Systems for Scalable Graph Analytics and Machine Learning: Trends and Methods

Da Yan, Lyuheng Yuan, Akhlaque Ahmad, Saugat Adhikari
Department of Computer Science, Indiana University Bloomington, IN, USA
{yanda,lyyuan,akahmad,adhiksa}@iu.edu

ABSTRACT

Graph-theoretic algorithms and graph machine learning models are essential tools for addressing many real-life problems, such as social network analysis and bioinformatics. To support large-scale graph analytics, graph-parallel systems have been actively developed for over one decade, such as Google's Pregel and Spark's GraphX, which (i) promote a think-like-a-vertex computing model and target (ii) iterative algorithms and (iii) those problems that output a value for each vertex. However, this model is too restricted for supporting the rich set of heterogeneous operations for graph analytics and machine learning that many real applications demand.

In recent years, two new trends emerge in graph-parallel systems research: (1) a novel think-like-a-task computing model that can efficiently support the various computationally expensive problems of subgraph search; and (2) scalable systems for learning graph neural networks. These systems effectively complement the diversity needs of graph-parallel tools that can flexibly work together in a comprehensive graph processing pipeline for real applications, with the capability of capturing structural features. This tutorial will provide an effective categorization of the recent systems in these two directions based on their computing models and adopted techniques, and will review the key design ideas of these systems. The slides can be found at https://sites.google.com/view/system4graph-tutorial.

1 TUTORIAL RELEVANCE AND GOALS

Two new trends emerge in graph-parallel systems research that can flexibly work together in a comprehensive graph processing pipeline for real applications: (1) a novel think-like-a-task computing model that can efficiently support the various computationally expensive problems of subgraph search; and (2) scalable systems for learning graph neural networks. This tutorial will provide an effective categorization of the recent systems in these two directions based on their computing models and adopted techniques, and will review the key design ideas of these systems.

This tutorial is closely relevant to the scope of EDBT. Specifically, graph analytics and graph machine learning have been important topics with EDBT in recent decades. For example, in EDBT 2024 alone, there are 12 publications with "graph" in the title. Besides, 3 out of the 17 demonstration papers, and 1 out of the 3 tutorials, are related to graph analytics and learning. However, it is important for graph application researchers to understand the recent graph systems research and development so that they can select the right and latest graph analytics and learning systems to solve their problems at hand, and the current tutorial is designed to fulfill this demand.

2 TARGET AUDIENCE & PREREQUISITES

The target audience for this tutorial includes anyone who are interested in large-scale graph analytics and machine learning, such as (1) researchers and practitioners who would like to understand how to choose the proper systems for their graph-related applications at hand, and (2) system developers who want to learn how to design graph-parallel systems. Our tutorial will be self-explanatory with minimal prerequisites, including a basic understanding of graph theory and some familiarity with GNNs.

3 PRIOR TUTORIALS AND DIFFERENCES

We have delivered this tutorial at IJCAI 2024, KDD 2024 [40], and CIKM 2024 [39]. This tutorial will cover more up-to-date systems not covered in previous tutorials, such as the systems for compressed training of graph neural networks. We will also deliver this tutorial in SDM 2025.

4 DETAILED TUTORIAL SCOPE

4.1 Introduction

Background and Motivation. Pioneered by Google's Pregel, a lot of graph-parallel systems have been developed in the past decade that adopt a think-like-a-vertex (TLAV) programming model and iterative computation model. However, TLAV systems are dedicated to scaling those graph problems that output a value for each vertex, such as random walks (PageRank, single-source SimRank) and graph traversal (breadth-first search, single-source shortest path), while many real problems concern subgraph structures, such as finding functional groups in bioinformatics, and finding communities in interaction networks for cybersecurity applications [9].

Figure 1 summarizes a typical pipeline for graph processing, consisting of a graph analytics phase and an optional graph machine learning (ML) phase. The analytic tasks either concern individual vertices (e.g., node scoring or classification), or concern substructures or even an entire graph (e.g., dense/frequent subgraph mining, graph classification). There are four analytics paths in the pipeline:

(1) **Vertex Analytics**, which outputs a score for each vertex, useful for applications such as biomolecule prioritization in network biology, or object ranking in recommender systems.
(2) **Vertex Analytics + ML**, where the analytics stage outputs vertex embeddings for downstream ML tasks. Vertex embeddings can be learned from the graph topology as in DeepWalk and node2vec, or the vertex features may come directly from the applications or be computed based on the graph topology (e.g., in- and out-degrees, clustering coefficient).
(3) **Structure Analytics**, which outputs subgraph structures (patterns or instances), useful for finding functional groups in network biology, and community detection.
(4) **Structure Analytics + ML**, where informative structures are extracted as features for graph classification/regression.

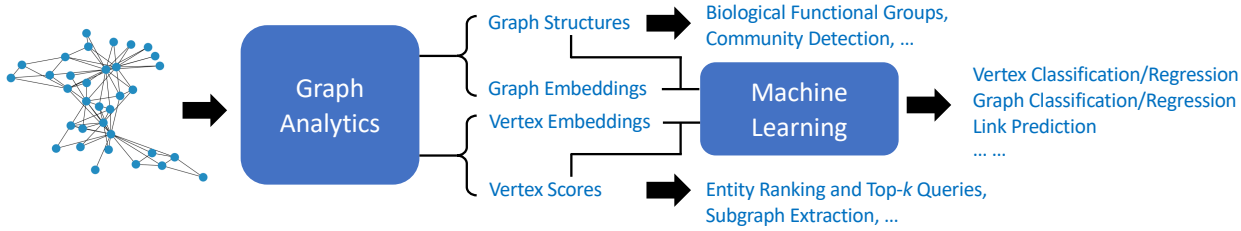

Figure 1: A Pipeline for Graph Analytics and Machine Learning

TLAV systems mainly address the scalability issue of vertex analytics (+ ML), with many killer applications in recommender systems and bioinformatics. However, many real problems concern subgraph structures, and they are actually more computationally challenging due to the exponential search space of subgraphs in a graph, but cannot be effectively accelerated by TLAV systems. For example, Chu and Cheng [6] noticed that for triangle counting, the state-of-the-art MapReduce algorithm takes 5.33 minutes using 1636 machines, while their serial external-memory algorithm takes only 0.5 minute. In fact, given an input graph $G = (V, E)$, TLAV systems are only efficient for iterative computations where each iteration has $O(|V| + |E|)$ cost and there are $O(\log |V|)$ iterations, giving a time complexity upper bound of $O\big((|V| + |E|) \log |V|\big)$ [33].

In recent years, two new trends emerge in graph-parallel systems research: (1) a lot of novel systems have been recently developed targeting the more compute-intensive subgraph search problems, which all adopt a subgraph-centric programming model (in contrast to vertex-centric); (2) graph neural networks (GNN) have boomed in various applications, and a number of scalable GNN systems have been developed. These two directions well cover the "Graph Structures" and "ML" components of the pipeline shown in Figure 1, but there currently lacks a comprehensive tutorial to survey and introduce these exciting new system advancements.

This tutorial aims to fill this gap. We offered a tutorial on graph-parallel systems in SIGMOD 2016 [32] but it mainly focused on TLAV systems. There were also recent tutorials on GNNs [2] but they focused on model design in real applications. In contrast, this tutorial introduces the parallel/distributed **system design** aspects of subgraph search and GNNs, so is unique and timely.

We remark that the two topics we cover here are related and important in order to fully explore the potential of various graph analytics tools in a real application pipeline. For example, frequent subgraph structural patterns have been found informative in conventional models for graph classification and regression [19, 21]. ML applications benefiting from having structural features include biochemistry [19], bioinformatics [20], and community detection [24], where structural features are found to outperform neural graph embedding methods. There are also works applying GNNs for approximate subgraph search, such as neural subgraph matching [41] and neural subgraph counting [27], where considering subgraph structures were found essential for good performance. Finally, Subgraph GNNs [3, 8] which model graphs as collections of subgraphs are found to be more expressive than regular GNNs.

4.2 Systems for Structure Analytics

Programmability is important for a graph-parallel system: the system should make it easy to implement a broad range of advanced parallel/distributed analytics, not much more difficult (if not easier) than their serial algorithm counterparts. TLAV systems are a good example, where the user-specified programs are often easier to implement than a serial algorithm from scratch.

Table 1: Systems for Subgraph Search: Summary of Features

	Problem Type: (Pattern-to-Instance) Subgraph Finding	Problem Type: (Pattern-to-Instance) FSM (A Big Graph)	Problem Type: (Instance-to-Pattern) FSM (transaction database)	Search Approach: Subgraph Materialization	Search Approach: Subgraph Backtracking	GPU Support
* SE/SM = Subgraph Enumeration/Matching; * EG = Entire Graph (Loaded to GPU); * PT = A Partition of a Graph (Loaded to GPU) Each Time						
Arabesque	✓	✓		✓		
RStream	✓	✓		✓		
Pangolin	✓	✓		✓		EG
Peregrine	✓	✓		✓		
Fractal	✓	✓			✓	
AutoMine	✓	✓			✓	
G-thinker	✓				✓	
G-thinkerQ						
G-Miner	✓				✓	
GraphPi	SE/SM				✓	
GraphZero	SE/SM				✓	
T-DFS	SE/SM				✓	EG
GSI, cuTS, STMatch, EGSM	SE/SM			✓		EG
PBE, VSGM, SGSI	SE/SM			✓		PT
G^2-AIMD	✓			✓		PT
DistGraph		✓		✓		
ScaleMine		✓			✓	
T-FSM		✓			✓	
PrefixFPM			✓		✓	

However, TLAV systems are not suitable for subgraph search, since computations are at individual vertices rather than subgraphs, and TLAV systems are for iterative computations with a time complexity of $O\big((|V| + |E|) \log |V|\big)$ [33].

In this tutorial, we will review a series of new systems proposed recently for subgraph search. Different from TLAV systems, these systems adopt a think-like-a-graph (TLAG) programming model, which extends valid small graph structures by one edge (or vertex) at a time to grow larger valid graph structures. However, most of these systems such as Arabesque [26], RStream [28] and Pangolin [5] adopt a breadth-first subgraph extension approach where subgraphs of size $(i + 1)$ cannot start their generation until all subgraphs with size i have been generated, which creates a lot of subgraph materialization cost and restricts scalability since the number of subgraph instances grows exponentially with the size. Some recent systems such as G-thinker [34, 35], G-Miner [4] and Fractal [7] resolve this issue by allowing depth-first subgraph-instance backtracking without actually materializing the instances. While these systems target one-time offline analytics, G-thinkerQ [43] efficiently supports interactive online querying where users continually submit subgraph queries

Table 2: Techniques of Distributed Systems for GNN Training

Systems \ Optimizations	Graph Data Communication	Operator's Scheduling	Model Computation and Synchronization	Other Optimizations	Full-Graph GNN
Euler		✓			
AliGraph		✓			
DistDGL	✓				
AGL	✓				
P^3	✓	✓	✓		
NeutronStar		✓			
ByteGNN	✓	✓			
DGCL	✓			✓	
BGL	✓	✓			
Sancus		✓	✓		
Dorylus		✓	✓	✓	
DistGNN	✓				✓
HongTu	✓				✓

with different query contents. AutoMine [18], GraphPi [22] and GraphZero [17] focus on subgraph enumeration/matching where different vertex matching order leads to different costs, and they adopt a compilation-based approach to generate subgraph enumeration code with a favorable vertex matching order.

More recently, some systems begin to explore the use of GPUs to further accelerate subgraph enumeration/matching, including BFS solutions such as GSI [47], cuTS [30], PBE [10], VSGM [12], SGSI [46] and G^2-AIMD [42], and DFS solutions such as STMatch [29], T-DFS [44] and EGSM [25].

Table 1 summarizes the key features of existing TLAG systems that we will cover in this tutorial.

4.3 Systems for Graph Machine Learning

Graph classification and regression have been conventionally solved by shallow-learning models such as support vector machines [19–21]. Recent advancement in deep learning has made graph neural networks (GNNs) (e.g., GCN, GAT) popular as downstream models for graph machine learning. GNNs operate by collecting the features of neighboring vertices and connected edges, and recursively aggregating them and transforming them into new vertex features. Taking the GraphSAGE model [11] as an example, where each graph convolution layer can be expressed as follows:

$$\mathbf{h}_{\mathcal{N}(v)}^{(k)} \leftarrow \text{AGGREGATE}_k\big(\{\mathbf{h}_u^{(k-1)} \mid \forall u \subseteq \mathcal{N}(v)\}\big),$$
$$\mathbf{h}_v^{(k)} \leftarrow \sigma\Big(\mathbf{W}^{(k)} \cdot \text{CONCAT}\big(\mathbf{h}_v^{(k-1)}, \mathbf{h}_{\mathcal{N}(v)}^{(k)}\big)\Big),$$

where $\mathcal{N}(v)$ denotes the set of v's neighboring vertices. For the k^{th} convolution layer, every vertex v first applies the AGGREGATE operation to obtain the feature vectors of its neighbors from the $(k-1)^{\text{th}}$ layer. Then, the aggregated result is merged with v's feature vector from the $(k-1)^{\text{th}}$ layer, followed by linear transformation with a trainable weight matrix $\mathbf{W}$. A non-linear activation function σ, such as the sigmoid function, is then applied to the output to obtain v's feature vector for the k^{th} layer. Here, we can see the each layer of GNN has two stages: *Graph Data Retrieving*, and *Model Computation and Synchronization*.

A large number of GNN training systems have been proposed in recent years. However, most of them are single-GPU systems, which cannot scale to industrial-scale large graphs. In this tutorial, we will focus on distributed GNN training systems. We will first present the challenges and then introduce a range of representative GNN systems with a variety of techniques designed to address the key performance bottlenecks. These systems are from both academic and industrial contexts, providing a comprehensive coverage.

Distributed GNN training presents unique challenges different from traditional machine learning tasks. Unlike training tasks in computer vision or natural language, GNN training requires access to neighborhood information that is not independent across training samples. As a result, the first challenge in distributed GNN training is the need for efficient (1) **Graph Data Communication**. Another challenge is (2) **Operator Scheduling** which needs to balance tasks among computing nodes, including subgraph sampling, neighborhood feature aggregation, and model learning operations such as loss computation, gradient calculation and parameter updating, while making optimal use of available resources. The third task is (3) **Model Computation and Synchronization**, where frequent synchronizations between nodes during training are needed to ensure consistency in model parameters, which often results in increased synchronization delay and communication overhead.

To tackle the aforementioned major challenges, many techniques have been proposed by various distributed GNN systems in recent years. Table 2 summarizes the existing distributed GNN systems that we will cover in this tutorial, along with their technique categories. We will also introduce recent works for compressed GNN training using various quantization techniques, such as EC-Graph [23], EXACT [15], F^2CGT [16] and Sylvie [48].

5 TUTORIAL OUTLINE

Our tutorial is structured as follows:

Opening: We will motivate this tutorial with the general graph processing pipeline, briefly mention the TLAV systems and their application scope, and the two new trends in graph-parallel systems research, i.e., subgraph search and GNNs.

Systems for Subgraph Finding: We will introduce subgraph finding (SF) problems and their applications. We will then review the systems for SF (including dedicated ones for subgraph enumeration/matching, some of which support GPUs) and explain their pros and cons.

Systems for Frequent Subgraph Mining (FSM): We will introduce FSM and its applications in both single-graph and transaction-database contexts, and review the systems for FSM and explain their pros and cons.

An overview on GNN systems: We will first give an overview on GNN training systems, the history and current trend.

Optimization techniques for GNN systems: We will then present various optimization techniques for distributed GNN systems, including optimizations on graph data communication, operator scheduling, model computation and synchronization.

Closing: We will summarize and envision the future trends of graph-parallel systems in both research and development.

6 PRESENTERS AND THEIR EXPERTISE

This tutorial will be delivered by Dr. Da Yan from the Department of Computer Science at Indiana University Bloomington, and his graph systems team members Lyuheng Yuan, Akhlaque Ahmad and Saugat Adhikari. Dr. Yan and his team have developed graph-analytics systems including G-thinker [9, 13, 34, 35], G-thinkerQ [43], PrefixFPM [36, 37], T-FSM [14, 45], G^2-AIMD [42] and T-DFS [44], and are the pioneers of the think-like-a-task computing model T-thinker. Dr. Yan also has extensive experience in

developing think-like-a-vertex (TLAV) graph systems and GNN. Dr. Yan and his team have a long history of developing graph-parallel systems, with dozens of related publications in top conferences such as SIGMOD, VLDB, KDD, ICDE, and top journals such as ACM TODS, VLDB Journal, IEEE TPDS and IEEE TKDE. Dr. Yan led the development of the BigGraph@CUHK platform [1] with many well-known systems following the TLAV model, a tutorial on TLAV systems in SIGMOD 2016 [32], and related books with prestigious publishers [31, 38]. Dr. Yan is the sole winner of Hong Kong 2015 Young Scientist Award in Physical/Mathematical science, and his graph systems research was funded by the DOE Office of Science Early Career Research Program in 2023.

Acknowledgment. This work was supported by DOE ECRP Award 0000278892, NSF OIA-2229394, OAC-2414474, OAC-2414185, and South Big Data Innovation Hub 2022 S.E.E.D.S. Award.

REFERENCES

[1] [n.d.]. BigGraph@CUHK. http://www.cse.cuhk.edu.hk/systems/graph/.
[2] [n.d.]. GNN Tutorials. https://graph-neural-networks.github.io/.
[3] Emily Alsentzer, Samuel G. Finlayson, Michelle M. Li, and Marinka Zitnik. 2020. Subgraph Neural Networks. In *NeurIPS*.
[4] Hongzhi Chen, Miao Liu, Yunjian Zhao, Xiao Yan, Da Yan, and James Cheng. 2018. G-Miner: an efficient task-oriented graph mining system. In *EuroSys*. ACM, 32:1–32:12.
[5] Xuhao Chen, Roshan Dathathri, Gurbinder Gill, and Keshav Pingali. 2020. Pangolin: An Efficient and Flexible Graph Mining System on CPU and GPU. *Proc. VLDB Endow.* 13, 8 (2020), 1190–1205.
[6] Shumo Chu and James Cheng. 2012. Triangle listing in massive networks. *ACM Trans. Knowl. Discov. Data* 6, 4 (2012), 17:1–17:32.
[7] Vinícius Vitor dos Santos Dias, Carlos H. C. Teixeira, Dorgival O. Guedes, Wagner Meira Jr., and Srinivasan Parthasarathy. 2019. Fractal: A General-Purpose Graph Pattern Mining System. In *SIGMOD Conference 2019*. ACM, 1357–1374.
[8] Fabrizio Frasca, Beatrice Bevilacqua, Michael M. Bronstein, and Haggai Maron. 2022. Understanding and Extending Subgraph GNNs by Rethinking Their Symmetries. In *NeurIPS*.
[9] Guimu Guo, Da Yan, M. Tamer Özsu, Zhe Jiang, and Jalal Khalil. 2020. Scalable Mining of Maximal Quasi-Cliques: An Algorithm-System Codesign Approach. *Proc. VLDB Endow.* 14, 4 (2020), 573–585.
[10] Wentian Guo, Yuchen Li, Mo Sha, Bingsheng He, Xiaokui Xiao, and Kian-Lee Tan. 2020. GPU-Accelerated Subgraph Enumeration on Partitioned Graphs. In *SIGMOD*. ACM, 1067–1082.
[11] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 1024–1034. https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html
[12] Guanxian Jiang, China Qihui Zhou, Tatiana Jin, Boyang Li, Yunjian Zhao, Yichao Li, and James Cheng. 2022. VSGM: View-Based GPU-Accelerated Subgraph Matching on Large Graphs. In *SC*. IEEE Computer Society, 739–753.
[13] Jalal Khalil, Da Yan, Guimu Guo, and Lyuheng Yuan. 2022. Parallel mining of large maximal quasi-cliques. *VLDB J.* 31, 4 (2022), 649–674.
[14] Jalal Khalil, Da Yan, Lyuheng Yuan, Jiao Han, Saugat Adhikari, Cheng Long, and Yang Zhou. 2024. FSM-Explorer: An Interactive Tool for Frequent Subgraph Pattern Mining From a Big Graph. In *ICDE*. IEEE, 5405–5408.
[15] Zirui Liu, Kaixiong Zhou, Fan Yang, Li Li, Rui Chen, and Xia Hu. 2022. EXACT: Scalable Graph Neural Networks Training via Extreme Activation Compression. In *ICLR*. OpenReview.net.
[16] Yuxin Ma, Ping Gong, Tianming Wu, Jiawei Yi, Chengru Yang, Cheng Li, Qirong Peng, Guiming Xie, Yongcheng Bao, Haifeng Liu, and Yinlong Xu. 2024. Eliminating Data Processing Bottlenecks in GNN Training over Large Graphs via Two-level Feature Compression. *Proc. VLDB Endow.* 17, 11 (2024), 2854–2866.
[17] Daniel Mawhirter, Sam Reinehr, Connor Holmes, Tongping Liu, and Bo Wu. 2021. GraphZero: A High-Performance Subgraph Matching System. *ACM SIGOPS Oper. Syst. Rev.* 55, 1 (2021), 21–37.
[18] Daniel Mawhirter and Bo Wu. 2019. AutoMine: harmonizing high-level abstraction and high performance for graph mining. In *SOSP*. ACM, 509–523.
[19] Shirui Pan and Xingquan Zhu. 2013. Graph Classification with Imbalanced Class Distributions and Noise. In *IJCAI*, Francesca Rossi (Ed.). IJCAI/AAAI, 1586–1592.
[20] Martin S. R. Paradesi, Doina Caragea, and William H. Hsu. 2007. Structural Prediction of Protein-Protein Interactions in Saccharomyces cerevisiae. In *IEEE BIBE*. IEEE Computer Society, 1270–1274.
[21] Hiroto Saigo, Sebastian Nowozin, Tadashi Kadowaki, Taku Kudo, and Koji Tsuda. 2009. gBoost: a mathematical programming approach to graph classification and regression. *Mach. Learn.* 75, 1 (2009), 69–89.
[22] Tianhui Shi, Mingshu Zhai, Yi Xu, and Jidong Zhai. 2020. GraphPi: high performance graph pattern matching through effective redundancy elimination. In *SC*. IEEE/ACM, 100.
[23] Zhen Song, Yu Gu, Jianzhong Qi, Zhigang Wang, and Ge Yu. 2022. EC-Graph: A Distributed Graph Neural Network System with Error-Compensated Compression. In *ICDE*. IEEE, 648–660.
[24] Andrew Stolman, Caleb Levy, C. Seshadhri, and Aneesh Sharma. 2022. Classic Graph Structural Features Outperform Factorization-Based Graph Embedding Methods on Community Labeling. In *SDM*. SIAM, 388–396.
[25] Xibo Sun and Qiong Luo. 2023. Efficient GPU-Accelerated Subgraph Matching. *Proc. ACM Manag. Data* 1, 2 (2023), 181:1–181:26.
[26] Carlos H. C. Teixeira, Alexandre J. Fonseca, Marco Serafini, Georgos Siganos, Mohammed J. Zaki, and Ashraf Aboulnaga. 2015. Arabesque: a system for distributed graph mining. In *SOSP*. ACM, 425–440.
[27] Hanchen Wang, Rong Hu, Ying Zhang, Lu Qin, Wei Wang, and Wenjie Zhang. 2022. Neural Subgraph Counting with Wasserstein Estimator. In *SIGMOD*. ACM, 160–175.
[28] Kai Wang, Zhiqiang Zuo, John Thorpe, Tien Quang Nguyen, and Guoqing Harry Xu. 2018. RStream: Marrying Relational Algebra with Streaming for Efficient Graph Mining on a Single Machine. In *OSDI*. USENIX Association, 763–782.
[29] Yihua Wei and Peng Jiang. 2022. STMatch: Accelerating Graph Pattern Matching on GPU with Stack-Based Loop Optimizations. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis, Dallas, TX, USA, November 13-18, 2022*, Felix Wolf, Sameer Shende, Candace Culhane, Sadaf R. Alam, and Heike Jagode (Eds.). IEEE, 53:1–53:13.
[30] Lizhi Xiang, Arif Khan, Edoardo Serra, Mahantesh Halappanavar, and Aravind Sukumaran-Rajam. 2021. cuTS: scaling subgraph isomorphism on distributed multi-GPU systems using trie based data structure. In *SC*. ACM, 69.
[31] Da Yan, Yingyi Bu, Yuanyuan Tian, and Amol Deshpande. 2017. Big Graph Analytics Platforms. *Found. Trends Databases* 7, 1-2 (2017), 1–195.
[32] Da Yan, Yingyi Bu, Yuanyuan Tian, Amol Deshpande, and James Cheng. 2016. Big Graph Analytics Systems. In *SIGMOD*, Fatma Özcan, Georgia Koutrika, and Sam Madden (Eds.). ACM, 2241–2243.
[33] Da Yan, James Cheng, Kai Xing, Yi Lu, Wilfred Ng, and Yingyi Bu. 2014. Pregel Algorithms for Graph Connectivity Problems with Performance Guarantees. *Proc. VLDB Endow.* 7, 14 (2014), 1821–1832.
[34] Da Yan, Guimu Guo, Md Mashiur Rahman Chowdhury, M. Tamer Özsu, Wei-Shinn Ku, and John C. S. Lui. 2020. G-thinker: A Distributed Framework for Mining Subgraphs in a Big Graph. In *ICDE 2020*. IEEE, 1369–1380.
[35] Da Yan, Guimu Guo, Jalal Khalil, M. Tamer Özsu, Wei-Shinn Ku, and John C. S. Lui. 2022. G-thinker: a general distributed framework for finding qualified subgraphs in a big graph with load balancing. *VLDB J.* 31, 2 (2022), 287–320.
[36] Da Yan, Wenwen Qu, Guimu Guo, and Xiaoling Wang. 2020. PrefixFPM: A Parallel Framework for General-Purpose Frequent Pattern Mining. In *ICDE 2020*. IEEE, 1938–1941.
[37] Da Yan, Wenwen Qu, Guimu Guo, Xiaoling Wang, and Yang Zhou. 2022. PrefixFPM: a parallel framework for general-purpose mining of frequent and closed patterns. *VLDB J.* 31, 2 (2022), 253–286.
[38] Da Yan, Yuanyuan Tian, and James Cheng. 2017. *Systems for Big Graph Analytics*. Springer. https://doi.org/10.1007/978-3-319-58217-7
[39] Da Yan, Lyuheng Yuan, Akhlaque Ahmad, and Saugat Adhikari. 2024. Systems for Scalable Graph Analytics and Machine Learning: Trends and Methods. In *CIKM*. ACM, 5547–5550.
[40] Da Yan, Lyuheng Yuan, Akhlaque Ahmad, Chenguang Zheng, Hongzhi Chen, and James Cheng. 2024. Systems for Scalable Graph Analytics and Machine Learning: Trends and Methods. In *KDD*. ACM, 6627–6632.
[41] Rex Ying, Zhaoyu Lou, Jiaxuan You, Chengtao Wen, Arquimedes Canedo, and Jure Leskovec. 2020. Neural Subgraph Matching. *CoRR* abs/2007.03092 (2020). arXiv:2007.03092 https://arxiv.org/abs/2007.03092
[42] Lyuheng Yuan, Akhlaque Ahmad, Da Yan, Jiao Han, Saugat Adhikari, Xiaodong Yu, and Yang Zhou. 2024. G^2-AIMD: A Memory-Efficient Subgraph-Centric Framework for Efficient Subgraph Finding on GPUs. In *ICDE*. IEEE, 3164–3177.
[43] Lyuheng Yuan, Guimu Guo, Da Yan, Saugat Adhikari, Jalal Khalil, Cheng Long, and Lei Zou. 2025. G-thinkerQ: A General Subgraph Querying System with a Unified Task-Based Programming Model. *IEEE Trans. Knowl. Data Eng., accepted and to appear* (2025).
[44] Lyuheng Yuan, Da Yan, Jiao Han, Akhlaque Ahmad, Yang Zhou, and Zhe Jiang. 2024. Faster Depth-First Subgraph Matching on GPUs. In *ICDE*. IEEE, 3151–3163.
[45] Lyuheng Yuan, Da Yan, Wenwen Qu, Saugat Adhikari, Jalal Khalil, Cheng Long, and Xiaoling Wang. 2023. T-FSM: A Task-Based System for Massively Parallel Frequent Subgraph Pattern Mining from a Big Graph. *Proc. ACM Manag. Data* 1, 1, 74:1–74:26.
[46] Li Zeng, Lei Zou, and M Tamer Özsu. 2022. SGSI–A Scalable GPU-friendly Subgraph Isomorphism Algorithm. *IEEE Transactions on Knowledge and Data Engineering* (2022).
[47] Li Zeng, Lei Zou, M. Tamer Özsu, Lin Hu, and Fan Zhang. 2020. GSI: GPU-friendly Subgraph Isomorphism. In *ICDE*. IEEE, 1249–1260.
[48] Meng Zhang, Qinghao Hu, Cheng Wan, Haozhao Wang, Peng Sun, Yonggang Wen, and Tianwei Zhang. 2024. Sylvie: 3D-Adaptive and Universal System for Large-Scale Graph Neural Network Training. In *ICDE*. IEEE, 3823–3836.