REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
04-02-2023	Final Report	10-Aug-2018 - 9-Aug-2022

4. TITLE AND SUBTITLE
Final Report: High-Performance Static and Streaming Tensor Factorization Algorithms

5a. CONTRACT NUMBER
W911NF-18-1-0344

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
611102

6. AUTHORS

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES
University of Minnesota - Minneapolis
450 McNamara Alumni Center
200 Oak Street SE
Minnepolis, MN 55455 -2070

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
73151-NC.11

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			George Karypis
UU	UU	UU	UU		19b. TELEPHONE NUMBER 612-625-4002

RPPR Final Report
as of 06-Feb-2023

Agency Code: 21XD

Proposal Number: 73151NC **Agreement Number: W911NF-18-1-0344**
INVESTIGATOR(S):

Name: George Karypis
Email: karypis@umn.edu
Phone Number: 6126254002
Principal: Y

Organization: **University of Minnesota - Minneapolis**
Address: 450 McNamara Alumni Center, Minnepolis, MN 554552070
Country: USA
DUNS Number: 555917996 EIN: 41-6007513
Report Date: 09-Nov-2022 Date Received: 04-Feb-2023
Final Report for Period Beginning 10-Aug-2018 and Ending 09-Aug-2022
Title: High-Performance Static and Streaming Tensor Factorization Algorithms
Begin Performance Period: 10-Aug-2018 **End Performance Period:** 09-Aug-2022
Report Term: 0-Other
Submitted By: George Karypis Email: karypis@umn.edu
Phone: (612) 625-4002
Distribution Statement: 1-Approved for public release; distribution is unlimited.

STEM Degrees: 4 **STEM Participants:**

Major Goals: The major goals of this project are the following: - Develop efficient and scalable serial and parallel algorithms for factorizing large, irregular, and sparse tensors. The research will address issues related to memory- and operation-efficient computations, streaming data and computations, dynamic sparsity, and cache- and locality-friendly tensor reorderings and partitionings. - Develop efficient and scalable serial and parallel algorithms for irregular computations arising when operating on large, sparse, and unstructured static and dynamic graphs. Examples of such computations arise in various network-science-related fundamental operations (e.g., triangle counting, truss-decomposition, clustering) and in computations arising in applications involving graph signal processing and graph neural network.

The overarching focus of the above research is to further our understanding on how irregular computations can be performed on emerging high-performing computing architectures.

Accomplishments: Parallel and streaming tensor decomposition algorithms: In many applications, a tensor's entries arrive in a streaming fashion for a potentially unbounded amount of time. Existing approaches for streaming sparse tensors are not practical for unbounded streaming because they rely on maintaining the full factorization of the data, which grows linearly with time. Our goal was to come up with effective streaming tensor factorization algorithms that work well with sparse streaming (`infinite') data without memory/complexity explosion. To this end, we developed CP-stream, an algorithm for streaming factorization in the model of the canonical polyadic decomposition which does not grow linearly in time or space, and is thus practical for long-term streaming. CP-stream incorporates user-specified constraints such as non-negativity which aid in the stability and interpretability of the factorization. An evaluation of CP-stream demonstrates that it converges faster than state-of-the-art streaming algorithms while achieving lower reconstruction error by an order of magnitude. We also evaluated it on real-world sparse datasets and demonstrated its usability in both network traffic analysis and discussion tracking. Our evaluation uses exclusively public datasets and our source code is released to the public as part of SPLATT, an open source high-performance tensor factorization toolkit.

Distributed-Memory Parallel Triangle Counting Algorithm: Triangle counting is a fundamental graph analytic operation that is used extensively in network science and graph mining. As the size of the graphs that needs to be analyzed continues to grow, there is a requirement in developing scalable algorithms for distributed-memory parallel systems. To address that, we developed an MPI-based distributed-memory algorithm for triangle counting using a set intersection-based approach. The key difference between our algorithm and previously proposed approaches is that it utilizes a 2D decomposition of the data and associated computations, which increases the concurrency that can be exploited and reduces the overall communication cost. Furthermore, our algorithm moves

the data among the processors by utilizing a sequence of communication steps that are similar to those used by Cannon's parallel matrix-matrix multiplication algorithm. This ensures that our algorithm is memory scalable and faces low communication overhead. We also include key optimizations that leverage the sparsity of the graph and the way the computations are structured. Some of these optimizations include enumerating the triangles using an ordering that specifically leverages hash-maps in the set intersection computation, changing the hashing routine for vertices based on the density of its adjacency list, and eliminating unnecessary intersection operations. We evaluate the performance of our algorithm on various real- world and synthetically generated graphs and compare it against other existing state-of-the-art approaches. Our experiments show that we obtain a relative speedup that range between 3.24 and 7.22 out of 10.56 across the datasets using 169 MPI ranks over the performance achieved by 16 MPI ranks. Moreover, the performance of our parallel algorithm compares favorably against those achieved by existing distributed memory algorithms that rely on 1D decomposition.

Streaming and batch algorithms for incremental truss decomposition: Real-world graphs change as new nodes and edges are added and existing nodes and edges are removed. As the graph updates with time, an important question to answer in network science is how the structure of the cohesive subgraphs (like the k-truss) change. Answering this question helps in detecting significant changes in the community structure in a social network as new relationships are formed and severed. While there are several serial and parallel algorithms for truss decomposition, these algorithms explore the entire graph. As a result, performing truss decomposition after every update can become computationally expensive. To address this problem, we developed a theory that provides an upper bound to the subset of the edges that need to be explored, such that the change in truss decomposition due to an update is guaranteed to contain within this subset. Using this, we developed an efficient incremental algorithm that explores a small set of edges. We showed that our algorithm exhibits a high degree of concurrency, which can be exploited by a parallel formulation. Finally, we extended the theory to efficiently perform batch updates. Using the batch algorithm, we can update the truss decomposition after a batch of edge updates faster than updating the truss decomposition after every edge update. We evaluated the performance of our algorithms on a sparse and a dense real-world dataset. The experiments showed that the incremental algorithm provides up to 250000 speedup when compared to using the non-incremental algorithm for performing edge insertion. Moreover, the batch algorithm consistently performs better than the incremental algorithm for batch updates, running up to 17.5 times faster in some cases. Note that our work considers the problem of updating the truss decomposition only when the stream consists of edge insertions and the theory can be extended to edge removals as well.

Distributed training of graph neural network models: Graph neural networks (GNN) have shown great success in learning from graph-structured data and they are increasingly used for very large graphs. However, scaling the computations associated with GNN training across multiple machines, is challenging. Distributed GNN training requires to read hundreds of neighbor vertex data to compute a single vertex representation, which accounts for majority of network traffic in distributed GNN training. In addition, neural network models are typically trained with synchronized stochastic gradient descent (SGD) to achieve good model accuracy. This requires the distributed GNN framework to generate balanced mini-batches, which is non-trivial given the irregular nature of the graphs. To address some of these challenges, we developed the core distributed graph processing components that can be used to develop efficient and scalable distributed GNN training. A result of this effort is DistDGL, a distributed GNN training framework that leverages the single-machine DGL framework. DistDGL deploys various performance-oriented optimizations. It distributes the graph data (graph structure and node/edge features) across all machines and run trainers, sampling servers, and in-memory key-value stores on the same set of machines. To achieve good model accuracy, DistDGL uses synchronous training and allows ego-networks forming the mini-batches to include non-local nodes. To reduce network communication, DistDGL adopts METIS to partition a graph with minimum edge cut and co-locate data with training computation. In addition, DistDGL deploys multiple load balancing optimizations to tackle the imbalance issue, including multi-constraint partitioning and two-level workload splitting. DistDGL for training GNNs on massive heterogeneous graphs in a mini-batch fashion, using distributed hybrid CPU/GPU training. DistDGLv2 places graph data in distributed CPU memory and performs mini-batch computation in GPUs. DistDGLv2 deploys an asynchronous mini-batch generation pipeline that makes computation and data access asynchronous to fully utilize all hardware (CPU, GPU, network, PCIe). The combination allows DistDGLv2 to train high-quality models while achieving high parallel efficiency and memory scalability. We demonstrate DistDGLv2 on various GNN workloads. Our results show that DistDGLv2 achieves 2-3X speedup over DistDGL and 18X speedup over Euler. It takes only 5-10 seconds to complete an epoch on graphs with hundreds of millions of vertices on a cluster with 64 GPUs.

Training Opportunities: The project has contributed to the training of the graduate students that are supported by it in the areas of algorithms, high-performance computing, scientific computing, network science, graph mining, big-data processing, graph machine learning, GPU programming, and graph neural networks.

RPPR Final Report
as of 06-Feb-2023

Results Dissemination: The dissemination of the work consisted of the published research papers reported later and software artifacts that were made available on github

https://github.com/KarypisLab
https://github.com/ShadenSmith/splatt

Honors and Awards: - Test of Time Award, SC 2021.
- Distinguished Contributions Award, PAKDD 2021.
- 10-year Highest Impact Award, ICDM 2020.
- Best Paper Award, MLG 2019.

Protocol Activity Status:

Technology Transfer: Nothing to Report

PARTICIPANTS:

Participant Type: Graduate Student (research assistant)
Participant: Maria Kalantzi
Person Months Worked: 9.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Ancy Tom
Person Months Worked: 15.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Saurav Manchanda
Person Months Worked: 6.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Shaden Smith
Person Months Worked: 3.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Rohit JV
Person Months Worked: 6.00 **Funding Support:**
Project Contribution:
National Academy Member: N

RPPR Final Report
as of 06-Feb-2023

Participant Type: Graduate Student (research assistant)
Participant: Costas Mavromatis
Person Months Worked: 6.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: PD/PI
Participant: George Karypis
Person Months Worked: 15.00 **Funding Support:**
Project Contribution:
National Academy Member: N

ARTICLES:

Publication Type: Journal Article Peer Reviewed: Y **Publication Status:** 1-Published
Journal: IEEE Transactions on Learning Technologies
Publication Identifier Type: DOI Publication Identifier: 10.1109/TLT.2022.3213635
Volume: 15 Issue: 6 First Page #: 757
Date Submitted: 2/4/23 12:00AM Date Published: 12/1/22 6:00AM
Publication Location:
Article Title: FERN: Fair Team Formation for Mutually Beneficial Collaborative Learning
Authors: Maria Kalantzi, Agoritsa Polyzou, George Karypis
Keywords: Collaborative learning, group fairness, hill climbing, partitioning, team formation
Abstract: Automated team formation is becoming increasingly important for a plethora of applications in open-source community projects, remote working platforms, as well as online educational systems. The latter case, in particular, poses significant challenges that are specific to the educational domain. Indeed, teaming students aims to accomplish far more than the successful completion of a specific task. It needs to ensure that all the members in the team benefit from the collaborative work, while also ensuring that the participants are not discriminated against with respect to their protected attributes, such as race and gender. Toward achieving these goals, this article introduces FERN, a fair team formation approach that promotes mutually beneficial peer learning, dictated by protected group fairness as equality of opportunity in collaborative learning.We formulate the problem as a multi-objective discrete optimization problem. We show this problem to be NP-hard and propose a heuristic hil
Distribution Statement: 1-Approved for public release; distribution is unlimited.
Acknowledged Federal Support: **Y**

Publication Type: Journal Article Peer Reviewed: Y **Publication Status:** 1-Published
Journal: Journal of Parallel and Distributed Computing
Publication Identifier Type: DOI Publication Identifier: 10.1016/j.jpdc.2017.11.016
Volume: 129 Issue: First Page #: 61
Date Submitted: 2/4/23 12:00AM Date Published: 7/1/19 5:00AM
Publication Location:
Article Title: Parallel cosine nearest neighbor graph construction
Authors: David C. Anastasiu, George Karypis
Keywords: Similarity search Neighborhood graph construction Bounded similarity graph Cosine similarity All-pairs Nearest neighbors Shared memory parallel
Abstract: The nearest neighbor graph is an important structure in many data mining methods for clustering, advertising, recommender systems, and outlier detection. Constructing the graph requires computing up to similarities for a set of objects. This high complexity has led researchers to seek approximate methods, which find many but not all of the nearest neighbors. In contrast, we leverage shared memory parallelism and recent advances in similarity joins to solve the problem exactly. Our method considers all pairs of potential neighbors but quickly filters pairs that could not be a part of the nearest neighbor graph, based on similarity upper bound estimates. The filtering is data dependent and not easily predicted, which poses load balance challenges in parallel execution. We evaluated our methods on several real-world datasets and found they work up to two orders of magnitude faster than existing methods, display linear strong scaling characteristics, and incur less than 1% load imbalan
Distribution Statement: 1-Approved for public release; distribution is unlimited.
Acknowledged Federal Support: **Y**

Publication Type: Journal Article Peer Reviewed: Y **Publication Status:** 1-Published
Journal: Proceedings of the AAAI Conference on Artificial Intelligence
Publication Identifier Type: DOI Publication Identifier: 10.1609/aaai.v34i05.6367
Volume: 34 Issue: 05 First Page #: 8472
Date Submitted: 2/4/23 12:00AM Date Published: 4/1/20 5:00AM
Publication Location:
Article Title: CAWA: An Attention-Network for Credit Attribution
Authors: Saurav Manchanda, George Karypis
Keywords: graph neural networks, citation analysis, network analysis
Abstract: Credit attribution is the task of associating individual parts in a document with their most appropriate class labels. It is an important task with applications to information retrieval and text summarization. When labeled training data is available, traditional approaches for sequence tagging can be used for credit attribution. However, generating such labeled datasets is expensive and time-consuming. In this paper, we present Credit Attribution With Attention (CAWA), a neural-network-based approach, that instead of using sentence-level labeled data, uses the set of class labels that are associated with an entire document as a source of distant-supervision. CAWA combines an attention mechanism with a multilabel classifier into an end-to-end learning framework to perform credit attribution. CAWA labels the individual sentences from the input document using the resultant attention-weights. CAWA improves upon the state-of-the-art credit attribution approach by not constraining a sentence
Distribution Statement: 1-Approved for public release; distribution is unlimited.
Acknowledged Federal Support: **Y**

CONFERENCE PAPERS:

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining
Date Received: 04-Feb-2023 Conference Date: 10-Aug-2022 Date Published: 11-Aug-2022
Conference Location: Washington DC USA
Paper Title: Distributed Hybrid CPU and GPU training for Graph Neural Networks on Billion-Scale Heterogeneous Graphs
Authors: Da Zheng , Xiang Song , Chengru Yang , Dominique LaSalle , George Karypis
Acknowledged Federal Support: **N**

RPPR Final Report
as of 06-Feb-2023

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: 2021 IEEE International Conference on Big Data (Big Data)
Date Received: 04-Feb-2023 Conference Date: 15-Dec-2021 Date Published: 15-Dec-2021
Conference Location: Orlando, FL, USA
Paper Title: Position-based Hash Embeddings For Scaling Graph Neural Networks
Authors: Maria Kalantzi, George Karypis
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: 2019 First International Conference on Graph Computing (GC)
Date Received: 04-Feb-2023 Conference Date: 25-Sep-2019 Date Published: 25-Sep-2019
Conference Location: Laguna Hills, CA, USA
Paper Title: Streaming and Batch Algorithms for Truss Decomposition
Authors: Venkata Rohit Jakkula, George Karypis
Acknowledged Federal Support: **N**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: ICPP 2019: 48th International Conference on Parallel Processing
Date Received: 04-Feb-2023 Conference Date: 14-Aug-2019 Date Published: 14-Aug-2019
Conference Location: Kyoto Japan
Paper Title: A 2D Parallel Triangle Counting Algorithm for Distributed-Memory Architectures
Authors: Ancy Tom , George Karypis
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: RecSys '19: Thirteenth ACM Conference on Recommender Systems
Date Received: 04-Feb-2023 Conference Date: 16-Oct-2019 Date Published: 16-Oct-2019
Conference Location: Copenhagen Denmark
Paper Title: Personalized diffusions for top-n recommendation
Authors: Athanasios N. Nikolakopoulos, Dimitris Berberidis, George Karypis , Georgios B. Giannakis
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing
Date Received: 04-Feb-2023 Conference Date: 17-Nov-2021 Date Published: 17-Nov-2021
Conference Location: Online and Punta Cana, Dominican Republic
Paper Title: Evaluating Scholarly Impact: Towards Content-Aware Bibliometrics
Authors: Saurav Manchanda, George Karypis
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: ECML/PKDD 2022
Date Received: 04-Feb-2023 Conference Date: 14-Sep-2022 Date Published:
Conference Location: Grenoble, France
Paper Title: Joint Learning of Hierarchical Community Structure and Node Representations: An Unsupervised Approach
Authors: Ancy Sarah Tom , Nesreen K. Ahmed , George Karypis
Acknowledged Federal Support: **Y**

RPPR Final Report
as of 06-Feb-2023

Partners

,

I certify that the information in the report is complete and accurate:
Signature: George Karypis
Signature Date: 2/4/23 6:08PM

1 Introduction

This report summarizes some of the work that is currently in progress that relates to developing GPU-based algorithms for a key kernel used in graph neural networks – layerwise sampling.

2 Problem Description and Contribution

In recent times, many systems and algorithms have been built to quickly extract mini-batches that are used to train a GNN model. Multiple generic systems have been built that focus on training GNNs, such as, DGL, PyG, NeuGraph, ROC, etc [Wang et al.(2019), Fey and Lenssen(2019), Ma et al.(2019), Jia et al.(2020)]. Natural graphs often exhibit a power-law degree distribution. In order to avoid encountering an exponential number of neighbors as the number of layers in the GNN increases, these systems sample a fixed number of neighbors of target vertices and utilize its features during training for the aggregate and combine steps.

Popular GNN systems perform sampling in different ways - (i) neighbor sampling: sample a fixed number of neighbors for each target vertex, (ii) layer-based sampling: sample a fixed number of neighbors from each layer, (iii) random walk algorithms. Recently, systems have emerged to reduce the time spent in sampling the neighbors of vertices in a GNN, some of them using the capabilities of GPUs to tackle sampling [Serafini and Guan(2021), Jangda et al.(2021), Zhang et al.(2021), Abadal et al.(2021), Wang et al.(2020), Lin et al.(2020), Bai et al.(2021), Pandey et al.(2020)] [Wang et al.(2021), Liu et al.(2021), Min et al.(2021)].

We note that many frameworks introduced recently are geared towards efficiently performing unbiased sampling to enable neighbor sampling. However, neighbor sampling algorithms encounter an exponential increase in neighbor vertices as the number of layers increases. To this end, we note that many recent GNNs proposed utilize biased layer-wise and layer-dependent sampling. These algorithms restrict the number of samples chosen for each layer in the GNN and thereby, limit the number of computations performed. That said, the existing frameworks do not take into account the defining characteristics of layer-based sampling algorithms for GNNs and thus, do not propose solutions that allow fast generation of mini-batches for relevant GNN models. Some of the main limitations involved in the existing approaches include: (i) inadequate memory management strategies, (ii) varying number of trials to sample distinct vertices in each layer, which are based on the degree of the target vertices, (iii) serial execution of training and mini-batch generation, and finally, (iv) serial generation of multiple mini-batches. To this end, we propose the following contributions in our work: (i) we present a framework to efficiently execute layer-based sampling algorithms. (ii) we use reservoir sampling to perform biased sampling. Reservoir sampling does not need multiple trials while sampling to achieve distinct results. (iii) we parallelize the sampling algorithm using GPUs which eliminates the multiple transfers of data from the CPU memory to the GPU memory.

3 Progress

We present a framework that efficiently executes the layer-based sampling algorithm on the GPUs to generate mini-batches for GNN training. In our framework, we focus on optimizing the layer-dependent sampling, called LADIES, described in [Zou et al.(2019)]. The LADIES sampling process samples, using importance sampling, a fixed number of vertices from the set of adjacent vertices of the seeds for each layer in the L-layer GNN. The input to the algorithm is a fixed-size number of seeds and the output is an L-length list of the bipartite graphs constructed as described below. This list of bipartite graphs is given as an input to the training algorithm.

The graph is stored in the CSR format in the memory. At a given layer, the neighboring vertices of the seeds are determined and the frequency of each of these vertices is considered as its weight. The vertices with non-zero weights are used in the sampling process. These weights are subsequently normalized and used as sampling probabilities for importance sampling. Importance sampling uses biased sampling where the selection of a vertex is biased by its weight. In our framework, Weighted Reservoir Sampling (WRS) is used to perform biased sampling. The seeds and their respective sampled vertices are used to train the GNN

model using the bipartite graph constructed between the seeds and the sampled vertices. The edges between these two sets of vertices are determined by the existing edges between these sets in the original graph. These edges decides the vertices used in the aggregation step of a seed. Finally, this set of sampled vertices is used as the seeds in the next layer of the GNN. Subsequently, the sampling process and the construction of bipartite graphs connecting sampled vertices to their respective seed vertices is repeated for each layer of the L layers. This is returned as a result of the sampling algorithm which the model can then use to train.

To efficiently execute the above described steps for a given layer on a GPU, we determine an appropriate strategy to handle the neighborhood vertices of the given set of seeds. Let the number of unique neighbor vertices with non-zero weights be denoted by nnz. We re-index the neighbor vertices such that they range from 0 to $nnz - 1$ and refer to these as local vertex ids. We use these local vertex ids as a mapping to determine if a vertex is included in the sample or not. In our implementation, we maintain an array of nnz flags such that if a neighbor vertex id has been sampled, we set the value at the position corresponding to the local vertex id as 1. Further, we recognize that performing the steps included in re-indexing, sampling and the creation of the bipartite graphs are easily done by using consecutive GPU threads and inbuilt optimized libraries. Thus, these result in load-balanced computations. This is in comparison to other methods that assign a block of threads to process the neighboring vertices of a given seed which subsequently lead to load imbalanced processing. Finally, once the samples have been determined, the creation of the bipartite graph is performed by using the array of flags which discards the neighbor vertices in the adjacency list of seeds that are not included in the sample list.

4 Results

Preliminary results of our approach has been detailed in Table 1. To study the scaling behavior of our approach, we conduct experiments with the number of seeds assigned values $2,000$, $10,000$, and $100,000$. Moreover, we set the number of sampled vertices at $1,000$ while using $2,000$ as the number of seeds and $5,000$ in other cases. We use $32, 64, 128, 256, 512$ and 1024 as the number of threads in a block. We conduct our experiments on *Orkut* and *Reddit* graph datasets. *Orkut* has 3072441 vertices and 117185082 edges, while *Reddit* has 231443 vertices and 23213838 edges.

The experiments were conducted on GPU `GeForce RTX 3090`. `GeForce RTX 3090` uses the Ampere microarchitecture, has 82 SMs each with 128 cores. Each SM has a register of size 256KB, L1 cache of size 128KB and L2 cache 6144KB. Global memory in the GPU is of size 24576MB. We use CUDA version 11.0.

Overall, we observe that the approach results in good performance and low runtimes for both the datasets. Although the number of seeds is increases from $2,000$ to $100,000$, the increase in respective runtimes is not as drastic. We conjecture that the runtimes obtained while using $2,000$ seeds is more influenced by system overheads since the number of operations performed for the computations are low. In each of the dataset, we note that using the number of threads as a multiple of 128 in a thread block results in lower runtimes as compared to using number of threads equal to 32 or 64. We observe that using 128 threads in a thread block allows the entire block to reside in the same SM. Since each SM has 128 cores, the latency incurred during processing is minimized.

References

[Abadal et al.(2021)] Sergi Abadal, Akshay Jain, Robert Guirado, Jorge López-Alonso, and Eduard Alarcón. 2021. Computing graph neural networks: A survey from algorithms to accelerators. *ACM Computing Surveys (CSUR)* 54, 9 (2021), 1–38.

[Bai et al.(2021)] Youhui Bai, Cheng Li, Zhiqi Lin, Yufei Wu, Youshan Miao, Yunxin Liu, and Yinlong Xu. 2021. Efficient Data Loader for Fast Sampling-Based GNN Training on Large Graphs. *IEEE Transactions on Parallel and Distributed Systems* 32, 10 (2021), 2541–2556.

Table 1: Runtime for obtaining results for a 1-layer GNN.

Dataset	#seeds	#samples	#threads in block	runtime (s)
Orkut	2,000	1,000	32	0.001132
Orkut	2,000	1,000	64	0.001120
Orkut	2,000	1,000	128	0.001099
Orkut	2,000	1,000	256	0.001117
Orkut	2,000	1,000	512	0.001109
Orkut	2,000	1,000	1024	0.001096
Orkut	10,000	5,000	32	0.001885
Orkut	10,000	5,000	64	0.001765
Orkut	10,000	5,000	128	0.001726
Orkut	10,000	5,000	256	0.001720
Orkut	10,000	5,000	512	0.001723
Orkut	10,000	5,000	1024	0.001750
Orkut	100,000	5,000	32	0.005933
Orkut	100,000	5,000	64	0.005405
Orkut	100,000	5,000	128	0.005169
Orkut	100,000	5,000	256	0.005146
Orkut	100,000	5,000	512	0.005163
Orkut	100,000	5,000	1024	0.005222
Reddit	2,000	1,000	32	0.001549
Reddit	2,000	1,000	64	0.001364
Reddit	2,000	1,000	128	0.001367
Reddit	2,000	1,000	256	0.001371
Reddit	2,000	1,000	512	0.001377
Reddit	2,000	1,000	1024	0.001471
Reddit	10,000	5,000	32	0.002432
Reddit	10,000	5,000	64	0.002254
Reddit	10,000	5,000	128	0.002217
Reddit	10,000	5,000	256	0.002228
Reddit	10,000	5,000	512	0.002231
Reddit	10,000	5,000	1024	0.002256
Reddit	100,000	5,000	32	0.008914
Reddit	100,000	5,000	64	0.007827
Reddit	100,000	5,000	128	0.007521
Reddit	100,000	5,000	256	0.007464
Reddit	100,000	5,000	512	0.007419
Reddit	100,000	5,000	1024	0.007835

[Fey and Lenssen(2019)] Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* (2019).

[Jangda et al.(2021)] Abhinav Jangda, Sandeep Polisetty, Arjun Guha, and Marco Serafini. 2021. Accelerating graph sampling for graph machine learning using gpus. In *Proceedings of the Sixteenth European Conference on Computer Systems*. 311–326.

[Jia et al.(2020)] Zhihao Jia, Sina Lin, Mingyu Gao, Matei Zaharia, and Alex Aiken. 2020. Improving the accuracy, scalability, and performance of graph neural networks with roc. *Proceedings of Machine Learning and Systems* 2 (2020), 187–198.

[Lin et al.(2020)] Zhiqi Lin, Cheng Li, Youshan Miao, Yunxin Liu, and Yinlong Xu. 2020. Pagraph: Scaling gnn training on large graphs via computation-aware caching. In *Proceedings of the 11th ACM Symposium on Cloud Computing*. 401–415.

[Liu et al.(2021)] Tianfeng Liu, Yangrui Chen, Dan Li, Chuan Wu, Yibo Zhu, Jun He, Yanghua Peng, Hongzheng Chen, Hongzhi Chen, and Chuanxiong Guo. 2021. BGL: GPU-Efficient GNN Training by Optimizing Graph Data I/O and Preprocessing. *arXiv preprint arXiv:2112.08541* (2021).

[Ma et al.(2019)] Lingxiao Ma, Zhi Yang, Youshan Miao, Jilong Xue, Ming Wu, Lidong Zhou, and Yafei Dai. 2019. Neugraph: parallel deep neural network computation on large graphs. In *2019 {USENIX} Annual Technical Conference ({USENIX}{ATC} 19)*. 443–458.

[Min et al.(2021)] Seung Won Min, Kun Wu, Sitao Huang, Mert Hidayetoğlu, Jinjun Xiong, Eiman Ebrahimi, Deming Chen, and Wen-mei Hwu. 2021. Large Graph Convolutional Network Training with GPU-Oriented Data Communication Architecture. *arXiv preprint arXiv:2103.03330* (2021).

[Pandey et al.(2020)] Santosh Pandey, Lingda Li, Adolfy Hoisie, Xiaoye S Li, and Hang Liu. 2020. C-SAW: A framework for graph sampling and random walk on GPUs. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–15.

[Serafini and Guan(2021)] Marco Serafini and Hui Guan. 2021. Scalable Graph Neural Network Training: The Case for Sampling. *ACM SIGOPS Operating Systems Review* 55, 1 (2021), 68–76.

[Wang et al.(2019)] Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, et al. 2019. Deep Graph Library: Towards Efficient and Scalable Deep Learning on Graphs. (2019).

[Wang et al.(2021)] Pengyu Wang, Chao Li, Jing Wang, Taolei Wang, Lu Zhang, Jingwen Leng, Quan Chen, and Minyi Guo. 2021. Skywalker: Efficient Alias-Method-Based Graph Sampling and Random Walk on GPUs. In *2021 30th International Conference on Parallel Architectures and Compilation Techniques (PACT)*. IEEE, 304–317.

[Wang et al.(2020)] Yuke Wang, Boyuan Feng, Gushu Li, Shuangchen Li, Lei Deng, Yuan Xie, and Yufei Ding. 2020. Gnnadvisor: An efficient runtime system for gnn acceleration on gpus. *arXiv preprint arXiv:2006.06608* (2020).

[Zhang et al.(2021)] Lizhi Zhang, Zhiquan Lai, Shengwei Li, Yu Tang, Feng Liu, and Dongsheng Li. 2021. 2PGraph: Accelerating GNN Training over Large Graphs on GPU Clusters. In *2021 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 103–113.

[Zou et al.(2019)] Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. 2019. Layer-dependent importance sampling for training deep and large graph convolutional networks. *Advances in neural information processing systems* 32 (2019).