An Exploration of β-Balanced Graph Problems in the Streaming Model

Oswaldo Ramirez Lauren Sands

Abstract

We investigate approximation of cuts in directed graphs in the streaming model. Unlike undirected graphs where $o(n^2)$ space sparsifiers can achieve a $(1 \pm \epsilon)$-approximation, the directed case provably requires $\Omega(n^2)$ space in general. Focusing on β-balanced directed graphs, we show that even computing the balance parameter β in one pass requires $\Omega(n^2)$ memory. We then survey existing streaming and sketching methods for approximating only the β-balanced cuts, and outline our thoughts on developing a one-pass streaming algorithm to preserve only the β-balanced cuts in a non-β-balanced graph.

1 Introduction

The minimum cut problem is a fundamental problem in graph theory with numerous real-world applications, particularly in the context of network reliability. In modern systems, networks are often modeled as graphs where nodes represent routers or systems, and edges correspond to the connections between them. The minimum cut of a graph represents the smallest set of edges that, if removed, would disconnect the graph. This problem has been studied in different variations, weighted and unweighted, on particular graphs such as networks. Therefore, understanding this problem provides critical insights into identifying vulnerabilities or weak points within a network.

While we started our focus on minimum cuts, a more general question is how to determine the value of any cut in a graph. This again has important applications to network reliability. In this case, we could determine how much redundancy or capacity exists between two groups of servers.

Furthermore, we study this problem in the streaming model in which we receive the edges one at a time. The task is still to compute the value of any cut in the graph. However, with massive graphs, we cannot store all of the edges. Instead, we must store enough information to construct an approximation of any cut. This is exactly the idea of graph sparsifiers introduced by Benczur and Karger [BK96]: the goal is to create a subgraph on the same vertex set using $o(n^2)$ memory that approximates all cut values up to a factor of ϵ . This problem has been studied extensively in the undirected case, but there are related open problems in the directed case that we discuss in this paper.

Because sparsifiers for directed graphs require $\Omega(n^2)$ memory, we will focus on the special case of β-balanced graphs. Algorithms to find cut sparsifiers for such graphs assume β is given, which naturally leads to the question of whether β could be computed in low space. We answer this in the negative, showing a lower bound of $\Omega(n^2)$ to compute β in a directed graph. This bound is tight, as clearly β can be computed if we store the entire graph in $O(n^2)$ space.

We also consider the question of approximating only the β-balanced cuts in a graph in the streaming model. More specifically, could we create a sparsifier that successfully approximates β-balanced cuts up to a factor of ϵ, but can fail for non-β-balanced cuts? Unfortunately, after many topic shifts, we were not left with enough time to answer this question concretely. However, we do provide a survey of what related results are known, and our thoughts about how we could apply those to develop an algorithm to solve this problem.

2 Preliminaries

Let $G = (V, E, w)$ be a weighted (directed) graph with n vertices and m edges, where each edge $e \in E$ has weight $w_e \geq 0$. We write $G = (V, E)$ if G is unweighted and leave out w. For two sets of nodes $S, T \subseteq V$, let $E(S, T) = \{(u, v) \in E : u \in S, v \in T\}$ denote the set of edges from S to T. Let $w(S, T) = \sum_{e \in E(S,T)} w_e$ denote the total weight of edges from S to T. For a node $u \in V$ and a set of nodes $S \subseteq V$, we write $w(u, S)$ for $w(\{u\}, S)$.

3 Related Work

3.1 Streaming Minimum Cut

The minimum cut problem has been studied extensively in multiple models (e.g., distributed, streaming, etc.) and variations (e.g., k-minimum cut, weighted). Within the streaming model, the problem has been explored in both single-pass and multi-pass settings. In this paper, we will focus on the single-pass case of the streaming model, where we can only observe the input stream once.

[DGL+24] showed a one-pass streaming algorithm that gives a $(1 \pm \epsilon)$-approximation to the minimum cut using $\tilde{O}(n/\epsilon)$ bits of space, and a corresponding lower bound. They also showed an algorithm to find the *exact* minimum-cut for random-order streams using $\tilde{O}(n)$ space. Finally, they showed a lower bound of $\Omega(n/\epsilon^2)$ for deterministic minimum cut algorithms, which matches the best-known upper bound up to polylog factors.

An open question is how to approach this problem in the fully-dynamic case. Although tight lower and upper bounds have been established for insertion-only streams, there are not currently tight bounds in the dynamic case (i.e., edge inserts and edge deletions).

[AGM12] showed a one-pass semi-streaming algorithm that constructs a graph sparsifier in the presence of both edge insertions and deletions using $\tilde{O}(n/\epsilon^2)$ bits. This can be used to get a $(1 \pm \epsilon)$ approximation to the minimum-cut in the dynamic case, leading to the best known upper bound of $\tilde{O}(n/\epsilon^2)$ bits. In [DGL+24], they show a $\Omega(n/\epsilon)$ lower bound for insertion-only streams, which must also be a lower bound for the dynamic case. The exact space complexity must be between these two values.

While we considered this question for a while, we eventually pivoted as it proved to be quite difficult.

3.2 Cut Sparsifiers for Directed Graphs

In the directed case, the value of a cut is the value of the edges from the S to the T side. In the worst case, $\Omega(n^2)$ space is required to construct a graph sparsifier for a directed graph [CLL+24].

Stream Type	Exact/Approx $(1+\epsilon)$	Upper Bound	Lower Bound
Adversarial	Exact	$O(n^2)$ (full graph)	$\Omega(n^2)$ [Zel11]
Adversarial	Approx, Deterministic	$\widetilde{O}(n/\epsilon^2)$ [BSS12]	$\Omega(n/\epsilon^2)^*$ [DGL+24]
Adversarial	Approx, Randomized	$\widetilde{O}(n/\epsilon)$ [DGL+24]	$\Omega(n/\epsilon)$ [DGL+24]
Random-Order	Exact	$\widetilde{O}(n)$ [DGL+24]	$\Omega(n)$ [CCM16]

Table 1: Minimum Cut Space Complexity in the Single-Pass Insertion-Only Streaming Setting

For example, in a bipartite graph, we can show that any edge must be included in the sparsifier by constructing a cut where it is the only edge from the S to the T side.

Therefore, we must consider special classes of graphs to develop sparsifiers in the directed case. Cen et. al. considered β-balanced graphs, defined as follows [CCPS21].

Definition 1 (β-Balanced Graphs)**.** *A graph $G = (V, E)$ is β-balanced if for every directed cut $(S, V \setminus S)$, the total weight of edges from S to $V \setminus S$ is at most β times that from $V \setminus S$ to S. That is, the ratio between incoming and outgoing edges in any cut is at most β.*

To $(1 \pm \epsilon)$-approximate directed cuts in β-balanced graphs, [CLL+24] showed a lower bound of $\widetilde{\Omega}(n\sqrt{\beta}/\epsilon)$ in the for-each model, and $\Omega(n\beta/\epsilon^2)$ in the for-all model. These match the upper bounds given in [CCPS21], so they are tight.

However the algorithms shown to achieve those upper bounds assume that β is given. This leads to a natural question of whether β can actually be calculated in low space. It seems that there are many applications where β would not be known ahead of time and we would like to be able to compute it on the fly. More formally, we pose the following question:

Question 1. *What is the exact space requirement to compute β for a directed graph in the streaming model?*

Clearly, β can be calculated with $O(n^2)$ space, since if we store all edges in the graph, we can determine β. Therefore, in Section 4, we simply try to show a matching lower bound. This unfortunately shows that we cannot calculate β on the fly with low space.

3.3 Spectral/Cut Sparsification in the Streaming Model

There has been significant progress on developing graph sparsifiers in the insertion-only streaming model for undirected graphs, particularly the semi-streaming model where $\widetilde{O}(n)$ space is allowed. Kelner et. al. devised a streaming algorithm that resparsifies every time the running sparsifier gets too large [KL11]. However, according to Cohen et. al., the proof of the algorithm is believed to be unfinished [CMP16]. They say that the row sampling depends on previous row samplings so it's incorrect to say that the procedure works as a single round of leverage score sampling. Therefore, their contribution is to use online ridged leverage scores, which is stream-friendly.

However, all these results focus on undirected graphs. There are still many open questions that tackle cut sparsifiers for directed graphs in the streaming model. While algorithms exist to create

sparsifiers for β-balanced graphs, as stated previously, there is an open question of whether a sparsifier could be created that preserves only the β-balanced cuts in a graph. We explore this question in Section 5.

4 Lower Bound on Calculating β in the Streaming Model

Theorem 1. *Any insertion-only streaming algorithm needs $\Omega(n^2)$ bits of memory to determine whether $\beta = 1$ in a directed simple graph. That is, whether the graph is Eulerian.*

Proof. We make use of the following result from communication complexity.

Lemma 1 (Index Lower bound [KN96])**.** *Suppose that Alice has a random string $u \in \{0,1\}^n$ and Bob has a random index $i \in [n]$. If Alice sends a single message to Bob from which Bob can recover u_i with probability at least $2/3$, where the probability is over both the randomness of the protocol and the input, then Alice must send $\Omega(n)$ bits to Bob.*

We will reduce the Index problem to the problem of determining whether $\beta = 1$. Assume for the sake of contradiction that we have an algorithm ALG which determines whether $\beta = 1$ on an n vertex graph using $o(n^2)$ space.

Suppose that Alice is given a random string $s \in \{0,1\}^{\binom{n}{2}}$. Each bit in this string can correspond to a particular pair of vertices $\{u, v\}$. Intuitively, each bit corresponds to a particular *undirected* edge in the graph on n vertices. The exact mapping from indices to edges is unimportant, so long as both Alice and Bob can efficiently compute the edges from the index, and vice versa.

Let Alice's stream $s(a)$ be the stream of edges containing both (u, v) and (v, u) iff the bit at index i corresponding to $\{u, v\}$ is 1. That is, Alice's stream represents a *directed* graph where both the forward and backward edges exist for any bits set to 1, and neither edge exists for any bits set to 0.

Say Bob wants to query index i in the string, which corresponds to edge $\{u, v\}$. Let Bob's stream $s(b)$ consist of just the edge (u, v). Note that this is only one of the edges between the two vertices, not both.

Now, Alice can run ALG$(s(a))$, and send the state of the algorithm to Bob. Note that this state is $o(n^2)$ bits. Bob can then determine ALG$(s(a), s(b))$ by sending his stream into ALG starting from the state sent to him. If the result is that the graph *is* β-balanced, then index i was a 0. If the graph *is not* β-balanced, then index i was a 1.

We know this answer is correct because Alice's stream must have represented a β-balanced graph since every edge that was added in the stream had the corresponding backwards edge also added. Since we only consider simple graphs, if Bob queried an index set to 1, a duplicate edge would have been added to the stream which would effectively be ignored because it does not change the final answer. Thus, the graph would still be β-balanced. However, if Bob queried an edge set to 0, the graph would no longer be β-balanced because only one of the edges between u and v was in the stream. We can consider the cut consisting of u on one side, and all other vertices on the other. This cut cannot be perfectly balanced because every edge has a corresponding back edge except the edge from u to v.

□

Corollary 1. *Any streaming algorithm needs $\Omega(n^2)$ bits of memory to compute β.*

Proof. This follows directly from Theorem 1. If we could calculate β in $o(n^2)$ space, then we would also be able to determine whether $\beta = 1$ in $o(n^2)$ space which would contradict Theorem 1. □

5 Approximation of β-Balanced Cuts in the Streaming Model

The restriction for the directed cut sparsifier in [CLL+24] is strong in that every cut must be β-balanced. Therefore, it begs the question what if a graph had most of its cuts β-balanced, with a few unbalanced cuts? More formally, we have the research question:

Question 2. *Can a sparsifier be created in the streaming model that preserves only β-balanced cuts in a graph, and can fail for non-β-balanced cuts?*

This question came from the following result, which we wanted to explore in the streaming model rather than the offline case:

Theorem 2 (Theorem 1.3 from [CCPS21])**.** *There is an $\widetilde{O}(m)$-time offline algorithm that, for any directed graph with n vertices, m edges, non-negative edge weights, and any $\beta \geq 1$, returns a weighted subgraph with $\widetilde{O}(\beta n/\epsilon^2)$ edges and preserves the values of all β-balanced cuts up to a factor of $1 \pm \epsilon$ for $\epsilon > 0$.*

While we were not able to develop such an algorithm, we will provide a summary below of the results we believe would be helpful to answer this question.

The result above, interestingly enough, is a corollary of a stronger result:

Theorem 3 ([CCPS21])**.** *Let $G = (V, E)$ be a directed graph where each edge e has weight $w_e \geq 0$, and let ϵ, β be parameters. There is an $\widetilde{O}(m)$-time algorithm that returns a weighted subgraph H that satisfies the following with high probability: for every α-cut U,*

$$\left(1 - \epsilon\sqrt{\frac{\alpha+1}{\beta+1}}\right) \cdot \delta_G(U) \leq \delta_H(U) \leq \left(1 + \epsilon\sqrt{\frac{\alpha+1}{\beta+1}}\right),$$

where $\delta_G(U)$ and $\delta_H(U)$ denote the cut value of U in G and H, respectively. The number of edges in H is $O(\beta n \log n/\epsilon^2)$ in expectation.

To achieve the algorithm we merely guarantee the directed graph to be β-balanced (i.e., $\alpha = \beta$ for every cut U). Therefore, we prserve the values of all β-balanced cuts up to a factor of $1 \pm \epsilon$ because

$$(1 - \epsilon) \cdot \delta_G(U) \leq \delta_H(U) \leq (1 + \epsilon) \cdot \delta_G(U) \quad \text{for } \alpha = \beta$$

To state the algorithm for the stronger result, we must first define edge-strength as it serves an important role in the edge sampling procedure.

Definition 2 (Strength)**.** *The strength of an edge e, denoted by k_e, is the largest k such that there exists a k-edge connected vertex-induced subgraph of G containing e.*

With access to the whole graph $G = (V, E, w)$, the algorithm first computes the approximation edge-strengths of each edge e, which is shown to be enough for their analysis and guarantees, using a lemma from [BK15]. Afterward, they set $\rho = \rho(d, n, \beta, \epsilon)$, and for each edge $e \in E$, they sample e

with probability $p_e = \rho \cdot w_e/\widetilde{k}_e$, where w_e, k_e is the weight and edge-strength, respectively. If the edge e is sampled, then they add e to their running sparsifier H with weight $w_e = \widetilde{k}_e/\rho$.

We would like to extend this approach to the streaming model, particularly the semi-streaming model, as it's more relaxed and preferable for graph algorithms. This has been tackled in multiple papers for undirected graphs, but not the directed case we are interested in. However, a particular paper stands out that we think would be useful to extend the algorithm by Cen et. al. to the semi-streaming model. Particularly, Goel et. al. approximate the edge-strengths in the semi-streaming model and use the edge-strengths to sample [GKK10]. The paper provides three theorems with accompanying algorithms. Let $G = (V, E, w)$ be a graph. Then,

Theorem 4 ([GKK10])**.** *Running Algorithm* 1 *from the paper with specific parameters, the resulting graph G' has $O(n \log^2 n/\epsilon^2)$ edges in expectation, and is an ϵ-sparsification of G with probability at least $1 - n^{-d+1}$.*

With more passes of the input stream, the following holds:

Theorem 5 ([GKK10])**.** *For any $\epsilon > 0$, there exists an $O(\log n)$-pass streaming algorithm that produces an ϵ-sparsification G' of G with at most $O(n \log^2 n/\epsilon^2)$ edges using $O((n/m) \log n + \log n \log\log n)$ time per edge.*

The final theorem, which seems the most promising for our extension, is the following:

Theorem 6 ([GKK10])**.** *For any $\epsilon > 0$ and $d > 0$, there exists a one-pass algorithm that given the edges of an undirected graph G streamed in adversarial order, produces an ϵ-sparsifier G' with $O(n \log^3 n/\epsilon^2)$ edges with probability at least $1 - n^{-d}$. The algorithm takes $O(\log\log n)$ amortized time per edge and uses $O(\log^2 n)$ space per node.*

Remark 1. *All these theorems have their tradeoffs but it guarantees that the sparsifier will have $\widetilde{O}(n/\epsilon^2)$ edges in expectation. Therefore, any of the algorithms from these theorems should do. Furthermore, we could potentially shave off $\log n$ factors with perhaps modern techniques (the paper is relatively old and much progress has been made in shrinking the size of the constructed sparsifier).*

After selecting the appropriate algorithm, we would need to analyze it under the cut-balance of directed graphs. A main concern while reviewing the literature was the fact that the lemmas and analysis from [BK15] were for undirected graphs. So, it wasn't clear if the lemmas would transfer over into the directed case. However, we believe with moderate confidence that they do. Specifically, [CCPS21] makes use of those lemmas when the directed graphs are restricted to a cut-balance. Thus, to resolve this question, the remaining step would be to analyze the same algorithm in the scope of the cut balance idea. Using these concepts, we hope to achieve a similar result to **Theorem 1.3** in [CCPS21], namely a weighted subgraph with $\widetilde{O}(\beta n/\epsilon^2)$ edges and preserves the values of all β-balanced cuts up to a factor of $1 \pm \epsilon$ for $\epsilon > 0$ in the semi-streaming model.

6 Conclusion

Computing the β in balanced-cut directed graphs is provably difficult in the streaming model. In Section 4, we showed that it requires $\Omega(n^2)$ bits of memory when β is unknown. In terms of streaming algorithms, we believe the guarantees of [CCPS21] can be lifted to the semi-streaming

model by making use of streaming algorithms presented by [GKK10]. With more time, we would analyze the semi-streaming algorithm(s) under cut-balanced directed graphs.

With more time, we would also investigate our third research question: finding a lower bound for the scenario described in Section 5. More formally:

Question 3. *What is the lower bound to compute a directed cut sparsifier, where the whole graph is not guaranteed to be β-balanced, and we only need to preserve β-balanced cuts?*

7 Acknowledgments

We would like to thank David Woodruff and Honghao Lin for the productive conversations and direction in this project.

References

[AGM12] Kook Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: Sparsification, spanners, and subgraphs. *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 03 2012.

[BK96] Andras A. Benczur and David R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In *The 28th Annual ACM Symposium on the Theory of Computing*, 03 1996.

[BK15] András A. Benczúr and David R. Karger. Randomized approximation schemes for cuts and flows in capacitated graphs. *SIAM Journal on Computing*, 44(2):290–319, 2015.

[BSS12] Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-Ramanujan sparsifiers. *SIAM J. Comput.*, 41(6):1704–1721, 2012.

[CCM16] Amit Chakrabarti, Graham Cormode, and Andrew McGregor. Robust lower bounds for communication and stream computation. *Theory of Computing*, 12(10):1–35, 2016.

[CCPS21] Ruoxu Cen, Yu Cheng, Debmalya Panigrahi, and Kevin Sun. Sparsification of Directed Graphs via Cut Balance. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 45:1–45:21, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[CLL+24] Yu Cheng, Max Li, Honghao Lin, Zi-Yi Tai, David P. Woodruff, and Jason Zhang. Tight lower bounds for directed cut sparsification and distributed min-cut, 2024.

[CMP16] Michael B. Cohen, Cameron Musco, and Jakub Pachocki. Online row sampling, 2016.

[DGL+24] Matthew Ding, Alexandro Garces, Jason Li, Honghao Lin, Jelani Nelson, Vihan Shah, and David P. Woodruff. Space complexity of minimum cut problems in single-pass streams, 2024.

[GKK10] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. Graph sparsification via refinement sampling, 2010.

[KL11] Jonathan A. Kelner and Alex Levin. Spectral Sparsification in the Semi-Streaming Setting. In Thomas Schwentick and Christoph Dürr, editors, *28th International Symposium on Theoretical Aspects of Computer Science (STACS 2011)*, volume 9 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 440–451, Dagstuhl, Germany, 2011. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[KN96] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1996.

[Zel11] Mariano Zelke. Intractability of min- and max-cut in streaming graphs. *Information Processing Letters*, 111(3):145–150, 2011.