

HiPerMotif: Novel Parallel Subgraph Isomorphism in Large-Scale Property Graphs

Mohammad Dindoost, Oliver Alvarado Rodriguez, Bartosz Bryg, Ioannis Koutis, and David A. Bader

Department of Computer Science

New Jersey Institute of Technology

Newark, NJ, USA

{md724, oaa9, bb474, ikoutis, bader}@njit.edu

Abstract—Subgraph isomorphism algorithms face significant scalability bottlenecks on large-scale property graphs due to inefficient vertex-by-vertex search that requires extensive exploration of early search tree levels where pruning is minimal. We present HiPerMotif, a hybrid parallel algorithm that overcomes these limitations through edge-centric initialization. HiPerMotif first reorders pattern graphs to prioritize high-connectivity vertices, then systematically identifies and validates all possible first-edge mappings before injecting these pre-validated partial states directly at search depth 2. This approach eliminates costly early exploration while enabling natural parallelization over independent edge candidates. Comprehensive evaluation against state-of-the-art baselines (VF2-PS, VF3P, Glasgow) demonstrates up to $66\times$ speedup on real-world networks and successful processing of massive datasets like the 150M-edge H01 human connectome that cause existing methods to fail due to memory constraints. Implemented in the open-source Arkouda/Arachne framework, HiPerMotif enables previously intractable large-scale network analysis in computational neuroscience and related domains.

I. INTRODUCTION

Subgraph isomorphism, critical for pattern detection in large-scale graphs, underpins discoveries in neuroscience [1], [2], systems biology [3], [4], social networks [5], [6], and cybersecurity [7], [8]. The fundamental challenge involves determining whether a small pattern graph exists within a much larger target network, a deceptively simple question that becomes computationally intractable as network sizes grow. Recurring substructures, or motifs, reveal network organization and function through tasks such as motif detection and network similarity analysis [9], [10].

Consider computational neuroscience applications in which researchers seek to understand brain connectivity by identifying synaptic motifs, for example, patterns of 3-10 neurons, within connectomes containing hundreds of millions of synaptic connections. Traditional algorithms must exhaustively explore billions of potential vertex mappings, often requiring days of computation or failing entirely due to memory constraints. Similar computational bottlenecks emerge across domains: fraud detection in financial networks, community analysis in social graphs, and pathway discovery in protein interaction networks. Analyzing massive graphs with millions of edges demands scalable tools for property graphs [11], [12] that support multiple vertex and edge attributes, significantly increasing computational complexity beyond simple structural matching.

Traditional subgraph matching algorithms follow a vertex-centric approach, starting from empty mappings and incrementally building solutions vertex-by-vertex through recursive backtracking. This strategy requires extensive exploration of early search tree levels, where pruning opportunities are limited and parallel efficiency is constrained due to insufficient independent work. The fundamental limitation lies in the initialization phase: algorithms spend considerable computational effort exploring unpromising partial mappings in the early search tree, where structural constraints provide minimal pruning power.

Existing algorithms struggle with five fundamental challenges: (1) inefficient candidate generation producing large search spaces; (2) rigid vertex-ordering heuristics that fail to adapt to graph-specific patterns; (3) high memory overhead tracking numerous partial states; (4) limited parallelization opportunities in early tree-search stages; and (5) unnecessary exhaustive enumeration when applications require only pattern existence or partial results. These limitations become critical for connectome motif detection, where graphs exceed 100 million edges and traditional methods fail completely.

In this paper, we propose HiPerMotif, a hybrid algorithm that fundamentally changes search initialization while preserving established tree-search benefits. After structurally reordering the pattern graph to prioritize high-degree vertices, HiPerMotif systematically identifies all possible first-edge mappings, validates candidates using efficient validators, and then injects validated partial mappings at depth 2 into traditional vertex-by-vertex search. This approach eliminates costly early search tree exploration while enabling natural parallelization over edge candidates, as each edge mapping can be validated independently before proceeding with traditional tree search.

Implemented in the open-source Arkouda/Arachne framework, HiPerMotif achieves up to $66\times$ speedup over state-of-the-art baselines and successfully processes massive datasets like the H01 connectome [13] that cause existing methods to fail due to memory and computational limitations.

The major contributions of this paper are as follows.

- 1) Hybrid edge-centric initialization with state injection at depth 2, eliminating costly early search exploration
- 2) Structural reordering strategy achieving up to $5\times$

- speedup independently
- 3) Efficient validators for rapid pruning in attribute-rich graphs
- 4) Parallel framework enabling analysis of previously intractable datasets

Complete technical details available at [14]. Section III presents the algorithm, Section IV evaluates performance, and Section V concludes with future directions.

II. BACKGROUND AND PRELIMINARIES

A. Subgraph Isomorphism Problem

Given graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with vertex labels $\alpha_i : V_i \rightarrow L_V$ and edge labels $\beta_i : E_i \rightarrow L_E$, subgraph isomorphism seeks an injective function $f : V_1 \rightarrow V_2$ such that:

$$\forall u, v \in V_1 : (u, v) \in E_1 \Leftrightarrow (f(u), f(v)) \in E_2, \quad (1)$$

$$\forall v \in V_1 : \alpha_1(v) = \alpha_2(f(v)), \quad (2)$$

$$\forall (u, v) \in E_1 : \beta_1(u, v) = \beta_2(f(u), f(v)). \quad (3)$$

Both isomorphism and monomorphism (relaxed edge constraint) are NP-complete with worst-case complexity $O(|V_2|^{|V_1|})$ [15]. Property graphs [11] with vertex and edge attributes significantly increase computational demands.

B. Existing Approaches

Subgraph isomorphism algorithms can be broadly categorized into tree-search methods that systematically explore partial mappings, index-based approaches that preprocess graphs for filtering, and specialized parallel implementations.

Tree-search algorithms dominate exact subgraph matching, using recursive backtracking to explore partial vertex mappings [16]. VF2 introduced frontier sets for pruning infeasible branches [17], while VF3 improved data structures for dense graphs [18]. The Glasgow Subgraph Solver employs constraint programming and conflict-directed backjumping [19], and LAD uses arc consistency preprocessing [20]. Recent methods like DP-iso [21] and VEQ [22] optimize backtracking with failing sets and dynamic equivalence classes, respectively.

The ordering of variables significantly affects the efficiency of tree-search [23]. Static strategies, such as the high-degree vertex selection of RI [24], precompute sequences, while dynamic methods, such as DP-iso, adapt during search [21]. Techniques such as CFL’s core-forest-leaf decomposition improve pruning by delaying leaf matching [25].

Parallel implementations address scalability on multi-core architectures. VF3P uses state cloning for near-linear scaling up to 16 cores [26], while SLF’s “Ask for Sharing” and “Low-depth Priority Sharing” reduce synchronization overhead [27]. Glasgow integrates parallel backjumping with low-overhead synchronization [19]. Dense graphs benefit more from parallelization due to larger search trees, but synchronization limits scalability beyond moderate core counts [26].

Index-based methods (e.g., FG-index [28]) preprocess graphs for rapid filtering but incur high memory overhead,

limiting scalability for large, attribute-rich graphs [29]. GPU-accelerated approaches such as GSI [30] leverage parallelism but struggle with irregular memory access [31].

Despite these advances, existing approaches share a fundamental limitation: they follow vertex-centric initialization that requires extensive exploration of early search tree levels where pruning opportunities are minimal, particularly problematic for large-scale networks.

C. VF2 Algorithm and VF2-PS Baseline

The VF2 algorithm represents one of the most widely adopted approaches for subgraph isomorphism due to its practical efficiency across diverse graph types. VF2 employs a sophisticated state-space search where each state represents a partial mapping between the query and the target graphs. The algorithm maintains frontier sets $(T_{in}^1, T_{out}^1, T_{in}^2, T_{out}^2)$ that track the boundary between mapped and unmapped vertices, enabling the focused exploration of promising search regions.

VF2’s efficiency stems from its candidate generation strategy and five feasibility rules that prune infeasible mappings early. However, VF2 follows a traditional vertex-by-vertex approach: starting from an empty mapping, it incrementally adds vertex pairs through recursive backtracking. This requires extensive exploration of early search tree levels where pruning opportunities are limited and parallelization is constrained.

Our previous VF2-PS algorithm [32] extends VF2 with parallel execution while maintaining comprehensive attribute support. VF2-PS uses thread-safe state cloning and parallel-safe data structures, preserving VF2’s state-space representation with partial mapping M and frontier sets. While demonstrating significant improvements over sequential VF2, fundamental limitations persist: the vertex-centric initialization requires expensive early tree exploration, lack of explicit domain tracking results in redundant exploration of infeasible branches, and static vertex ordering prevents adaptation to graph-specific patterns, motivating HiPerMotif’s edge-centric innovations.

III. HIPERMOTIF ALGORITHM

Traditional algorithms face a fundamental bottleneck: starting from empty mappings, they must explore extensive early search tree levels where pruning is minimal and parallelization is constrained. HiPerMotif eliminates this bottleneck by systematically identifying and validating all possible first-edge mappings, then injecting these pre-validated partial states directly at depth 2. Fig. 1 illustrates the complete workflow.

A. Structural Reordering and MVE Selection

We reorder the pattern graph G_1 using the ranking function $\sigma(v) = (\text{totaldeg}(v), \text{outdeg}(v))$, placing the highest degree vertex first. This creates the Matching-optimal Viable Edge (MVE) - the first edge $(0, 1)$ becomes optimal for initialization due to high connectivity and pruning potential.

Theorem 1 (Complexity of Structural Reordering). *The algorithm 1 has time complexity $O(|V_1|^2)$ and space complexity $O(|V_1| + |E_1|)$, where $n = |V_1|$.*

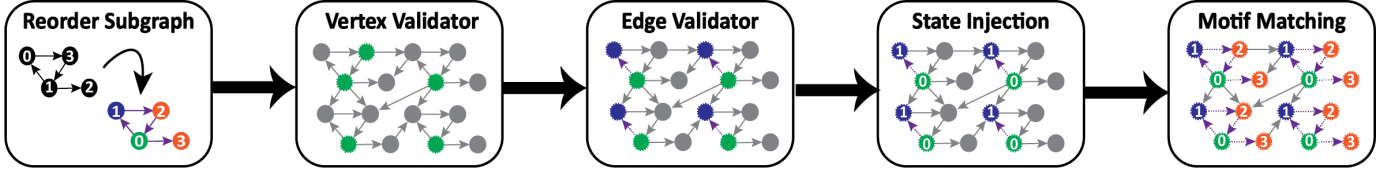


Fig. 1. HiPerMotif's workflow: structural reordering, MVE selection, vertex/edge validation, and state injection.

Algorithm 1 Structural Reordering

```

1: Compute  $\sigma(v)$  for all  $v \in V_1$ 
2:  $v^* \leftarrow \arg \max_{v \in V} \sigma(v)$ , place at position 0
3: while  $|\text{placed}| < |V_1|$  do
4:   Select highest-ranked neighbor of last-placed vertex
5:   If no neighbors, select highest-ranked remaining vertex
6: end while

```

Proof. Computing degree metrics requires scanning all edges, taking $O(|V_1| + |E_1|)$ time. The main loop executes $|V_1| - 1$ iterations. In each iteration, identifying unplaced out-neighbors takes $O(\deg(u))$, and finding the maximum-ranked vertex requires $O(|V_1|)$ time in the worst case, as $\sigma(w)$ is computed for all unplaced vertices. The `Swap` operation updates all edges involving w^* , taking $O(|E_1|)$ time across all iterations, since each edge is updated at most twice. Thus, the total time complexity is $O(|V_1| \cdot |V_1| + |E_1|) = O(|V_1|^2 + |E_1|)$, dominated by $O(n^2)$ for dense graphs. The space complexity includes the graph ($O(|V_1| + |E_1|)$), \mathcal{R} ($O(|V_1|)$), and auxiliary arrays ($O(|V_1|)$), resulting in $O(|V_1| + |E_1|)$. \square

B. Edge-Centric Validation

Two efficient validators ensure early pruning of infeasible mappings:

Vertex Validator: Identifies candidates $v \in V_2$ for vertex 0, checking attribute matches and degree constraints $T_{\text{in}}^v \geq T_{\text{in}}^0, T_{\text{out}}^v \geq T_{\text{out}}^0$.

Edge Validator: Verifies if edge $(u, v) \in E_2$ maps to MVE $(0, 1)$, checking:

- Attribute compatibility for vertices and edges
- Degree thresholds: $|T_{\text{in}}^v| \geq |T_{\text{in}}^1|, |T_{\text{out}}^v| \geq |T_{\text{out}}^1|$
- Bidirectionality requirements
- Neighbor overlap: $|N_u \cap N_v| \geq |N_0 \cap N_1|$

C. Hybrid Algorithm with State Injection

As shown in the Algorithm 2 for HiPerMotif Core,

Key Innovation: Instead of starting VF2-PS from empty mapping, HiPerMotif pre-validates all $(u, v) \rightarrow (0, 1)$ edge mappings and directly initializes VF2-PS with $\text{core}[0] = u, \text{core}[1] = v$ at depth 2.

D. Correctness and Complexity

Theorem 2 (Correctness & Completeness). *HiPerMotif finds all valid embeddings that VF2-PS would find, satisfying subgraph isomorphism constraints.*

We provide detailed proofs in [14].

Algorithm 2 HiPerMotif Core

```

1: for all  $e \in E_2$  do ▷ Parallel over edges
2:    $u \leftarrow \text{src}(e), v \leftarrow \text{dst}(e)$ 
3:   if  $\text{vertexFlag}[u] \wedge (u \neq v)$  then
4:      $s \leftarrow \text{new State}(|V_2|, |V_1|)$ 
5:     if  $\text{EdgeValidator}(u, v, s)$  then
6:        $s.\text{core}[0] \leftarrow u; s.\text{core}[1] \leftarrow v; s.\text{depth} \leftarrow 2$ 
7:        $M_{\text{new}} \leftarrow \text{VF2-PS}(s, 2)$  ▷ Continue from
8:        $M \leftarrow M \cup M_{\text{new}}$ 
9:     end if
10:  end if
11: end for

```

Complexity: HiPerMotif is correct and complete, finding all valid embeddings as VF2-PS when (u, v) matches the MVE $(0, 1)$. The time complexity is $O(|E_2| \cdot |V_2|)$ in the worst case, but the sparse graphs approach $O(|E_2|)$. The space complexity is $O(|V_1| + |V_2|)$ per state, less than domain-aware methods such as LAD [20]. Reordering reduces candidate vertices, boosting pruning, as verified in experiments.

E. Arachne Implementation

HiPerMotif extends the open-source Arkouda/Arachne framework [33], [34] with:

- **Double-index format** for $O(1)$ edge access [35], [36]
- **Contiguous attribute arrays** enabling $O(1)$ vertex and $O(\log d)$ edge attribute access [36]
- **Chapel concurrency** [37] for parallel scalability
- **Enhanced property graph support** for arbitrary vertex/edge attributes

IV. EXPERIMENTAL EVALUATION

We evaluated HiPerMotif against state-of-the-art parallel subgraph matching algorithms: VF2-PS [32], VF3P [26], and Glasgow Subgraph Solver [19] in synthetic and real-world datasets. All experiments report mean execution times of 5 independent runs with confidence intervals of 95%. The experiments were carried out on dual AMD EPYC 7713 processors (128 cores total, 2.0GHz) with 512GB RAM.

A. Datasets

Synthetic Graphs: We developed several families of synthetic graphs for systematic evaluation. Erdős-Rényi random graphs with edge probabilities $P = 0.0005$ (sparse), $P = 0.005$ (medium), and $P = 0.05$ (dense), spanning 5,000

to 130,000 vertices. Scale-free networks using the Barabási-Albert model with parameters $\alpha = 0.41$, $\beta = 0.54$, $\gamma = 0.05$, $\delta_{in} = 0.2$, $\delta_{out} = 0.2$. Small-world networks using the Watts-Strogatz model with $k = 10$, $p = 0.01$. Attributes assigned using uniform random distribution.

Real-World Networks: Neuroscience connectomes (Fly-Wire [38]–[41]: 139,255 vertices/2,700,513 edges; Hemibrain [42]: 21,739 vertices/3,550,403 edges), communication networks (EU email [43]: 265,214 vertices/420,045 edges), social networks (Twitter [44]: 81,306 vertices/1,768,149 edges), and the massive H01 [13] human cortex dataset (50K vertices/150M edges).

Query Patterns: Both established network motifs commonly used in network science literature [9], [45], also those in our previous work [46], and randomly generated subgraphs from 3 to 20 nodes to provide comprehensive structural coverage.

B. Synthetic Graph Performance

Fig. 2 shows that HiPerMotif exhibits clear performance advantages as graph size increases, particularly on medium to dense networks. On smaller or sparse graphs, HiPerMotif shows expected overhead due to parallel coordination costs, as it is specifically designed for large-scale networks where parallel processing capabilities can be fully utilized.

Scale-free and small-world networks (Figs. 3 and 4) demonstrate HiPerMotif’s effectiveness across diverse topologies, effectively handling heterogeneous degree distributions and clustered structures characteristic of real-world systems.

C. Effectiveness of Structural Reordering

Fig. 5 demonstrates the consistent effectiveness of our MVE reordering strategy through a comprehensive analysis using 60 random graphs with edge probabilities of 0.0005 to 0.1 and query patterns from 3 to 20 nodes. Reordering does not introduce performance degradation while achieving peak speedup of 5 \times and average improvement of 1.74 \times over baseline VF2-PS.

D. Real-World Network Performance

Neuroscience Connectomes:

Figures 6 and 7 show that HiPerMotif demonstrates exceptional performance on biological neural networks. Across five different randomly generated subgraph patterns, the algorithm consistently outperforms all baselines, achieving peak speedups of 66.47 \times on FlyWire and 65.71 \times on Hemibrain compared to worst-performing baselines.






Communication and Social Networks: On EU email networks, HiPerMotif achieves peak speedup of 5.92 \times , while Twitter social networks show consistent competitive performance, validating the approach’s versatility across diverse network domains beyond biological systems [5], [6].

E. Large-Scale Network Analysis: H01 Dataset

To demonstrate HiPerMotif’s capability on truly massive networks, we evaluated performance on the H01 dataset, containing 50K vertices and 150 Millions edges representing

a cubic millimeter of human cortex. This dataset represents one of the largest connectome reconstructions available and poses significant computational challenges that exceed the capabilities of existing subgraph matching algorithms.

TABLE I
PERFORMANCE OF HiPERMOTIF ON THE H01 LARGE-SCALE DATASET.
BASELINE ALGORITHMS COULD NOT COMPLETE EXECUTION DUE TO
MEMORY AND COMPUTATIONAL CONSTRAINTS.

Motif	H01 (seconds)
	571.94
	1011.62
	21.23
	363.54
	1209.82

The H01 results represent a significant achievement in computational neuroscience, as existing baseline algorithms (VF2-PS, VF3P, Glasgow) were unable to complete execution on this massive dataset due to memory limitations and computational complexity. HiPerMotif successfully processed all test motifs, with execution times ranging from 21 seconds to approximately 20 minutes, demonstrating practical feasibility for large-scale connectome analysis. These performance gains are particularly attributable to our Vertex Validator component, which enables rapid pruning of infeasible search paths and significantly reduces the computational complexity on such massive networks. This capability gap highlights the fundamental scalability advantages of our approach and opens new possibilities for analyzing the largest available brain reconstruction datasets.

F. Parallel Speedup Analysis

Fig. 8 demonstrates HiPerMotif’s robust parallel scalability, particularly for large-scale graphs where near-linear speedup is frequently achieved. The Vertex Validator enables rapid pruning with minimal synchronization overhead [47], [48], while large-scale graphs naturally generate substantial parallel workloads [49], [50]. For smaller graphs, modest gains reflect fundamental parallel computing constraints consistent with prior work [47], [51].

V. CONCLUSION

This paper presented HiPerMotif, a hybrid algorithm for subgraph isomorphism that addresses fundamental scalability limitations of existing approaches on large-scale, attribute-rich directed graphs. Through edge-centric initialization with

Performance Analysis on Erdős-Rényi Random Graphs

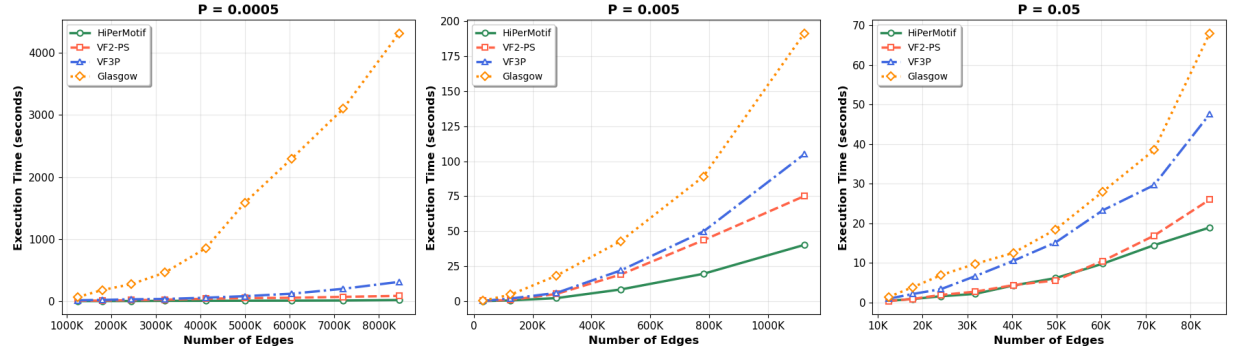


Fig. 2. Performance comparison on Erdős-Rényi random graphs with varying densities. HiPerMotif demonstrates superior scalability on larger graphs but shows overhead on sparse graphs with few nodes, as it is designed to handle massive and large-scale networks.

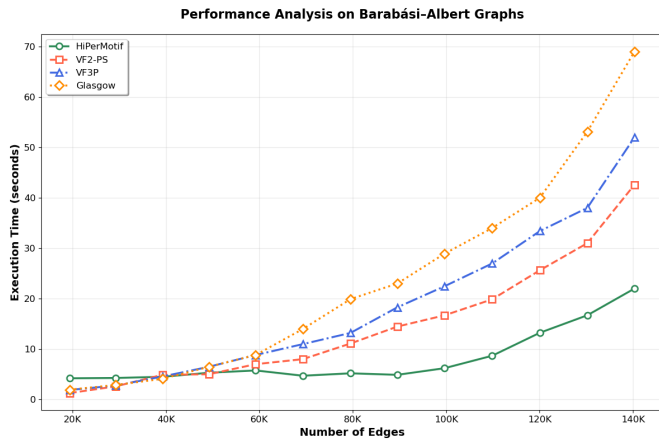


Fig. 3. Performance on scale-free networks. HiPerMotif maintains performance advantage on power-law degree distributions typical of real-world systems.

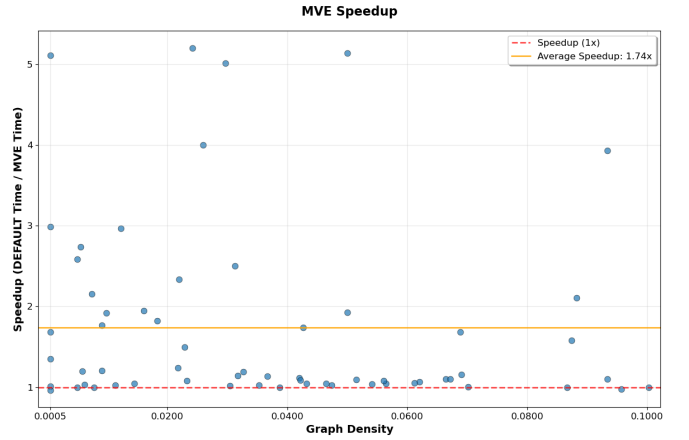


Fig. 5. Performance improvement achieved by MVE reordering on VF2-PS across diverse graph topologies and query patterns. Peak speedups of 5x and average improvement of 1.74x.

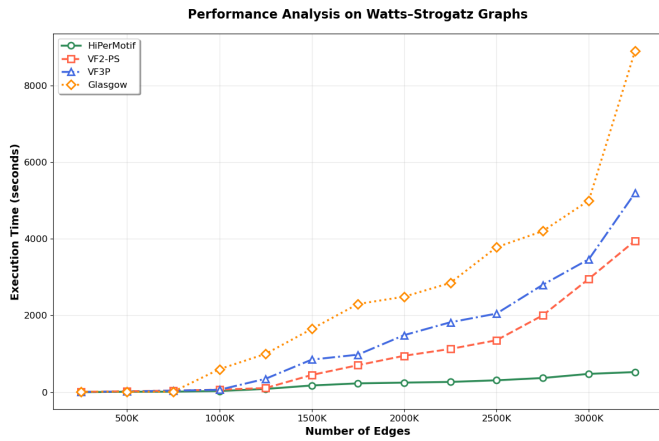


Fig. 4. Performance on Watts-Strogatz small-world networks. HiPerMotif demonstrates superior performance starting from approximately 250K edges.

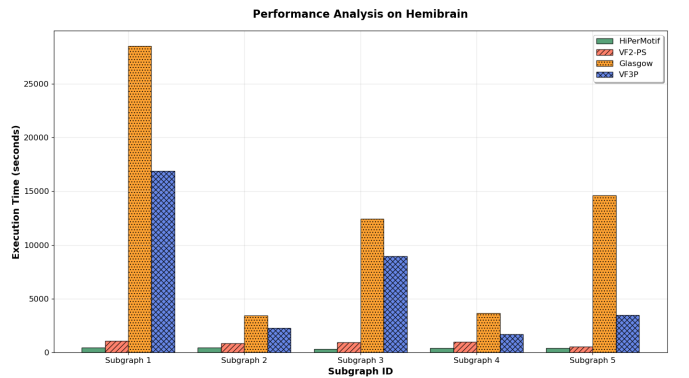


Fig. 6. Performance comparison on Hemibrain connectome dataset across five randomly generated subgraphs.

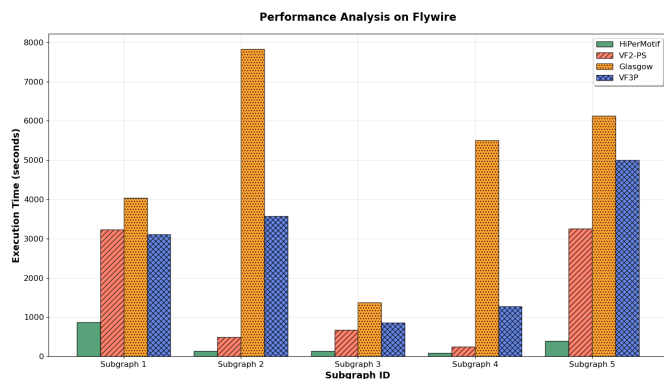


Fig. 7. Performance comparison on FlyWire connectome dataset across five randomly generated subgraphs.

state injection, structural reordering, and efficient validation mechanisms, HiPerMotif achieves significant performance improvements on large networks while enabling analysis of previously intractable datasets.

Our experimental evaluation demonstrates that HiPerMotif outperforms state-of-the-art algorithms on medium to large-scale networks, achieving speedups of up to 66× on real-world datasets where baselines complete execution, and 5× improvement through structural reordering alone. In particular, HiPerMotif successfully processes massive networks, such as the H01 connectome, that cause existing algorithms to fail due to memory constraints, representing a breakthrough for computational neuroscience applications.

HiPerMotif shows performance overhead on small graphs due to parallel coordination costs and advanced data structure overhead. The algorithm is specifically designed for large-scale networks where these overheads are amortized by substantial parallelization benefits, representing a conscious design choice prioritizing scalability over small-graph performance.

The broader impact extends beyond performance improvements, allowing research communities working with large-scale networks to tackle previously computationally prohibitive problems in computational neuroscience, social network analysis, and systems biology.

Future research directions include extending HiPerMotif to dynamic graphs, incorporating machine learning-guided edge selection, and developing adaptive mechanisms.

Complete technical details and extended evaluation available at [14]. HiPerMotif is open source and publicly available at <https://github.com/Bears-R-Us/arkouda-njit> with execution scripts and documentation.

ACKNOWLEDGMENT

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

REFERENCES

[1] O. Sporns, “Graph theory methods: applications in brain networks,” *Dialogues in clinical neuroscience*, vol. 20, no. 2, pp. 111–121, 2018.

[2] D. C. Van Essen, S. M. Smith, D. M. Barch, T. E. Behrens, E. Yacoub, K. Ugurbil, W.-M. H. Consortium *et al.*, “The WU-Minn human connectome project: an overview,” *Neuroimage*, vol. 80, pp. 62–79, 2013.

[3] S. Mangan and U. Alon, “Structure and function of the feed-forward loop network motif,” *Proceedings of the National Academy of Sciences*, vol. 100, no. 21, pp. 11 980–11 985, 2003.

[4] A.-L. Barabási, N. Gulbahce, and J. Loscalzo, “Network medicine: a network-based approach to human disease,” *Nature reviews genetics*, vol. 12, no. 1, pp. 56–68, 2011.

[5] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow, “The anatomy of the Facebook social graph,” *arXiv preprint arXiv:1111.4503*, 2011.

[6] M. Fire and R. Puzis, “Organization mining using online social networks,” *Networks and Spatial Economics*, vol. 16, no. 2, pp. 545–578, 2016.

[7] L. Akoglu, H. Tong, and D. Koutra, “Graph based anomaly detection and description: a survey,” *Data mining and knowledge discovery*, vol. 29, pp. 626–688, 2015.

[8] C. C. Noble and D. J. Cook, “Graph-based anomaly detection,” *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 631–636, 2003.

[9] U. Alon, “Network motifs: theory and experimental approaches,” *Nature Reviews Genetics*, vol. 8, no. 6, pp. 450–461, 2007.

[10] J. Lauri, “Subgraphs as a measure of similarity,” *Structural analysis of complex networks*, pp. 319–334, 2011.

[11] R. Angles, “The property graph database model,” in *Proceedings of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management, Cali, Colombia, May 21-25, 2018*, ser. CEUR Workshop Proceedings, D. Olteanu and B. Poblete, Eds., vol. 2100. CEUR-WS.org, 2018.

[12] M. A. Rodriguez and P. Neubauer, “The property graph database model,” *Computing Research Repository*, 2010.

[13] A. Shapson-Coe, M. Januszewski, D. R. Berger, A. Pope, Y. Wu, T. Blakely, R. L. Schalek, P. H. Li, S. Wang, J. Maitin-Shepard *et al.*, “A petavoxel fragment of human cerebral cortex reconstructed at nanoscale resolution,” *Science*, vol. 384, no. 6696, p. eadk4858, 2024.

[14] M. Dindoost, O. A. Rodriguez, B. Bryg, I. Koutis, and D. A. Bader, “HiPerMotif: Novel parallel subgraph isomorphism in large-scale property graphs,” 2025. [Online]. Available: <https://arxiv.org/abs/2507.04130>

[15] S. A. Cook, “The complexity of theorem-proving procedures,” in *Logic, Automata, and Computational Complexity: The Works of Stephen A. Cook*, 2023, pp. 143–152.

[16] J. R. Ullmann, “An algorithm for subgraph isomorphism,” *Journal of the ACM (JACM)*, vol. 23, no. 1, pp. 31–42, 1976.

[17] 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.

[18] V. Carletti, P. Foggia, A. Saggese, and M. Vento, “Challenging the time complexity of exact subgraph isomorphism for huge and dense graphs with VF3,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 804–818, 2017.

[19] 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.

[20] C. Solnon, “Alldifferent-based filtering for subgraph isomorphism,” *Artificial Intelligence*, vol. 174, no. 12-13, pp. 850–864, 2010.

[21] M. Han, H. Kim, G. Gu, K. Park, and W.-S. Han, “Efficient subgraph matching: Harmonizing dynamic programming, adaptive matching order, and failing set together,” in *Proceedings of the 2019 International Conference on Management of Data*, 2019, pp. 1429–1446.

[22] H. Kim, Y. Choi, K. Park, X. Lin, S.-H. Hong, and W.-S. Han, “Versatile equivalences: Speeding up subgraph query processing and subgraph matching,” in *Proceedings of the 2021 international conference on management of data*, 2021, pp. 925–937.

[23] Z. Zhang, Y. Lu, W. Zheng, and X. Lin, “A comprehensive survey and experimental study of subgraph matching: trends, unbiasedness, and interaction,” *Proceedings of the ACM on Management of Data*, vol. 2, no. 1, pp. 1–29, 2024.

[24] V. Bonnici, R. Giugno, A. Pulvirenti, D. Shasha, and A. Ferro, “RI: A graph indexing framework to speed-up graph mining,” in *Proceedings of*

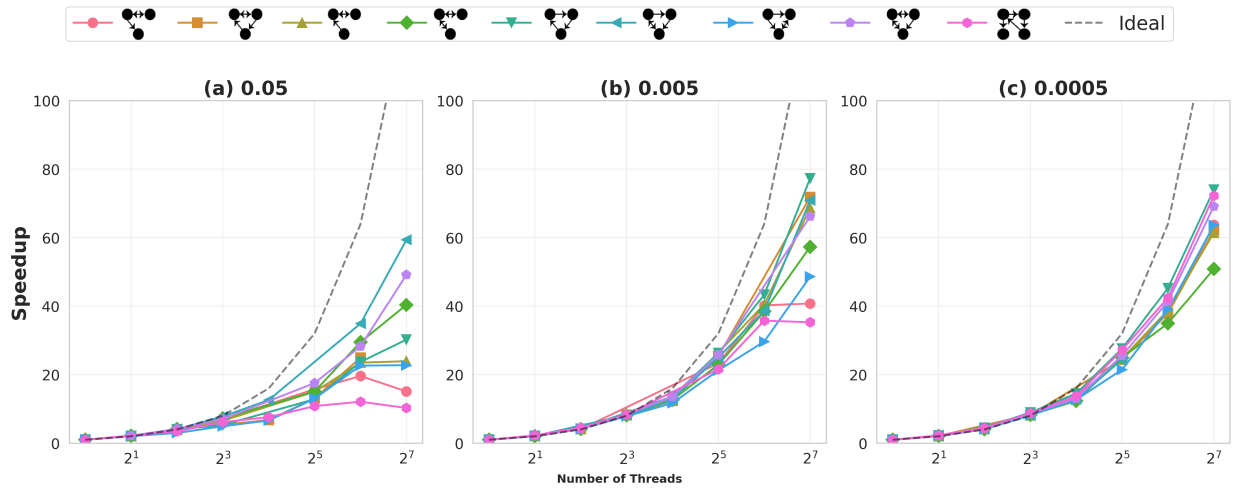


Fig. 8. Parallel speedup of HiPerMotif across Erdős-Rényi networks with varying vertex counts and edge probabilities.

- the *International Conference on Pattern Recognition Applications and Methods*, vol. 1, 2013, pp. 144–151.
- [25] F. Bi, L. Chang, X. Lin, L. Qin, and W. Zhang, “Efficient subgraph matching by postponing cartesian products,” in *Proceedings of the 2016 International Conference on Management of Data*, 2016, pp. 1199–1214.
- [26] V. Carletti, P. Foggia, P. Ritrovato, M. Vento, and V. Vigilante, “A parallel algorithm for subgraph isomorphism,” in *Graph-Based Representations in Pattern Recognition: 12th IAPR-TC-15 International Workshop, GBRPR 2019, Tours, France, June 19–21, 2019, Proceedings 12*. Springer, 2019, pp. 141–151.
- [27] W. Liang, W. Dong, and M. Yuan, “SLF: A passive parallelization of subgraph isomorphism,” *Information Sciences*, vol. 623, pp. 900–914, 2023.
- [28] J. Cheng, Y. Ke, W. Ng, and A. Lu, “FG-index: towards verification-free query processing on graph databases,” in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, 2007, pp. 857–872.
- [29] F. Katsarou, N. Ntarmos, and P. Triantafyllou, “Hybrid algorithms for subgraph pattern queries in graph databases,” in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 656–665.
- [30] L. Zeng, L. Zou, M. T. Özsu, L. Hu, and F. Zhang, “GSI: GPU-friendly subgraph isomorphism,” in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 1249–1260.
- [31] H.-N. Tran, J.-j. Kim, and B. He, “Fast subgraph matching on large graphs using graphics processors,” in *Database Systems for Advanced Applications*, M. Renz, C. Shahabi, X. Zhou, and M. A. Cheema, Eds. Cham: Springer International Publishing, 2015, pp. 299–315.
- [32] M. Dindoost, O. A. Rodriguez, S. Bagchi, P. Pauliuchenka, Z. Du, and D. A. Bader, “VF2-PS: Parallel and scalable subgraph monomorphism in Arachne,” in *Proceedings of the 28th Annual IEEE High Performance Extreme Computing Conference (HPEC)*. 28th Annual IEEE High Performance Extreme Computing Conference (HPEC), 2024.
- [33] O. A. 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.
- [34] Z. Du, O. A. Rodriguez, J. Patchett, and D. A. Bader, “Interactive graph stream analytics in Arkouda,” *Algorithms*, vol. 14, no. 8, p. 221, 2021.
- [35] Z. Du, O. Alvarado Rodriguez, J. Patchett, and D. A. Bader, “Interactive graph stream analytics in Arkouda,” *Algorithms*, vol. 14, no. 8, 2021. [Online]. Available: <https://www.mdpi.com/1999-4893/14/8/221>
- [36] O. A. 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.
- [37] 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.
- [38] S. Dorkenwald, A. Matsliha, A. R. Sterling, P. Schlegel, S.-C. Yu, C. E. McKellar, A. Lin, M. Costa, K. Eichler, Y. Yin *et al.*, “Neuronal wiring diagram of an adult brain,” *Nature*, vol. 634, no. 8032, pp. 124–138, 2024.
- [39] P. Schlegel, Y. Yin, A. S. Bates, S. Dorkenwald, K. Eichler, P. Brooks, D. S. Han, M. Gkantia, M. Dos Santos, E. J. Munnely *et al.*, “Whole-brain annotation and multi-connectome cell typing of *Drosophila*,” *Nature*, vol. 634, no. 8032, pp. 139–152, 2024.
- [40] Z. Zheng, J. S. Lauritzen, E. Perlman, C. G. Robinson, M. Nichols, D. Milkie, O. Torrens, J. Price, C. B. Fisher, N. Sharifi *et al.*, “A complete electron microscopy volume of the brain of adult *Drosophila melanogaster*,” *Cell*, vol. 174, no. 3, pp. 730–743, 2018.
- [41] J. Buhmann, A. Sheridan, C. Malin-Mayor, P. Schlegel, S. Gerhard, T. Kazimiers, R. Krause, T. M. Nguyen, L. Heinrich, W.-C. A. Lee *et al.*, “Automatic detection of synaptic partners in a whole-brain *Drosophila* electron microscopy data set,” *Nature methods*, vol. 18, no. 7, pp. 771–774, 2021.
- [42] L. K. Scheffer, C. S. Xu, M. Januszewski, Z. Lu, S.-y. Takemura, K. J. Hayworth, G. B. Huang, K. Shinomiya, J. Maitlin-Shepard, S. Berg *et al.*, “A connectome and analysis of the adult *Drosophila* central brain,” *elife*, vol. 9, p. e57443, 2020.
- [43] J. Leskovec, J. Kleinberg, and C. Faloutsos, “Graph evolution: Densefication and shrinking diameters,” *ACM transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, pp. 2–es, 2007.
- [44] J. Leskovec and J. McAuley, “Learning to discover social circles in ego networks,” *Advances in neural information processing systems*, vol. 25, 2012.
- [45] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, “Network motifs: simple building blocks of complex networks,” *Science*, vol. 298, no. 5594, pp. 824–827, 2002.
- [46] M. F. Shewarega, J. Troidl, O. A. Rodriguez, M. Dindoost, P. Harth, H. Haberkern, J. Stegmaier, D. Bader, and H. Pfister, “MoMo-combining neuron morphology and connectivity for interactive motif analysis in connectomes,” *bioRxiv*, pp. 2025–07, 2025.
- [47] C. McCreesh and P. Prosser, “A parallel, backjumping subgraph isomorphism algorithm using supplemental graphs,” in *International conference on principles and practice of constraint programming*. Springer, 2015, pp. 295–312.
- [48] V. Carletti, P. Foggia, A. Greco, A. Saggese, and M. Vento, “The VF3-light subgraph isomorphism algorithm: when doing less is more effective,” in *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshop, S+ SSPR 2018, Beijing, China, August 17–19, 2018, Proceedings 9*. Springer, 2018, pp. 315–325.
- [49] G. M. Amdahl, “Validity of the single processor approach to achieving large scale computing capabilities,” in *Proceedings of the April 18-20, 1967, spring joint computer conference*, 1967, pp. 483–485.
- [50] J. L. Gustafson, “Reevaluating Amdahl’s law,” *Communications of the ACM*, vol. 31, no. 5, pp. 532–533, 1988.
- [51] J. Cheng, Y. Ke, and W. Ng, “Efficient query processing on graph databases,” *ACM Transactions on Database Systems (TODS)*, vol. 34, no. 1, pp. 1–48, 2009.