Brief Announement: Applications of Uniform Sampling: Densest Subgraph and Beyond*

Hossein Esfandiari
University of Maryland
Hossein@cs.umd.edu

MohammadTaghi Hajiaghayi†
University of Maryland
hajiagha@cs.umd.edu

David P. Woodruff
IBM Almaden
dpwoodru@us.ibm.com

ABSTRACT

In this paper we provide a framework to analyze the effect of uniform sampling on graph optimization problems. Interestingly, we apply this framework to a general class of graph optimization problems that we call *heavy subgraph problems*, and show that uniform sampling preserves a $1-\epsilon$ approximate solution to these problems. This class contains many interesting problems such as densest subgraph, directed densest subgraph, densest bipartite subgraph, d-max cut, and d-sum-max clustering. As an immediate impact of this result, one can use uniform sampling to solve these problems in streaming, turnstile or Map-Reduce settings. Indeed, our results by characterizing heavy subgraph problems address Open Problem 13 at the IITK Workshop on Algorithms for Data Streams in 2006 regarding the effects of subsampling, in the context of graph streams.

Recently Bhattacharya et al. in STOC 2015 provide the first one pass algorithm for the densest subgraph problem in the streaming model with additions and deletions to its edges, i.e., for dynamic graph streams. They present a $(0.5-\epsilon)$-approximation algorithm using $\tilde{O}(n)$ space, where factors of ϵ and $\log(n)$ are suppressed in the $\tilde{O}$ notation. In this paper we improve the $(0.5-\epsilon)$-approximation algorithm of Bhattacharya et al. by providing a $(1-\epsilon)$-approximation algorithm using $\tilde{O}(n)$ space.

1. INTRODUCTION

In this paper we consider a general class of graph optimization problems that we call *heavy subgraph problems* in the streaming setting with additions and deletions, i.e., in dynamic graph streams. We show that many interesting problems such as densest subgraph, directed densest subgraph, densest bipartite subgraph, d-max cut, and d-sum-max clustering fit in this general class of problems. To the best of our knowledge, we are the first to consider densest bipartite subgraph and d-sum-max clustering in the streaming setting.

Finding the densest subgraph of a graph is one of the fundamental problems in computer science. This problem has many applications across different areas that deal with massive datasets such as Community detection [5, 19, 9, 7], Link spam detection [11], distance query indexing [8, 13], analyzing communication in social networks [9], and, Computational biology [20] among others. We refer to [16] for a survey of applications of the densest subgraph problem.

In an instance of the *densest subgraph problem* we are given a graph G and want to find a subgraph $sol \subseteq G$, that maximizes $\frac{|E_{sol}|}{|V_{sol}|}$, where V_{sol} and E_{sol} are the vertex set and the edge set of sol, respectively. Similarly, in an instance of the *directed densest subgraph problem* we are given a directed graph $G=(V_G, E_G)$ and want to find a pair $A, B \subseteq V_G$, that maximizes $\frac{|E(A,B)|}{\sqrt{|A|\cdot|B|}}$, where $E(A,B)$ is the number of edges from A to B in E_G.

There are polynomial time algorithms for the densest subgraph problem in the classical setting [6, 10, 12, 15]. Specifically, Charikar [6] studies the densest subgraph problem in the classical setting and provides an exact algorithm with $\tilde{O}(nm)$ running time, where n and m are the number of vertices and edges, respectively. He also provides a 0.5-approximation algorithm with running time $O(n+m)$. To the best of our knowledge, this is the fastest known constant factor approximation algorithm for the densest subgraph problem. Moreover, he provides a 0.5-approximation algorithm for the directed densest subgraph problem as well.

Later, Bahmani, Kumar and Vassilvitskii [3], consider the densest subgraph problem and the directed densest subgraph problem in the streaming setting with only insertions of edges. For both problems, they present streaming algorithms with a $\frac{1}{2(1+\epsilon)}$-approximation factor using $\log_{1+\epsilon}(n)$ passes over the input. To the best of our knowledge, their results for directed graphs are the only non-trivial results for the directed densest subgraph problem in the streaming setting, prior to our work. Recently, Bahmani, Goel and Munagala [2] improve this result and provide a $(1-\epsilon)$-approximation algorithm using $O(\log(\frac{n}{\epsilon^2}))$ passes over the input.

Very recently, Bhattacharya et al. [4] present the first single pass streaming algorithm for dynamic graph streams.

*A full version of this paper is available at: http://arxiv.org/abs/1506.04505

†Supported in part by NSF CAREER award CCF-1053605, NSF BIGDATA grant IIS-1546108, NSF AF:Medium grant CCF-1161365, DARPA GRAPHS/AFOSR grant FA9550-12-1-0423, and another DARPA SIMPLEX grant.

SPAA '16 July 11-13, 2016, Pacific Grove, CA, USA

ACM ISBN 978-1-4503-4210-0/16/07.

DOI: http://dx.doi.org/10.1145/2935764.2935813

Their first algorithm provides a $(0.5-\epsilon)$-approximation using $\tilde{O}(n)$ bits of space. The update time of this algorithm is $\tilde{O}(1)$, though the query time is inefficient. They provide a second algorithm with a $(0.25-\epsilon)$-approximation factor for which the update time and query time are only $\tilde{O}(1)$, again using $\tilde{O}(n)$ bits of space.

In the *d-max cut* problem, we are given a graph G, and we want to decompose the vertices of G into d partitions such that the number of edges between different partitions is maximized. To the best of our knowledge, we are the first to consider this problem for general d in the streaming setting. A restricted version of this problem where $d = 2$, is the classic max cut problem. One can store a sparsifier [1] of the input graph in $\tilde{O}(n)$ space, and preserve a $(1-\epsilon)$-approximation of the max cut. However, it is not clear if sparsifiers preserve d-max cut or not. Recently, Kapralov, Khanna and Sudan [14] show that any $(1-\epsilon)$-approximation to the max cut problem in the streaming setting requires $n^{1-O(\epsilon)}$ space.

1.1 Our Results

In this paper, we first consider the densest subgraph problem in the streaming setting where we have both insertions and deletions to the edges as they arrive in the stream, i.e., in a dynamic graph stream. We improve the $(0.5-\epsilon)$-approximation algorithm of Bhattacharya et al. (STOC'15) [4] by providing a $(1-\epsilon)$-approximation algorithm for this problem using $\tilde{O}(n)$ space. Indeed, our algorithm simply samples $\tilde{O}(n)$ edges uniformly at random, and finds the densest subgraph on the sampled graph. We also achieve update time $\tilde{O}(1)$. To achieve this, we use min-wise independent hashing together with fast multi-point polynomial evaluation.

THEOREM 1.1. *There exists a semi-streaming algorithm in dynamic graph streams for the densest subgraph problem with space $\tilde{O}(n)$ which gives a $(1-\epsilon)$-approximate solution, with probability $1-1/n$. The update time is $\tilde{O}(1)$.*

We note that Bhattacharya et al. [4] provide a $(0.25-\epsilon)$ approximation streaming algorithm with update time and query time $\tilde{O}(1)$ as well. Our update time matches their $\tilde{O}(1)$ update time, which is important since it multiplies the time to process each stream update. We note, though, that they provide a better query time, which is useful if one repeatedly queries the data structure, as may often be the case in the study of dynamic graph algorithms. However, especially in the data stream setting, which is the setting considered in this paper, getting a $(1-\epsilon)$-approximation instead of a $(0.25-\epsilon)$-approximation is often more important. This is true if one is only interested in querying the data structure at the end of the stream, or at a few intermediate positions, for which one can amortize the cost of the query, which is at most $\tilde{O}(n)$, over the next n stream updates. We also note that the query time of [4] et al. can be as large as $\tilde{\Omega}(n)$, that is, it is proportional to the current number of nodes of the densest subgraph, so to take advantage of it one should apply it to graphs with small densest subgraphs.

Next, we extend our results to a general family of graph optimization problems that we call *heavy subgraph problems*. Interestingly, we show that by uniformly sampling edges we obtain enough information about the solution of any heavy subgraph problem. Since the solution of a heavy subgraph problem itself may be as large as the whole graph, here we just claim that we can estimate the size of the optimum solution. However, in some cases, like for the densest subgraph problem, it might be possible to also obtain the optimum solution itself, and not just the size, from the sampled graph.

A graph optimization problem is defined by, an input graph G, a set of feasible solutions Sol_G, which are subgraphs of G, and an objective function $f : Sol \to \mathbb{R}$. In a graph optimization problem we aim to find a solution $sol \in Sol_G$ that maximizes f. In fact, the number of feasible solutions for a graph G may be exponential in the size of G. We say a graph optimization problem on graph G is a (γ, l)-*heavy subgraph problem* if there exist l [not necessarily disjoint] sets $Sol_G^1, Sol_G^2, \ldots, Sol_G^l$, such that $Sol_G = \cup_{k=1}^{l} Sol_G^k$ and for any k:

- **Local Linearity:** There exists a number $f_k \geq 1$ such that for any solution $sol \in Sol_G^k$, we have $f(sol) = f_k \cdot |E_{sol}|$, where E_{sol} is the edge set of sol. Without loss of generality we assume $f_1 \geq f_2 \geq \cdots \geq f_l = 1$.
- **Hereditary Property:** For any spanning subgraph $H \subseteq G$, we have $sol_H \in Sol_H^k$ if and only if there exists a solution $sol_G \in Sol_G^k$ such that $sol_H = sol_G \cap H$.
- γ **Bound:** γ is chosen such that the optimum solution is lower bounded by $\gamma \log(|Sol_G^k|) f_k \frac{m}{n}$.

Let $\mathcal{P}(\gamma, l)$ be a heavy subgraph problem, and let Alg be an α-approximation algorithm for $\mathcal{P}$. Algorithm 1 samples $O(\frac{n\delta \log(l)}{\gamma\epsilon^2})$ edges of the input graph and runs Alg on the sampled graph. Interestingly, the following Theorem shows Algorithm 1 is an $(\alpha-\epsilon)$-approximation algorithm for $\mathcal{P}$ on G.

THEOREM 1.2. *Let $\mathcal{P}(\gamma, l)$ be a heavy subgraph problem. Let G be an arbitrary graph, and let Alg be an α-approximation algorithm for $\mathcal{P}$. With probability $1-e^{-\delta}$, Algorithm 1 is an $(\alpha-\epsilon)$-approximation algorithm for $\mathcal{P}$ on G, using $O(\frac{n\delta \log(l)}{\gamma\epsilon^2})$ space.*

Finally, we show several applications of Theorem 1.2. Indeed, we show that directed densest subgraph, densest bipartite subgraph, d-max cut and d-sum-max clustering all fits in the general family of heavy subgraph problems, and thus, Theorem 1.2 holds for them.

THEOREM 1.3. *The following statements hold.*

- *Densest bipartite subgraph is a $(\gamma = \frac{1}{2(\log(n)+1)}, l = n)$ heavy subgraph problem.*
- *Directed densest subgraph is a $(\gamma = \frac{1}{2\sqrt{n}\log(n)}, l = n^2)$ heavy subgraph problem.*
- *d-max cut is a $(\gamma = \frac{1}{2\log(d)}, l = 1)$ heavy subgraph problem.*
- *d-sum-max clustering is a $(\gamma = \frac{n-2d}{n\log(d)}, l = 1)$ heavy subgraph problem.*

In fact, understanding the structure of the problems that can be solved using sampling, and specifically uniform sampling, is a well-motivated challenge, which was highlighted as a direction in the IITK Workshop on Algorithms for Data

Algorithm 1 A General Algorithm

Input: A graph G, a heavy subgraph problem $\mathcal{P}(\gamma, l)$ and an α approximation algorithm Alg for $\mathcal{P}$.
Output: An $\alpha - \epsilon$ estimator of $\mathcal{P}$ on graph G, w.pr. $1 - e^{-\delta}$.

Set $C = \frac{12n(4+\delta)\log(l)}{\gamma\epsilon^2}$
if $|E| \leq C$ **then**
  Return $Alg(G)$.
else
  Sample C edges uniformly at random, without replacement from G.
  Let H be the sampled graph.
  Return $\frac{|E|}{C} Alg(H)$.

Streams in 2006. Our structural results, as well as our characterization for heavy subgraph problems, give partial answers to this open question in the context of graphs.

In simultaneous and independent work McGregor et al. [17] present a single pass $(1-\epsilon)$-approximation algorithm for the densest sugraph problem in the dynamic graph streaming model with update time $\tilde{O}(1)$, that uses $\tilde{O}(n)$ space. Also, simultaneously and independently of our work, Mitzenmacher et al. [18] show if one samples each edge with a small probability, then with high probability the sampled graph preserves the densest subgraph of the input graph. Mitzenmacher et al. do not provide a way to implement the sampling efficiently in dynamic graph streams.

2. REFERENCES

[1] K. J. Ahn, S. Guha, and A. McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of the 31st symposium on Principles of Database Systems*, pages 5–14. ACM, 2012.

[2] B. Bahmani, A. Goel, and K. Munagala. Efficient primal-dual graph algorithms for mapreduce. In *Algorithms and Models for the Web Graph - 11th International Workshop, WAW 2014, Beijing, China, December 17-18, 2014, Proceedings*, pages 59–78. Springer, 2014.

[3] B. Bahmani, R. Kumar, and S. Vassilvitskii. Densest subgraph in streaming and mapreduce. *Proceedings of the VLDB Endowment*, 5(5):454–465, 2012.

[4] S. Bhattacharya, M. Henzinger, D. Nanongkai, and C. E. Tsourakakis. Space-and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *STOC*, 2015.

[5] G. Buehrer and K. Chellapilla. A scalable pattern mining approach to web graph compression with communities. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 95–106. ACM, 2008.

[6] M. Charikar. Greedy approximation algorithms for finding dense components in a graph. In *Approximation Algorithms for Combinatorial Optimization*, pages 84–95. Springer, 2000.

[7] J. Chen and Y. Saad. Dense subgraph extraction with application to community detection. *Knowledge and Data Engineering, IEEE Transactions on*, 24(7):1216–1230, 2012.

[8] E. Cohen, E. Halperin, H. Kaplan, and U. Zwick. Reachability and distance queries via 2-hop labels. *SIAM Journal on Computing*, 32(5):1338–1355, 2003.

[9] Y. Dourisboure, F. Geraci, and M. Pellegrini. Extraction and classification of dense communities in the web. In *Proceedings of the 16th international conference on World Wide Web*, pages 461–470. ACM, 2007.

[10] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing*, 18(1):30–55, 1989.

[11] D. Gibson, R. Kumar, and A. Tomkins. Discovering large dense subgraphs in massive graphs. In *Proceedings of the 31st international conference on Very large data bases*, pages 721–732. VLDB Endowment, 2005.

[12] A. V. Goldberg. *Finding a maximum density subgraph.* University of California Berkeley, CA, 1984.

[13] R. Jin, Y. Xiang, N. Ruan, and D. Fuhry. 3-hop: a high-compression indexing scheme for reachability query. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 813–826. ACM, 2009.

[14] M. Kapralov, S. Khanna, and M. Sudan. Streaming lower bounds for approximating max-cut. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1263–1282. SIAM, 2015.

[15] S. Khuller and B. Saha. On finding dense subgraphs. In *Automata, Languages and Programming*, pages 597–608. Springer, 2009.

[16] V. E. Lee, N. Ruan, R. Jin, and C. Aggarwal. A survey of algorithms for dense subgraph discovery. In *Managing and Mining Graph Data*, pages 303–336. Springer, 2010.

[17] A. McGregor, D. Tench, S. Vorotnikova, and H. T. Vu. Densest subgraph in dynamic graph streams. In *Mathematical Foundations of Computer Science 2015*, pages 472–482. Springer, 2015.

[18] M. Mitzenmacher, J. Pachocki, R. Peng, C. Tsourakakis, and S. C. Xu. Scalable large near-clique detection in large-scale networks via sampling. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 815–824. ACM, 2015.

[19] M. E. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[20] B. Saha, A. Hoch, S. Khuller, L. Raschid, and X.-N. Zhang. Dense subgraphs with restrictions and applications to gene annotation graphs. In *Research in Computational Molecular Biology*, pages 456–472. Springer, 2010.