Scaling Exact Substructure Extraction Beyond the Five-Vertex Graphlet Wall

Mohammad Dindoost, Bartosz Bryg, and David A. Bader
Department of Computer Science
New Jersey Institute of Technology
Newark, NJ, USA
[md724,bb474,bader]@njit.edu

***Abstract*—Exact substructure counts make graph neural networks (GNNs) provably more expressive than the first-order Weisfeiler–Leman (1-WL) limit on message passing, but two practical walls confine them to small graphs: combinatorial graphlet counters stop at a fixed catalog of patterns on at most five vertices, and exact extraction at scale was widely treated as too costly. We treat exact substructure-feature extraction as a parallel-systems problem. Using HiPerMotif, an edge-centric parallel subgraph-isomorphism engine in the open-source Arkouda/Arachne framework, we convert isomorphism mappings into per-vertex *orbit* features by normalizing with each pattern's automorphism group. On a single 128-core shared-memory node, the extraction strong-scales with counts bit-identical across thread counts, reaches a multi-million-vertex, 10^8-edge graph (a single-node capacity result), and passes the five-vertex graphlet wall by extracting patterns no fixed-catalog counter expresses, such as induced long cycles (C_6–C_8); we chart this coverage boundary against ORCA, ESCAPE, and PGD. Two reproducible constructs ground the expressivity payoff: the ten-class circulant skip-link (CSL) benchmark, whose 1-WL-identical classes cannot be fully separated by any size-≤ 5 graphlet system but can be separated by induced long cycles, and the cospectral Shrikhande/4×4-rook pair, which a single clique count distinguishes though 1-WL and the adjacency/Laplacian spectrum cannot. The two results demonstrate separate capabilities, each shown in its own regime, not their intersection: the size-4 orbit features give a capacity-independent accuracy benefit on structure-driven graph classification, validated against a dimension-matched random control, with strong native features able to mask this benefit, while HiPerMotif gives feasible at-scale extraction of the beyond-wall patterns. The size-4 orbit counts are verified against the ORCA oracle and the long cycles against an independent enumerator.**

***Index Terms*—parallel graph analytics, subgraph isomorphism, motif counting, graph neural networks, Weisfeiler–Leman, expressivity**

I. Introduction

Substructure statistics, the counts of small patterns each vertex participates in, carry the signal in many graph domains, from functional motifs in connectomes and biological networks to recurring patterns in social and cybersecurity graphs. They are also what the dominant graph-learning models cannot see on their own. Message-passing graph neural networks (GNNs) have a well-characterized expressivity limit because their power to distinguish non-isomorphic graphs is bounded by the first-order Weisfeiler–Leman (1-WL) test [1], [2]: two graphs with the same 1-WL coloring are indistinguishable regardless of depth, width, or training, and as a direct consequence such networks cannot count many simple substructures, including triangles, cycles beyond length three, and cliques [3]. Making these counts available as features is therefore well motivated.

The established remedy is to inject substructure information directly into the model: Graph Substructure Networks and related approaches augment each vertex with counts of its participation in a fixed family of small patterns, provably exceeding 1-WL [4]. The bottleneck is not the model but the *counting*: prior work evaluates exact counts almost entirely on small molecular graphs, and at larger scale learned counters approximate them in place of exact computation [5], trading exactness for tractability. The open problem is therefore exact substructure counting *at scale*, not a new architecture.

Recent parallel engines directly address exact subgraph isomorphism at scale. HiPerMotif [6] shifts search initialization onto validated edges and injects partial states deeper in the classical VF2 search [7]; built in the open-source Arachne framework [8] on Arkouda [9] and Chapel [10], it enumerates exact pattern instances on property graphs far larger than prior substructure-aware GNN work has used. Combinatorial graphlet counters supply exact statistics only over a fixed catalog of patterns up to five vertices; a general isomorphism engine instead yields the explicit embeddings of *arbitrary* patterns, including the induced long cycles benchmarked here, that lie beyond that catalog. This lets us ask two distinct questions: can exact per-vertex orbit features be *extracted* at scale, and *where* does their exactness actually improve a GNN?

The answers fall on two axes. On the *accuracy* axis, exact orbit features help where structure drives the label; on the *systems* axis, exact extraction scales to a full 128-core node and reaches patterns beyond the five-vertex graphlet wall. We do not claim the two axes meet in a single large-graph accuracy gain: the contribution is that exact structural features are both *useful where structure drives the task* and *feasible at scale*. The size-4 orbit counts are validated against the ORCA graphlet oracle [11] and the long cycles against an independent enumerator.

Contributions. Our HiPerMotif engine [6] contributes one primitive: the parallel enumeration of a pattern's embeddings in a large graph. Everything below is new to this work and is built on top of that primitive, not inside the engine.

- We add an orbit-feature extraction layer on top of HiPerMotif's raw embeddings: it collapses each pattern's embeddings over automorphism-orbit positions and normalizes by $|\mathrm{Aut}(H)|$, the pattern's symmetry-group size, to yield validated per-vertex *orbit* features for expressive GNNs, where an orbit groups symmetry-interchangeable pattern vertices (Sec. IV).
- We make the expressivity payoff concrete with two exactly reproducible cases (Sec. III): the CSL benchmark, where every class shares one 1-WL coloring, and the cospectral Shrikhande/rook pair, which no adjacency- or Laplacian-spectral invariant separates but a single clique count does.
- On the accuracy axis (Sec. VI-A, Sec. VI-B) we show exact orbit features give a capacity-independent benefit on structure-driven tasks, while stating the boundary that on feature-rich node classification the benefit vanishes; with native features removed, the structural signal reappears even on a 169K-vertex graph.
- On the systems axis (Sec. VI-C) we give a strong-scaling study of exact 15-orbit extraction on one 128-core node, with counts bit-identical across thread counts, reaching real-world graphs such as roadNet-CA ($\sim$2M vertices).
- Because the extractor is a general isomorphism engine, not a fixed-catalog counter, it reaches patterns beyond the five-vertex graphlet wall (Sec. VI-D): we extract and strong-scale induced long cycles (C_6–C_8) and traverse a 10^8-edge graph on one node. We chart the resulting coverage boundary against ORCA, ESCAPE, and PGD, and implement an independent cycle oracle to validate correctness beyond ORCA's five-vertex range.

II. Background and Related Work

Message-passing GNNs update each vertex representation from its neighbors', a computation Xu et al. [1] and Morris et al. [2] showed is at most as discriminative as first-order Weisfeiler–Leman (1-WL) color refinement: two graphs with the same 1-WL coloring are indistinguishable to any such network, and even the most expressive variants (e.g. GIN) match but do not exceed this bound. A direct consequence is an inability to count many simple substructures, including triangles, longer cycles, and cliques [3]. Ways to raise the ceiling include higher-order networks that operate on vertex tuples to climb the k-WL hierarchy at rapidly growing cost [2], [12] and subgraph methods that run a GNN over a bag of derived subgraphs [13]–[15]. We follow the lighter route of augmenting each vertex with counts of its participation in a fixed family of small patterns, which provably exceeds 1-WL while leaving the architecture unchanged [4], because it turns the added cost into a one-time, parallelizable feature-extraction step rather than a change to the model.

The bottleneck is then the *counting*. Graph Substructure Networks [4] attach per-vertex orbit counts over a chosen pattern set, and Chen et al. [3] characterize which substructures message passing can and cannot count, but exact counts are obtained almost only on small molecular graphs; to scale, DeSCo [5] and related methods *learn* to predict counts approximately, trading exactness for speed [16]. A distinct line counts graph *homomorphisms* rather than subgraph instances as a WL-comparable invariant [17], [18]; we count induced subgraphs, which carry different information. Exact combinatorial counters compute small-pattern statistics directly: ORCA [11] counts graphlet orbits up to five vertices via a linear system, ESCAPE [19] counts five-vertex subgraphs by reduction to smaller ones, and PGD [20] is a fast parallel counter for three- and four-vertex graphlets. These are *counters* over a fixed catalog: ORCA returns per-vertex orbit counts up to five vertices (exactly the size-four orbit features this paper uses, which we compute with it), while ESCAPE and PGD return graph-level frequencies. What no counter provides is the explicit per-embedding main-graph vertex mapping needed to extend orbit features beyond that catalog (six or more vertices, labeled, or directed). We therefore use ORCA as a correctness oracle and a coverage boundary, not a speed rival: it is exact and, for the patterns it supports, lighter than enumeration, while the patterns it cannot express delimit the regime our approach serves.

Obtaining per-vertex instances of arbitrary patterns is subgraph isomorphism, which is NP-hard in general. Classical solvers such as VF2 [7] use constraint propagation and matching-order heuristics, and the Glasgow subgraph solver [21] couples constraint programming with strong inference on shared-memory parallel hardware; beyond exact solvers, parallel graph-mining systems enumerate pattern instances at scale [22], [23]. Arachne [8] provides property-graph structures on top of Arkouda [9] and the Chapel parallel language [10]; within it, our VF2-PS [24] and HiPerMotif [6] engines perform parallel subgraph matching, the latter shifting search initialization onto validated edges and injecting partial states deeper in the search tree. Unlike the combinatorial counters, HiPerMotif returns the main-graph vertex mapping of every embedding, the per-vertex instance information that lets us collapse embeddings into automorphism-orbit features (Sec. IV) on a single shared-memory node.

III. The Expressivity Gap, Concretely

We make the problem and the payoff exact on two constructs whose classes are indistinguishable to every message-passing GNN; all numbers here are computed directly, independent of any trained network.

The CSL benchmark [25], [26] consists of 4-regular circulant graphs $\mathrm{CSL}(n, s)$ that differ only in a skip length s. For $n = 41$ and the ten standard skips $s \in \{2, 3, 4, 5, 6, 9, 11, 12, 13, 16\}$ the graphs are vertex-transitive, so 1-WL assigns every vertex the same color and all ten classes collapse to a single 1-WL representation; any message-passing GNN is therefore capped at $1/10$ on the ten-way task, which GIN [1], the maximally 1-WL-expressive MPNN, attains empirically. That cycle counts separate CSL is itself known [3], [4], [25]; our focus here is specifically on *coverage*. We measure, for each exact feature family, how many of the ten graphs receive a *distinct* count vector, an information-theoretic

TABLE I

CSL DISTINCT-SIGNATURE COVERAGE: OF THE TEN 1-WL-IDENTICAL CSL CLASSES, HOW MANY RECEIVE A *DISTINCT* EXACT COUNT VECTOR UNDER EACH FEATURE FAMILY (A CLASSIFIER-INDEPENDENT, INTRA-CLASS-CONSISTENT QUANTITY). EVERY SIZE-≤ 5 GRAPHLET SYSTEM SATURATES AT $5/10$; ONLY CYCLES SPANNING MORE THAN FIVE VERTICES REACH $10/10$.

Exact feature family	dim	distinct sigs / 10
degree, wedge, triangle (legacy)	3	2
ORCA size-4 orbits	15	3
ORCA size-≤ 5 orbits (maximum)	73	5
induced cycles C_3–C_8	6	**10**

quantity independent of any classifier or GNN: graphs sharing a vector are indistinguishable to every measurable function of that feature family. Degree, wedge, and triangle counts realize 2 distinct vectors; the full size-4 orbit statistic (ORCA, 15 orbits) realizes 3; the *complete* size-≤ 5 orbit statistic (ORCA at its maximum, 73 orbits) realizes only 5 (Table I). Thus $5/10$ is a hard ceiling for *every* size-≤ 5 graphlet system, ORCA, ESCAPE [19], and PGD [20] alike, not a weakness of a particular model: resolving the remaining cells provably requires a feature that is not a function of ≤ 5-vertex induced subgraph counts, i.e. one that observes ≥ 6 vertices jointly. Exact counts of induced (chordless) cycles up to length 8 are one such completing family: in a chordless k-cycle every *connected* induced subgraph on fewer than k vertices is a path, so the cycle is invisible to any ≤ 5-graphlet system (which counts only connected graphlets) regardless of the counting procedure used. Adding the chordless 6-, 7-, and 8-cycle counts lifts the distinct-vector count from $5/10$ to $10/10$, and a linear classifier on the cycle-length features reaches 100% five-fold cross-validation accuracy (a GNN with the same features likewise reaches 100%); the accuracies merely confirm that the separating information is usable. We report induced counts because that is what an induced subgraph-isomorphism engine returns; we verified numerically on the ten CSL graphs that all-simple cycle counts give the same $10/10$ and 100%. Spectral positional encodings also separate CSL [26], and we do not claim cycle features are the only route; rather, induced subgraph counts are a different invariant (they are not determined by the adjacency spectrum, unlike closed-walk counts), they are exact and localizable to vertices, and, as the next sections show, an engine that extracts them generalizes to arbitrary, labeled, and larger patterns at main-graph sizes spectral shortcuts do not address.

CSL is a controlled probe that isolates the cycle-length signal, not our only evidence that long-range cyclic structure carries task-relevant information. Cycle and ring features are known to improve real benchmarks: subgraph-isomorphism counts of cycles help on molecular property prediction [4], message passing augmented with explicit rings improves molecular regression and classification (ZINC, ogbg-molhiv) [27], and long-range benchmarks where ring and long-distance structure dominate motivate features beyond short-range message passing [28]. Our contribution is orthogonal to these: not that long cycles matter, which is established, but that an exact subgraph-isomorphism engine can extract them at main-graph sizes and pattern sizes prior substructure-aware GNN work has not reached.

A sharper case is the Shrikhande graph and the 4×4 rook's graph, both strongly regular with parameters $(16, 6, 2, 2)$. They are non-isomorphic but share the same 1-WL coloring, and, being strongly regular with identical parameters, they are cospectral, so no invariant derived from the adjacency spectrum, including counts of closed walks, can separate them. Both contain exactly 32 triangles. Yet the number of 4-cliques differs sharply: the rook's graph contains 8 (one per row and column), while the Shrikhande graph contains 0. A single exact substructure count thus distinguishes a pair that defeats 1-WL and every spectral feature. This quantity is directly returned by a subgraph-isomorphism engine.

IV. METHOD: HIPERMOTIF AS AN ORBIT-FEATURE ENGINE

Our pipeline produces, for a main graph G and a library $\mathcal{H}$ of pattern graphs, both graph-level motif counts and per-vertex orbit features, using HiPerMotif for all enumeration. A key step is correct normalization by pattern symmetry.

A. Patterns, automorphisms, and orbits

For each pattern $H \in \mathcal{H}$ we precompute, once with networkx, its automorphism group $\mathrm{Aut}(H)$ and the partition of $V(H)$ into automorphism *orbits*; two pattern vertices are in the same orbit if some automorphism maps one to the other. A subgraph isomorphism search reports every distinct mapping, so each embedded copy of H in G is found $|\mathrm{Aut}(H)|$ times. The true number of copies is therefore the raw count divided by $|\mathrm{Aut}(H)|$. For vertex features, a main-graph vertex's per-orbit tally, summed over all of the orbit's positions, is similarly divided by the full $|\mathrm{Aut}(H)|$ so that each embedded copy contributes once per participating vertex. The full $|\mathrm{Aut}(H)|$, not the stabilizer $|\mathrm{Aut}(H)|/|O_j|$, is the correct divisor: collapsing the tally over the $|O_j|$ positions of an orbit already multiplies each true copy by $|O_j|$, which the full $|\mathrm{Aut}(H)|$ exactly cancels, leaving the number of copies in which the vertex plays an orbit-j role (for K_4: triangle role $18/6{=}3$, 4-clique role $24/24{=}1$). These quantities are small and are computed once per pattern.

B. Extraction with HiPerMotif

We build the main graph and each pattern as Arachne property graphs [29] and run HiPerMotif's edge-centric search. The engine can return either the aggregate number of isomorphisms, which gives graph-level motif counts after dividing by $|\mathrm{Aut}(H)|$, or, for each embedding, the main-graph vertices matched to the n pattern vertices, from which we form the per-vertex orbit features by the tally-and-divide above (Algorithm 1).

We validate the extractor against ORCA, an exact combinatorial graphlet oracle: for the size-4 library, HiPerMotif's per-vertex orbit counts equal ORCA's on every orbit, on complete and cycle graphs, the cospectral Shrikhande and

Algorithm 1 Exact orbit features via HiPerMotif

Require: main graph G, pattern library $\mathcal{H}$
Ensure: vertex-feature matrix $X \in \mathbb{R}^{|V(G)| \times F}$

build Arachne property graph G_p from G
for each pattern $H \in \mathcal{H}$ **do**
  compute $\mathrm{Aut}(H)$ and orbits $O_1, \ldots, O_r$ ▷ once, offline
  build property graph H_p from H
  isos $\leftarrow$ SubgraphIso(G_p, H_p) ▷ per-embedding vertex mappings
  reshape *isos* into $k \times n$ embeddings ▷ k = raw mappings, $n = |V(H)|$; column j = pattern vertex j
  for each orbit O_j with divisor $b_j = |\mathrm{Aut}(H)|$ **do**
    for each pattern position $p \in O_j$ **do** ▷ collapse positions within the orbit
      increment $X[\,\mathit{isos}[\cdot, p],\ (H, j)\,]$
    end for
    $X[:, (H, j)] \leftarrow X[:, (H, j)]/b_j$
  end for
end for
return $\log(1 + X)$ ▷ motif counts are heavy-tailed

rook pair, and the Cora and PROTEINS benchmarks. Matching ORCA's *induced* graphlet counts requires induced search: we enforce that every pattern non-edge maps to a main-graph non-edge, and the equality holds on the raw orbit counts, before the $\log(1+X)$ transform of Algorithm 1. Therefore, the two engines are interchangeable as feature sources. We compute the features for the accuracy experiments with ORCA, the lighter oracle, and reserve HiPerMotif for the at-scale extraction study, where its parallel enumeration of per-vertex instances is the capability ORCA lacks.

C. Use in a GNN

The orbit feature matrix is concatenated with the GNN's input vertex features and the combined matrix is consumed by any standard architecture. Because the features are exact and precomputed, the architecture and message-passing depth are unchanged; the only added cost is the wider input layer and the one-time extraction we study at scale. The features are permutation equivariant by construction, since orbit counts depend only on graph structure.

V. Experimental Methodology

Scaling experiments run on one node of NJIT's Wulver cluster (two AMD EPYC 7753, 128 cores total, 2.45 GHz, 512 GB); accuracy experiments run on a single NVIDIA RTX 4070 Ti SUPER, with orbit features computed by ORCA on CPU. Each number is the mean over 10 seeds (accuracy) or, for scaling, three repeats per point; we treat the spread as a consistency check, not a significance test. We report test accuracy (mean $\pm$ std) and, for extraction, total wall-clock time with strong-scaling speedup and efficiency, obtained by varying the core count from 1 to 128 (limiting worker threads and relaunching the engine per point).

The feature library is the 15 automorphism orbits of the nine connected graphlets on two to four vertices (edge, 3-path, triangle, 4-path, claw, paw, 4-cycle, diamond, 4-clique) with $|\mathrm{Aut}(H)|$ normalization (Sec. IV), computed with ORCA, which HiPerMotif reproduces byte-for-byte. Accuracy uses five feature sets that share identical hyperparameters so the feature set is the only variable: a base GCN (the 1-WL baseline), the 15-orbit library, a dimension-matched random control (*Orbit-Rand*, isolating structure from added capacity), a degree-only control (orbit 0 alone), and, on the small graphs, the legacy degree/wedge/triangle set. The accuracy datasets are the Planetoid citation graphs (Cora, Citeseer, Pubmed; node classification) [30], the TU molecular datasets (PROTEINS, NCI1, ENZYMES; graph classification) [31], and the large graph ogbn-arxiv (169,343 vertices, 1,157,799 undirected edges, 40 classes, 128-dim features) [32]; the CSL and Shrikhande/rook constructs of Sec. III are theory-defined, not real-world evidence. The scaling study uses random d-regular graphs of $2{\times}10^4$ to 10^6 vertices (uniform workload), a Barabási–Albert graph (load-imbalance case), and the roadNet-CA road network [33] ($1.97{\times}10^6$ vertices, $2.77{\times}10^6$ edges).

VI. Results

A. Local accuracy: exact features vs. a capacity-matched control

We first establish that exact orbit features carry a real, capacity-independent benefit where structure drives the label. Table II reports test accuracy for conditions sharing identical hyperparameters and differing only in the feature set. Exact features beat the base GCN on four of six datasets ($A > 0$), clearly outside seed noise on two (ENZYMES $+0.070$, NCI1 $+0.067$) and nominally on Cora ($+0.009$) and PROTEINS ($+0.015$), concentrated on graph classification. A degree-only control (orbit 0 alone) shows this gain is higher-order structure, not merely degree: Orbit-15 beats degree-only on all three graph tasks (ENZYMES $+0.067$, NCI1 $+0.024$, PROTEINS $+0.015$), and on ENZYMES degree alone barely moves the base GCN (0.297 vs. 0.294) while the full orbits reach 0.364. The capacity-matched random control corroborates that the gain is structural rather than extra dimensionality (B up to $+0.173$); since that control is itself somewhat destructive, we read A as the primary effect. On the node-classification tasks degree instead carries what little signal exists, and the richer library does not beat the simpler legacy set on accuracy alone on these small graphs, where higher-order orbits are sparse and become abundant only in the large-graph regime HiPerMotif targets. There are two important negatives: on Pubmed exact features are net-harmful versus the base GCN ($A = -0.042$, still above random $B = +0.087$), and on Citeseer they confer no benefit on any axis.

B. Accuracy on a large graph: a boundary and an isolation diagnostic

We test whether exact orbit features help on a graph two orders of magnitude larger than the standard benchmarks. On

TABLE II

EXACT GRAPHLET-ORBIT FEATURES (ORBIT-15), CONCATENATED TO EACH DATASET'S STANDARD INPUT FEATURES, VS. THE BASE GCN ON THOSE FEATURES ALONE (GCN), THE LEGACY 3-FEATURE SET (LEGACY-3), A CAPACITY-MATCHED RANDOM CONTROL (ORBIT-RAND), AND A DEGREE-ONLY CONTROL (DEG-ONLY, ORBIT 0 ALONE, ISOLATING HIGHER-ORDER STRUCTURE FROM DEGREE). ALL CONDITIONS SHARE ONE BACKBONE, DIFFERING ONLY IN THE CONCATENATED STRUCTURAL BLOCK. TEST ACCURACY, MEAN $\pm$ STD OVER 10 SEEDS. A: DOES ADDING EXACT STRUCTURE BEAT THE BASE GCN? B: IS THE GAIN STRUCTURAL, NOT CAPACITY? GAPS ARE CONSISTENCY INDICATORS (FROM UNROUNDED MEANS), NOT SIGNIFICANCE TESTS.

Dataset	Task	GCN	Legacy-3	Orbit-15	Orbit-Rand	Deg-only	A=O$-$GCN	B=O$-$Rand
Cora	node	0.782 ± 0.004	0.781 ± 0.010	$\mathbf{0.791 \pm 0.010}$	0.763 ± 0.009	0.783 ± 0.010	$+0.009$	$+0.028$
Citeseer	node	0.669 ± 0.008	0.668 ± 0.009	0.651 ± 0.015	0.653 ± 0.011	0.674 ± 0.009	-0.018	-0.002
Pubmed	node	0.754 ± 0.007	0.763 ± 0.008	0.712 ± 0.011	0.625 ± 0.023	0.762 ± 0.008	-0.042	$+0.087$
PROTEINS	graph	0.705 ± 0.017	0.716 ± 0.022	$\mathbf{0.720 \pm 0.017}$	0.636 ± 0.015	0.705 ± 0.028	$+0.015$	$+0.084$
NCI1	graph	0.691 ± 0.012	0.729 ± 0.021	$\mathbf{0.758 \pm 0.013}$	0.614 ± 0.014	0.734 ± 0.014	$+0.067$	$+0.144$
ENZYMES	graph	0.294 ± 0.034	0.324 ± 0.038	$\mathbf{0.364 \pm 0.027}$	0.191 ± 0.031	0.297 ± 0.040	$+0.070$	$+0.173$

TABLE III

ACCURACY ON THE LARGE GRAPH OGBN-ARXIV (169K VERTICES); TEST ACCURACY MEAN $\pm$ STD OVER 10 SEEDS. *TOP:* NATIVE FEATURES (REAL-WORLD SETTING). *BOTTOM:* NATIVE FEATURES REPLACED BY A CONSTANT, SO STRUCTURE IS THE ONLY SIGNAL (AN ISOLATION DIAGNOSTIC, NOT REAL-WORLD ACCURACY).

Condition	Test accuracy
Native features (real-world setting)	
GCN (native only)	0.694 ± 0.002
Orbit-15 (exact)	0.696 ± 0.002
Orbit-Rand (15)	0.693 ± 0.002
Features ablated to a constant (isolation diagnostic)	
GCN (structure only)	0.210 ± 0.009
Orbit-Rand (15)	0.220 ± 0.007
Deg-only (orbit 0)	0.262 ± 0.001
Orbit-15 (exact)	$\mathbf{0.410 \pm 0.007}$

ogbn-arxiv in its native setting (Table III, top), the exact features, the capacity-matched random control, and the base GCN all lie within 0.003 of one another (orbit$-$random $= +0.003$): the 128-dimensional native features saturate a two-layer GCN, and 15 additional dimensions, whether structural or random, do not meaningfully move it. We report this as a clear boundary on where local structure helps: not on feature-rich transductive node classification.

To separate "structure does not help here" from "structure is masked here," we re-run with the native features replaced by a constant, so that graph structure is the only signal (the same isolation used for CSL in Sec. III). Now the exact orbit features nearly double accuracy over both the structure-only GCN and the capacity-matched random control (0.410 vs. 0.220, far outside the seed-to-seed spread; Table III, bottom). Exact local structure therefore carries large, capacity-independent vertex-label signal even at this scale. A degree-only control isolates how much of this is degree: orbit 0 alone reaches 0.262, well above the random control but far below the full orbits (0.410), so the signal is dominated by higher-order structure, not degree, and the random control rules out added capacity as the cause.

C. At-scale extraction: strong scaling

We next measure how the one-time extraction cost parallelizes on a single shared-memory node (128 cores). To isolate parallel scaling from work imbalance, we use random d-regular graphs of increasing size (d=6), which spread the size-4 subgraph workload uniformly across vertices; Table IV and Fig. 1 report the total time to extract the full 15-orbit library. Each graph's own speedup curve improves monotonically in core count out to 128, with two regimes: below 8 cores the apparent efficiency exceeds 100% (the $1 \rightarrow 8$-core speedup is $\approx 9\times$ on all three graphs), and above 8 cores efficiency falls into the 35–55% range for the comparatively cheap size-4 search. The above-100% efficiency below 8 cores is a baseline artifact, not genuine algorithmic scaling: the single-core run is memory-bound, so measuring speedup against it overstates parallel gains at low core counts. We therefore treat the 35–55% efficiency beyond 8 cores, where the machine behaves normally, as the representative measure of parallel scaling for this workload. The whole-range ($1 \rightarrow 128$) speedup therefore peaks at the mid-size graph ($80\times$ on reg-200k) and is smaller at both ends: $68\times$ on reg-20k and $64\times$ on the largest graph (reg-1M, 10^6 vertices, $3{\times}10^6$ edges: $2629\,\mathrm{s}$ to $41.2\,\mathrm{s}$). Every point is the mean of three runs. At 128 cores, where size-4 totals are only seconds, run-to-run variation is larger than for the cycle workload (reg-1M spanned 37.8–$43.7\,\mathrm{s}$ over three runs, mean $41.2\,\mathrm{s}$, the value tabulated), so the size-4 whole-range speedups should be read as approximate. Each scaling table reports speedup from the smallest core count at which the workload completes on every graph: size-4 extraction is feasible at one core, whereas single-core C_6–C_8 extraction is not on the largest graph (Sec. VI-D), so the cycle table uses a two-core baseline. The per-vertex orbit counts are bit-identical across all thread counts from 1 to 128: parallelism changes only the assembly order of embeddings, not the exact counts, which we verify against the ORCA oracle. As a real-world reference point, the full 15-orbit library on the roadNet-CA road network ($\sim$2M vertices, 2.77M edges) extracts in $12.3\,\mathrm{s}$ at 128 cores.

On scale-free graphs the extraction is instead governed by *work* imbalance rather than the near-uniform per-task cost of the regular graphs: a small number of high-degree hubs generate the overwhelming majority of open-pattern (claw/path) embeddings, so a Barabási–Albert graph (10^4 vertices, $\sim 5{\times}10^4$ edges, the scale of reg-20k) scales only $5\times$ at 128 cores. The dominant cost tracks the size-4 subgraph count, not $|E|$.

TABLE IV

STRONG SCALING OF EXACT 15-ORBIT EXTRACTION ON ONE 128-CORE NODE, ON RANDOM 6-REGULAR GRAPHS (UNIFORM WORKLOAD); ROADNET-CA IS A REAL-WORLD REFERENCE POINT, MEASURED AT 128 CORES ONLY. TIMES ARE THE TOTAL OVER ALL 15 ORBITS, MEAN OVER RUNS; ORBIT COUNTS ARE BIT-IDENTICAL ACROSS ALL THREAD COUNTS.

Graph	Vertices	1 core (s)	128 cores (s)	Speedup
reg-20k	2×10^4	50.7	0.74	68×
reg-200k	2×10^5	529.2	6.58	80×
reg-1M	1×10^6	2628.6	41.2	64×
roadNet-CA	1.97×10^6	–	12.33	–

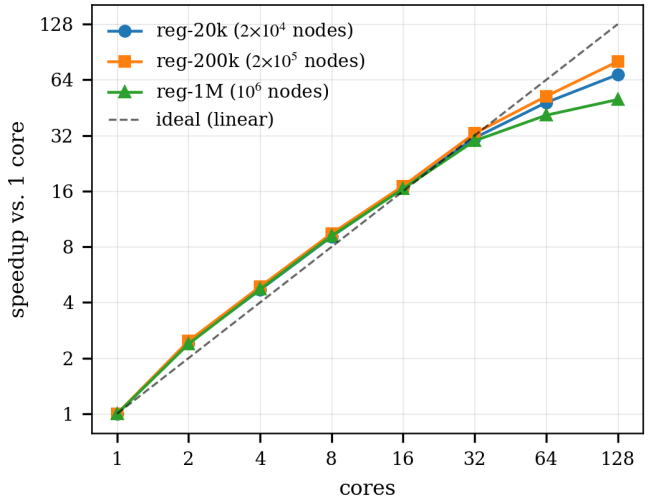

Fig. 1. Strong scaling of HiPerMotif 15-orbit extraction on random 6-regular graphs of 2×10^4 to 10^6 vertices, one 128-core node: speedup versus cores against the ideal-linear reference. All graphs scale monotonically to 128 cores; see text for the whole-range speedups.

D. Beyond size-5 graphlets: long cycles at scale

The features above are size-4 orbits, which ORCA also produces. The capability that distinguishes a general subgraph-isomorphism engine is exact extraction of patterns *beyond* the size-$\leq$ 5 graphlet wall, the regime that separates CSL (Sec. III). We extract induced cycles of length 6, 7, and 8, patterns no fixed size-$\leq$ 5 graphlet counter can express, and validate the counts against an independent enumerator (`cycle_oracle.py`, which counts chordless induced cycles directly with NetworkX's `simple_cycles` and compares them to the engine's $n_{\mathrm{emb}}/2k$) on graphs small enough to enumerate exhaustively, reg(n=200) and the ten CSL graphs at n=41; ORCA cannot serve as the oracle here because its catalog ends at five vertices, and NetworkX does not scale to the largest graphs, where correctness instead relies on the exact induced search and the thread-invariance of the counts. As in Sec. VI-C, the same engine binary runs at every core count, so the reported speedups are intrinsic to one code path, not a comparison against a separate serial implementation.

Cycle extraction parallelizes well because, unlike open patterns (claws and paths), cycle embeddings through a vertex do not grow combinatorially with its degree, so work does not concentrate on hubs. Figure 2 and Table V show strong scaling of the combined C_6–C_8 extraction on random 6-regular graphs. Single-core C_6–C_8 extraction on the 10^6-vertex graph is itself infeasible (C_7 alone exceeds five hours and C_8 exceeds a day), and the measured single-core points on the smaller graphs show above-100% efficiency at the $1 \rightarrow 2$ step (the same memory-bound one-core-baseline effect); from two cores onward scaling is near-linear, so we baseline at two cores, the smallest core count feasible on every graph, and report parallel efficiency over the $2 \rightarrow 128$ range. reg-200k improves from 11360 s at 2 cores to 227 s at 128 cores (50× over 64× the cores, 78% efficiency), reg-20k scales identically (51×, 80%), and reg-1M, the largest graph, scales best (56880 s to 1099 s, 52×, 81% efficiency), confirming that cycle extraction stays compute-bound and parallelizes cleanly as the graph grows. All three curves track the ideal-linear reference and remain monotonic to 128 cores, demonstrating the benefit of parallelism. Across three runs at 128 cores the cycle totals are stable (within 5%: reg-20k 22.5–23.1 s, reg-200k 222.0–232.5 s). At fixed core count the time scales linearly in the number of edges (a 10× larger graph costs 10.0× at 128 cores); the induced short-cycle *count* in a random regular graph is asymptotically independent of n, so the regular graphs isolate parallel and problem-size scaling, while enumeration growth appears on the real-world graph. On the roadNet-CA road network (1.97M vertices, 2.77M edges) the engine enumerates 5.1M induced C_6–C_8 embeddings (raw isomorphism mappings; each chordless k-cycle is reported $2k$ times) in 45.5 s at 128 cores, a throughput of ∼112k mappings per second.

TABLE V

STRONG SCALING OF EXACT INDUCED C_6–C_8 EXTRACTION ON RANDOM 6-REGULAR GRAPHS, ONE 128-CORE NODE; ROADNET-CA IS A REAL-WORLD REFERENCE POINT, MEASURED AT 128 CORES ONLY. TIMES ARE THE C_6–C_8 TOTAL, MEAN OVER RUNS; COUNTS ARE BIT-IDENTICAL ACROSS THREAD COUNTS. SPEEDUP IS RELATIVE TO 2 CORES (SINGLE-CORE DOES NOT COMPLETE ON THE LARGEST GRAPH).

Graph	Vertices	2 cores (s)	128 cores (s)	Speedup
reg-20k	2×10^4	1158.0	22.7	51×
reg-200k	2×10^5	11360.1	227.4	50×
reg-1M	1×10^6	56879.9	1098.6	52×
roadNet-CA	1.97×10^6	–	45.5	–

To probe the graph size the engine reaches on one node, we extract triangle, C_4, and C_6 (C_6 as the representative beyond-ORCA pattern; C_7 and C_8 are omitted at this size for cost) on a random 6-regular graph of 3.3×10^7 vertices and 10^8 edges. The full traversal completes in 18.8 min at 128 cores on a single 512 GB node without exhausting its memory; the cost is dominated by search over the graph (the random-regular structure yields few short cycles), so this is a capacity result, the engine traverses a 10^8-edge graph within one node, rather than an enumeration-throughput result.

The relationship to combinatorial counters is one of cover-

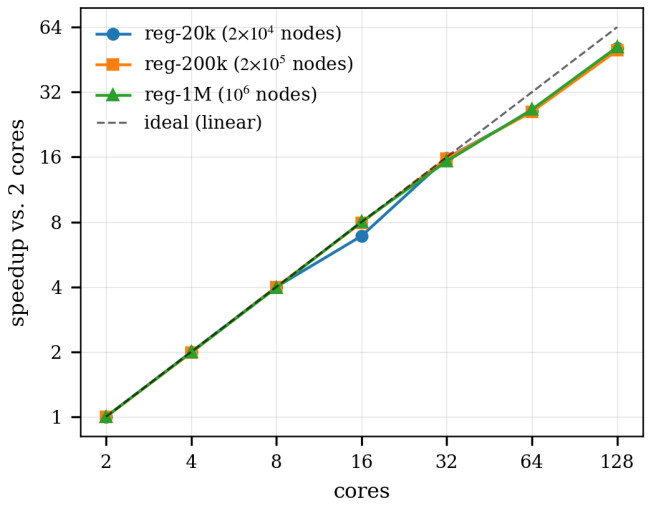

Fig. 2. Strong scaling of exact induced C_6–C_8 extraction on random 6-regular graphs (2×10^4, 2×10^5, 10^6 vertices), one 128-core node: speedup vs. cores relative to 2 cores, against the ideal-linear reference. Same runs as Table V.

TABLE VI
COVERAGE BOUNDARY (SEE TEXT). ORCA IS THE FASTER, LIGHTER ORACLE FOR THE ≤ 5-VERTEX GRAPHLETS BOTH CLASSES SUPPORT (1.96 S, 0.81 GB ON REG-200K, SINGLE CORE); NO COUNTER PRODUCES INDUCED CYCLES OF LENGTH ≥ 6, WHICH HIPERMOTIF EXTRACTS ON THE SAME GRAPH IN 227 S AT 128 CORES. ESCAPE AND PGD RETURN GRAPH-LEVEL *FREQUENCIES*, NOT THE *PER-VERTEX* DECOMPOSITION THIS PAPER USES. ✓ = SUPPORTED, × = NOT SUPPORTED; † MARKS CAPABILITIES THE ENGINE SUPPORTS BUT THAT WE DO NOT BENCHMARK IN THIS PAPER (SIZE-4 ORBITS AND C_6–C_8, WITHOUT A DAGGER, ARE THE ONES DEMONSTRATED HERE).

Pattern class	ORCA	ESCAPE	PGD	HiPerMotif
size-4 graphlet counts	✓	✓	✓	✓
size-5 graphlet counts	✓	✓	×	✓†
per-vertex orbit features	✓	×	×	✓
induced C_6–C_8	×	×	×	✓
labeled / directed	×	×	×	✓†

age, not speed (Table VI): for the ≤ 5-vertex graphlets both classes support, ORCA is the faster, lighter oracle and we use it for correctness throughout; for induced cycles of length ≥ 6, ORCA, ESCAPE, and PGD produce no such output in their standard public implementations, so HiPerMotif's extraction time is the price of a capability the counters lack, not a slowdown relative to them.

This is exactly the capability the CSL benchmark demands (Sec. III): separating its ten 1-WL-identical classes requires a feature spanning ≥ 6 vertices (Table I), which the size-≤ 5 graphlets of the accuracy table (Table II) cannot supply but induced cycles up to length 8 do, so that separation is a beyond-graphlet-wall result, not a property of the size-4 library.

E. Head-to-head against a general subgraph-isomorphism solver

The scaling results above are intrinsic: the same engine binary at every core count. To test whether a mature general-purpose solver would serve equally well, we compare against the Glasgow Subgraph Solver [21], a leading publicly available constraint-programming engine for subgraph isomorphism, on the same main graphs and patterns (induced search, solution counting, up to 128 cores). Glasgow is the right comparator because it is in HiPerMotif's category, a general solver that enumerates induced embeddings of an arbitrary pattern, rather than a fixed-graphlet counter such as ORCA, ESCAPE, or PGD. To ensure a fair comparison, we confirmed on reg-20k that the two engines return identical raw induced-embedding counts for every pattern (each chordless k-cycle reported $2k$ times; $C_6/C_7/C_8$: 15,900/77,728/386,000, and all size-4 graphlets), so wall-clock time measures the same work; this also re-confirms the long-cycle counts against an independent engine.

TABLE VII
HEAD-TO-HEAD AGAINST THE GLASGOW SUBGRAPH SOLVER [21]: INDUCED C_6–C_8 TOTAL, 128 CORES, MEAN OF THREE RUNS PER ENGINE. OOM = OUT-OF-MEMORY KILL (EXIT 137) ON THE 512 GB NODE. HIPERMOTIF TIMES MATCH TABLE V.

Graph	Vertices	Glasgow (s)	HiPerMotif (s)
reg-20k	2×10^4	21.5	22.7
reg-200k	2×10^5	3825.4	227.4
reg-1M	1×10^6	OOM	1098.6
roadNet-CA	1.97×10^6	OOM	45.5

Glasgow is genuinely competitive at small scale. On reg-20k its constraint propagation extracts the C_6–C_8 total in 21.5 s at 128 cores, marginally ahead of HiPerMotif's 22.7 s, and it wins outright on the hardest single cycle (C_8: 9.0 s vs. 18.6 s). The engines diverge at larger scales for two reasons: parallel scaling and memory consumption. First, parallelism: from one to 128 cores on reg-20k Glasgow's times barely improve, and several *degrade* (e.g. the path P_4 rises from 17.3 to 26.2 s, reflecting limited parallel scaling), whereas HiPerMotif's size-4 total falls from 50.7 to 0.74 s over the same range. By reg-200k the gap is decisive: at 128 cores HiPerMotif is faster on every pattern, from 13× on C_8 to over 3,600× on the dense size-4 cliques; its size-4 graphlet total of ≈6.6 s beats Glasgow's ≈2,811 s (≈425×) and its C_6–C_8 total of 227 s beats Glasgow's 3,825 s by 17× (Table VII). Second, memory consumption becomes the limiting factor: on every graph at or above 10^6 vertices Glasgow exhausts the 512 GB node and is terminated by the kernel out-of-memory killer (exit 137) before producing any result, consistent with growth in its solver state, while HiPerMotif extracts C_6–C_8 on reg-1M in 1,099 s and on roadNet-CA in 45.5 s, and traverses the 10^8-edge graph of Sec. VI-D on the same node. The contribution is not that HiPerMotif is uniformly faster, it is not at one core on small graphs, but that exact extraction at this scale is feasible with a parallel engine and infeasible for a leading general solver, which never reaches the regime at all.

VII. Conclusion

Standard message-passing GNNs are blind to substructure by construction, and the established fix, exact substructure features, has been limited by the cost of exact extraction at scale. We showed that a parallel exact subgraph isomorphism engine, HiPerMotif in Arachne, serves as an orbit-feature extractor with correct automorphism normalization, and we made the expressivity payoff concrete on constructs that defeat 1-WL and the adjacency/Laplacian spectrum. Our findings separate cleanly into two axes that do not meet. On accuracy, exact orbit features give a capacity-independent benefit on structure-driven graph classification (clearest on NCI1 and ENZYMES) but not on feature-rich node classification, where strong native features saturate the model. On systems, the extraction is feasible at a scale and a pattern coverage prior substructure-aware GNN work has not reached: it strong-scales to 128 cores with bit-identical counts, traverses a 10^8-edge graph on one node, and extracts induced long cycles beyond the five-vertex graphlet wall that combinatorial counters cannot express. The exactness is validated throughout, against ORCA for the size-4 orbits and an independent enumerator for the long cycles. Future work includes comparing exact and learned-approximate feature quality on the same graphs, mapping the per-motif compute-versus-accuracy frontier, and extending the extraction across multiple nodes for graphs beyond a single shared-memory machine. HiPerMotif and the Arachne framework (Arkouda-njit) are open source and available at https://github.com/Bears-R-Us/arkouda-njit.

Acknowledgment

This research was funded in part by NSF grant numbers CCF-2109988, OAC-2402560, and CCF-2453324.

References

[1] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[2] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks," in *Proceedings of the AAAI conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 4602–4609.

[3] Z. Chen, L. Chen, S. Villar, and J. Bruna, "Can graph neural networks count substructures?" *Advances in Neural Information Processing Systems*, vol. 33, pp. 10 383–10 395, 2020.

[4] G. Bouritsas, F. Frasca, S. Zafeiriou, and M. M. Bronstein, "Improving graph neural network expressivity via subgraph isomorphism counting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 657–668, 2022.

[5] T. Fu, C. Wei, Y. Wang, and R. Ying, "Desco: Towards generalizable and scalable deep subgraph counting," in *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, 2024, pp. 218–227.

[6] M. Dindoost, O. Alvarado Rodriguez, B. Bryg, I. Koutis, and D. A. Bader, "HiPerMotif: Novel parallel subgraph isomorphism in large-scale property graphs," in *2025 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2025, pp. 1–7.

[7] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub) graph isomorphism algorithm for matching large graphs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 10, pp. 1367–1372, 2004.

[8] O. Alvarado Rodriguez, Z. Du, J. Patchett, F. Li, and D. A. Bader, "Arachne: An Arkouda package for large-scale graph analytics," in *2022 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2022, pp. 1–7.

[9] M. Merrill, W. Reus, and T. Neumann, "Arkouda: interactive data exploration backed by chapel," in *Proceedings of the ACM SIGPLAN 6th on Chapel Implementers and Users Workshop*, 2019, pp. 28–28.

[10] B. L. Chamberlain, D. Callahan, and H. P. Zima, "Parallel programmability and the chapel language," *The International Journal of High Performance Computing Applications*, vol. 21, no. 3, pp. 291–312, 2007.

[11] T. Hočevar and J. Demšar, "A combinatorial approach to graphlet counting," *Bioinformatics*, vol. 30, no. 4, pp. 559–565, 2014.

[12] H. Maron, H. Ben-Hamu, H. Serviansky, and Y. Lipman, "Provably powerful graph networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[13] B. Bevilacqua, F. Frasca, D. Lim, B. Srinivasan, C. Cai, G. Balamurugan, M. M. Bronstein, and H. Maron, "Equivariant subgraph aggregation networks," *arXiv preprint arXiv:2110.02910*, 2021.

[14] F. Frasca, B. Bevilacqua, M. Bronstein, and H. Maron, "Understanding and extending subgraph GNNs by rethinking their symmetries," *Advances in Neural Information Processing Systems*, vol. 35, pp. 31 376–31 390, 2022.

[15] L. Zhao, W. Jin, L. Akoglu, and N. Shah, "From stars to subgraphs: Uplifting any GNN with local structure awareness," *arXiv preprint arXiv:2110.03753*, 2021.

[16] C. I. Kanatsoulis and A. Ribeiro, "Counting graph substructures with graph neural networks," in *International Conference on Learning Representations*, 2024.

[17] H. Dell, M. Grohe, and G. Rattan, "Lovász meets Weisfeiler and Leman," *arXiv preprint arXiv:1802.08876*, 2018.

[18] H. Nguyen and T. Maehara, "Graph homomorphism convolution," in *International Conference on Machine Learning*. PMLR, 2020, pp. 7306–7316.

[19] A. Pinar, C. Seshadhri, and V. Vishal, "ESCAPE: Efficiently counting all 5-vertex subgraphs," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 1431–1440.

[20] N. K. Ahmed, J. Neville, R. A. Rossi, and N. Duffield, "Efficient graphlet counting for large networks," in *2015 IEEE international conference on data mining*. IEEE, 2015, pp. 1–10.

[21] C. McCreesh, P. Prosser, and J. Trimble, "The Glasgow subgraph solver: using constraint programming to tackle hard subgraph isomorphism problem variants," in *International Conference on Graph Transformation*. Springer, 2020, pp. 316–324.

[22] C. H. Teixeira, A. J. Fonseca, M. Serafini, G. Siganos, M. J. Zaki, and A. Aboulnaga, "Arabesque: A system for distributed graph mining," in *Proceedings of the 25th Symposium on Operating Systems Principles*, 2015, pp. 425–440.

[23] K. Jamshidi, R. Mahadasa, and K. Vora, "Peregrine: A pattern-aware graph mining system," in *Proceedings of the Fifteenth European Conference on Computer Systems*, 2020, pp. 1–16.

[24] M. Dindoost, O. Alvarado Rodriguez, S. Bagchi, P. Pauliuchenka, Z. Du, and D. A. Bader, "VF2-PS: Parallel and scalable subgraph monomorphism in arachne," in *2024 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2024, pp. 1–9.

[25] R. Murphy, B. Srinivasan, V. Rao, and B. Ribeiro, "Relational pooling for graph representations," in *International Conference on Machine Learning*. PMLR, 2019, pp. 4663–4673.

[26] V. P. Dwivedi, C. K. Joshi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson, "Benchmarking graph neural networks," *Journal of Machine Learning Research*, vol. 24, no. 43, pp. 1–48, 2023.

[27] C. Bodnar, F. Frasca, N. Otter, Y. Wang, P. Lio, G. F. Montufar, and M. Bronstein, "Weisfeiler and lehman go cellular: Cw networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 2625–2640, 2021.

[28] V. P. Dwivedi, L. Rampášek, M. Galkin, A. Parviz, G. Wolf, A. T. Luu, and D. Beaini, "Long range graph benchmark," *Advances in Neural Information Processing Systems*, vol. 35, pp. 22 326–22 340, 2022.

[29] O. Alvarado Rodriguez, F. V. Buschmann, Z. Du, and D. A. Bader, "Property graphs in arachne," in *2023 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2023, pp. 1–7.

[30] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *International conference on machine learning*. PMLR, 2016, pp. 40–48.

[31] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, "TUDataset: A collection of benchmark datasets for learning with graphs," *arXiv preprint arXiv:2007.08663*, 2020.

[32] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on

graphs," *Advances in Neural Information Processing Systems*, vol. 33, pp. 22 118–22 133, 2020.

[33] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters," *Internet Mathematics*, vol. 6, no. 1, pp. 29–123, 2009.