and has a known predecessor b. Keeping a fixed, filter its messages for a neighbor of the moving endpoint and subtract exactly the predecessor's message copy; only one forward direction remains until the moving endpoint becomes a skeleton neighbor of a. At an internal position (a, t_j), filter for the second-source End bit, whose resulting pivots are exactly the two S-neighbors t_{j-1}, t_{j+1} of t_j. The previous step supplies the joint type of t_{j-1}, so Lemma 12 removes one matching predecessor contribution and supplies the next. By Lemma 8 no outside skeleton contributes at this nonadjacent S-root, the End bits make the two scans stop together, and no intermediate vertex is revisited because the scan follows a simple cycle.

Starting from the original skeleton edge (a, b), this traverses the complementary b–a path and recovers a bijection of the whole cycle that preserves every directed skeleton-edge color and all diagonal colors. For a nonadjacent cycle pair every pivot lies in this unique S and is a neighbor of one endpoint,

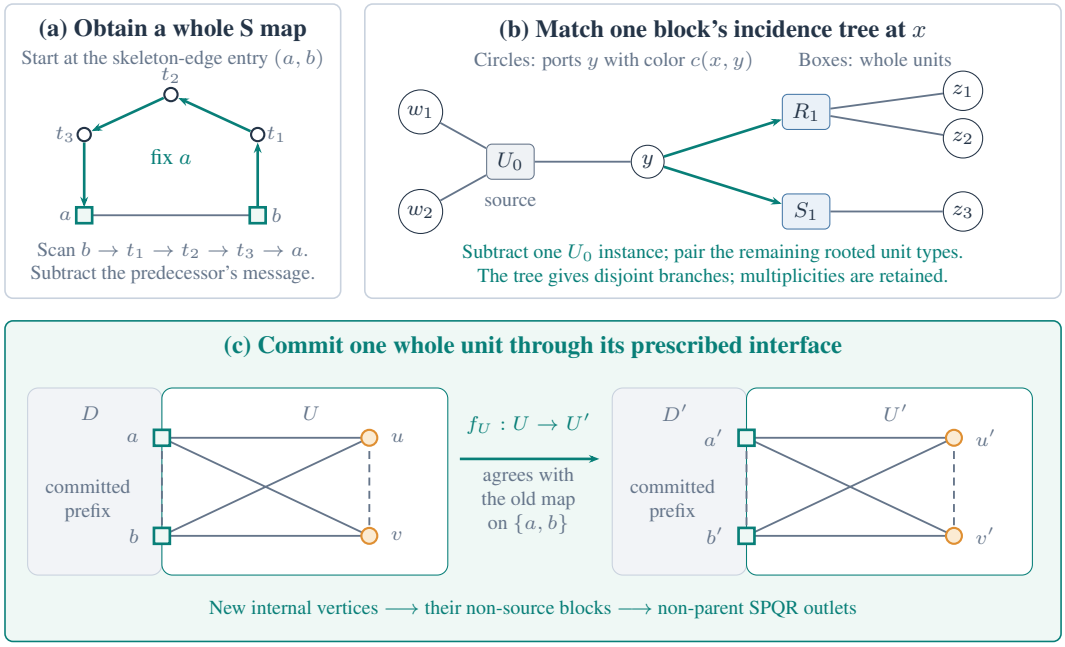

Figure 6: **One step of the local-to-global construction.** (a) The S scan fixes a and follows the complementary path from b back to a, subtracting the predecessor's message at each step. (b) One component of the incidence forest at x represents one incident block. At port y, remove one already matched source-unit instance U_0, then match the remaining units by their rooted types. Each unit correspondence uses a whole-unit map. (c) A new unit meets the committed prefix only in its parent interface, shown as $\{a, b\}$. The new map agrees there with the old map; orange vertices are newly committed. Dashed skeleton edges mark two-vertex outlets. New vertices receive their non-source block responsibilities before the non-parent SPQR outlets are processed.

and the two initial colors agree because such a pair is a nonedge of the original graph on both sides. Induction on refinement rounds therefore shows that the cycle bijection preserves every internal pair color: source pairs that are skeleton edges already have equal stable colors, and every other source falls under the inductive hypothesis. The actual skeleton edge roles are preserved as well, since an S edge has three internal pivots in a triangle and four in a longer cycle, while a virtual edge has an additional outside pivot that a real skeleton edge lacks.

The scan produces a whole-cycle map before any new vertices are committed to the global construction. When a nonadjacent initial root is used, both directions around the cycle must be scanned and joined at their common endpoints, since a single arc does not cover the cycle.

F.2 The incidence forest at a vertex

For an endpoint root (x, y), let $K(U; x, y)$ be the isomorphism type of unit U with all its stable auxiliary pair colors, skeleton roles, and ordered entry. For an S this type is counted only at skeleton-edge entries, whereas for an R every distinct internal pair is allowed. The complete-unit certificate count and the preceding cycle scan show that $c(x, y)$ determines the multiset of these types, and a bridge is recognized by a total pivot count of two and carries its evident rooted map.

Fix a vertex x and form a bipartite incidence graph $\mathcal{I}_x$ whose port nodes are the End-neighbors y, each carrying the color $c(x, y)$. Every R containing x contributes one unit node adjacent to all its other vertices as ports, every S containing x contributes one unit node adjacent only to its two cycle neighbors of x, and a bridge contributes a unit node adjacent to its other endpoint.

Lemma 21 (Incidence forest)**.** *$\mathcal{I}_x$ is a forest, and its connected components correspond exactly to the blocks incident to x. Equal port colors permit a rooted correspondence of the relevant components, with each unit correspondence realized by a whole-unit color-preserving map. An already matched source unit may be fixed and excluded from further traversal.*

Proof. Within a non-bridge block the skeletons containing x form a connected subtree by the carrier property. A P whose other pole is y is represented by the port y, and a direct S/R adjacency through $\{x, y\}$ is represented by an inserted port y. Since an S/R has at most one virtual edge on a given pole pair and reduced SPQR trees have no adjacent P-nodes, a given port takes the form of a P-star, a direct connection, or a leaf, which makes the incidence graph a tree. Different blocks share only x, so their ports do not mix.

At a matched port $(x, y) \leftrightarrow (x', y')$ the local type multisets agree, so pair units by type and choose a whole-unit map fixing x, y. That map determines the other port images and preserves their colors. At each new port, subtract one copy of the actual source unit type and pair the remaining types. The forest has no alternative path back to an already fixed port, so no additional interface has silently been imposed; each unit and port is visited once, and the equal residual multiplicities give coverage on both sides. When a source unit map is given, remove that node and perform the same construction independently in its port branches. □

The diagonal message at x lists all port colors together with their multiplicities. Choosing a port in any remaining component, finding a matching port on the other side, and applying Lemma 21 matches the whole component, whose full port multiset is then subtracted; iterating matches the incident blocks. When a source unit is already mapped, its component is matched and subtracted first, and the remaining components are exactly the non-source blocks. Block grouping is thus recovered from colors rather than assumed to be synchronized across graphs by a canonical decomposition.

F.3 Responsibilities and the one-step invariant

We use the rooted task construction of Appendix A.3: a vertex's CUT responsibility is attached only at the first bag of its carrier, every virtual-edge instance is preserved, and every P-family is kept as one dispatch task. These rules, together with their separation and coverage proof, are prerequisites of the following invariant.

A state consists of a connected prefix of fully matched bags, the union f of their maps, the unprocessed tasks with their parent interfaces and source instances, and the vertices whose cut responsibilities have already been assigned. The invariant is:

1. Every committed bag map preserves all stable auxiliary pair colors and real/virtual/nonedge roles; parent and child maps agree on their intersection.
2. The committed prefix and pending tasks cover the graph. Each pending interior is new and meets the committed part only in its declared parent interface.
3. A P-family is either awaiting one dispatch or has its non-source branch instances assigned exactly once. A child cannot reopen its parent's sibling responsibilities.
4. Each new vertex has one block–cut responsibility, with its source block specified by its introducing unit.

The unprocessed regions are not assumed to be isomorphic.

For a new one-vertex cut task, the incidence-forest result excludes its source block and pairs the remaining blocks, and a bridge is mapped directly. For a non-bridge block choose a corresponding entrance unit from its matched incidence component and take its rooted local map. Only that unit is committed; the maps used temporarily in the forest argument are not all committed at once.

For a virtual-edge outlet $\{a, b\}$ of a committed unit, the local rooted type multiset pairs all incident S/R units and removes the single source instance. There is then either a unique directly adjacent unit or a P-family whose remaining branches are dispatched once. With q incident S/R units, the direct case is $q = 2$ with no original edge ab; otherwise the P also carries the original edge when present. Both q and the original edge bit are preserved, so the candidate units lie in the correct unprocessed SPQR responsibilities rather than being arbitrary same-type units elsewhere in the graph.

An R is matched by Lemma 4 and an S by the whole-cycle construction, and each map fixes precisely its parent interface. Committing the new internal vertices, processing their non-source blocks, and then processing the non-parent SPQR outlets never reopens the cut tasks of previously introduced interface endpoints. The new bag map preserves all outlet colors, so the same local-type and block-grouping arguments remain applicable at the next step.

Operational order and allowed choices. The resulting finite procedure is summarized below.

1. At a matched diagonal root, match incident blocks using the incidence forest. For a new block, select an entrance unit inside its assigned component of that forest.
2. At a two-vertex outlet, use the root's unit-type multiset to remove the actual source instance and assign the remaining branches. A P assignment is recorded once.
3. Obtain one whole S or R map extending the parent interface. Its existence comes from the cycle argument or Lemma 4; a finite enumeration of unit bijections can find such a map. It need not solve any future task before being selected.
4. Commit this map, assign each new vertex's non-source CUT task, then process the non-parent SPQR outlets. Reuse existing interface vertices without reopening their CUT tasks.

Only whole-unit maps that preserve all stable auxiliary pair colors and local skeleton roles are allowed at step 3, and the temporary maps used to establish an incidence-forest correspondence are not precommitted vertex images. Names select and distinguish actual instances on each side; Lemma 12 justifies the equal-type subtraction, and the forest places the surviving choices in the assigned responsibilities. The color argument establishes the existence of an admissible next map, of which finite search is one implementation rather than the existence proof.

F.4 Why old records remain consistent

Let D be the union of the committed bags. An unprocessed bag can meet D only inside the first parent interface on its path to the prefix. When this interface is a singleton, its old coordinates contribute only the self-triple; when it is a pole pair, the parent S/R rule already contains the complete cube on that pair. For bookkeeping a P-bag may also be assigned its two-pole cube, since every such triple is already generated by an incident S/R and the diagonal rule. Hence

$$\widehat{P} \cap D^2 = \bigcup_{U \text{ committed}} V(U)^2, \qquad \widehat{T} \cap D^3 = \bigcup_{U \text{ committed}} T_U. \tag{18}$$

The committed local maps and their inverses preserve these sets and all their pair colors, so they form a partial graph isomorphism, because every true edge is active and atomic adjacency is retained. For each committed active root they also preserve the exact already committed pivot set and its joint-color contributions, and subtracting these contributions removes actual message copies including their multiplicities, not an entire color class. Adding a neighboring bag preserves (18), since its intersection with the old prefix is exactly its parent interface. This proves the single-step assertion in Lemma 5.

F.5 Termination, coverage, and disconnected graphs

Initialize with corresponding diagonal roots and use their incident block grouping. Each step consumes one previously unprocessed bridge, skeleton, cut task, or P-dispatch in a finite responsibility tree, and no source task is reopened and no branch is duplicated, so the process terminates. Since each parent pairs all non-source children with matching multiplicities, termination covers both connected components. The bag maps agree on intersections, their new interiors map to disjoint new interiors, and their union is a bijection preserving every true edge whose inverse has the same property; it is therefore a rooted component isomorphism preserving all vertex diagonal colors.

Equal graph signatures contain equal diagonal-color multisets. Choosing one remaining component and a same-colored root in the other graph, constructing its component isomorphism, and subtracting the full diagonal-color multiset of that component, we repeat until the multisets are empty; isolated vertices are the one-root, zero-outlet case. This establishes the forward implication in Theorem 3.

G Dense colors determine the exact Stitch selector

Fix a common dense stable coloring d on densely indistinguishable graphs. Relations that are unions of its color classes are closed under Boolean operations, transpose, and relational composition: the number of witnesses to $R(a, z) \wedge S(z, b)$ is a sum of joint-message multiplicities at (a, b), and finite transitive closure is readable by repeatedly taking Boolean union and composition until stabilization. None of these operations uses vertex names as colors. Figure 7 illustrates the normalization and the subsequent exact filtering of dense messages.

G.1 A COMMON COLOR-DEFINED BAG REPRESENTATION

For connected inputs, Kiefer & Neuen (2022, Definition 5.4, Theorem 5.6 and Appendix A) provide a rooted tree decomposition with adhesion at most two whose torsos are cycles or triconnected graphs, with small bags treated separately. Its bag-membership and parent relations use common dense color classes on equivalent graphs, so this direction relies on the external color-defined decomposition theorem and is independent of the named simulation. Concretely, there exist color sets C_{bags} and C_{edges} depending only on the dense equivalence class such that membership of $d(v,x)$ in the former means $x \in \beta(t_v)$ and membership of $d(v,w)$ in the latter means that t_v is the parent of t_w. The source counts graphs of order at most two as triconnected; we keep those as small bags and classify a three-vertex complete torso as a cycle. Requiring a parent bag not to contain its child ensures that every bag has a nonempty representative class. If t_v is the highest bag containing v, write

$$B(v,x) = [x \in \beta(t_v)], \quad D(v,w) = [t_v \text{ is the parent of } t_w], \quad Q(v,w) = [t_v = t_w]. \tag{19}$$

The same-bag relation belongs to the representation; equivalently, it is obtained by comparing the two bag rows and their intersection sizes. Both $m(v) = |\{w : Q(v,w)\}|$ and $k(v) = |\beta(t_v)|$ are read from $d(v,v)$. For distinct a, b, the number of bags of size at least three containing both endpoints is

$$n(a,b) = \sum_{v} \frac{\mathbf{1}[B(v,a) \wedge B(v,b) \wedge k(v) \geq 3]}{m(v)}. \tag{20}$$

The joint message of (a,b) reads each indicator after transposition of the first source, and the diagonal color of the pivot supplies its weight; each representative class contributes one.

G.2 RECOVERING TORSOS AND REDUCING CONSECUTIVE CYCLE BAGS

For distinct endpoints define

$$A(a,b) = E_G(a,b) \ \vee \ [n(a,b) \geq 2]. \tag{21}$$

On each large bag U this is its torso edge relation. For the nontrivial direction, if another large bag W contains a, b, its biconnected torso supplies an a–b path through its other vertices, and replacing virtual steps by their outside witnesses gives a path outside U, hence a torso edge. Conversely, a component outside U adjacent to both a and b must attach through a carrier path containing both; if all other bags on that two-vertex carrier subtree were merely $\{a,b\}$, the component would attach through a branch missing one pole, a contradiction. Some other large bag therefore contains the pair. Small redundant bags must be excluded from (20), since they witness no other nontrivial branch.

Using (21), the torso degree of every member is readable at a representative root: at (v,u) count the pivots z satisfying $B(v,z)$ and $A(z,u)$, and the row of a representative then tests whether all member degrees equal two. A large bag all of whose torso degrees are two is a preliminary cycle bag, and every other large bag is a preliminary rigid bag. Bag-intersection sizes are read by counting the z with $B(v,z) \wedge B(w,z)$.

The parent interface of the bag represented by v is

$$I_v = \{x : B(v,x) \wedge \neg Q(v,x)\}. \tag{22}$$

and has at most two elements. If $I_v = \{a,b\}$, then at the root (v,a) the unique non-diagonal second-source message whose pivot belongs to I_v recovers $d(b,a)$, so the bag diagonal color can test the shared-pole predicate $Z(a,b) = [n(a,b) = 2 \wedge \neg E_G(a,b)]$. Indeed, the representative row lists each $a \in I_v$ together with the color of (v,a); that color identifies the unique other interface pivot and hence the value of Z, and both interface choices read the same symmetric value.

Mark a parent–child bag edge when both bags are preliminary cycles, their intersection is $\{a,b\}$, and $Z(a,b)$ holds. Also mark both edges of a small-bag intermediary $\{a,b\}$ with exactly two adjacent preliminary cycle bags and the same predicate; this test uses representative weights when counting adjacent bags. Taking the equivalence closure of these marked edges together with Q gives a color-readable relation F on representatives, and

$$B^*(v,x) = [\exists w \ (F(v,w) \wedge B(w,x))] \tag{23}$$

is the merged membership relation, while $m^*(v) = |[v]_F|$ is the new representative mass.

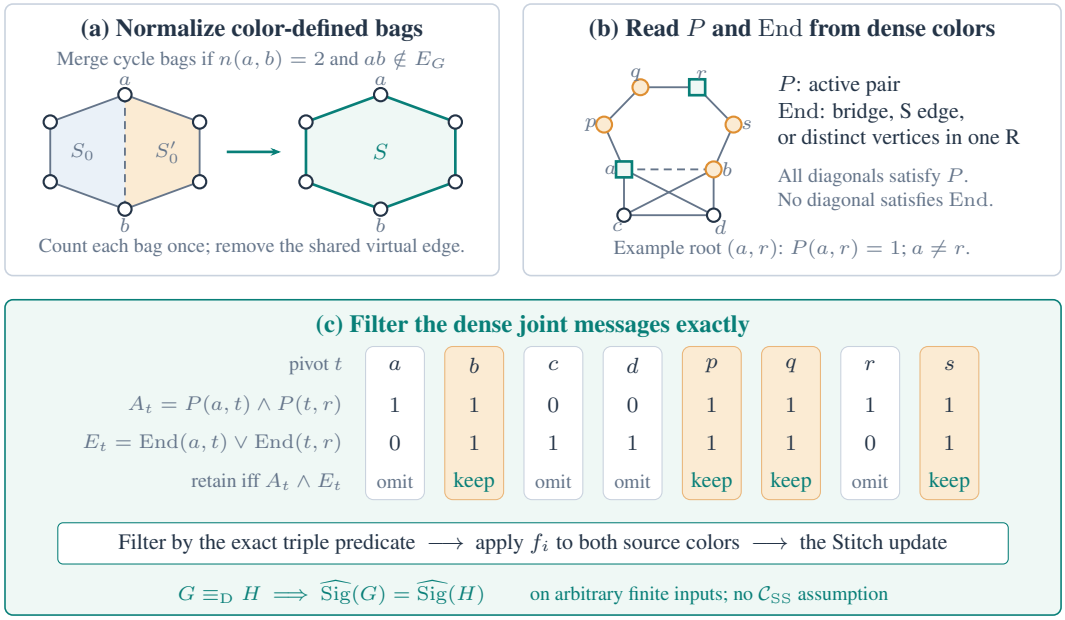

Figure 7: **Dense colors recover the exact sparse update.** (a) In a separate normalization example, two preliminary cycle bags sharing $\{a, b\}$ merge when they are the only two large bags on that pair and ab is not an original edge. (b) On the example from Figure 1, dense colors read activity P and the nonreflexive relation End. (c) At root (a, r), the two source pairs must be active and at least one must satisfy End. The retained pivots are exactly $\{b, p, q, s\}$; c, d fail source activity, while the endpoints a, r fail the End test. Applying the round-i projection to the retained source colors reproduces the sparse joint multiset with its multiplicities. For arbitrary roots, the full predicate in (26) additionally includes the all-equal triple.

These merged bags are exactly the reduced S/R bags. Indeed, at a pole pair appearing in n large bags each bag supplies one nontrivial branch: if $n = 2$ with no original edge the two branches connect directly, and otherwise a P-node joins them and carries the original edge when present. The resulting decomposition has unique true-edge ownership and connected vertex carriers, and its only remaining reductions are adjacent cycle bags, exactly the marked cases above. Merging them deletes the shared virtual edge and produces one cycle, whereas a third branch or an original edge blocks such a merge; the tree structure ensures that different merges identify no additional vertices. Rigid bags remain unchanged, and the resulting reduced decomposition is the reduced SPQR decomposition of each block. This identification uses the precise skeleton axioms and uniqueness convention of Appendix A.2, not an assumed identity of the original bags with SPQR nodes, so the normalization yields the reduced SPQR representation required by the selector.

G.3 Activity, endpoints, and triple membership

Use the weighted counting formula with B^*, m^* to read the number $n^*(a, b)$ of reduced S/R bags containing both endpoints and the number $n_R^*(a, b)$ of such R-bags. For distinct endpoints,

$$P(a,b) = E_G(a,b) \vee [n^*(a,b) > 0], \tag{24}$$

$$\mathrm{End}(a,b) = E_G(a,b) \vee [n_R^*(a,b) > 0] \vee [n^*(a,b) \geq 2]. \tag{25}$$

All diagonals are active and none satisfies End, and the activity formula includes every original edge, hence every bridge. In the endpoint formula, an S true edge satisfies adjacency, an S virtual edge is shared with another S/R, and a nonadjacent S pair belongs only to its own S and is not an original edge, while R-pairs and bridges are covered by the other terms.

It remains to verify (26). For three distinct coordinates the three activity predicates imply that the carriers intersect pairwise, so the subtree Helly property gives a common bag, necessarily an S or an R. An R permits the entire cube, and within an S an internal nonadjacent pair cannot become an

End pair through a different bag by the carrier argument above, so the End test says exactly that one source is an S edge, as required by (1). With two distinct coordinates the non-diagonal End test gives precisely the allowed endpoint pivot or diagonal-root neighbor pivot, and the all-equal case is explicitly included. This proves the exact Boolean factorization, with no multiplicities introduced by overlapping generators.

G.4 Round-by-round projection and disconnected inputs

The initial auxiliary color is a function of d. Suppose $\widehat{c}_i = f_i(d)$ on active pairs. The root color $d(a, b)$ and its two source colors determine all tests in (26), so filtering the stable joint messages by these tests and applying f_i to both sources gives exactly the next auxiliary update; a common f_{i+1} therefore exists. Activity is itself a dense-color predicate, so equal dense color multiplicities imply equal auxiliary graph signatures at every round and at stability.

For disconnected graphs, connectivity is the reflexive transitive closure of the original edge relation and is therefore dense-color readable. Filtering dense messages to one component simulates its local dense refinement, and a global diagonal color determines the local graph-color histogram of its component by counting that component's diagonal colors and their full row histograms. Each component comparison type can therefore use the common bag relations supplied above, with bag relations false across components. This avoids choosing an asymmetric global root and extends the projection to arbitrary disconnected inputs; the empty graph is immediate, and (9) follows.

G.5 Selector factorization used in the main comparison

Write $P(a, b)$ for activity and define $\mathrm{End}(a, b)$, for distinct endpoints, to mean that the pair lies on a bridge, on an S skeleton edge, or within an R-skeleton; it is false on the diagonal. The selector satisfies

$$(a, t, b) \in \widehat{T}_G \quad \Longleftrightarrow \quad (a = t = b) \ \vee \big(P(a, b) \wedge P(a, t) \wedge P(t, b) \\ \wedge (\mathrm{End}(a, t) \vee \mathrm{End}(t, b))\big). \tag{26}$$

Using the color-defined decomposition of Kiefer & Neuen (2022), the reduced S/R bags and these two binary relations can be recovered from dense colors, and (26) then filters the dense joint messages exactly.

H Finite execution and the compared selectors

Proof of Proposition 2. Source closure follows from the generating bags. The retained old color makes the finite partition refine monotonically, and an isomorphism preserves every generator at every round. After accounting for the n self-triples, a bridge contributes at most six further triples and an R at most $r_R^3 - r_R$. For a fixed S-pivot t, either a or b is one of the two cycle neighbors of t, and inclusion–exclusion gives $2s_S + 2s_S - 4$ ordered choices of (a, b); summing over t yields the S term. The pair bound follows the same way, counting diagonals once. The cycle has no other generator, so its count is exact. □

H.1 An exact discrete implementation

The specification below makes the ordering of decomposition, refinement, and qualification explicit. It applies to arbitrary finite inputs, since qualification restricts the domain of the theorem rather than the executability of the auxiliary computation.

1. Construct the block–cut and reduced SPQR data with the convention of Appendix A. Record original adjacency separately from skeleton adjacency. Isolated vertices need no block or skeleton.
2. Generate a set of ordered triples. Insert every (v, v, v), every bridge cube, and every R cube. For each S, pivot t, and skeleton neighbor u of t, insert (u, t, b) for every $b \in V(S)$ and (a, t, u) for every $a \in V(S)$. This enumerates precisely the last term of (1), including virtual neighbors. Duplicate insertions do not create additional messages.

3. Project triples to their first and third coordinates to obtain the active-pair set. Index each root's row by its distinct pivots, and store references to its two source states. Every pair internal to an S is active: any skeleton neighbor of its first coordinate is a legal pivot. Every selected triple lies in a bridge, S, R, or singleton domain, all of whose pairs are active. This proves source closure, including the S case.
4. Initialize active states by the ordered pair's atomic equality and original-adjacency bits. For a comparison, recolor the two input state sets jointly, using the old color and the sorted multiset of ordered source-color pairs as the exact dictionary key. Both sources are read from the previous round; no in-place update is permitted. Save the dictionary if colors will subsequently be decoded.
5. Stop when the joint partition no longer splits, and compare the two color histograms over active states. To evaluate membership in $\mathcal{C}_{\mathrm{SS}}$, subsequently form each A_R, compute its local coherent closure, and check the stated configuration conditions; this membership check leaves the sparse colors unchanged.

If the compared state sets have total size M and the initial joint partition has k_0 nonempty classes, there are at most $M - k_0$ strict refinements, and one further update confirms stability. Because the old-color field prevents merging, equality of successive class counts is a sufficient stopping test under this exact encoding; at that point each class has a fixed multiset of source classes, so further updates can only rename classes. Numerical equality of independently chosen color identifiers is not a valid stopping or comparison criterion. The empty state set is handled immediately, and an isolated vertex has one diagonal state and one self-message.

The bounds in Proposition 2 count stored pair states and selected message incidences per round, and their derivation does not multiply by the number of times a triple is generated. An implementation may use sorting or a dictionary to deduplicate generators and encode multisets; those operations, the decomposition construction, feature dimensions, iteration count, and the separate qualification problem fall outside any claim of constant work per selected triple.

H.2 The baseline domains used in the comparisons

Let $\mathcal{C}$ be the set of connected components. The interaction domains of the two prior methods are

$$T_{\mathrm{CoSp}} = \bigcup_{C\in\mathcal{C}} \{(a,t,b) \in V(C)^3 : |\{a,t,b\}| \le 2\} \cup \bigcup_{\text{blocks } B} V(B)^3, \tag{27}$$

$$T_{\mathrm{DA}} = \{(v,v,v) : v \in V\} \cup \bigcup_{B\in\mathcal{B}_{\mathrm{br}}} V(B)^3 \cup \bigcup_{U\in\mathcal{S}\cup\mathcal{R}} V(U)^3. \tag{28}$$

These translate the neighborhood rule of Co-Sparsify and the support rule of DA into the triple notation used here (Chen et al., 2026; Anonymous Authors, 2026). All unions deduplicate triples, and the active pairs are their root projections. In particular, Co-Sparsify retains *all* ordered pairs within a connected component and all repeated-coordinate triples on them, including pairs across cut vertices; it is not merely a union of block cubes. Under our Q-suppressed convention, the explicit native-edge and exposed-pole supports of DA are already covered by bridge or S/R cubes, which gives the displayed equivalent formula. For graph comparison, Co-Sp groups active states by connected component, records the diagonal and off-diagonal color multisets separately within each component, and takes the multiset of these component records, whereas Dense, DA, and Stitch use the multiset of all their active-pair colors. These comparisons concern the specified exact discrete refinements; they do not identify previous neural features or learned readouts with the present atomic-initialized algorithm. Hence $\widehat{T} \subseteq T_{\mathrm{DA}} \subseteq T_{\mathrm{CoSp}}$. The first inclusion changes only long S computation, since on triangles the S rule together with the self-triples is the complete cube. These containments explain the counts but by themselves establish no expressivity ordering, because each selector also depends on graph structure. The reverse comparison for Stitch is instead Appendix G.

For Figure 1 the block has vertex set $\{a,b,c,d,p,q,r,s\}$, R vertex set $\{a,b,c,d\}$, and S cycle a,p,q,r,s,b,a, and the shared ab is virtual and not an original edge. At the root (a,r), Co-Sparsify uses all eight vertices and DA uses the six S vertices, while Stitch keeps exactly the neighbors of a or r in that S, namely $\{b,p,q,s\}$; neither a nor r is a legal Stitch pivot there. This is a single fixed-root example, whereas Proposition 2 and Table 9 count the complete computation.

I Exact boundary example and message-level verification

I.1 Why complete rigid-skeleton refinement alone is insufficient

Let Q be the 4×4 rook graph and L the Shrikhande graph, both strongly regular with parameters $(16, 6, 2, 2)$. For a graph F, let $\mathrm{Sub}(F)$ replace each edge by a path of length two, and set $G = \mathrm{Sub}(Q)$ and $H = \mathrm{Sub}(L)$. Each graph then has sixteen original vertices, forty-eight subdivision vertices, and ninety-six edges, and its SPQR decomposition consists of a central R-skeleton on the original vertices together with one triangular S-skeleton per subdivided edge.

Proposition 22 (A loss of graph distinction outside the guarantee)**.** *The auxiliary Stitch refinement has equal stable graph signatures on G and H, whereas dense 2-FWL distinguishes them. DA uses the same selector as auxiliary Stitch on these two inputs and also fails to distinguish them.*

Proof. Every S-skeleton in this construction is a triangle, so the sparse S rule together with the self-triples is its complete cube. The selectors of auxiliary Stitch and DA therefore coincide: the central 16^3 cube and the forty-eight triangular cubes, with duplicates removed. Their stable colors have six common types, namely the original-vertex and subdivision-vertex diagonals, two kinds of distinct original-vertex pairs according to skeleton adjacency, and the two orientations of an incident original/subdivision pair. The sizes and joint-message tables of these types agree between the two graphs; Appendix J lists the table explicitly.

For dense refinement, let W be the adjacency matrix of a subdivided graph. Its second round determines $(W^2)_{ef}$ and $(W^4)_{ef}$, and for distinct subdivision vertices e, f the condition

$$(W^2)_{ef} = 0, \qquad (W^4)_{ef} = 4 \tag{29}$$

means that the corresponding base edges are disjoint and their four endpoints form a K_4. The rook graph has forty-eight ordered pairs satisfying this condition and the Shrikhande graph none, so the dense graph signatures differ. □

The example rules out unrestricted equivalence for the auxiliary selector and lies outside the class of Theorem 3. The actual colored central R-configurations in this pair are algebraically isomorphic but non-isomorphic, so they fail the separability requirement. The example also shows what the sparse computation misses, namely comparisons between two subdivision vertices near the rigid core, and it contradicts neither the stated structural nor the finite-instance guarantees of DA.

I.2 Exact refinement protocol and results

All results in this section use equality/true-adjacency initialization, exact multiset encoding, and a shared palette across the two graphs. We stop when the joint color partition no longer splits, and separately assigned integer color names are never compared. The reproduction script generates the graphs and selectors directly, while Appendix H specifies source closure, exact execution, and the compared baseline domains.

The sparse partition stabilizes after two updates, counting the update that confirms the joint partition no longer splits, whereas dense refinement first separates the graphs at round two and stabilizes after five updates. The shared dense palette has $10, 19, 50, 52, 52$ classes across these rounds. Because the example combines a large reduction in interactions with a loss of expressivity, it cleanly separates computational savings from a preservation guarantee.

J Exact boundary witness and reproducibility

J.1 The two base graphs

Both bases have vertex set $\mathbb{Z}_4^2$. The rook graph joins distinct vertices that agree in one coordinate, and the Shrikhande graph is the Cayley graph with connection set $D = \{\pm(1, 0), \pm(0, 1), \pm(1, 1)\}$. Both have sixteen vertices, are 6-regular, and have two common neighbors for any two distinct vertices, adjacent or not. For the rook graph these are the two other points on a common row or column, or the two opposite corners when both coordinates differ; for the Cayley graph the same

Table 6: Exact counts and refinement rounds on the 64-vertex boundary example. Pairs and triples are per input graph. Stabilization includes the final no-split update; a dash means no distinction. These are interaction counts, not runtimes.

Refinement	Pairs	Triples	First distinction	Stable round	Separates?
Stitch (auxiliary)	496	5,008	–	2	No
DA	496	5,008	–	2	No
Dense 2-FWL	4,096	262,144	2	5	Yes
Co-Sp	4,096	262,144	2	5	Yes
One-hop Plus	4,096	262,144	2	5	Yes

count is $|D \cap (g + D)| = 2$ for each nonzero $g \in \mathbb{Z}_4^2$, verified directly from the six listed elements. Every vertex neighborhood in the Shrikhande graph induces a six-cycle (at zero its cyclic order is $(1,0),(1,1),(0,1),(-1,0),(-1,-1),(0,-1)$) and therefore contains no K_4. In the rook graph every clique of order at least three lies in one row or one column, so its K_4s are exactly its four rows and four columns. These explicit definitions make both the parameter calculation and the dense witness checkable without a graph database.

J.2 The six stable message types

Let A denote an original-vertex diagonal, B a subdivision-vertex diagonal, C a pair of distinct adjacent vertices in the *base* graph, D a distinct nonadjacent base pair, E an incident original-to-subdivision pair, and F its reverse. No other active pairs occur in this example. Table 7 lists the type sizes and messages, where a coefficient denotes message multiplicity.

Table 7: A common stable partition for auxiliary Stitch and DA on both subdivided graphs. C refers to base adjacency, although the original-vertex pairs are nonedges in the subdivided input.

Type	Size	Joint messages
A	16	$(A,A) + 6(C,C) + 9(D,D) + 6(E,F)$
B	48	$(B,B) + 2(F,E)$
C	96	$(A,C) + (C,A) + 2(C,C) + 3(C,D) + 3(D,C) + 6(D,D) + (E,F)$
D	144	$(A,D) + (D,A) + 2(C,C) + 4(C,D) + 4(D,C) + 4(D,D)$
E	96	$(A,E) + (C,E) + (E,B)$
F	96	$(F,A) + (F,C) + (B,F)$

The strongly regular parameters determine every entry. The atomic partition is coarser than these six types, and the first update separates all six: original and subdivision degrees distinguish the diagonals, skeleton-adjacent original pairs have one extra low pivot, and incidence orientations differ in their diagonal-source positions. The common table then shows that no later update splits a type, which establishes the sparse indistinguishability independently of a finite stopping heuristic.

J.3 The dense witness

If subdivision vertices e, f correspond to disjoint base edges, they have no common neighbor in the subdivided graph, so $(W^2)_{ef} = 0$, while every length-four walk between them corresponds to a base edge joining an endpoint of e to an endpoint of f. Thus $(W^4)_{ef} = 4$ precisely when all four cross edges exist, which gives a K_4 together with e, f. The rook graph has eight such cliques, its four rows and four columns, and each clique supplies three unordered pairs of disjoint edges, or six ordered pairs, for forty-eight witnesses in total; the Shrikhande graph has none. The reproduced graph definitions check these counts directly. The first dense round reads W^2 and the second reads $(W^4)_{ef} = \sum_z (W^2)_{ez}(W^2)_{zf}$ from its joint messages, while endpoint degrees identify the subdivision vertices, so the difference is visible in the graph signature.

On the central R the inherited sparse colors already distinguish base adjacency from nonadjacency, and its actual coherent configuration is the rank-three configuration of the base strongly regular graph. These configurations share their intersection parameters but are not isomorphic, since one graph contains a K_4 and the other does not, and they are therefore nonseparable.

J.4 Reproduction protocol

The repository-root script `experiments/reproduce.py` constructs the two base graphs on $\mathbb{Z}_4^2$, joining rook vertices when one coordinate agrees and using the Shrikhande connection set $\{\pm(1,0), \pm(0,1), \pm(1,1)\}$. It subdivides each edge once, explicitly generates the cubes described in Proposition 22, and deduplicates the resulting triples. Joint exact recoloring assigns integer identifiers to complete tuples consisting of the old color and the sorted multiset of ordered source-color pairs, and the identifier assignment is shared between the graphs. Because old colors are retained, equality of successive joint class counts certifies stabilization of the partition.

The script checks source closure, the common six-type table, the dense walk witness, the exact cycle formula on small cycles, and the cycle and boundary counts reported in Tables 9 and 6, and it records the refinement traces in `experiments/results.json`, also relative to the repository root. These are reproducible witnesses and implementation checks; the general expressivity statements rest on the proofs of the preceding appendices. The larger exact study suite independently reconstructs this example, and Appendix P gives its complete protocol and results.

Scope of the measurements. No trained-model accuracy, GPU memory, preprocessing time, or wall-clock speedup is inferred from these counts. Prediction scores and per-run ZINC-full times appear in Appendix Q, and the separate preliminary timing summary is in Appendix R, whose measurement scope remains to be documented. Such comparisons would require a concrete sparse implementation of (1), matched initialization and network capacity, preprocessing-inclusive timings, and separately reported qualification costs or an independently certified input class.

K Decomposition-log coverage: data, computation, and scope

K.1 Source snapshot and reproducibility

The supplied snapshot consists of 26 decomposition-log files whose filesystem modification dates fall in April 2025. We report 25 non-BREC collections and keep BREC separate as a synthetic diagnostic log. Table 8 lists every reported collection, including the lower-coverage ENZYMES case (Borgwardt et al., 2005; Dobson & Doig, 2003), and the collection names are the source filenames rather than an assertion that all available graph benchmarks were surveyed. The sources can overlap: ZINC and ZINC-full are not independent populations, and FRANKENSTEIN and Mutagenicity (Debnath et al., 1991) have identical aggregate topology counts in this snapshot. We therefore neither sum the rows into a count of unique graphs nor claim that the collections are independent samples.

The reproduction entry point is `experiments/paper_tools/summarize_logs.py`, relative to the repository root; it accepts the log directory and an output directory and emits the JSON audit together with both table fragments. The audit records the SHA-256, byte size, graph count, split and segment counts, per-graph maximum-R histogram, and parsing and reconciliation issues of each input, reading existing graph summaries rather than recomputing decompositions. The 25 reported logs have consecutive per-segment indices, matching index and summary counts, no positive R-count without an R maximum, and R-presence counts agreeing with the supplied segment footers. BREC restarts indices by synthetic family in a format the split parser does not recognize; its raw audit is retained but excluded from the coverage claims and the table.

K.2 Metric and interpretation

With $q(G)$ as defined in Section 5, we report

$$\mathrm{coverage}_{13}(\mathcal{D}) = \frac{\#\{G \in \mathcal{D} : q(G) \le 13\}}{|\mathcal{D}|}.$$

Each line beginning with the per-graph `N` field is one graph record, and split-level minima and maxima are excluded. The field `Rmax` gives the maximum R order of that graph. When no R fields are present and `NR` is not positive, we read the graph as having no R, an interpretation supported by consistency with the R-presence footer of the log. A graph passes the sufficient size test exactly when it has no R or its `Rmax` is at most thirteen, so the test constrains every R in the graph rather than a typical R or

the average R size. Because the summaries omit the size of each individual R, they do not support an R-instance size histogram.

ZINC (Irwin et al., 2012) records 10,000 training, 1,000 validation, and 1,000 test graphs, of which 39, 4, and 7 contain R. QM9 (Ramakrishnan et al., 2014) has 130,831 graph records and 12,562 with R. PCQM4Mv2 (Hu et al., 2020) has 3,228,606 training, 150,000 validation, and 73,545 test records in this snapshot; these counts describe the supplied files rather than all possible releases or test partitions. The audit cannot independently certify the original SPQR implementation, the graph conversion, the hydrogen treatment, or the feature processing, and those preprocessing choices must match any subsequent neural evaluation before the counts are attributed to its exact inputs.

Table 8: Full reported log coverage. "Small R" counts graphs with at least one R and no R larger than thirteen. No-R plus Small R gives the exact qualifying count; percentages are rounded to two decimals. Zero in Max. R denotes absence of R throughout the log.

Log	Graphs	No-R	Small R	Coverage	Max. R
BZR	405	400	5	100.00%	4
COX2	467	467	0	100.00%	0
DHFR	756	756	0	100.00%	0
ENZYMES	600	4	436	73.33%	57
FRANKENSTEIN	4,337	4,024	312	99.98%	14
Mutagenicity	4,337	4,024	312	99.98%	14
aqsol	9,833	9,705	128	100.00%	12
molbace	1,513	1,456	57	100.00%	4
molbbbp	2,039	1,950	89	100.00%	7
molchembl	456,309	449,903	6,402	100.00%	22
molclintox	1,477	1,429	48	100.00%	9
molesol	1,128	1,112	16	100.00%	12
molfreesolv	642	640	2	100.00%	6
molhiv	41,127	39,429	1,690	99.98%	22
mollipo	4,200	4,122	78	100.00%	7
molmuv	93,087	92,102	985	100.00%	7
molpcba	437,929	432,881	5,047	100.00%	14
molsider	1,427	1,373	54	100.00%	9
moltox21	7,831	7,702	129	100.00%	10
moltoxcast	8,576	8,436	140	100.00%	10
PCQM4Mv2	3,452,151	3,435,408	16,743	100.00%	12
Peptides-func	15,535	14,749	786	100.00%	8
QM9	130,831	118,269	12,562	100.00%	8
ZINC	12,000	11,950	50	100.00%	7
ZINC-full	249,456	248,242	1,214	100.00%	9

K.3 Relation to the theorem and to learning experiments

The small-R interpretation explicitly assumes E of Section 4, whereas no-R membership is unconditional. The main theorem compares two unlabelled graphs within $\mathcal{C}_{\mathrm{SS}}$. Neither a successful structural certificate nor this audit asserts identification against every outside-class graph or identification of all possible feature labelings of a single topology. Adding molecular features in a neural model requires a separate, matched experimental comparison, and a labelled theorem requires its own statement and proof; a large R or an unverified general configuration does not by itself show that the method fails.

Structural coverage, exact refinement, and learned prediction are separate forms of evidence (Appendix Q). The documented shared learning settings and the DA score comparison appear in that appendix, and a Stitch hardware evaluation should separately report preprocessing time, per-epoch or per-step time, inference time, peak memory, and retained pair and triple counts. The structural theorem promises neither that optimization reaches the injective realization nor that a smaller interaction count yields a proportional hardware speedup.

L THE COMPLETE ONE-HOP PLUS DEFINITION

STITCH2-FWL PLUS extends each R-skeleton by *one decomposition hop* to retain additional interactions near its interfaces. Two S/R skeletons are neighbors for this purpose if they are directly adjacent in the reduced SPQR tree or connected through the same P-node, which includes every other S/R neighbor of that P-node. For a central R, its extension H_R contains its own vertices, all vertices of each neighboring R, and the entry buffers of each neighboring S. If the entry into S is its virtual edge xy, the buffers are

$$\{x, y, x_-\} \cup \{x, y, y_+\}, \tag{30}$$

where x_- and y_+ are the other cycle neighbors of x and y; for a triangular S the two sets coincide. The extension is applied once and never crosses a cut vertex into another block.

The original one-hop Plus rule also retains both endpoints as pivots for nonadjacent S-roots, so its complete selector can be written compactly as

$$T_G^+ = \widehat{T}_G \cup \bigcup_S \{(a, a, b), (a, b, b) : a, b \in V(S),\ a \neq b\} \cup \bigcup_R V(H_R)^3. \tag{31}$$

All unions are deduplicated, and initialization, joint aggregation, and graph readout are unchanged; the domain is defined for all finite graphs without a $\mathcal{C}_{\mathrm{SS}}$ qualification check, and source closure, termination, and isomorphism invariance follow directly from these generators. The additional interaction count is at most $2\sum_S s_S(s_S - 3) + \sum_R |V(H_R)|^3$, before overlap removal.

On the pair in Proposition 22 the central extension contains all subdivision vertices, so $H_R = V(G)$ and Plus is exactly dense refinement, which restores the distinction on this example. This gives a concrete role to the extra neighborhood interactions without a universal equivalence theorem for the one-hop rule. In particular, complete cubes in overlapping H_R regions do not by themselves prove that their conditional comparisons compose globally, and graph-level equivalence of the one-hop variant on arbitrary inputs remains open. The main theorem and all Stitch results are independent of this supplementary extension.

M NEURAL REALIZATION AND THE DISCRETE COMPARISON

The selector can be used in a pair-state neural architecture with

$$h_{i+1}(a, b) = \phi_i \left(h_i(a, b), \sum_{t \in \widehat{N}_G(a,b)} \psi_i(h_i(a, t), h_i(t, b)) \right), \tag{32}$$

and a permutation-invariant readout over the active pairs. With identical atomic initialization, each layer is bounded by the corresponding discrete refinement, by induction on the pair colors. For the general function class in (32), injective encodings can realize the discrete updates on a finite input collection, and an injective readout can realize their graph signatures. The stable theorem requires sufficient depth and is not a guarantee for an arbitrary fixed-depth or finite-width trained network. Additional structural role features would change the initialization and require a separate comparison, and predictive accuracy, optimization, and realized runtime are not established by this discrete argument. The adjacency-power prediction results in Appendix Q provide separate empirical evidence, while additional initialization features would require a correspondingly specified discrete comparison.

The layer comparison and finite realization. At initialization the neural state is a function g_0 of the atomic pair color. Suppose $h_i = g_i(\widehat{c}_i)$ on active pairs. Equal next-round discrete colors have equal old colors and equal multisets of ordered source-color pairs, so applying the same ψ_i and summing gives equal aggregates and the same ϕ_i gives equal new states; a common function g_{i+1} therefore exists. A permutation-invariant readout of equal state multisets cannot separate equal discrete graph signatures.

For the converse, on a finite input collection and a finite number of rounds the encountered source-state pairs form a finite set. Assign a distinct basis vector to each ordered source-state pair via ψ_i, so the sum is its exact multiset count vector, and choose ϕ_i injective on the finitely many combinations

of old state and count vector encountered. The resulting states represent precisely the discrete colors by induction, and an injective histogram readout represents the graph signature. Taking at least the maximum stabilization depth on that finite collection realizes its stable distinctions. The construction lets the function capacity and depth depend on the finite collection; it establishes an expressive realization for (32) rather than showing that the particular fixed-width DA backbone used in the experiments learns these encodings. □

N Research context and comparative analysis

This appendix develops the comparison of Section 8. It separates three choices: the order of a state, the range of its updates, and the structural regions in which those updates are performed. A guarantee must also specify its target, whether recovering a graph property, counting selected patterns, matching a refinement, or identifying all graphs in a class; these targets cannot substitute for one another.

N.1 State order and interaction cost

The foundational correspondence between message passing and 1-WL gives a precise baseline for graph distinction (Gilmer et al., 2017; Xu et al., 2019; Morris et al., 2019), and higher-order tuple models strengthen that baseline. PPGN, in particular, uses second-order states and matrix multiplication to obtain 3-WL graph-distinguishing power (Maron et al., 2019), which separates representation order from interaction cost: a pair-state model can store quadratically many states while still combining cubically many pair–pivot triples in a direct update. Morris et al. (2023) survey the broader landscape, including invariant and equivariant architectures, and further surveys cover expressivity and higher-order models (Zhang et al., 2023a; Besta et al., 2026). The correspondence with logic was made precise for message passing and for higher-order refinement (Barceló et al., 2019; Grohe, 2021), and systematic comparisons of GNN extensions appear in Papp & Wattenhofer (2022); Geerts & Reutter (2022).

Stitch addresses the retained triple domain rather than increasing state order. Its bounds count interactions in the explicit refinement, not a lower bound on every possible implementation of dense 2-FWL, so they make no claim against fast matrix multiplication, numerical kernel optimizations, or alternative neural implementations. Similarly, the ideal injective realization in Appendix M is separate from predictive performance at a fixed network width and depth.

N.2 Message redundancy and structural encodings

RFGNN propagates bounded-length path information through redundancy-free message flows (Chen et al., 2022), and its notion of message-flow redundancy concerns repeated propagation along paths. Stitch instead removes selected pair–pivot interactions and proves when the resulting graph signature retains dense distinctions, so the two mechanisms should be distinguished even though both reconsider the computation performed by message passing.

Chen et al. (2025) study local connectivity, spectral, and distance encodings for enhanced subgraph learning in 2-FWL GNNs; classical comparisons of combinatorial and spectral invariants provide an earlier perspective on such encodings (Fürer, 2010). Graph transformers take a complementary route, injecting structure through positional or structural encodings rather than through higher-order states, and their expressivity has been analyzed relative to message-passing baselines (Ma et al., 2023; Black et al., 2024; Zhu et al., 2023). That encoding perspective complements the interaction-domain perspective studied here. Our discrete theorem fixes atomic initialization, whereas the learned model uses the adjacency-power initialization documented in Appendix Q; we do not attribute that experimental feature specification to an equivalence result for arbitrary structural encodings.

N.3 Local refinement and replacement rules

Morris et al. (2020b, Section 3, Theorem 2) prove that counting-augmented local δ-k-WL matches their global δ-k-WL on connected graphs, where the augmentation records color multiplicities beyond the local replacement neighborhood. This result therefore cannot be read as permission to delete all nonlocal messages from an unchanged dense update, although their unrolling-tree argument is an

important precedent for relating local refinement to global information. Stitch's additional obligation is to recover the particular block–cut and SPQR responsibilities from its own retained messages.

SpeqNets controls computation through tuples whose induced supports have a bounded number of connected components (Morris et al., 2022, Section 3), and its support constraint provides a structural notion of locality. Distance-restricted FWL instead truncates the pair domain by shortest-path distance and proves short-cycle counting guarantees (Zhou et al., 2023, Sections 3–4). These examples illustrate two different savings, reducing represented states and reducing the updates available to each state, whereas Stitch uses complete pair domains inside its S/R units but reduces the S pivot domain.

The (k, t)-FWL$^+$ framework permits general equivariant replacement sets (Feng et al., 2023, Section 3). Its N^2-FWL instance uses joint two-vertex replacements, and for a sufficiently large neighborhood parameter Corollary 4.2 guarantees at least 3-WL power with quadratic state space, which establishes strong global expressivity with a local replacement mechanism. The specific contribution of Stitch is to establish sufficient conditions for deleting interactions from the single-pivot 2-FWL update according to connectivity pieces. We neither claim that connectivity is the only successful notion of locality nor infer a universal expressivity ordering between these update mechanisms.

N.4 Rooted views, subgraph communication, and counting

Nested GNNs replace rooted-tree views by rooted subgraphs (Zhang & Li, 2021), ESAN formalizes equivariant processing of subgraph collections including information exchange across them (Bevilacqua et al., 2022), and SUN derives a broader node-based architecture from the symmetries of those collections while establishing a 3-WL upper bound for the studied family (Frasca et al., 2022). All three include mechanisms for communicating across subgraph views.

The SWL hierarchy makes the remaining distinction precise (Zhang et al., 2023b, Theorem 5.2): for its specified schemes, the strongest node-marking variant SSWL is strictly weaker than symmetrized localized FWL(2), which is itself strictly weaker than dense FWL(2). These are statements about unrestricted inputs and particular algorithms, so they neither preclude equivalence on a restricted graph class nor apply to every possible subgraph policy or higher-order encoder. Stitch provides such a class-conditioned comparison for a different selector, using explicit shared one- and two-vertex interfaces.

Substructure counting asks which graph statistics survive a representation (Chen et al., 2020; Kanatsoulis & Ribeiro, 2024). Substructure-aware GNNs and subgraph-isomorphism counting inject local pattern statistics directly (Zeng et al., 2023; Bouritsas et al., 2023); I^2-GNN uses additional identifiers to improve cycle counting (Huang et al., 2023), ESC-GNN precomputes structural information to reduce repeated subgraph computation (Yan et al., 2024), and Loopy WL studies a hierarchy with cycle and cactus-homomorphism counting guarantees (Paolino et al., 2024). Such criteria can expose distinctions a fixed WL test misses, so counting power should not be presented as uniformly below 3-WL; matching a specified list of counts does not, however, establish equivalence to every graph distinction of dense 2-FWL. Stitch's S reconstruction concerns a cycle together with its recursively attached branches, and its guarantee is a statement about the composed graph.

N.5 Recognizing connectivity and allocating computation by connectivity

Block–cut and SPQR trees provide the classical decomposition structure used here (Holm et al., 2018; Fink & Rutter, 2023; Holberg, 1992). The general graph-isomorphism problem is instead solved by dedicated algorithms rather than by refinement (Babai, 2016; McKay & Piperno, 2014); Stitch keeps refinement as the decision procedure and uses decomposition only to choose where to refine. For dense refinement, Kiefer et al. (2019); Kiefer & Neuen (2022) relate graph identification to colored component information and show how WL detects the relevant decomposition structure, and GD-WL further demonstrates that distance information can encode biconnectivity properties (Zhang et al., 2023c). Distance-based representations can therefore encode separator information, and the architectural question in this paper is whether the computation selected at and around those separators preserves the desired distinctions.

PlanE is a particularly close conceptual predecessor (Dimitrov et al., 2023, Sections 4–6). It uses block–cut and SPQR encoders, informed by planar canonical representations, to learn complete

invariants of planar graphs, and it already supports the principle that suitable local descriptions can be composed along a decomposition. Stitch instead obtains its local descriptions from retained common-pivot messages and requires a separate dense-color readability argument for the selector, and its class is defined by the actual colored rigid configurations rather than by planarity. These differences concern the available information and the proof obligations; they do not establish dominance over PlanE.

N.6 The precise advance over Co-Sparsify and DA

Co-Sparsify supplies the direct block-based preservation result (Chen et al., 2026), and its original construction also keeps component-wide pair states and degenerate messages, so it is not accurately described by a union of block cubes alone. DA goes further by retaining SPQR-induced supports, shared interfaces, and its specified repeated-coordinate messages (Anonymous Authors, 2026). Connectivity-based routing and the use of separator pairs are therefore inherited ingredients.

The new comparison has two parts. Within S-skeletons, complete cubes are reduced to the exact neighbor selector, whose quadratic interaction count is accompanied by a reconstruction argument; around R-skeletons, the subdivided-graph example shows that complete internal cubes need not retain all relevant joint information. The actual colored-R conditions bridge that gap for $\mathcal{C}_{\mathrm{SS}}$. Separability and Schurianity are established notions (Evdokimov & Ponomarenko, 2000; Fuhlbrück et al., 2021), used here in two distinct roles, namely realizing a local relation correspondence and aligning its prescribed ordered interface. For the small-order certificate, the enumeration premise applies to the colored coherent configuration itself.

Finally, the target is graph-level 3-WL expressivity: the theorem compares graph signatures and constructs a whole-graph isomorphism within $\mathcal{C}_{\mathrm{SS}}$, without requiring equality of all sparse and dense pair partitions. The counterexample prevents extending that conclusion to the unrestricted auxiliary selector, and the original one-hop Plus variant repairs the stated example without closing this general gap. The results thus specify a preservation regime and an explicit obstruction outside it, which Plus addresses through additional R-neighborhood interactions.

O When Stitch already suffices

The presence of rigid skeletons does not by itself make Plus necessary, and thirteen vertices is a sufficient threshold rather than the boundary of Stitch's capability. We collect sufficient conditions under which base Stitch preserves dense graph distinctions. For the class-based statements, both inputs are finite simple unlabelled graphs in the stated class, as in Theorem 3. "Suffices" refers to preservation of dense 2-FWL graph distinctions; it is not a prediction that a trained neural model will attain a particular task score.

O.1 No rigid skeletons: unrestricted size and depth

All no-R graphs belong to $\mathcal{C}_{\mathrm{SS}}$ without premise E, by Corollary 6. Examples include trees, forests, cycles, cactus graphs, and more generally graphs of treewidth at most two. The guarantee places no bound on graph order, cycle length, the number of blocks or skeletons, branching degree, or decomposition depth. In particular, long cycles and many interconnected S/P units do not require the R-neighborhood extension. The proof reconstructs whole cycles together with their attachments and combines them along the responsibility structure; it does not merely count short cycles.

O.2 Small rigid skeletons: many R units remain admissible

Under the explicit enumeration premise E, define

$$\mathcal{C}_{\leq 13} = \{G : \ |V(R)| \leq 13 \text{ for every R-skeleton of } G\}.$$

The small-R consequence of Theorem 3 gives $\mathcal{C}_{\leq 13} \subseteq \mathcal{C}_{\mathrm{SS}}$. The bound concerns each R separately, not the sum of their sizes or the number of interface vertices in the whole graph. It allows arbitrarily many R-skeletons, long S-cycles, and arbitrarily deep chains or branching patterns in a valid reduced decomposition.

For example, arbitrarily many K_5 or $K_{3,3}$ pieces can be joined at cut vertices, together with bridges and cycles. Their R-skeleton orders remain five or six, although the whole graph can be arbitrarily large and nonplanar. Legal SPQR compositions whose R-skeletons all have at most thirteen vertices are covered as well; one checks the actual reduced decomposition rather than assuming that any gluing preserves the original pieces. The sufficient regime is therefore restricted neither to planar graphs nor to graphs with few rigid pieces, and all such comparisons use base Stitch without invoking Plus.

This certificate also has a computational interpretation. If $|V(R)| \le r_0$ for every R, then

$$\sum_R |V(R)|^3 \le r_0^2 \sum_R |V(R)|.$$

For fixed r_0 the rigid contribution is linear in the number of R-bag vertex occurrences, while cycle contributions remain quadratic in their sizes and preprocessing and refinement depth remain separate costs. The same local size parameter therefore informs both a sufficient expressivity condition and the allocation of computation.

O.3 Large rigid skeletons: size is not the general criterion

The definition of $\mathcal{C}_{\mathrm{SS}}$ checks the actual colored configuration $\mathcal{X}_R(G)$, not merely the order or the uncolored isomorphism type of R. Any size of R is allowed when this configuration is separable and Schurian. A simple sufficient certificate is that every stable diagonal color class of $\mathcal{X}_R(G)$ is a singleton: then every ordered pair is its own basic relation, so the configuration is discrete, its algebraic correspondences are realized by the induced vertex bijections, and its pair classes are the orbits of the identity group. Both required properties follow. If every R in a graph has such a certificate, the main theorem applies without E and without a size bound.

The distinguishing colors must come from the specified refinement and the actual skeleton relations; unique vertex identifiers cannot be added to manufacture this certificate. More general separable and Schurian configurations are also admitted by Definition 1. Conversely, failing the small-size test or this singleton test does not show that a configuration fails the general condition. There is consequently no rule that "an R larger than thirteen requires Plus."

O.4 A whole-graph R already receives dense computation

A separate observation applies without any configuration qualification. If the input itself is a single R-skeleton, then the auxiliary Stitch selector contains $V(G)^3$ and is a subset of that same set, hence

$$\widehat{T}_G = V(G)^3, \qquad \widehat{P}_G = V(G)^2.$$

Its update, initialization, and readout are exactly those of dense 2-FWL, so the two procedures agree at every round. For any two such inputs, auxiliary Stitch and dense refinement therefore have identical distinguishing power, regardless of whether either R is identifiable or Schurian. This is an update-level fact outside the restricted-domain definition of Stitch2-FWL where necessary. It gives no sparsification saving on that input and does not assert that 2-FWL identifies every three-connected graph. With no neighboring skeleton to add, the one-hop extension provides no additional interactions.

O.5 Practical interpretation and the role of Plus

The structural audit in Section 5 checks the easy size certificate at the graph level. Every recorded graph in ZINC, QM9, ZINC-full, PCQM4Mv2, and Peptides-func passes it. These are coverage claims for the supplied topologies under the documented parsing and preprocessing scope; the small-R implication uses E and the class theorem retains its two-sided comparison condition. They are not claims about every dataset release or every feature labeling.

A practical order of checks is:

1. If the graph has no R, use the no-R guarantee.
2. If all R-skeletons have at most thirteen vertices, use the size certificate under E.

3. For larger R-skeletons, seek a certificate for the actual colored configurations; the theorem imposes no size cutoff.
4. If no certificate has been obtained, record the guarantee as unverified. This does not prove failure or make Plus mandatory.

The single-R dense case provides an additional direct comparison when applicable. The expensive general qualification check is a way to certify a theorem, not an extra operation fed into the sparse update.

Plus addresses additional R-neighborhood relations outside the demonstrated preservation regime, including the explicit boundary example. It is not a prerequisite for the no-R, small-R, or general qualifying-configuration guarantees. Nor does moving to Plus presently provide a proved universal fallback, since its original one-hop equivalence remains an open question in this paper. The positive conclusion is substantial without that claim: Stitch's sufficient regimes admit unrestricted total size, long cycles, many rigid pieces, and deep compositions, and a readily checked regime covers all recorded inputs in several molecular collections.

P EXACT EXPERIMENT SETTINGS AND COMPLETE RESULTS

This appendix documents the controlled long-S study, the enumerated small-graph and official BREC comparisons, and the independent rerun of the boundary example. All results use the original selectors, including the original one-hop Plus in the boundary study; no block-completion variant is used. The experiments below rerun neither the real-data decomposition audit nor any neural training, and the separately supplied prediction results under the DA protocol are documented in Appendix Q.

P.1 INPUTS AND SELECTION

Controlled long S. The `geng` enumeration covers all unlabelled connected graphs on four through eight vertices, namely 6, 21, 112, 853, and 11,117 graphs, for 12,109 in total; filtering to three-connected graphs leaves 2,545 cores. Within common stable 1-WL buckets we retain the 144 core pairs distinguished by Dense, so the controlled study tests preservation of known dense distinctions under subdivision rather than the fraction of arbitrary graph pairs that Dense distinguishes. All 144 pairs are included at every $q \in \{0, 1, 2, 4, 8, 16, 32\}$. For a core with n_0 vertices and m_0 edges the expanded input has $n_0 + qm_0$ vertices and $(q + 1)m_0$ edges, because the core edge is replaced by the path rather than kept as an additional true edge.

The attachment check uses the triangular-prism/$K_{3,3}$ pair at $q = 0, 1, 2, 4$ with three variants: no attachment, a length-two pendant path at every original core vertex, or a K_4 block sharing that vertex and carrying a length-two pendant path at one of its new vertices. This gives twelve cases, including the four unadorned controls. Attachments follow the same rule on each graph without assuming a named correspondence between the cores, and every method distinguishes every case while one independent random relabelling check per case and method passes.

Small-graph expressivity. The same four-to-eight-vertex enumeration supplies all 332 pairs of distinct graphs sharing a common stable 1-WL signature, and no Dense or Stitch outcome is used to select this set. All four methods distinguish all 332 pairs, and two random relabellings per case and method, each with a newly computed decomposition, pass. The controlled core pairs and the small-graph pairs are related populations, so they are not combined into an independent-sample accuracy estimate.

Official BREC. We use the 400 graph pairs in `brec_v3.npy` from the official BREC Release archive (Wang & Zhang, 2024). The array contains 51,200 graph6 records, 25,600 comparison records and 25,600 reliability-control records. Each pair has 32 relabelled comparisons and 32 reliability comparisons, and all are checked for every method, with no pair or official check omitted. These are exact-refinement checks on the benchmark graphs; no neural training, RPC threshold, or Hotelling T^2 statistic is used.

P.2 EXACT REFINEMENT AND COST DEFINITIONS

We use the Sage Hopcroft–Tarjan SPQR backend (Hopcroft & Tarjan, 1973), retain true and virtual edge instances separately, and validate the decomposition representation. Initial colors encode only equality and original adjacency, and at every round the new color includes the old color together with the complete multiset of ordered source-color pairs under a common pivot. For each method the two compared graphs share the encoding dictionary; Co-Sp uses its component-grouped diagonal/off-diagonal readout, while the other methods read the multiset of all active-pair colors. Appendix H specifies these baseline domains.

For case $i = (G_i, H_i)$ and method M, define

$$I_{M,i} = |\mathcal{T}_M(G_i)| + |\mathcal{T}_M(H_i)|, \qquad K_{M,i} = T_{M,i} I_{M,i},$$

where $T_{M,i}$ counts the joint refinement updates through stabilization, including the final update with no class split and excluding initialization. For any stated case collection $\mathcal{I}$ the reductions relative to DA are

$$1 - \frac{\sum_{i \in \mathcal{I}} I_{\mathrm{Stitch},i}}{\sum_{i \in \mathcal{I}} I_{\mathrm{DA},i}}, \qquad 1 - \frac{\sum_{i \in \mathcal{I}} K_{\mathrm{Stitch},i}}{\sum_{i \in \mathcal{I}} K_{\mathrm{DA},i}}.$$

Input graphs appearing in several pairs contribute once per comparison occurrence, so these denominators match the experiment rather than counting unique underlying graphs. The domains are fixed through refinement, and overlapping selector contributions are deduplicated before counting. For reference, on a standalone s-cycle Stitch retains $4s^2 - 3s$ interactions and s^2 pair states, while Dense and DA retain s^3 interactions and s^2 states.

Table 9: Exact single-cycle domain counts. These are analytic counts, separate from the measured expanded-core cases below.

Cycle size	Dense / DA	Stitch	Retained
6	216	126	58.33%
16	4,096	976	23.83%
64	262,144	16,192	6.18%
256	16,777,216	261,376	1.56%

P.3 Complete long-S grid

Table 10 reports every subdivision level, and at each level all four methods distinguish all 144 pairs. For $q \geq 2$ both the per-round reduction and the cumulative improvement hold in each of the 144 cases, with no exceptions hidden by aggregation. The range of stabilization rounds is taken across cases and is not a confidence interval.

Table 10: Complete long-S grid, with 144 pairs in each row. Reductions are relative to DA and aggregate both inputs of every pair. "Strict" counts pairs with smaller Stitch domains on both sides. No S is present at $q = 0$.

q	S order	Strict	Per-round reduction	Cumulative reduction	DA rounds	Stitch rounds
0	–	0	0.00%	0.00%	3–4	3–4
1	3	0	0.00%	0.00%	5–7	5–7
2	4	144	13.75%	13.76%	5–7	5–7
4	6	144	37.62%	27.72%	5–7	6–8
8	10	144	61.57%	43.28%	5–7	8–10
16	18	144	78.39%	54.39%	5–7	12–14
32	34	144	88.44%	60.98%	6–7	20–22

At $q = 32$ the aggregate retained pair counts are 5,370,418 for both DA and Stitch, and their aggregate per-round interaction counts are 182,714,390 and 21,117,218, for cumulative counts of 1,152,759,428 and 449,842,242 respectively. Dense and Co-Sp coincide on these biconnected inputs and retain 43,701,955,862 interactions per round in aggregate. The extra Stitch reduction is therefore a reduction within the skeleton computation already selected by DA; it claims no additional pair-state memory reduction over DA in this family.

P.4 BREC categories and structural strata

Table 11 retains every official category, including those on which all four methods leave all pairs indistinguishable. Agreement here means equality of the pairwise stable decision, not equality of internal color partitions across different algorithms.

Every method has the same decisions in each stratum. The 188 pairs outside the simple size certificate have at least one input with an R of order greater than thirteen, which does not imply failure of the general separability/Schurian qualification. On the full BREC set the aggregate per-round interaction counts are 145,537,140 for DA and 145,325,118 for Stitch, an additional reduction of 0.15%; on the 96-pair strict-reduction subset they are 64,734,834 and 64,522,912, a reduction of 0.33%. BREC therefore primarily tests retained distinctions, whereas the controlled long-S experiment measures the savings specifically targeted by the new selector. Among the small-graph cases 19 pairs have both inputs no-R, all 332 meet the small-R criterion, and ten have strict reduction on both inputs; all are distinguished, and these small-graph strata likewise overlap.

Table 11: Complete BREC category results. Columns report distinguished graph pairs.

Category	Pairs	Dense	Co-Sp	DA	Stitch
Basic	60	60	60	60	60
Regular	100	50	50	50	50
Extension	100	100	100	100	100
CFI	100	60	60	60	60
4-vertex condition	20	0	0	0	0
Distance regular	20	0	0	0	0
Total	400	270	270	270	270

Table 12: BREC structural strata. The no-R row is contained in the size-certificate row; the strict-reduction row overlaps the other rows. They must not be added. The size criterion is checked on both inputs, and its interpretation as a guarantee retains premise E.

Stratum	Pairs	Dense	Stitch	Agreement
Both inputs no-R	4	4	4	4/4
Both max-R ≤ 13 (includes no-R)	212	212	212	212/212
Outside the simple size certificate	188	58	58	188/188
Strict reduction on both inputs	96	72	72	96/96

P.5 Boundary check and original one-hop Plus

The independent study runner reconstructs the 64-vertex pair in Proposition 22, computes its Sage decomposition, and evaluates Dense, Co-Sp, DA, Stitch, and the original one-hop Plus. The counts and outcomes agree with the direct boundary construction in Appendix J. Dense, Co-Sp, and Plus first distinguish at round two and stabilize at round five, whereas DA and Stitch remain indistinguishable and stabilize at round two. The independently computed walk witness is 48 versus zero, and two random relabelling checks pass for each method; Table 6 gives the per-graph counts. Here the one-hop H_R covers the whole graph, so the repair uses the full dense domain; it is evidence neither of a sparse computational saving for Plus on this example nor of its general equivalence.

P.6 Implementation checks and reproducible artifacts

The run uses Python 3.12.13, Sage 10.8, NetworkX 3.6.1, and a Cython 3.2.4/C++17 refinement backend on macOS arm64. Full signatures are compared after hash lookup, and sorting and run-length encoding preserve every joint-message multiplicity; tests deliberately force hash collisions and compare native with Python round-by-round colors. Dense pivot domains are represented implicitly, without allocating cubic Python objects, while their reported interaction counts remain the complete cubic counts. Signature caching is limited to 64 MiB, with uncached signatures recomputed exactly.

Identical domains may share an exact transition computation, certified from the nested domains on each input and, for Co-Sp readout reuse, from the connected-input condition. This reuse is performed within each actual graph comparison rather than by assuming official relabelled graphs are isomorphic and skipping their checks, and such rows report no independently measured runtime. The results therefore do not support a wall-clock comparison between these implementations.

The recorded suite has 36 passing tests, covering source closure, exact domain construction, native/Python agreement, Sage and reference decomposition checks, and detection of deliberately corrupted official controls. The domain and count tests cover all 1,253 unlabelled graphs on at most seven vertices, including empty and disconnected graphs, and the small-graph, attachment, and boundary results agree with the earlier Python artifacts. The completion verifier checks expected case IDs, absence of duplicates, input sizes, stable statuses, cumulative-count arithmetic, official checks, and source-version consistency. Every reported case completed, and no resource-limited result is treated as indistinguishable. All full configurations use a 512-round limit, a 3,600-second limit per refinement call, and per-input domain limits of 2,000,000 pairs and 2,000,000,000 triples.

Q PREDICTION UNDER THE DA EXPERIMENTAL PROTOCOL

Q.1 MATCHED ARCHITECTURE AND OPTIMIZATION

To compare the interaction selectors under the same learning setup, Stitch follows the dataset-specific architecture, features, optimization, splits, and training protocol of DA-PPGN (Anonymous Authors, 2026). Only the retained pair–pivot interaction domain changes; the neural backbone and the task-level settings are shared. The configuration follows the "DA-PPGN Architecture" and "Neural Evaluation Protocol" sections of the DA supplement.

Pair initialization and readout. Let $\widetilde{A} = D^{-1/2}AD^{-1/2}$, with zero inverse degree for isolated vertices and $\widetilde{A}^0 = I$. The structural channels of an active pair are

$$\mathbf{s}_{u,v} = [(\widetilde{A}^0)_{u,v}, \ldots, (\widetilde{A}^q)_{u,v}].$$

We use $q = 22$ for both ZINC settings and $q = 18$ for TU, giving 23 and 19 structural channels respectively. Pair initialization combines a learned projection of these channels with node features on the diagonal and edge features on true edges, while virtual edges receive no edge features. The backbone forms same-pivot messages by an elementwise product of two source-pair MLP outputs, sums over the retained pivots, and applies the DA signed square-root transform with $\epsilon = 10^{-6}$ before a residual update MLP. Readout concatenates the sum of diagonal states with the off-diagonal sum divided by $|V|$, followed by an output MLP.

Table 13: **Shared Stitch/DA learning settings.** ZINC-full is the setting called ZINC-250k in DA. All tasks use the same diagonal-sum/off-diagonal-average readout. q is the highest adjacency power, including power zero.

Dataset	q	Width	Layers	Batch	Epochs	Warm-up	Weight decay
ZINC-12k	22	80	10	32	2000	50	10^{-5}
ZINC-full	22	80	10	128	2000	50	10^{-5}
TU	18	64	4	32	200	10	0

All tasks use AdamW (Kingma & Ba, 2015) with initial learning rate 10^{-3} and minimum rate 10^{-6}, linear warm-up followed by cosine decay, zero dropout, and gradient clipping at 1.0. There is no gradient accumulation, and the data loader uses zero workers. ZINC uses categorical node and edge embeddings with vocabularies $21/4$ for ZINC-12k and $28/4$ for ZINC-full together with an L_1 loss. TU uses linear encoders for available attributes and dummy embeddings otherwise, with binary cross-entropy as its loss except for six-class ENZYMES, which uses cross-entropy.

Q.2 SPLITS, HARDWARE, AND RESULT PROVENANCE

ZINC uses the official splits, and TU uses ten stratified folds with split seed 42. Training seeds follow $1000, 2000, \ldots$, and the supplied ZINC-full records contain the five seeds from 1000 through 5000. The reference environment is Ubuntu 22.04 LTS with an Intel Core i7-14700K CPU, 64 GB RAM, and an NVIDIA RTX 4090 with 24 GB memory. The software versions are Python 3.12, PyTorch 2.6, PyTorch Geometric 2.6 (Fey & Lenssen, 2019), CUDA 12.6, and SageMath 10.6 for the block–cut and SPQR decomposition.

Table 2 combines Stitch's supplied summaries, `tud_summary.tex`, `zinc_summary.tex`, and `zincfull.md`, with the DA column of the reference manuscript's graph-level prediction table. The DA values are taken from that manuscript; we did not rerun the DA baseline. All five source files, their checksums, the protocol transcription, and a table-generation script are preserved in `experiments/prediction/`. The neural results remain separate from the exact-refinement receipt in Appendix P.

Q.3 UNITS AND AGGREGATION

For ZINC-12k we preserve the supplied mean 0.05032 and SD 0.001386, rounding the latter to 0.00139 in the table. Both TU means and SDs are multiplied by 100. The supplied SDs round to

1.28, 1.13, 1.37, and 4.76 percentage points for NCI1, NCI109, FRANKENSTEIN, and ENZYMES respectively, and table entries use two decimals with ties rounded up. TU results are ten-fold means and SDs under the shared protocol. Because the reference specifies no SD denominator for TU or ZINC-12k, the supplied SDs are retained without conversion to another convention.

Five-seed ZINC-full results. Table 14 preserves the supplied rows and recomputes each error column using

$$\bar{m} = \frac{1}{5}\sum_{i=1}^{5} m_i, \qquad s = \sqrt{\frac{1}{4}\sum_{i=1}^{5}(m_i - \bar{m})^2}.$$

The test mean and sample SD are 0.0190584 and 0.00248071, consistent with the supplied summary 0.01906 ± 0.00248, and the validation mean and sample SD are 0.0147002 and 0.00068469, with recomputations limited to the precision of the displayed seed rows. The "normalized MAE" column of the source is retained for provenance; its definition is not supplied and it is not the main task score.

Table 14: **Stitch's supplied ZINC-full runs under the DA protocol.** Seed rows retain source precision; mean and sample SD are recomputed. "Normalized MAE" and "best epoch" retain the source labels. Times are per-run wall-clock hours.

Seed	Test MAE	Val. MAE	Normalized MAE	Best epoch	Run time (h)
1000	0.022568	0.015087	0.011017	1993	36.3
2000	0.019052	0.014934	0.009300	1963	39.5
3000	0.019102	0.013791	0.009324	1978	36.3
4000	0.015552	0.014211	0.007592	1999	39.8
5000	0.019018	0.015478	0.009284	1969	36.4
Mean	0.019058	0.014700	0.009303	1980.4	37.66
Sample SD	0.002481	0.000685	0.001211	—	—

The mean recorded best epoch is 1980.4. Per-run times range from 36.3 to 39.8 hours and sum to 188.3 hours using the rounded records. This sum is accumulated run wall time, not the elapsed time of a parallel campaign or a measured speedup over DA. The memory and timing reductions of the DA paper are not Stitch measurements, and the separate preliminary timing summary is reported in Appendix R.

Q.4 Interpretation and reproducibility limits

The comparison fixes the dataset-specific learning setup while changing the interaction domain. On ZINC-12k Stitch's mean is 0.05032 against the reported DA mean 0.050, and on ZINC-full the means are 0.01906 and 0.018. Stitch's reported TU means are lower than DA's by 0.78, 1.07, 0.33, and 7.83 percentage points on NCI1, NCI109, FRANKENSTEIN, and ENZYMES respectively. These are descriptive differences, not tests of statistical significance or non-inferiority; in particular, the ENZYMES result does not support a claim that prediction is preserved on every dataset.

Full run-level reproducibility would still require an explicit checkpoint-selection rule, the TU inner-validation allocation, the ZINC-12k repetition count, and the SD conventions for summary-only results, none of which the reference provides. The auxiliary normalization and epoch-index convention of the supplied full-ZINC log also remain unspecified. Table 13 records the shared settings that are documented, and these missing details limit exact run-level reproduction. The empirical initialization uses adjacency-power channels, so the atomic-initialization comparison in Appendix M applies only under its stated assumptions.

R REPORTED RUNTIME MEASUREMENTS

Table 15 preserves the supplied timing summary in full. Each cell lists Co-Sp, DA, and Stitch in that order, all times are in seconds, and the source reports preprocessing, refinement, and total time separately, rounded to two decimals. We retain the supplied totals rather than recomputing them from rounded components.

Table 15: **Complete reported timing summary.** Each cell follows Co-Sp / DA / Stitch. The final column retains the source values; its precise definition is pending confirmation. Values below one are consistent with relative time, not the usual inverse-time speedup factor.

Dataset	Preprocessing (s)	Refinement (s)	Total (s)	Source ratio
FRANKENSTEIN	0.11 / 0.24 / 0.24	0.20 / 0.04 / 0.04	0.30 / 0.28 / 0.28	1.00 / 0.19 / 0.18
NCI1	0.24 / 0.40 / 0.39	0.52 / 0.08 / 0.07	0.76 / 0.48 / 0.46	1.00 / 0.15 / 0.14
NCI109	0.26 / 0.40 / 0.39	0.52 / 0.08 / 0.07	0.78 / 0.48 / 0.46	1.00 / 0.15 / 0.13
ENZYMES	0.51 / 0.65 / 0.65	0.92 / 0.24 / 0.24	1.43 / 0.89 / 0.89	1.00 / 0.26 / 0.26
ZINC-12k	0.12 / 0.29 / 0.29	0.22 / 0.05 / 0.04	0.33 / 0.34 / 0.34	1.00 / 0.22 / 0.21
ZINC-full	0.12 / 0.30 / 0.30	0.22 / 0.05 / 0.05	0.33 / 0.35 / 0.35	1.00 / 0.22 / 0.21
QM9	0.07 / 0.24 / 0.24	0.11 / 0.03 / 0.03	0.18 / 0.27 / 0.27	1.00 / 0.24 / 0.23

Interpretation. Stitch's reported refinement time is lower than Co-Sp's on all seven datasets and close to DA's at the displayed precision. The total includes preprocessing: Stitch has a smaller reported total on the four TU datasets but a larger one on both ZINC settings and QM9, so the refinement-stage advantage does not imply a total-time advantage on every dataset. The source ratio column is neither recomputed from rounded times nor converted into claimed speedup factors.

Measurement scope and provenance. The supplied summary does not specify whether each time is per graph, batch, iteration, or dataset, nor whether refinement refers to the discrete algorithm or to a neural forward computation. It also lacks the hardware, repetition count, variability, and aggregation rule of this timing experiment. These details must be supplied before the table can support a reproducible hardware comparison, and the prediction environment in Appendix Q is not assumed to be the timing environment. Raw records are saved in `experiments/runtime/reported-runtime.json` and `experiments/runtime/reported-runtime.csv`, relative to the manuscript directory.