POSTER: An Efficient Uncertain Graph Processing Framework for Heterogeneous Architectures

Heng Zhang
zhangheng17@iscas.ac.cn
Institution of Software, Chinese Academy of Sciences
University of Sydney

Lingda Li
lli@bnl.gov
Brookhaven National Laboratory
New York, USA

Donglin Zhuang
donglinzhuang@outlook.com
University of Sydney
Sydney, Australia

Rui Liu
ruiliu@cs.uchicago.edu
University of Chicago
Chicago, USA

Shuang Song
shuangsong@fb.com
Facebook Inc.
Montain View, USA

Dingwen Tao
dingwen.tao@wsu.edu
Washington State University
Washington, USA

Yanjun Wu
yanjun@iscas.ac.cn
Institution of Software, Chinese Academy of Sciences
Beijing, China

Shuaiwen Leon Song
shuaiwen.song@sydney.edu.au
University of Sydney
Sydney, Australia

Abstract

Uncertain or probabilistic graphs have been ubiquitously used in many emerging applications. Previously CPU based techniques were proposed to use sampling but suffer from (1) low computation efficiency and large memory overhead, (2) low degree of parallelism, and (3) nonexistent general framework to effectively support programming uncertain graph applications. To tackle these challenges, we propose a general uncertain graph processing framework for multi-GPU systems, named `BPGraph`. Integrated with our highly-efficient path sampling method, `BPGraph` can support a wide range of uncertain graph algorithms' development and optimization. Extensive evaluation demonstrates a significant performance improvement from `BPGraph` over the state-of-the-art uncertain graph sampling techniques.

CCS Concepts: • **Computing methodologies** → **Parallel programming languages**; • **Computer systems organization** → *Parallel architectures.*

Keywords: uncertain graph, path sampling, GPU

ACM Reference Format:
Heng Zhang, Lingda Li, Donglin Zhuang, Rui Liu, Shuang Song, Dingwen Tao, Yanjun Wu, and Shuaiwen Leon Song. 2021. POSTER: An Efficient Uncertain Graph Processing Framework for Heterogeneous Architectures. In *26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '21), February 27–March 3, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3437801.3441584

PPoPP '21, February 27–March 3, 2021, Virtual Event, Republic of Korea

ACM ISBN 978-1-4503-8294-6/21/02…$15.00
https://doi.org/10.1145/3437801.3441584

1 Introduction

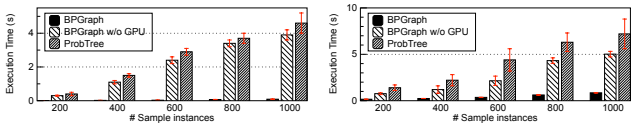
Uncertainty is intrinsic to a wide spectrum of applications, which inevitably applies to graph data such as noisy measurement in bio-informatics [2, 9], connection probability in peer-to-peer network [8], congestion prediction in traffic network [7], etc. In the literature, uncertain graphs have been widely utilized to represent these uncertainties, in which, the existence of the connection is supposed to be independently indeterminate, and is formulated as a probabilistic edge. Solving an uncertain graph problem requires to iterate through all its instances, a.k.a., possible worlds. As the number of possible worlds grows exponentially to the number of edges, it is unrealistic to get exact solutions.

Previous works on uncertain graph analysis have sought to sampling based methods to get approximate solutions on uncertain graphs [4, 5, 11, 14]. With a reasonable amount of samples, an approximate solution can be estimated with an accuracy guarantee. The three main limitations of these existing approaches are as follows.

First, *poor computation and memory efficiency caused by unnecessary edge sampling*. Our evaluation shows that the existing approaches suffer from significant performance and memory overhead, especially for large uncertain graphs.

Table 1. *Uncertain graph datasets.*

Dataset	*Vertices*	*Edges*	*Size*	*Edge Prob*
netHept[1]	15,233	62,774	921KB	0.04±0.04
gnutellaP2P[1]	62,586	147,892	1.8MB	0.23±0.20
coauthor-DBLP[1]	540,486	15,245,729	331MB	0.11±0.09
soc-twitter[10]	28,504,110	531,000,244	14GB	0.46±0.28
uk-2005[1]	39,454,748	936,364,284	20GB	0.32±0.25



(a) *netHept* **(b)** *gnutellaP2P*

Figure 1. *GPU-accelerated performance comparison.*

Second, *lack of utilization for modern general-purpose accelerators* (e.g., GPUs) that are widely available in high performance systems. Packed with massive parallelism and high-bandwidth memory, modern GPU is an attractive computing platform for uncertain graph processing [3, 6, 12, 13, 15].

Third, *lack of a general support for developing efficient uncertain graph applications*. Previous uncertain graph processing solutions only provide ad-hoc optimizations on one or several applications, while do not provide a general programmable interface for users to effectively implement a wide-range of applications. In summary, an efficient, scalable, and programmable uncertain graph processing framework is desirable.

2 Contributions

Our *novel path sampling method* aims to eliminate unnecessary edge sampling at runtime and improve computation and memory efficiency. It can effectively identify useless edges before performing sampling, and only considers edges that are on a path between the source and target vertices to be useful.

Centered around this sampling method, we present `BPGraph`, an efficient, scalable, and programmable uncertain graph processing framework that helps implement path sampling on GPUs. `BPGraph` provides a general programming interface so that users can implement various uncertain graph processing applications with ease. Additionally, it optimizes the data organization and computation patterns to better map uncertain graph processing onto GPUs, and is able to scale on multi-GPU systems.

3 Experiments

We perform the experiments on a NVIDIA DGX server with 8 NVIDIA V100 GPUs. Each GPU has 80 Streaming Multiprocessors, 32GB global memory and 768KB L2 cache. The system also consists of two 10-core Intel(R) Xeon(R) CPU E5-2698 v4, and 504GB DDR4 main memory, running with Ubuntu 18.04 (kernel v4.15.0) and CUDA 10.0.

Datasets. Table 1 shows the real-world graphs with a broad range of sizes and features. The edge probabilities in datasets come from real-world application criteria or are assigned uniformly and randomly within a specified value range.

Parameter Setting. For query pairs, we select 10 different source vertices, uniformly at random from the datasets. Next, the target vertices are chosen from n hops from the source vertices, uniformly at random, in which n is randomly selected from 2 to the diameter. The reported results of s-t query are calculated by averaging those of pairs. Initially, the value K, i.e., # samples, is 100. It increases at a step of 200 till the results converge.

State-of-the-arts. We evaluate our system by comparing with four state-of-the-art methods: *MC-Sampling (MC)* [14], *BitEdge-Aware Sampling (BA)* [16], *ProbTree (PT)* [11], *DistR (DR)* [4]. We compare their performance for three applications: source-to-target reliability query, k-nearest neighbor (KNN), and any-pair shortest path.

Performance Comparison. Compared to multi-CPU versions of *MC*, *BA*, *PT*, and *DR*, *BPGraph* significantly outperforms them in all applications. For the s-t query, *BPGraph* is on average 39× and 30× faster than entirety sampling methods, i.e., MC sampling and BA Sampling, respectively. Compared to the partition sampling method (ProbTree), *BPGraph* is 26× faster. For KNN and shortest path, *BPGraph* presents even higher speedups, e.g., it achieves an average speedup of 69× and 43× compared to MC sampling and ProbTree respectively. It is because these two applications have much larger workloads due to recursive traversal and sampling. For the reason of *BPGraph*'s superior performance, it is because 1) *BPGraph* only traverses the graph once and also samples useful edges only, 2) the high-parallelism computing capability of GPU over multi-core CPUs, and 3) the high memory bandwidth of the NVIDIA V100 GPU over the CPU.

Moreover, to show the benefits of path sampling, we implement a version of CPU-based *BPGraph* without utilizing GPUs. Figure 1 compares the performance of *BPGraph* on GPU, *BPGraph* on CPU, and ProbTree, which performs best among four state-of-the-art methods. Even without GPU acceleration, *BPGraph* on CPU still achieves better performance than ProbTree, which indicates the effectiveness and efficiency of our proposed path sampling method. Also, we observe that the performance of *BPGraph* on GPU achieves over 20× improvement compared with *BPGraph* on CPU.

Scalability over Multiple GPUs. We evaluate the scalability of *BPGraph* with 2 large graphs, i.e., *soc-twitter* and *uk-2005*, on 8 GPUs. The performance does scale well as we add more GPUs. Specifically, the performance of multi-GPU execution on *soc-twitter* achieves average 1.4×, 3.0× speedup

from 1 GPU to 2 GPUs, 8 GPUs. *BPGraph* reduces the communication overhead by using the vertex status array to mark the master/slave vertices for value synchronization, which minimizes unnecessary GPU communication. On the *uk-2005* datase, *BPGraph* achieves almost 4.2× speedup using 8 GPUs (6207s) over using 1 GPU (1498s). This is because the comparatively peer-to-peer communication gives us a benefit of scalability on larger uncertain graph datasets on multi-GPU servers.

4 Conclusion

In this work, we develop *BPGraph*, a novel GPU-based uncertain graph analytic system. It produces over 58× speedup over the state-of-the-art multi-core CPU-based uncertain graph algorithms, and 20× over a GPU-implementation of MC-sampling. This is achieved thanks to our proposed *path sampling* methodology and its efficient GPU implementation. For future work, we plan to extend *BPGraph* with distributed configuration based on high-speed storage and networks, to enable for even larger-scale uncertain graph processing.

5 Acknowledgments

This work was partially supported by Australian Research Council (ARC) Discovery Project DP210101984 and University of Sydney Computer Science start-up funding; Grant No. 62002350 from the National Natural Science Foundation of China, Grant ZDBS-LY-JSC038 from the Key Research Program of Frontier Sciences, Chinese Academy of Sciences.

References

[1] http://law.di.unimi.it/webdata/. LAW web dataset. (http://law.di.unimi.it/webdata/).

[2] Michael O Ball. 1986. Computational complexity of network reliability analysis: An overview. *IEEE Transactions on Reliability* 35, 3 (1986), 230–239.

[3] Tal Ben-Nun, Michael Sutton, Sreepathi Pai, and Keshav Pingali. 2017. Groute: An asynchronous multi-GPU programming model for irregular computations. In *Proceedings of the 22nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. ACM, 235–248.

[4] Yurong Cheng, Ye Yuan, Lei Chen, Guoren Wang, Christophe Giraud-Carrier, and Yongjiao Sun. 2016. Distr: A distributed method for the reachability query over large uncertain graphs. *IEEE Transactions on Parallel and Distributed Systems* 27, 11 (2016), 3172–3185.

[5] George S Fishman. 1986. A comparison of four Monte Carlo methods for estimating the probability of st connectedness. *IEEE Transactions on reliability* 35, 2 (1986), 145–155.

[6] Sungpack Hong, Tayo Oguntebi, and Kunle Olukotun. 2011. Efficient parallel graph exploration on multi-core CPU and GPU. In *Parallel Architectures and Compilation Techniques (PACT), 2011 International Conference on*. IEEE, 78–88.

[7] Ming Hua and Jian Pei. 2010. Probabilistic path queries in road networks: traffic uncertainty aware path selection. In *Proceedings of the 13th International Conference on Extending Database Technology*. 347–358.

[8] Ruoming Jin, Lin Liu, Bolin Ding, and Haixun Wang. 2011. Distance-constraint reachability computation in uncertain graphs. *Proceedings of the VLDB Endowment* 4, 9 (2011), 551–562.

[9] Arijit Khan and Lei Chen. 2015. On uncertain graphs modeling and queries. *Proceedings of the VLDB Endowment* 8, 12 (2015), 2042–2043.

[10] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is Twitter, a social network or a news media?. In *WWW '10: Proceedings of the 19th international conference on World wide web* (Raleigh, North Carolina, USA). ACM, New York, NY, USA, 591–600.

[11] Silviu Maniu, Reynold Cheng, and Pierre Senellart. 2017. An Indexing Framework for Queries on Probabilistic Graphs. *ACM Transactions on Database Systems* 42, 2 (2017).

[12] Duane Merrill, Michael Garland, and Andrew Grimshaw. 2012. Scalable GPU graph traversal. In *ACM SIGPLAN Notices*, Vol. 47. ACM, 117–128.

[13] Duane Merrill, Michael Garland, and Andrew Grimshaw. 2015. High-performance and scalable GPU graph traversal. *ACM Transactions on Parallel Computing* 1, 2 (2015), 14.

[14] Panos Parchas, Francesco Gullo, Dimitris Papadias, and Franceseco Bonchi. 2014. The pursuit of a good possible world: extracting representative instances of uncertain graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on management of data*. 967–978.

[15] Dipanjan Sengupta, Shuaiwen Leon Song, Kapil Agarwal, and Karsten Schwan. 2015. GraphReduce: processing large-scale graphs on accelerator-based systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 28.

[16] Zhaonian Zou, Faming Li, Jianzhong Li, and Yingshu Li. 2017. Scalable Processing of Massive Uncertain Graph Data: A Simultaneous Processing Approach. In *IEEE International Conference on Data Engineering*.