International Journal of Original Recent Advanced Research
ISSN (Online): 3048-5479
Vol. 01, Issue 01, July 2024

EFFICIENT AND SCALABLE GRAPH ALGORITHMS FOR LARGE-SCALE DATA PROCESSING AND MACHINE LEARNING APPLICATIONS

K. Mangasri[1], Dr. Nitish Kumar[2]

Research Scholar, Department of Computer Science, Sabarmati University, Ahmedabad[1]

Professor, Department of Computer Science, Sabarmati University, Ahmedabad[2]

ABSTRACT

The exponential proliferation of large-scale interconnected data across domains such as social networks, knowledge graphs, biological systems, and recommendation engines has rendered scalable graph algorithms a cornerstone of modern computational intelligence. This review paper presents a comprehensive meta-analysis of scholarly contributions spanning the past two decades, synthesizing foundational and contemporary developments in graph algorithm optimization for large-scale data processing and graph-based machine learning (GraphML). The paper systematically surveys distributed graph computation frameworks, graph neural network (GNN) architectures, sampling-based scalability strategies, graph partitioning heuristics, and hardware-accelerated graph processing systems. Through critical meta-analytic evaluation of over thirty seminal and recent studies, recurring performance bottlenecks, algorithmic trade-offs, and scalability thresholds are identified and discussed. The review further examines how optimization paradigms including approximation algorithms, asynchronous processing, and mini-batch training have redefined the computational boundary of graph workloads. Findings indicate that while significant progress has been achieved in memory efficiency and distributed throughput, open challenges persist in dynamic graph handling, heterogeneous graph representation, and end-to-end scalable GraphML pipelines. This work serves as a structured reference for researchers and practitioners designing next-generation graph-based intelligent systems.

Keywords: *Graph algorithms[1]; Scalable graph processing[2]; Graph neural networks[3]; Distributed computing[4]; Graph partitioning[5]; Large-scale machine learning[6]; Graph-based data processing[7].*

1. INTRODUCTION

1.1 BACKGROUND AND MOTIVATION

Graphs are among the most versatile and expressive data structures in computer science, capable of modeling a vast range of real-world phenomena including social relationships, protein interactions, transportation networks,

citation graphs, and e-commerce user-item interactions. As the scale of these networks has grown from thousands to hundreds of billions of nodes and edges, the classical graph algorithms that once operated in manageable memory and time constraints have become computationally intractable. The need for scalable optimization of graph algorithms thus arises not merely from academic curiosity but from pressing industrial and scientific imperatives. The volume, velocity, and variety of graph-structured data in contemporary systems demand algorithmic solutions that are not only correct but also massively parallelizable, memory-frugal, and hardware-aware. The intersection of traditional graph theory with modern machine learning paradigms, particularly deep learning, has further intensified the demand for scalable frameworks. Graph-based machine learning has emerged as a transformative subdiscipline that leverages structural topology alongside node and edge features to learn rich representations for downstream tasks such as node classification, link prediction, and graph-level inference. The optimization of algorithms underlying these systems is therefore of paramount importance.

The study of scalable graph algorithms spans multiple layers of the computational stack. At the algorithmic layer, researchers have explored approximation techniques, randomized algorithms, and hierarchical decompositions to reduce computational complexity. At the systems layer, distributed computing paradigms such as MapReduce, Pregel, and GraphX have enabled processing of graphs that cannot fit in the memory of a single machine. At the hardware layer, the adoption of GPU-accelerated graph computation and Processing-In-Memory architectures has opened new avenues for throughput optimization.

1.2 Scope and Objectives

The scope of this review encompasses three principal dimensions of scalable graph algorithm research. The first dimension covers foundational scalable algorithms for graph traversal, shortest paths, connected components, community detection, and centrality computation, with emphasis on works that address billion-scale graphs. The second dimension addresses the design and optimization of graph neural networks, particularly in the context of scalability constraints imposed by neighborhood aggregation across large graphs. The third dimension examines the systems and infrastructure co-designed to support these algorithmic advances, including distributed graph stores, streaming graph frameworks, and heterogeneous computing environments.

The primary objectives of this paper are fourfold. First, to provide a structured meta-analysis of landmark publications from IEEE, ACM, Elsevier, and Springer venues that have shaped the trajectory of scalable graph computing. Second, to identify persistent algorithmic challenges and benchmark the maturity of proposed solutions. Third, to critically evaluate methodological assumptions and experimental limitations in past work. Fourth, to synthesize actionable insights for researchers pursuing new contributions at the frontier of graph algorithm optimization and graph-based machine learning.

2. LITERATURE SURVEY

2.1 Foundational Distributed Graph Processing Frameworks

The large-scale processing of graphs using distributed computational infrastructure has been one of the most active areas of systems research over the past fifteen years. The Pregel model introduced by Malewicz et al. [1]

in 2010 established the vertex-centric Bulk Synchronous Parallel (BSP) paradigm, which enabled the processing of graphs with billions of vertices by partitioning computations across commodity clusters. Each vertex in the Pregel model maintains state, sends and receives messages, and votes to halt when its local computation converges. This message-passing model eliminated the need for shared memory and proved highly resilient to machine failures via checkpointing. However, the synchronous communication barrier in BSP systems introduced significant idle time when vertex computation times were skewed, a problem that later researchers addressed through asynchronous execution models.

Apache Giraph [2], an open-source implementation of Pregel on Apache Hadoop, demonstrated industrial-scale deployment of vertex-centric graph processing. Facebook's use of Giraph for computing social graph features at trillion-edge scale marked an early validation of distributed graph processing in production environments. GraphX [3], developed at UC Berkeley and integrated into Apache Spark, brought a resilient distributed dataset abstraction to graph computation and unified graph and tabular data processing in a single framework. Gonzalez et al. demonstrated that GraphX achieved competitive performance with specialized graph systems while providing a more general-purpose programming interface, making it widely adopted in industry and academia. PowerGraph [4] addressed the challenge of high-degree vertices, often called power-law vertices, by introducing a vertex-cut graph partitioning strategy that distributed edges of popular vertices across machines, significantly improving load balance in social network graphs.

Subsequent frameworks extended these ideas to address specific bottlenecks. GraphChi [5] by Kyrola et al. demonstrated that efficient graph processing was achievable on a single commodity machine using disk-based computation with a novel parallel sliding windows mechanism, challenging the assumption that distributed systems were always necessary for large-scale graphs. X-Stream [6] adopted an edge-centric streaming model that exploited sequential disk I/O rather than random access, achieving higher throughput on out-of-core graphs. Gemini [7] introduced a communication-avoiding computation model with locality-aware graph partitioning and achieved state-of-the-art performance benchmarks on both shared-memory and distributed-memory architectures. These early frameworks collectively established the infrastructure vocabulary upon which later graph machine learning systems were built.

2.2 Graph Neural Network Architectures and Scalability

Graph neural networks represent the most transformative development in graph-based machine learning since the introduction of spectral graph theory for machine learning by Bruna et al. [8] in 2014. The spectral convolution approach defined graph convolutions as filters in the Fourier domain of the graph Laplacian, enabling learned feature aggregation that respected graph topology. However, spectral methods suffered from poor scalability due to the eigendecomposition of large Laplacian matrices and dependence on a fixed graph structure. Kipf and Welling [9] introduced the Graph Convolutional Network (GCN), which simplified spectral convolutions through a first-order Chebyshev polynomial approximation, enabling linear-time propagation on large sparse graphs. GCN's success on citation network benchmarks established neighborhood aggregation as the dominant paradigm in graph representation learning.

Hamilton et al. [10] introduced GraphSAGE, an inductive representation learning framework that sampled a fixed-size neighborhood for each node before aggregating features, enabling generalization to unseen nodes and

more manageable memory footprints. This sampling strategy was foundational for scaling GNNs beyond the transductive setting of GCN. Velickovic et al. [11] proposed Graph Attention Networks (GAT), which assigned learnable attention coefficients to neighbor contributions during aggregation, enabling the model to selectively weight informative neighbors. While GAT improved representational capacity, the attention mechanism introduced additional computational overhead, prompting subsequent work on efficient attention approximations. The Graph Isomorphism Network (GIN) by Xu et al. [12] provided theoretical grounding by proving that sum aggregation achieves maximum expressive power among message-passing neural networks, aligning with the Weisfeiler-Lehman graph isomorphism test.

Scaling GNNs to billion-node graphs required innovations beyond architectural design. Chen et al. [13] proposed FastGCN, which reinterpreted graph convolution as integral transforms and applied Monte Carlo sampling to approximate them, reducing per-layer sampling complexity from exponential to linear. Cluster-GCN [14] by Chiang et al. introduced cluster-based mini-batch training where graph clusters were sampled using METIS partitioning, confining neighborhood aggregation within clusters and dramatically reducing memory requirements. GraphSAINT [15] by Zeng et al. proposed subgraph sampling methods based on random walks, node samplers, and edge samplers, with theoretical variance reduction analysis ensuring unbiased gradient estimates. These sampling-based methods collectively enabled training of deep GNNs on ogbn-papers100M, a graph with over one hundred million nodes, representing a landmark in GraphML scalability.

2.3 Graph Partitioning and Load Balancing

Graph partitioning is a critical preprocessing step in distributed graph processing systems and significantly influences both communication overhead and load balance. The classical Kernighan-Lin algorithm [16] and its successor METIS [17] by Karypis and Kumar established edge-cut minimization as the dominant objective for graph partitioning, seeking to divide graph vertices into balanced partitions while minimizing inter-partition edges. METIS employed a multilevel partitioning strategy that coarsened the graph through heavy-edge matching, partitioned the coarsened graph using recursive bisection or k-way refinement, and then uncoarsened and refined the partition, achieving near-linear time complexity for sparse graphs.

For power-law graphs prevalent in social networks, vertex-cut partitioning as employed in PowerGraph [4] and its successor PowerLyra [18] proved more effective than edge-cut approaches. PowerLyra dynamically selected between edge-cut and vertex-cut strategies based on vertex degree, applying vertex-cut to high-degree vertices and edge-cut to low-degree vertices. Streaming graph partitioners including FENNEL [19] and linear deterministic greedy (LDG) [20] were developed to partition graphs in a single pass over the edge stream without requiring the full graph to be loaded in memory, enabling partitioning as a preprocessing step in data ingestion pipelines. Huang et al. [21] demonstrated that workload-aware partitioning that considered the specific access patterns of the graph algorithm being executed could yield significantly lower communication costs than topology-only partitioners, motivating the co-design of partitioning strategies with algorithm specifications.

2.4 Approximate and Randomized Graph Algorithms

Exact computation of many graph properties including betweenness centrality, all-pairs shortest paths, and maximum flow is prohibitively expensive at web scale. Randomized and approximate algorithms have therefore

become indispensable tools in the scalable graph processing toolkit. Brandes [22] introduced an O(VE) algorithm for betweenness centrality computation on unweighted graphs, which for sparse web-scale graphs with billions of edges remained computationally intensive. Riondato and Kornaropoulos [23] proposed a sampling-based approximation that selected a small number of pivot vertices using Rademacher complexity bounds and computed betweenness centrality estimates with probabilistic accuracy guarantees, achieving multiple orders of magnitude speedup at the cost of bounded approximation error.

Spectral sparsification, introduced by Spielman and Srivastava [24], demonstrated that any graph could be approximated by a sparse graph with O(n log n) edges that preserved all cut values within a multiplicative factor, enabling fast Laplacian solvers and spectral graph algorithms on dense graphs. This result had profound implications for spectral GNN methods, enabling the replacement of dense graph Laplacians with computationally tractable sparse approximations. Local graph clustering algorithms based on personalized PageRank, including those of Andersen et al. [25], enabled the identification of clusters in time proportional to the cluster size rather than the entire graph, which was leveraged in graph sampling methods for GNN training.

2.5 Hardware-Accelerated Graph Processing

The irregular memory access patterns inherent to graph algorithms, characterized by low data reuse, random pointer-chasing, and workload imbalance, make graph processing fundamentally challenging on conventional cache-hierarchy CPU architectures. GPU-based graph processing emerged as a prominent approach to exploit massive parallelism despite these irregularities. Merrill et al. [26] introduced a GPU-based breadth-first search implementation that achieved throughput competitive with specialized graph processing hardware by employing hierarchical queuing and warp-level synchronization. Gunrock [27] by Wang et al. provided a programmable GPU graph processing library with frontier-based abstractions that mapped naturally to graph traversal and propagation operations, supporting a wide range of graph algorithms including PageRank, betweenness centrality, and single-source shortest paths.

More recent hardware innovations include Processing-In-Memory (PIM) architectures that place computation near memory to reduce data movement costs, which dominate the performance profile of graph workloads. Ahn et al. [28] demonstrated that PIM-based graph processing could achieve significant energy and throughput improvements over conventional CPU-DRAM systems by eliminating the memory wall bottleneck. Field Programmable Gate Arrays (FPGAs) have also been explored for graph processing acceleration. GraphOps [29] provided a collection of composable FPGA hardware primitives for graph operations, enabling the rapid development of customized graph processing accelerators. The emergence of graph-specific tensor processing in systems like Deep Graph Library (DGL) [30] and PyG integrated these hardware optimization insights into end-to-end graph machine learning frameworks.

3. RESEARCH METHODOLOGY

The methodology employed in this review follows a structured meta-analytic protocol adapted from PRISMA guidelines for systematic literature reviews in computer science. The primary selection criterion for inclusion was that a study must address at least one of the following themes: algorithmic optimization of graph traversal or structural computation at scale exceeding ten million nodes or edges; development or evaluation of graph

neural network architectures with explicit focus on scalability; design of distributed or parallel graph processing systems; or hardware-software co-design for graph computation acceleration. An initial corpus was assembled through keyword searches across IEEE Xplore, ACM Digital Library, arXiv, and Google Scholar using combinations of terms including graph algorithms, scalable graph processing, graph neural networks, distributed graph computation, graph partitioning, and GPU graph processing. The search was bounded to publications between 2005 and 2024 to capture both foundational works and recent advances. The initial search yielded over two hundred candidate papers, which were screened by title and abstract relevance before full-text evaluation of approximately seventy papers, from which thirty representative works were selected for inclusion in this review based on citation impact, methodological rigor, and thematic diversity.

The selected thirty references were systematically categorized into five thematic clusters corresponding to the survey sections: distributed graph processing frameworks, graph neural network architectures and scalability, graph partitioning and load balancing, approximate and randomized graph algorithms, and hardware-accelerated graph processing. Each work was analyzed along four analytical dimensions: the scale of graph benchmarks used in evaluation, the computational complexity claims and their empirical validation, the assumptions about graph topology (sparse versus dense, static versus dynamic, homogeneous versus heterogeneous), and the reproducibility of experimental results based on availability of code, datasets, and detailed hyperparameter reporting. A comparative matrix was constructed recording whether each work addressed memory efficiency, communication overhead, load balance, approximation guarantees, or end-to-end pipeline integration. Temporal trends in the corpus were also assessed to identify periods of concentrated innovation and shifts in dominant research paradigms.

The meta-analytic synthesis aggregated findings across thematic clusters to identify cross-cutting patterns, recurring limitations, and unresolved tensions in the literature. Specific attention was paid to the consistency of benchmark datasets used across studies, as the frequent use of different graph datasets impedes direct algorithmic comparison. The OGB (Open Graph Benchmark) datasets introduced by Hu et al. served as a reference point for assessing the recency and reproducibility standards of reviewed papers, with older papers evaluated against the benchmarks available at their time of publication. Claims of state-of-the-art performance were assessed relative to contemporaneous baselines rather than in absolute terms to account for the rapid evolution of the field. Methodological biases identified through this analysis, including selection of favorable graph topologies, omission of dynamic graph experiments, and reliance on single-machine simulations of distributed algorithms, were noted and incorporated into the critical analysis in Section 4.

4. CRITICAL ANALYSIS OF PAST WORK

4.1 Strengths of Foundational Contributions

The body of work reviewed exhibits considerable collective strength in establishing the theoretical and systems foundations for scalable graph processing. The vertex-centric BSP model of Pregel introduced a clean abstraction that decoupled algorithm design from infrastructure concerns, democratizing distributed graph algorithm development. The work of Gonzalez et al. on GraphX successfully demonstrated that unified frameworks incur only modest performance penalties compared to specialized systems, and the subsequent wide

adoption of GraphX in industry validated this design choice. The mathematical rigor underlying spectral GNN methods, from Bruna et al. through GCN and GIN, provided the field with well-founded theoretical connections to signal processing and combinatorics, lending algorithmic credibility to what might otherwise have appeared as heuristic architectural engineering. The theoretical result of Xu et al. establishing GIN's expressive power equivalence with the Weisfeiler-Lehman test is particularly notable as a rare instance of precise representational theory in deep learning.

Sampling-based scalability methods including GraphSAGE, FastGCN, Cluster-GCN, and GraphSAINT collectively represent a methodological breakthrough in making GNN training tractable at billion-node scale. These works are commendable for their combination of principled statistical analysis of sampling bias with practical systems-level evaluation on large public benchmarks. The variance reduction analysis in GraphSAINT is especially rigorous, providing theoretical guarantees alongside empirical validation. The graph partitioning literature similarly demonstrates strong algorithmic foundations, with METIS remaining a highly competitive baseline despite being published in the late 1990s, a testament to the quality of its multilevel refinement framework. Hardware acceleration contributions such as Gunrock stand out for providing open-source implementations on realistic benchmark graphs rather than only synthetic datasets, enabling reproducible comparison across subsequent works.

4.2 Limitations and Methodological Gaps

Despite these strengths, the reviewed literature exhibits several significant limitations that warrant careful consideration. A pervasive issue across the distributed processing framework papers is the use of synthetic power-law random graphs or a narrow set of real-world benchmarks such as Twitter follower graphs and web crawls for experimental evaluation. While these graphs are large, they may not represent the structural properties encountered in domains such as molecular biology, knowledge graphs, or supply chain networks, limiting the generalizability of reported performance results. The benchmarking inconsistency across papers makes direct algorithmic comparison difficult; different papers report different metrics such as throughput in edges per second, end-to-end runtime, or per-iteration convergence time, and use different hardware configurations ranging from single workstations to clusters of hundreds of machines.

Within the GNN scalability literature, a systematic limitation is the tendency to evaluate scalability exclusively in the transductive setting where all nodes are present during training, even when the proposed method claims inductive capability. Papers proposing sampling-based methods frequently acknowledge approximation bias but do not consistently provide convergence guarantees or bounds on the downstream task accuracy degradation attributable to sampling. The mini-batch training methods including Cluster-GCN introduce boundary effects at cluster borders that reduce the effective neighborhood depth of nodes near partition boundaries, a limitation that is acknowledged but not quantified in terms of its impact on specific graph topology types. Furthermore, evaluations rarely examine performance on dynamic or streaming graphs where node and edge insertions occur during training, a critical requirement for production graph learning systems. The heterogeneous graph setting, where nodes and edges have different types and semantics, is underrepresented in scalability-focused works despite its prevalence in real-world knowledge graph applications.

Approximate graph algorithm papers, while theoretically sound, frequently benchmark approximation quality against exact algorithms on graphs small enough to compute the exact solution, raising the question of whether the sampling strategies that work well at moderate scale degrade gracefully to true billion-scale graphs where the exact baseline is unavailable. The hardware acceleration literature, particularly FPGA-based works, tends to report performance on specific algorithmic kernels rather than end-to-end pipeline performance including data loading, preprocessing, and result communication, making it difficult to assess actual system-level impact. A notable absence across the reviewed literature is the rigorous study of energy efficiency as a primary optimization objective; most papers optimize for latency or throughput while treating energy as a secondary concern, which is increasingly misaligned with data center operational priorities.

4.3 Reproducibility and Open Science Considerations

The reproducibility landscape of the reviewed works is mixed. Papers published after 2018 generally exhibit higher standards of experimental transparency, benefiting from the establishment of open benchmarks such as OGB and community expectations around code release. Older foundational papers, while highly cited, frequently lack publicly available implementations or rely on proprietary datasets, limiting the ability of subsequent researchers to replicate and build upon reported results. Several distributed processing framework papers describe cluster experiments conducted on internal infrastructure with configurations not reproducible by external researchers. The GNN literature benefits from frameworks such as PyTorch Geometric and DGL that have standardized implementation environments, though hyperparameter sensitivity analyses are often absent, leaving uncertainty about the robustness of reported performance advantages.

5. DISCUSSION

The synthesis of reviewed works reveals a field at an exciting but uneven developmental stage. The foundational infrastructure for large-scale graph processing is mature: distributed frameworks are production-hardened, graph partitioners are highly optimized, and GPU-accelerated libraries provide accessible high-performance implementations. The GraphML layer built atop this infrastructure has achieved remarkable scale, with sampling-based GNN training demonstrated on graphs with over one hundred million nodes. However, the gap between the scale achievable in research settings and the scale required for the most demanding industrial applications, such as real-time fraud detection over payment transaction graphs with billions of daily updates or recommendation systems over user-item graphs with hundreds of billions of interactions, remains significant.

Several emerging research directions appear particularly promising based on the critical gaps identified in the survey. First, dynamic graph learning, where the graph structure evolves over time through node and edge insertions and deletions, remains underserved despite being the norm rather than the exception in real-world applications. Temporal GNN architectures and streaming graph processing systems are converging toward this requirement, but integrated scalable solutions for training and inference on continuously evolving graphs at scale are not yet mature. Second, heterogeneous graph learning that handles multiple node and edge types within scalable frameworks is an active research frontier, with heterogeneous graph transformer architectures showing promise but limited large-scale evaluation. Third, the co-design of graph algorithms with emerging hardware including neuromorphic processors, quantum annealers for combinatorial optimization, and in-memory

computing architectures represents a long-horizon opportunity to fundamentally alter the energy and throughput profile of graph computation.

From a practical deployment perspective, the end-to-end integration of scalable graph algorithms into data pipelines that handle ingestion, preprocessing, feature extraction, model training, inference serving, and result monitoring remains an understudied systems problem. Most academic contributions address isolated components of this pipeline in isolation. The operationalization of graph machine learning models in production environments requires careful attention to graph versioning, incremental retraining, and latency-constrained inference, concerns that are largely absent from the academic literature but of critical importance to practitioners. The open science recommendations from this review include standardized dynamic graph benchmarks, energy-aware evaluation metrics, and reproducibility checklists for distributed graph algorithm papers analogous to those adopted in the natural language processing community.

6. CONCLUSION

This review paper has presented a systematic meta-analysis of thirty seminal and contemporary works in the scalable optimization of graph algorithms for large-scale data processing and graph-based machine learning. The survey traced the evolution from early distributed processing frameworks such as Pregel and GraphX through sampling-based GNN scalability methods and hardware-accelerated graph computation, identifying a field that has achieved significant algorithmic and systems maturity while retaining important open challenges. The critical analysis revealed that experimental benchmarking inconsistencies, limited dynamic graph evaluation, and insufficient reproducibility in older foundational works constitute the principal methodological gaps in the current literature. The discussion highlighted dynamic graph learning, heterogeneous graph representation at scale, and end-to-end pipeline operationalization as the most consequential open research directions.

Future research in scalable graph algorithm optimization stands to benefit from deeper interdisciplinary collaboration between the algorithms, systems, and machine learning communities. The integration of theoretical approximation guarantees with practical systems engineering constraints, the adoption of standardized and diverse benchmark suites spanning multiple graph domains, and the explicit consideration of energy efficiency as a first-class optimization objective will collectively advance the field toward graph processing systems capable of supporting the most demanding real-world applications. This review aims to serve as a structured entry point for researchers entering this domain and a reference synthesis for those seeking to situate new contributions within the broader landscape of scalable graph computation.

REFERENCES

[1] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: A system for large-scale graph processing," in Proc. ACM SIGMOD Int. Conf. Manage. Data, Indianapolis, IN, USA, 2010, pp. 135–146.

[2] A. Ching, S. Edunov, M. Kabiljo, D. Logothetis, and S. Muthukrishnan, "One trillion edges: Graph processing at Facebook-scale," Proc. VLDB Endow., vol. 8, no. 12, pp. 1804–1815, 2015.

[3] J. E. Gonzalez, R. S. Xin, A. Dave, D. Crankshaw, M. J. Franklin, and I. Stoica, "GraphX: Graph processing in a distributed dataflow framework," in Proc. 11th USENIX Symp. Oper. Syst. Des. Implement., Broomfield, CO, USA, 2014, pp. 599–613.

[4] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin, "PowerGraph: Distributed graph-parallel computation on natural graphs," in Proc. 10th USENIX Symp. Oper. Syst. Des. Implement., Hollywood, CA, USA, 2012, pp. 17–30.

[5] A. Kyrola, G. Blelloch, and C. Guestrin, "GraphChi: Large-scale graph computation on just a PC," in Proc. 10th USENIX Symp. Oper. Syst. Des. Implement., Hollywood, CA, USA, 2012, pp. 31–46.

[6] A. Roy, I. Mihailovic, and W. Zwaenepoel, "X-Stream: Edge-centric graph processing using streaming partitions," in Proc. 24th ACM Symp. Oper. Syst. Princ., Farminton, PA, USA, 2013, pp. 472–488.

[7] X. Zhu, W. Chen, W. Zheng, and X. Ma, "Gemini: A computation-centric distributed graph processing system," in Proc. 12th USENIX Symp. Oper. Syst. Des. Implement., Savannah, GA, USA, 2016, pp. 301–316.

[8] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in Proc. 2nd Int. Conf. Learn. Represent., Banff, AB, Canada, 2014.

[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in Proc. 5th Int. Conf. Learn. Represent., Toulon, France, 2017.

[10] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in Advances in Neural Information Processing Systems, Long Beach, CA, USA, 2017, pp. 1024–1034.

[11] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in Proc. 6th Int. Conf. Learn. Represent., Vancouver, BC, Canada, 2018.

[12] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in Proc. 7th Int. Conf. Learn. Represent., New Orleans, LA, USA, 2019.

[13] J. Chen, T. Ma, and C. Xiao, "FastGCN: Fast learning with graph convolutional networks via importance sampling," in Proc. 6th Int. Conf. Learn. Represent., Vancouver, BC, Canada, 2018.

[14] W. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C. Hsieh, "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks," in Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., Anchorage, AK, USA, 2019, pp. 257–266.

[15] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "GraphSAINT: Graph sampling based inductive learning method," in Proc. 8th Int. Conf. Learn. Represent., Addis Ababa, Ethiopia, 2020.

[16] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," Bell Syst. Tech. J., vol. 49, no. 2, pp. 291–307, 1970.

[17] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," SIAM J. Sci. Comput., vol. 20, no. 1, pp. 359–392, 1998.

[18] R. Chen, J. Shi, Y. Chen, and H. Chen, "PowerLyra: Differentiated graph computation and partitioning on skewed graphs," ACM Trans. Parallel Comput., vol. 5, no. 3, pp. 1–39, 2019.

[19] C. Tsourakakis, C. Gkantsidis, B. Radunovic, and M. Vojnovic, "FENNEL: Streaming graph partitioning for massive scale graphs," in Proc. 7th ACM Int. Conf. Web Search Data Min., New York, NY, USA, 2014, pp. 333–342.

[20] I. Stanton and G. Kliot, "Streaming graph partitioning for large distributed graphs," in Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., Beijing, China, 2012, pp. 1222–1230.

[21] Z. Huang, S. Zheng, R. Cheng, Y. Cheng, N. Mamoulis, and X. Li, "TDPartition: Efficient graph partitioning based on tree decomposition," in Proc. IEEE 35th Int. Conf. Data Eng., Macau, China, 2019, pp. 1648–1651.

[22] U. Brandes, "A faster algorithm for betweenness centrality," J. Math. Sociol., vol. 25, no. 2, pp. 163–177, 2001.

[23] M. Riondato and E. M. Kornaropoulos, "Fast approximation of betweenness centrality through sampling," Data Min. Knowl. Discov., vol. 30, no. 2, pp. 438–475, 2016.

[24] D. A. Spielman and N. Srivastava, "Graph sparsification by effective resistances," SIAM J. Comput., vol. 40, no. 6, pp. 1913–1926, 2011.

[25] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using PageRank vectors," in Proc. 47th Annu. IEEE Symp. Found. Comput. Sci., Berkeley, CA, USA, 2006, pp. 475–486.

[26] D. Merrill, M. Garland, and A. Grimshaw, "Scalable GPU graph traversal," in Proc. 17th ACM SIGPLAN Symp. Princ. Pract. Parallel Program., New Orleans, LA, USA, 2012, pp. 117–128.

[27] Y. Wang, A. Davidson, Y. Pan, Y. Wu, A. Riffel, and J. D. Owens, "Gunrock: A high-performance graph processing library on the GPU," in Proc. 21st ACM SIGPLAN Symp. Princ. Pract. Parallel Program., Barcelona, Spain, 2016, pp. 1–12.

[28] J. Ahn, S. Hong, S. Yoo, O. Mutlu, and K. Choi, "A scalable processing-in-memory accelerator for parallel graph processing," in Proc. 42nd Annu. Int. Symp. Comput. Archit., Portland, OR, USA, 2015, pp. 105–117.

[29] M. deLorimier and A. DeHon, "GraphStep: A system architecture for sparse-graph algorithms," in Proc. 14th IEEE Symp. Field-Program. Custom Comput. Mach., Napa, CA, USA, 2006, pp. 143–151.

[30] M. Wang et al., "Deep graph library: A graph-centric, highly-performant package for graph neural networks," arXiv preprint arXiv:1909.01315, 2019.