Enhanced Streaming Algorithms for the Maximum Directed Cut Problem Using Smoothed Snapshots

1st Divya Sri Bevara
Dept of Computer Science
University of North Texas
Denton, USA
divyasribevara@my.unt.edu

2nd Parthiv Kumar Gunjari
Dept of Computer Science
University of North Texas
Denton, USA
parthivkumargunjari@my.unt.edu

***Abstract*—In this paper we are addressing the problem of Maximum Directed Cut (Max-DICUT) which is the finding the best way to divide the vertices of directed graph so that we can maximize the number of edges from one part to another. The main motive of the project is to introduce the better version of single pass streaming algorithm that uses $O(\sqrt{n})$ space. Here, n is number of vertices to obtain 0.483 approximation. In previous proposal 0.45 is the approximation. In our proposed method we used first order of snapshots in order to create a simplified graph structure. By using these snapshots, we can include non-streaming Max-DICUT algorithms for using streaming environment. We result a smoothed estimations for these snapshots we results in maintaining sub-linear complexity through which we can ensure the accuracy through estimation methods and this approach is not used for streaming algorithms, we can also use in the fields such as machine learning, network optimization and data mining, where it has an ability to process large graphs efficiently. Furthermore, through this approach we introduced a new algorithm to solve constraint satisfaction problems (CSPs) in streaming environments with balancing algorithm complexity and with limited computing resources.**

***Index Terms*—Streaming algorithm, Max-DICUT, Graph theory, Refined Snapshots**

I. Introduction

In computer science and in graph theory, the maximum directed cut (Max-DICUT) problem plays major role and has a significant impact on machine learning, network optimization and data mining. In today's, world is getting digitalized in every aspect henceforth large amounts of data will be generated, and most of the issues will be related in processing of large graphs with limited amount of memory.

In this project we divide the vertices of the directed graph into two subsets in a way where the number of edges heading from the first subset to the second subset is maximized. The maximization of the subset can be done for the directed graph G with m edges and n vertices only. The primary key feature of this problem is interlined with the constraint satisfaction problem (CSPs). CSPs is important developing and understanding new algorithms and address computational challenges. As we know in directed graph, we divide the vertices into two non-empty sets where the value of the cut is the total weight of edges which as one end point in each of the sets.

As matter of fact, for the algorithms using Max-DICUT using the streaming model is the process where each edge comes one by one and developed with the best space complexity and proficient approximation. Using previous algorithm 0.45 is the approximation ratio by using $O(\log n)$ space. In our proposed method we used large datasets because of its lower approximation and scalability issues.

In this project we introduced a new streaming algorithm that uses $O(\sqrt{n})$ space, and we obtained 0.483 approximation. The improvement from previous algorithm is because of the first order of snapshot which improves the algorithm's approximation in the streaming model.

Let us understand about first order snapshots. The first order snapshots are designed to capture the required things which are used in minimizing space requirements and it helps in simulating in non-streaming Max-DICUT algorithms, which is used in reduce the gap between non-streaming and streaming models. This method is very efficient in processing large amount of data as these snapshots are created by sampling edges and vertices then smoothen the result to reduce all the inaccuracies.

A. Background

Approximating graphs especially directed graphs for the Max-DICUT problem is very challenging and it is crucial because it used in various field like network design, machine learning and data analysis. In this generation we are using large amount of data, and we are representing large-scale graphs with limited memory. In traditional way we used a greater number of computational resources and made unrealistic application possible. With streaming model, we are processing large amount of data in bits sequentially with limited memory and found alternative methods to maintain the accuracy and efficiency which is furthermore used to address complex algorithms that is used to balance these limitations and provide scalable solutions.

B. Recent Work

In previous studies, Max-DICUT focused on achieving sublinear space complexity with practical approximations. In

this study the main focus is on upgrades, expansion, bounded-degree graphs and the other graphs too. In this study they faced many challenges like dealing with larger graphs and make the method more complex.

In the current study we deal with the first order snapshots that are represented with graphical representations. Through this approach we have a better understanding of the graph topologies and give us and the best version of the algorithm as it has $O(\sqrt{n})$ constraint, which improves performance in approximation and efficiently handles large datasets, which is important for the applications like road maps and vast social networks.

II. Theorem

For every $d \in N$, there exists a streaming algorithm that provides a 0.483-approximation for the Max-DICUT value of a graph with a maximum degree of d using $\tilde{O}_d(\sqrt{n})$ space.

In order to understand this theorem, for estimating the approximation that is 0.483 times we have natural numbers d for streaming algorithms. The approximation is valid irrespective with the graphs. This means it does not depend on the nodes and it will be easy for us to obtain the approximation even when we have the graph with vertices that have highest degree equal to the d which means each vertex is connected to at most d and the other vertices.

The memory required to compute this algorithm is the square root of number of vertices n in the graph and its space complexity is $\tilde{O}_d(\sqrt{n})$, and the graph depends on the factors that are depending on d. We obtain the approximate solution for the Max-DICUT for the graph that has bounded degree cases even though we have limited memory. Through this theorem we conclude that we can deal easily with the large graphs even though we do not have too many connections between the vertices.

III. Lemma

A. Lemma A

An informal version of Lemmas III.11 and III.17 states: If we can achieve an approximation ratio α for Max-DICUT by examining a snapshot of a graph G, and $\widehat{A}$ is a (ω, δ) pointwise estimate of the refined snapshot $A = RSnap_{G,d,t}$, then $\widehat{A}$ can be used to achieve an $(\alpha - \epsilon)$ approximation for Max-DICUT. Here, $\epsilon = O(\delta(kl)^2 + \omega\lambda + 1/n)$, where k, l, and λ represent the number of degree classes in d, the number of bias classes in t, and the maximum width of any interval in t, respectively.

This lemma implies that a refined snapshot $\widehat{A}$, which is an estimate of A, can still provide a good approximation $(\alpha - \epsilon)$ for Max-DICUT. The error margin ϵ is influenced by various properties of the graph and the accuracy of our estimate, such as the number of degree classes, bias classes, and the width of intervals in the graph. Thus, even with an imperfect graph, we can find a near-optimal solution using these snapshots.

Through this lemma we achieved a better approximation (α) for the Max-DICUT. For this problem we used a detailed snapshot for the graph that is $(\alpha - \epsilon)$ for that we roughly estimated a snapshot $(\widehat{A})$. In our observation we come to know about the error margin (ϵ) is depending on the properties of the graph and accuracy of our estimation. For example the number of degree classes, bias classes and the width in between the intervals in the graph. Finally, we concluded that, if the graph is not perfect but still we can find a solution with the snapshots in the graph and find out the best possible solution.

B. Lemma B

Let $X_1, X_2, \ldots, X_n$ be independent random variables that can only take the values 0 or 1, and let X be the sum of these variables, defined as $X = \sum_{i=1}^{n} X_i$. For any δ in the range $0 \leq \delta \leq 1$, the probability that the absolute difference between X and its expected value $E[X]$ is at least $\delta E[X]$ is at most $2\exp\left(-\frac{\delta^2 E[X]}{3}\right)$.

In this lemma we assume that the sum of n number of small and independent random variables that is (0 or 1) is improbable to be in a distance which is expected to be, and the probability of significant deviation is decreasing exponentially as the deviation value rises, which means that the large derivative values are comparatively very less and this results in the measuring the probability and statistics, and find the consistency of the summation of random variables.

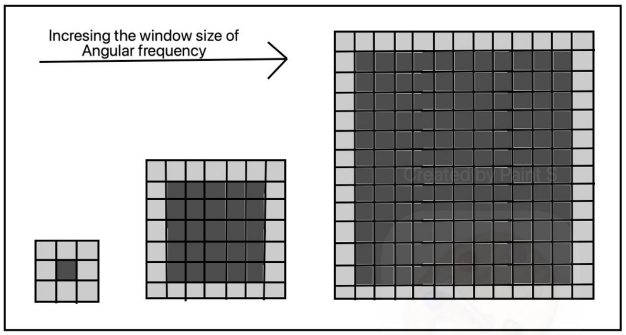

Fig. 1. Increasing the window size of Angular frequency

As we can see from the image, size of the window w is increasing. The bigger $w \times w$ square (dark gray) is reducing the effect of border (light gray). We can observe the area of the square rises sooner than its perimeter. In terms of guessing the Max-DICUT value in a graph, using larger windows can cause errors due to continuousness traditions. The best select of w strikes carefully to balance these effects.

IV. Definition

A. Definition A

A set of hash functions $H = \{h : [n] \to [m]\}$ is considered k-wise independent if it satisfies the following conditions: For every $x \in [n]$ and $a \in [m]$, and $h \sim H(n, m)$ uniformly, the probability that $h(x) = a$ is $\frac{1}{m}$. Additionally, for any distinct $x_1, \ldots, x_k$ in $[n]$, and $h \sim H(n, m)$ uniformly, $h(x_1), \ldots, h(x_k)$ are independent random variables.

A k-wise independent set of hash functions can be addresses in two cases. In the first case, for every input it has an equal chance of plotting to any output. So, this guaranteeing uniform distribution.In the second case, k – wise has independent set of hash function inputs, which means the hash value of one input does not provide any information about the hash values of the other inputs. This plays a major role as it is important because it ensures that input hash function is distributed uniformly and without predictable patterns. This make sure that the application is more preferable to choose as it uses hashing and randomized algorithms.

B. Definition B

Given an array A of size $k \times l$, the matrix Proj(A) of size $m \times l$ is formed by summing over the first and second coordinates of A. Specifically, each element $(\text{Proj}(A))\,(i, j)$ in the new matrix is the sum of all elements $A(a, b, i, j)$ where a and b range from 1 to k. Here, we create a smaller matrix by taking reference from the larger multi- dimensional array. These multi-dimensional arrays are formed by adding up the elements. We focus on third and fourth coordinates of the array and sum all the values that are shared with these coordinates. So, ignore the first and second coordinates. Through this process we reduce the dimensions of the data and maintain specific information related to the third and fourth coordinates.

V. Algorithm

VI. Table of Notations

TABLE I
TABLE OF NOTATIONS

Notation	Value	Description
G	(U, D)	U , D are vertices and edges in the Graph
p_e	[0, 1]	It the Probability of sampling each edge based on weight
k	Integer	No. of refinement steps
S	Set	It is the set of sampled edges
V	Set	V is the Set of vertices in sampled edges
Snapshot	Matrix	Snapshot of the given graph
Max-DICUT	Float	The Estimated value of Max Directed Cut

Algorithm 1 Estimating the MAX-DICUT with Weighted Edge Sampling and Progressive Refinement

Require: Graph $G = (U, D)$, p_e, k are sampling probability and refinement steps
Ensure: Make sure Max-DICUT value
1: At first initialize the empty sets $S \leftarrow \emptyset$ and $V \leftarrow \emptyset$
2: **for** each edge in $(t, u) \in D$ **do**
3: Now Compute the weight of the edge $w \leftarrow$ compute_edge_weight(t, u)
4: **if** random.random() ¡ $p_e * w$ **then**
5: $S \leftarrow S \cup \{(t, u)\}$
6: $V \leftarrow V \cup \{t, u\}$
7: **end if**
8: **end for**
9: we should construct the initial snapshot using edges and vertices (S ,V)
10: **for** each and every refinement step i from 1 to k **do**
11: Now Update snapshot by adding additional edges based on the requirement (connectivity and weight)
12: For the updated snapshot recompute the biases and degrees
13: To reduce noise and improve accuracy we should apply localized smoothing
14: **end for**
15: Finally estimating the Max-DICUT value from the final refined snapshot
16: **return** It should be the estimated Max-DICUT value

VII. Future Work

Finally, adapting the algorithm for other graph-related problems like influence maximization and community detection, as well as leveraging new computing architectures such as quantum computing, could lead to further efficiency and performance improvements. Further, we deal with the distributed and parallel versions of the algorithm to address scalability for large graphs which is robust in nature and deal with the corrupted data which improves practical effectiveness of the algorithm. Furthermore, in future we provide a better insights in the algorithm's regarding performance and its limitations and can be useful in validating its performance in real world datasets.

VIII. Conclusion

From this paper we can conclude the demonstration of enhanced streaming algorithm for approximating the Max-DICUT problem using smoothed snapshots and techniques. Finally, we have projected a new algorithm for solving the problems based on vertex sampling combined with edge sampling which allows us handle large graphs with flexible precision and approximation. In this paper we have presented the approximation ratio of 0.483 and it uses sub-linear space which is more effectual than conventional methods. In real time applications like big data, and when we have limited memory, can be significantly gain from this development

method. So, for the future development we are expected that this research will lay a strong basis into stream algorithms and their applications in network optimization, data mining etc., and challenging effective graph processing.

IX. References

References

[1] D. Kogan and R. Krauthgamer, "Sketching cuts in graphs and hypergraphs," in *Proceedings of the 6th Annual Conference on Innovations in Theoretical Computer Science (ITCS 2015, Rehovot, Israel, January 11-13, 2015)*. Association for Computing Machinery, 2015, pp. 367–376.

[2] M. Kapralov, S. Khanna, and M. Sudan, "Approximating matching size from random streams," in *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2014, Portland, OR, USA, January 5-7, 2014)*. USA: Society for Industrial and Applied Mathematics, Jan. 2014, pp. 734–751.

[3] V. Guruswami, A. Velingker, and S. Velusamy, "Streaming Complexity of Approximating Max 2CSP and Max Acyclic Subgraph," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX 2017, Berkeley, CA, USA, August 16-18, 2017)*, ser. LIPIcs, K. Jansen, J. D. P. Rolim, D. Williamson, and S. S. Vempala, Eds., vol. 81. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, Aug. 2017, pp. 8:1–8:19.

[4] M. Kapralov, S. Khanna, M. Sudan, and A. Velingker, "(1 + (1)) approximation to MAX-CUT requires linear space," in *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017, Barcelona, Spain, January 16-19, 2017)*. Society for Industrial and Applied Mathematics, Jan. 2017, pp. 1703–1722.

[5] V. Guruswami and R. Tao, "Streaming Hardness of Unique Games," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX 2019, Cambridge, MA, USA, September 20-22, 2019)*, ser. LIPIcs, D. Achlioptas and L. A. Végh, Eds., vol. 145. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, Sep. 2019, pp. 5:1–5:12.