Pseudodeterministic Algorithms for Minimum Cut Problems

Aryan Agarwala
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

Nithin Varma
University of Cologne, Germany

Abstract

In this paper we present efficient pseudodeterministic algorithms for both the global minimum cut and minimum s-t cut problems. The running time of our algorithm for the global minimum cut problem is asymptotically better than the fastest sequential deterministic global minimum cut algorithm (Henzinger, Li, Rao, Wang; SODA 2024).

Furthermore, we implement our algorithm in streaming, PRAM, and cut-query models, where no efficient deterministic global minimum cut algorithms are known.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Pseudorandomness and derandomization

Keywords and phrases Minimum Cut, Pseudodeterministic Algorithms

Digital Object Identifier 10.4230/LIPIcs.ITCS.2026.4

Acknowledgements The authors thank Danupon Nanongkai for extensive discussions.

1 Introduction

Randomization is one of the cornerstones in the design of algorithms and has been effective in the design of simple and fast algorithms for a variety of computational problems. The advantage of deterministic algorithms, on the other hand, is their *replicability*, where multiple runs of the same algorithm on the same input result in the same output. The question of whether randomness results in provably faster algorithms than their deterministic counterparts is fundamental to the field of theoretical computer science.

The investigation of pseudodeterminism, initiated by Gat and Goldwasser [10] has implications to the aforementioned line of work [14, 30]. A pseudodeterministic algorithm for a search problem is a randomized algorithm which outputs the same answer with high probability. For the global minimum cut problem, a randomized algorithm only needs to output with high probability *some* minimum cut, while a pseudodeterministic algorithm needs to output, with high probability, the *same* minimum cut. Formally, an algorithm $\mathcal{A}$ is pseudodeterministic if for all inputs x, with probability $\rho \geq \frac{2}{3}$, we have $\mathcal{A}(x) = s(x)$, where $s(x)$ is a canonical solution for x. Pseudodeterministic algorithms capture the replicability property of deterministic algorithms, while being as efficient and simple as randomized algorithms, thereby giving us the best of both worlds. Pseudodeterministic algorithms have been studied for the construction of primes [35, 7, 34], bipartite matching [15], non-bipartite matching [2], matroid intersection [12], undirected connectivity [22], and more. It is an important open direction as to which problems have efficient pseudodeterministic algorithms.

1.1 Pseudodeterministic Global Minimum Cut Algorithm

In this work, we investigate the design of pseudodeterministic algorithms for the well-studied problem of global minimum cut. The global minimum cut problem takes as input a weighted undirected graph G on n vertices and m edges, and produces as output a partition of the

vertices of G which minimises the total weight of edges crossing the partition. This is closely related to the minimum s-t cut problem, which produces as output a partition of vertices of G such that the vertices s and t lie on different sides of the partition. Naively, the global minimum cut problem can be solved using $n-1$ calls to a minimum s-t cut algorithm [18]. Thus, it was initially believed that global minimum cut is a harder variant of the minimum s-t cut problem. Eventually, Nagamochi and Ibaraki [33] and Hao and Orlin [23] showed that global minimum cut can be solved with running time matching the best known minimum s-t cut algorithms at the time. Karger then initiated a line of work on *randomized* algorithms for the global minimum cut problem [25, 28, 26], eventually culminating in a $O(m \log^3 n)$ time algorithm, which is faster than the best minimum s-t algorithms even 25 years later. Thus, global minimum cut became one of the classical examples of how randomness can lead to significantly faster and simpler algorithms. Gawrychowski et al. [11] improved a log factor in Karger's algorithm, giving a $O(m \log^2 n)$ time randomized algorithm for global minimum cut. Recently, Henzinger et al. [24] obtained an $\tilde{O}(m)$[1] time deterministic algorithm for global minimum cut. We note that their algorithm, while matching the runtime of randomized algorithms upto polylog factors, seems to have a high exponent in the log term and is far more complicated than Karger's near-linear time algorithm [26].

We present a simple and efficient pseudodeterministic algorithm for the global minimum cut problem that uses, as black-box, any randomized algorithm for global minimum cut.

▶ **Theorem 1.** *Consider a randomized algorithm $\mathcal{A}$ that takes as input a simple graph $G = (V, E)$ with edge weights given by $w : E \to \mathbb{Z}^+$, runs in time $t(n, m)$ and outputs, with probability at least $\frac{2}{3}$, a minimum cut of G with respect to w. Then there exists a pseudodeterministic algorithm $\mathcal{A}'$ that, on input $G = (V, E, w)$, runs in time $O(t(n, m + n) \log(n) \log\log(n))$, and with probability at least $\frac{2}{3}$, outputs a specific global minimum cut of G.*

Our result, in conjunction with the algorithm of Gawrychowski et al. [11], immediately implies a pseudodeterministic global minimum cut algorithm with running time $O(m \log^3 n \log\log n)$, which is significantly faster than the best deterministic algorithm. The relative simplicity of our algorithm is an additional advantage.

1.2 Streaming, Cut Query, and Parallel Models

Since its introduction, pseudodeterminism has been studied in many contexts, such as query complexity [14, 6, 17], streaming algorithms [21, 4, 16], parallel algorithms [15, 2, 13], space-bounded computation [22], and more[2]. In this paper, we provide efficient implementations of our pseudodeterministic global minimum cut algorithm in the streaming, cut query and parallel models. Our pseudodeterministic algorithms are significantly more efficient than the best-known deterministic algorithms in the respective models. In fact, as we discuss below, there exist large gaps between randomized and deterministic algorithms in all these models, unlike in the sequential model.

1. **Streaming**: In this model, the input graph consists of a stream of insertions or deletions of edges and their associated weights. Specifically, each stream element is either of the form *insert* (e, w_e), where $e \in E$ and $w_e \in \mathbb{Z}^+$, or of the form *delete* e. Note that we

[1] The $\tilde{O}$ hides polylog(n) factors.
[2] For a better overview of the field, we refer the reader to the doctoral thesis of Grossman [20].

consider streaming of simple graphs, and therefore, an edge once inserted cannot be inserted again until it is deleted. The typical goal is to compute the global minimum cut using as little space and making as few passes over this stream as possible.
A randomized exact streaming algorithm for global minimum cut using $\tilde{O}(n)$ space and $\log(n)$ passes was shown by Mukhopadhyay and Nanongkai [31]. As far as we are aware, there is no deterministic algorithm for this problem using $O(n^{1.99})$ space and making $\tilde{O}(1)$ many passes.

▶ **Theorem 2.** *There exists a pseudodeterministic streaming algorithm for global minimum cut that makes $O(\log^2 n)$ passes over the input and uses $\tilde{O}(n)$ space.*

2. **Cut Query**: Here, access to the graph is permitted only via an oracle which takes as input a partition of the vertices into two parts and produces as output the total weight of edges which cross this partition. The goal is to compute the global minimum cut making as few queries to this oracle as possible.
A randomized exact algorithm for global minimum cut using $\tilde{O}(n)$ many cut queries was shown by Mukhopadhyay and Nanongkai [31]. As far as we are aware, the best deterministic algorithm is a corollary of the algorithm by Grebinski and Kucherov [19] and reconstructs the entire graph using $O(n^2/\log n)$ many queries.

▶ **Theorem 3.** *There is a pseudodeterministic global minimum cut algorithm that makes $\tilde{O}(n)$ cut queries.*

3. **Parallel**: We consider the Common-Read-Exclusive-Write (CREW) Parallel Random Access Machine (PRAM) model, where there are p processors connected to a shared random access memory. An arbitrary number of processors can read from a memory cell, whereas only one processor can write on a memory cell in a single time step.
In this model, a randomized exact algorithm for global minimum cut with $\tilde{O}(m)$ work and $\tilde{O}(1)$ depth was given by Anderson and Blelloch [3]. As far as we are aware, the only deterministic NC algorithm for this problem is by Karger and Motwani [27]. While they do not explicitly calculate the work of their algorithm, it is certainly a huge polynomial, and seems to be worse than even, say, $O(n^{10})$.

▶ **Theorem 4.** *There is a pseudodeterministic global minimum cut algorithm in the PRAM model that does $\tilde{O}(m)$ work and is of depth $\tilde{O}(1)$.*

1.3 Our Techniques

A key tool in our algorithm is built upon the *isolation lemma*. Introduced by Mulmuley, Vazirani, and Vazirani [32], the isolation lemma states that for any family of objects built on a ground set E, such as perfect matchings of a bipartite graph, if one assigns random polynomially bounded integer weights to each element of E, the minimum weight object, i.e., perfect matching, will be unique with high probability. Derandomizing the isolation lemma has since been a major open problem, with a long line of work on it (See e.g. [5, 8, 9, 37, 1])

Goldwasser and Grossman [15], in their pseudodeterministic NC algorithm for bipartite perfect matching, gave a pseudodeterministic algorithm for assigning edge weights to a bipartite graph such that the minimum weight perfect matching is unique with high probability. Then, using a randomized NC algorithm for minimum weight perfect matching on the input graph with their constructed edge weights, they obtain a pseudodeterministic NC algorithm for bipartite perfect matching. Anari and Vazirani [2] do the same for non-bipartite matching, and Ghosh et al. [13] for linear matroid intersection.

In this work, we follow a similar route as the algorithm of [15]. Our pseudodeterministic algorithm takes as input a weighted graph, and applies perturbations to the weights in order to make the global minimum cut of the graph unique. We mention that the previous algorithms that follow this approach in [15, 2] are slower than their randomized counterparts by a polynomial factor. Thus, ours is the first instance of this approach that preserves computational efficiency.

The implementations of our algorithm in the four models that we discussed (sequential, streaming, cut query, parallel) lose only a logarithmic factor in efficiency in all of those models. Thus, combining our algorithm with the randomized algorithms in [11, 31, 3], we obtain near optimal pseudodeterministic algorithms for global minimum cut across many models of computation. Moreover, while the isolation lemma has previously found applications in parallel [32, 8, 9, 37], space-bounded [36, 38], catalytic [1, 29] and sequential [32] models, this is the first application of isolation that we are aware of in streaming and cut query models.

We note that our exact approach can also give similar blackbox statements for the more general problem of submodular function minimisation. For succinctness, we do not state this explicitly in the paper.

1.4 Organization

The paper is organized as follows. Section 2 contains basic notation and definitions that we use throughout. Section 3 presents a pseudodeterministic algorithm for the minimum s-t cut problem and introduces some of the key ideas that we use in this paper. Section 4 contains our pseudodeterministic global minimum cut algorithm and its analysis. Finally, Section 5 contains the details of the implementation of our algorithm in the streaming, cut query and parallel models.

2 Preliminaries

In this section, we introduce some basic notation and definitions. Let $G = (V, E)$ be an undirected graph, where $V = \{1, \ldots, n\}$. Let $\emptyset \subsetneq S \subsetneq V$ be a subset of vertices. Any such vertex set is said to be a *cut-set* of the graph. The *cut* formed by S is the set

$$\text{cut}_G(S) = \{e \in E \mid e = (u, v), u \in S, v \notin S\}.$$

The cut-sets S and $V \setminus S$ both represent $\text{cut}_G(S)$; thus, we sometimes refer to this cut by $(S, V \setminus S)$.

Let $w : E \to \mathbb{Z}^+$ be an edge weight assignment. For a subset $E' \subseteq E$, we use $w \restriction_{E'}$ to denote the restriction of the weight function w to the set E'. The *weight* of a cut $(S, V \setminus S)$, denoted $w(\text{cut}_G(S))$, is equal to the sum of the weights of the edges in $\text{cut}_G(S)$. A cut $(S, V \setminus S)$ is a *global minimum cut* of G if its weight equals $\min_{\emptyset \subsetneq S \subsetneq V} w(\text{cut}_G(S))$.

▶ **Fact 5.** *The function $w(\text{cut}_G(\cdot)) : 2^V \to \mathbb{Z}^+$ defined above is a submodular function. That is, for sets $S, T \subseteq V$, it holds that $w(\text{cut}_G(S \cap T)) + w(\text{cut}_G(S \cup T)) \leq w(\text{cut}_G(S)) + w(\text{cut}_G(T))$.*

Let $s, t \in V$ be special source and sink vertices. A cut $(S, V \setminus S)$ is an s-t cut if $s \in S$ and $t \notin S$. An s-t cut $(S, V \setminus S)$ of G is a *minimum s-t cut* if its weight equals $\min_{\emptyset \subsetneq S \subsetneq V, s \in S, t \notin S} w(\text{cut}_G(S))$.

We now introduce the concept of stitched weights, which allow us to perturb weights while maintaining integrality. This has been used in many prior works (eg. [9, 2, 15, 37]).

▶ **Definition 6** (Stitched Weights). *Let $G = (V, E)$ be a graph and $w, w' : E \to \mathbb{Z}^+$ be two edge weight assignments. Let $|w'|$ be any value such that $|w'| \geq \sum_{e \in E} w'(e)$. We define the stitching $[w, w'] : E \to \mathbb{Z}^+$ to be an edge weight assignment defined as follows:*

$$[w, w'](e) = (|w'| + 1)w(e) + w'(e)$$

For the sake of notational simplicity, while referring to the stitched weight of a subset $A \subseteq E$, we may denote $[w, w'](A)$ by a tuple $(w(A), w'(A))$. This notation is useful due to the observation that the ordering of stitched weights behaves exactly like the lexicographic ordering of tuples (w, w').

▶ **Observation 7.** *Let $A, B \subseteq E$, then:*

1. $[w, w'](A) \leq [w, w'](B) \iff w(A) < w(B)$ *or* $\{w(A) = w(B)$ *and* $w'(A) \leq w'(B)\}$, *and*
2. $[w, w'](A) = [w, w'](B) \iff w(A) = w(B)$ *and* $w'(A) = w'(B)$.

Observation 7 immediately implies the following fact.

▶ **Fact 8.** *Let $G = (V, E)$ be a graph and $w, w' : E \to \mathbb{Z}^+$ be edge weight assignments. Any (s-t) minimum cut of G under $[w, w']$ is also a (s-t) minimum cut of G under w.*

▶ **Definition 9** (E^s_{star}). *Let V be the vertex set of a graph and $s \in V$ be a source vertex. The edge set E^s_{star} is defined to be the set of edges corresponding to the star graph on the vertex set V with the vertex s as the center. That is, $E^s_{\text{star}} = \{(s, v) \mid v \in V, v \neq s\}$.*

▶ **Definition 10** (w_s). *Let $G = (V, E)$ be a graph and $s \in V$ be a source vertex. The edge weight assignment $w_s : E \to \mathbb{Z}^+$ is defined by $e \mapsto \mathbb{I}[e \in E^s_{\text{star}}]$. That is,*

$$w_s(e) = \begin{cases} 1, & \text{if } e \in E^s_{\text{star}} \\ 0, & \text{otherwise.} \end{cases}$$

Note that $n \geq \sum_{e \in E} w_s(e)$. Thus, we may set $|w_s|$ to n when stitching w_s (see Definition 6).

3 Warm-up: Pseudodeterministic Minimum *s-t* Cut

In this section, we present our pseudodeterministic algorithm for the minimum s-t cut problem and prove the following theorem. Specifically, we provide a generic transformation from randomized algorithms to pseudodeterministic algorithms.

▶ **Theorem 11.** *Let $\mathcal{A}$ be a randomized algorithm that:*

1. *Takes as input a simple graph $G = (V, E)$ along with positive integer edge weights $w : E \to \mathbb{Z}^+$, and*
2. *Outputs an s-t cut (S, T) of G such that*

$$\Pr[(S, T) \text{ is a minimum } s\text{-}t \text{ cut of } G \text{ under edge weights } w] \geq \rho$$

Then, the algorithm $\mathsf{PD}^{\mathcal{A}}_{s,t}$ is a pseudodeterministic algorithm which:

1. *Takes as input a simple graph $G = (V, E)$ along with positive integer edge weights $w : E \to \mathbb{Z}^+$, and*
2. *There exists a fixed minimum s-t cut (S^*, T^*) of G satisfying $\Pr[\mathsf{PD}^{\mathcal{A}}_{s,t}(G, w) = (S^*, T^*)] \geq \rho$.*

Algorithm 1 Pseudodeterministic Algorithm $\mathsf{PD}^{\mathcal{A}}_{s,t}$ for Minimum s-t Cut.

Input: $G = (V, E)$ be the input graph with edge weights $w : E \to \mathbb{Z}^+$; oracle access to randomised algorithm $\mathcal{A}$ for weighted minimum s-t cut

Output: Minimum s-t cut (S, T)

1: Extend w to the domain $E \cup E^s_{\text{star}}$ by $w(e) = 0$ for all $e \in E^s_{\text{star}}$.
2: Run $\mathcal{A}$ on $G' = (V, E \cup E^s_{\text{star}})$ with edge weights $[w, w_s]$. Let the cut returned be (S, T).
3: Output (S, T).

The following claim simply observes that adding weight 0 edges to a graph does not change the value of any cuts.

▷ Claim 12. Let $G' = (V, E \cup E^s_{\text{star}})$ be a graph with edge weights $w : E \cup E^s_{\text{star}} \to \mathbb{Z}^+$ such that $w(e) = 0 \ \forall e \notin E$. Then (S, T) is a minimum s-t cut of G' if and only if (S, T) is a minimum s-t cut of $G = (V, E)$ with edge weights $w \restriction_E$.

Proof. Simply note that for any cut (S, T) of G',

$$w(\text{cut}_{G'}(S)) = \sum_{e \in \text{cut}_{G'}(S) \cap E} w(e) + \sum_{e \in \text{cut}_{G'}(S) \setminus E} w(e) = \sum_{e \in \text{cut}_{G'}(S) \cap E} w(e) = \sum_{e \in \text{cut}_G(S)} w \restriction_E (e)$$

$$= w \restriction_E (\text{cut}_G(S))$$

Thus, because each cut in G' with edge weights w have exactly the same weight as the corresponding cut in G with edge weights $w \restriction_E$, the minimum cuts of G' with weights w and G with weights $w \restriction_E$ are exactly the same. ◁

▶ **Fact 13.** *Let $G = (V, E)$ be a graph with edge weights $w : E \to \mathbb{Z}^+$ and a source vertex s and sink vertex t. Let $S \subseteq V$ and $S' \subseteq V$ be cut-sets that both correspond to minimum s-t cuts in G under edge weights w, with $s \in S' \cap S$. Then $S \cup S'$ is also a cut-set corresponding to a minimum s-t cut in G under edge weights w.*

Proof. It is a simple fact of submodular functions that $w(\text{cut}_G(S \cap S')) + w(\text{cut}_G(S \cup S')) \leq w(\text{cut}_G(S)) + w(\text{cut}_G(S'))$. Since S and S' are both minimisers of $w(\text{cut}_G(S))$, both terms on the left must also be minimisers. Thus, $S \cup S'$ is also a cut-set corresponding to a minimum cut of G under edge weights w. ◀

▷ Claim 14. Let $G' = (V, E \cup E^s_{\text{star}})$ be a graph with edge weights $w : E \cup E^s_{\text{star}} \to \mathbb{Z}^+$. Let $s \in V$ be a fixed source vertex. For any $t \in V \setminus \{s\}$, there exists a unique minimum s-t cut in G' with edge weights $[w, w_s]$.

Proof. Fix some $t \in V \setminus \{s\}$. Assume for the sake of contradiction that G' under edge weights $[w, w_s]$ admits multiple minimum weight s-t cuts. Let these cuts be (S_1, T_1) and (S_2, T_2). Assume that they have weight (W, W'). Note that $W' = |T_1| = |T_2|$. Consider now the cut $(S_1 \cup S_2, T_1 \cap T_2)$.

By Fact 8, we know that (S_1, T_1) and (S_2, T_2) are both also minimum cuts of G under edge weights w. This implies, by Fact 13, that $(S_1 \cup S_2, T_1 \cap T_2)$ is also a minimum cut of G under edge weights w. Thus,

$$w(\text{cut}_G(S_1 \cup S_2)) = W.$$

Moreover,

$$w_s(\text{cut}_G(S_1 \cup S_2)) = |T_1 \cap T_2| < |T_1| = |T_2| = w_s(\text{cut}_G(S_1)) = w_s(\text{cut}_G(S_2)).$$

Thus, we have

$$[w, w_s](\text{cut}_G(S_1 \cup S_2)) < [w, w_s](\text{cut}_G(S_1)) = [w, w_s](\text{cut}_G(S_2))$$

This contradicts the minimality of (S_1, T_1) and (S_2, T_2). This completes the proof. ◁

Proof of Theorem 11. Consider $G' = (V, E \cup E^s_{\text{star}})$ with edge weights $[w, w_s]$. By Claim 14, G' with weights $[w, w_s]$ has a unique minimum s-t cut. Let it be (S^*, T^*). By Claim 12 and Fact 8, (S^*, T^*) is a minimum cut of G with respect to weights w.

Now, simply note that $\mathsf{PD}^{\mathcal{A}}_{s,t}(G, w) = \mathcal{A}(G', [w, w_s])$. With probability $\geq \rho$, the output $\mathcal{A}(G', [w, w_s])$ is a minimum cut of G' under weights $[w, w_s]$. The only such minimum cut is (S^*, T^*). Thus, with probability $\geq \rho$, the output $\mathsf{PD}^{\mathcal{A}}_{s,t}(G, w) = (S^*, T^*)$. This completes the proof. ◀

▶ Remark 15. As far as we are aware, the most efficient minimum s-t cut algorithms in all models use s-t max flow algorithms. Given an s-t max flow, it is easy to obtain a canonical minimum s-t cut by simply finding the cut-set S of vertices reachable from s in the residual graph formed by the maximum flow. Thus, we instead focus on global minimum cut, where much more efficient algorithms than max flow are known.

4 Pseudodeterministic Global Minimum Cut

In this section, we design pseudodeterministic algorithms for the global minimum cut problem. Specifically, we prove the following.

▶ **Theorem 16.** *Let $\mathcal{A}$ be a randomised algorithm which:*

1. *Takes as input a simple graph $G = (V, E)$ along with positive integer edge weights $w : E \to \mathbb{Z}^+$, and*
2. *Outputs a cut (S, T) of G such that*

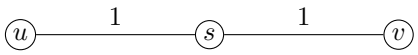
$$\Pr[S \text{ is a minimum cut of } G \text{ under edge weights } w] \geq \rho$$

Then, the algorithm $\mathsf{PD}^{\mathcal{A}}_{\text{global}}$ is a pseudodeterministic algorithm which:

1. *Takes as input a simple graph $G = (V, E)$ along with positive integer edge weights $w : E \to \mathbb{Z}^+$, and*
2. *There exists a fixed minimum cut (S^*, T^*) of G such that*

$$\Pr[\mathsf{PD}^{\mathcal{A}}_{\text{global}}(G, w) = (S^*, T^*)] \geq 1 - (1 - \rho) \cdot O(\log n).$$

The structure of global minimum cuts, for our purpose, is slightly more complicated than minimum s-t cuts. The following example of a simple path illustrates why our approach for minimum s-t cut does not work immediately.



Simply note that both $\{s, v\}$ and $\{s, u\}$ are global minimum cuts of the transformation $G' = (V, E \cup E^s_{\text{star}})$ under edge weights $[w, w_s]$.

The following lemma sheds light on the structure of these counterexamples.

▶ **Lemma 17.** *Let $G = (V, E)$ be a simple graph with edge weights $w : E \to \mathbb{Z}^+$. Let $s \in V$ be a special source vertex. Consider the transformed graph $G' = (V, E \cup E^s_{\text{star}})$ with edge weights $[w, w_s]$. Let $\{(S_1, T_1), \ldots, (S_k, T_k)\}$ be the global minimum cuts of G' under edge weights $[w, w_s]$ such that $s \in S_i$ $\forall i \in [k]$. Then:*

1. *For all $i, j \in [k]$ such that $i \neq j$, we have $T_i \cap T_j = \emptyset$.*
2. *For all $i, j \in [k]$, we have $|T_i| = |T_j|$.*

Proof. Claim 14 implies that the Ts must be pairwise disjoint. Else, let (S_i, T_i) and (S_j, T_j) both be minimum cuts such that $s \in S_i \cap T_j$ and $t \in T_i \cap T_j$. These are both minimum s-t cuts, which contradicts the uniqueness of the minimum s-t cut.

Now, consider any (S_i, T_i) and (S_j, T_j). Note that since these are both minimum cuts, in particular they have the same weight with respect to both w and w_s (by Observation 7). Thus, $w_s(\text{cut}_{G'}(S_i)) = w_s(\text{cut}_{G'}(S_j)) \implies |T_i| = |T_j|$. This completes the proof. ◀

In order to exploit this property, we now introduce a second type of edge set and weight assignment.

▶ **Definition 18** ($w_s^{<i}$)**.** *Let $G = (V, E)$ be a graph and $s \in V$ be a source vertex. The weight assignment $w_s^{<i} : E \to \mathbb{Z}^+$ is defined as*

$$w_s^{<i}(e) = \begin{cases} 1, & \text{if } e = (s, v) \text{ and } v < i \\ 0, & \text{otherwise.} \end{cases}$$

Note that $n \geq \sum_{e \in E} w_s^{<i}(e)$. Thus, we may set $|w_s^{<i}| = n$ when stitching $w_s^{<i}$ (see Definition 6).

We now need to work with three weight assignments instead of two, and so we introduce the concept of repeated stitching.

▶ **Definition 19** (Repeated Stitching)**.** *Let $G = (V, E)$ be a graph and $w_1, w_2, w_3 : E \to \mathbb{Z}^+$ be edge weights. The weight assignment $[w_1, w_2, w_2]$ is defined as the repeated stitching $[[w_1, w_2], w_3]$.*

▷ Claim 20. Let $G = (V, E)$ be a graph with edge weights w, and let s be a source vertex. Let $G' = (V, E \cup E^s_{\text{star}})$, and let $\{(S_1, T_1), \ldots, (S_k, T_k)\}$ be the set of global minimum cuts in G' according to weights $[w, w_s]$, such that $s \in S_i \ \forall i \in [k]$. Furthermore, let $\mathcal{T} = \bigcup_{i \in [k]} T_i = \{t_1, t_2, \ldots, t_m\}$.

For any $x \in V$:

1. If $x > t_m$, then every minimum cut (S, T) of G' w.r.t. weights $[w, w_s, w_s^{<x}]$ (with $s \in S$) satisfies $w_s^{<x}(\text{cut}_{G'}(S)) = |T|$.
2. If $x \leq t_{m-1}$, then either every minimum cut (S, T) of G' w.r.t. weights $[w, w_s, w_s^{<x}]$ (with $s \in S$) satisfies $w_s^{<x}(\text{cut}_{G'}(S)) < |T| - 1$, or G' under weights $[w, w_s, w_s^{<x}]$ has multiple minimum cuts (or both).
3. If $t_{m-1} < x \leq t_m$, then the minimum cut (S, T) of G' w.r.t. weights $[w, w_s, w_s^{<x}]$ is unique and satisfies $w_s^{<x}(\text{cut}_{G'}(S)) = |T| - 1$. Moreover, (S, T) is exactly the cut (S_i, T_i) such that $t_m \in T_i$.

Proof. By Lemma 17 then, all T_i are disjoint and equal sized. By Fact 8, any minimum cut (S, T) of G' under weights $[w, w_s, w_s^{<x}]$, is a minimum cut of G' under weights $[w, w_s]$. Thus, $(S, T) = (S_i, T_i)$ for some $i \in [k]$, and more importantly $w_s^{<x}(S) = \min_{i \in [k]} w_s^{<x}(S_i)$.

For any cut S_i, we have $w_s^{<x}(\text{cut}_{G'}(S_i)) = |T_i \cap \{v \in V \mid v < x\}| = |T_i \setminus \{v \in V \mid v \geq x\}| = |T| - |\{v \in T \mid v \geq x\}|$.

Now, we handle the cases individually:

1. If $x > t_m$, we have that $|\mathcal{T} \cap \{v \in V \mid v \geq x\}| = \emptyset$, which implies that $\forall i \in [k]$, $w_s^{<x}(\text{cut}_{G'}(S_i)) = |T_i|$. Thus, $w_s^{<x}(\text{cut}_{G'}(S)) = |T|$.
2. If $x \leq t_{m-1}$, we have $|\mathcal{T} \setminus \{v \in V \mid v \geq x\}| \geq 2$.

   a. If there exists $i \in [k]$ such that $T_i \cap \{v \in V \mid v \geq x\} \geq 2$, then $w_s^{<x}(\text{cut}_{G'}(S)) \leq w_s^{<x}(\text{cut}_{G'}(S_i)) \leq |T_i| - 2 < |T_i| - 1 = |T| - 1$. Thus, we end up in the first case.
   b. If instead we have that $\forall i \in [k]$, $|T_i \cap \{v \in V \mid v \geq x\}| \leq 1$, then since the T_i are disjoint, there must exist $i, j \in [k]$ such that $|T_i \cap \{v \in V \mid v \geq x\}| = |T_j \cap \{v \in V \mid v \geq x\}| = 1$, and we thus have that both S_i and S_j are minimum cuts of G' under weights $[w, w_s, w_s^{<x}]$. This is the second case, where the minimum cut is not unique.
3. If $t_{m-1} < x \leq t_m$. Let (S_i, T_i) be the cut such that $t_m \in T_i$. Simply note that for all $j \neq i$, $w_s^{<x}(\text{cut}_{G'}(S_j)) = |T_j| = |T|$, whereas $w_s^{<x}(\text{cut}_{G'}(S_i)) = |T_i| - 1 = |T| - 1$. Thus, we have that $(S, T) = (S_i, T_i)$ is the unique minimum cut of G' under weights $[w, w_s, w_s^{<x}]$ and it satisfies $w_s^{<x}(\text{cut}_{G'}(S)) = |T| - 1$.

This completes the proof. ◁

The last ingredient we need is an algorithm that checks whether the minimum cut of a graph is unique. We describe it in Algorithm 2.

Algorithm 2 Uniqueness-Test$^{\mathcal{A}}$.

Input: Graph $G = (V, E)$ with edge weights $w : E \to \mathbb{Z}^+$; oracle access to a randomised algorithm $\mathcal{A}$ for the global minimum cut problem on simple weighted graphs.
Output: A unique cut in $G = (V, E)$ w.r.t. w if it exists and **False** otherwise

1: Run $\mathcal{A}$ on G with edge weights w.
2: Let the cut returned be (S, T) such that $s \in S$.
3: Let $G' = (V, E \cup E^s_{\text{star}})$. Extend w to the domain $E \cup E^s_{\text{star}}$ by $w(e) = 0$ for all $e \in E^s_{\text{star}} \setminus E$.
4: Let $w_1 : E \cup E^s_{\text{star}} \to \mathbb{Z}^+$ be the edge weight assignment defined by $w_1(e) = 1$ if $e = (s, v) \in E^s_{\text{star}} \cap \text{cut}_{G'}(S)$, and $w_1(e) = 0$, otherwise.
5: Select an arbitrary vertex $t \in T$.
6: Let $\bar{G} = (V, E \cup E^t_{\text{star}})$.
7: Let $w_2 : E \cup E^t_{\text{star}} \to \mathbb{Z}^+$ be the edge weight assignment defined by $w_2(e) = 1$ if $e = (t, v) \in E^t_{\text{star}} \cap \text{cut}_{G'}(S)$, and $w_2(e) = 0$, otherwise.
8: Run $\mathcal{A}$ on G' with edge weights $[w, w_1]$. Let the cut returned be (S', T') such that $s \in S'$.
9: Run $\mathcal{A}$ on $\bar{G}$ with edge weights $[w, w_2]$. Let the cut returned be $(\bar{S}, \bar{T})$ such that $s \in \bar{S}$.
10: **return** (S, T) if $S = S' = \bar{S}$ and **False** otherwise.

▶ **Theorem 21.** *Let $\mathcal{A}$ be a randomised algorithm which:*
1. *Takes as input a simple graph $G = (V, E)$ along with positive integer edge weights $w : E \to \mathbb{Z}^+$, and*
2. *Outputs an cut (S, T) of G such that*

$$\Pr[S \text{ is a minimum cut of } G \text{ under edge weights } w] \geq \rho$$

Then, Uniqueness-Test$^{\mathcal{A}}$ is a randomised algorithm which:
1. *Takes as input a simple graph $G = (V, E)$ along with positive integer edge weights $w : E \to \mathbb{Z}^+$, and*
2. *Decides whether the minimum cut of G is unique with probability $\geq 1 - 3(1 - \rho)$,*
3. *Using only $O(1)$ calls to $\mathcal{A}$.*

Proof. First, note that Uniqueness-Test$^{\mathcal{A}}$ makes exactly 3 calls to $\mathcal{A}$. Assume that both calls return a minimum cut. This happens with probability $\geq 1 - 3(1 - \rho)$.

If the minimum cut (S,T) of G w.r.t. weights w is unique, then the cuts (S',T') and $(\bar{S},\bar{T})$ must both be equal to (S,T). This is true for (S',T') because it is the minimum cut of $G' = (V, E \cup E^s_{\text{star}})$ under $[w, w_1]$, which by Fact 8 is a minimum cut of $G' = (V, E \cup E^s_{\text{star}})$ under w, which by Claim 12, is a minimum cut of G under w. The same proof works for $(\bar{S},\bar{T})$. Thus, in this case, Uniqueness-Test$^{\mathcal{A}}$ correctly decides that the minimum cut of G under weights w is unique.

Assume that the minimum cut of G under weights w is not unique. Let (S,T) be one of them, and let $(\hat{S},\hat{T})$ be another. If $(S',T') \neq (S,T)$, then Uniqueness-Test$^{\mathcal{A}}$ correctly decides that the minimum cut of G under weights w is not unique. Thus, assume that $(S',T') = (S,T)$. We will show that in this case $(\bar{S},\bar{T}) \neq (S,T)$. First, we have $w(\text{cut}_{G'}S) = w(\text{cut}_{G'}\hat{S})$. Thus, if $(S',T') = (S,T)$, we must have $w_1(\text{cut}_{G'}S) \leq w_1(\text{cut}_{G'}\hat{S}) \iff |T| \leq |\hat{T} \cap T| \implies T \subsetneq \hat{T}$. Thus, note that we have $t \in \hat{T}$ and $|\hat{S}| < |S|$. This implies $w_2(\text{cut}_{\bar{G}}S) = |S| > |\hat{S}| = w_2(\text{cut}_{\bar{G}}\hat{S})$. Thus, $(\bar{S},\bar{T}) \neq (S,T)$. This shows that Uniqueness-Test$^{\mathcal{A}}$ correctly decides that the minimum cut of G w.r.t. weights w is not unique. ◀

Next, we describe our pseudodeterministic global minimum cut algorithm in Algorithm 3. It runs both an arbitrary global minimum cut algorithm $\mathcal{A}$ as well as Uniqueness-Test$^{\mathcal{A}}$. We then complete the proof of Theorem 16.

Algorithm 3 Pseudodeterministic Algorithm $\mathsf{PD}^{\mathcal{A}}_{\text{global}}$ for Global Minimum Cut.

Input: Graph $G = (V,E)$ with edge weights $w : E \to \mathbb{Z}^+$; oracles access to $\mathcal{A}$, a randomised algorithm for the global minimum cut problem on simple weighted graphs
Output: Global Minimum Cut (S,T)

1: Fix an arbitrary vertex $s \in V$ to be the source vertex.
2: Extend w to the domain $E \cup E^s_{\text{star}}$ by $w(e) = 0$ for all $e \in E^s_{\text{star}} \setminus E$.
3: Define $G' = (V, E \cup E^s_{\text{star}})$.
4: If G' has a unique global minimum cut (S,T) under weights $[w, w_s]$, output (S,T) and terminate.
5: $l \leftarrow 1, u \leftarrow n$
6: **while** $l \leq u$ **do** ▷ Do binary search over the range $[l = 1, u = n]$.
7: Let $x = \lfloor (l+u)/2 \rfloor$ and $(S,T) = \mathcal{A}(G', [w, w_s, w_s^{<x}])$
8: If (S,T) is the unique global minimum cut in G' with weights $[w, w_s, w_s^{<x}]$, and $w_s^{<x}(\text{cut}_{G'}(S)) = |T| - 1$, output (S,T) and terminate.
9: If $w_s^{<x}(\text{cut}_{G'}(S)) = |T|$, let $u = x - 1$ and continue the binary search.
10: Else, let $l = x + 1$ and continue the binary search.
11: **end while**

Proof of Theorem 16. Assume that every time $\mathsf{PD}^{\mathcal{A}}_{\text{global}}$ uses a randomised algorithm as a subroutine, it returns the correct answer. This happens with probability $\geq \rho$ for each use of $\mathcal{A}$, and with probability $\geq 1 - (1-\rho)\cdot 3$ for the uniqueness test. Since we use the uniqueness test and $\mathcal{A}$ at most $O(\log n)$ many times, by a union bound, all subroutines work correctly with probability $\geq 1 - (1-\rho)\cdot O(\log n)$.

Now, let $(S_1,T_1),\ldots,(S_k,T_k)$ be the minimum cuts of $G' = (V, E \cup E^s_{\text{star}})$ under weights $[w, w_s]$ (such that $\forall i \in [k],\ s \in S_i$), and let $\mathcal{T} = \bigcup_{i\in[k]} T_i = \{t_1,\ldots,t_m\}$. Let $i \in [k]$ be such that $t_m \in T_i$. Lemma 17 shows that the T_j are all disjoint, and thus this choice of i is unique. We will show that $\mathsf{PD}^{\mathcal{A}}_{\text{global}}$ returns $(S^*,T^*) = (S_i,T_i)$.

First, if $k = 1$, then Step 4 of Algorithm 3 necessarily returns $(S^*, T^*) = (S_i, T_i)$. Thus, assume that $k > 1$. In this case, we perform the binary search. Now, simply note that the conditions of Claim 20 ensure that our binary search procedure finds $x \in (t_{m-1}, t_m]$, at which point G' w.r.t. weights $[w, w_s, w_s^{<x}]$ has a unique minimum cut, which is (S_i, T_i), and $\mathsf{PD}^{\mathcal{A}}_{\text{global}}$ algorithm outputs this in Step 8.

Thus, if every randomised subroutine works correctly, $\mathsf{PD}^{\mathcal{A}}_{\text{global}}$ outputs (S^*, T^*). As discussed in the first paragraph of the proof, this happens with probability $\geq 1 - (1 - \rho) \cdot O(\log n)$. This completes the proof. ◀

5 Streaming, Cut Query, and Parallel Models

In this section, we will briefly argue that Algorithm 3 can be implemented efficiently across sequential, streaming, cut query, and parallel models.

5.1 Sequential

Say $\mathcal{A}$ runs in time $T(n, m)$ on a graph G with n vertices and m edges. Clearly, Algorithm 2 can be implemented such that it runs in $O(m + n + T(n, m + n))$ time. Then, Algorithm 3 can be implemented to run in $O((m + n + T(n, m + n)) \log n \log \log n))$ time. Combined with the randomized $O(m \log^2 n)$ time algorithm of [11], this gives us a pseudodeterministic $O(m \log^3 n \log \log n)$ time algorithm for global minimum cut, thereby proving Theorem 1. This additional $\log \log n$ factor is in order to boost the success probability of $\mathcal{A}$ from $2/3$ to $1 - 1/\Omega(\log n)$. We incur a similar cost in the other models, but it is hidden under the tilde notation.

5.2 Parallel

Both Algorithm 3 and Algorithm 2 involve only the following non-trivial steps:

1. Adding the edges E^s_{star} of a star graph to the graph G. This can be done with $\tilde{O}(n + m)$ work and $\tilde{O}(1)$ depth by simply adding (s, v) for all $v \in V$ in parallel. It is easy to avoid adding a duplicate edge by maintaining a lookup table indexed by $v \in V$, indicating whether $(s, v) \in E$.
2. Stitching a weight assignment w' with a weight assignment w. Our algorithm only does this when n is an upper bound on $\sum_{e \in E} w'(e)$. Thus, we simply need to replace the weight w of each edge with $(n + 1) \cdot w(e) + w'(e)$. By doing this in parallel for each $e \in E$, this takes $\tilde{O}(n + m)$ work and $\tilde{O}(1)$ depth.
3. Testing whether two cut-sets S and S' are equal. There are many easy ways to do this with $\tilde{O}(n + m)$ work and $\tilde{O}(1)$ depth.
4. Computing the weight of a cut with respect to an edge weight assignment w. By simply iterating over all edges in parallel, and then adding up their contribution to the cut in a tree-like fashion, this can be done with $\tilde{O}(n + m)$ work and $\tilde{O}(1)$ depth.

Say $\mathcal{A}$ takes work $W(n, m)$ and depth $D(n, m)$ on a graph with n vertices and m edges. With the aforementioned observations, it's easy to see that Algorithm 2 and Algorithm 3 can be implemented with work $\tilde{O}(n + m + W(n, m + n))$ and time $\tilde{O}(D(n, m + n))$. The $\tilde{O}$ absorbs the $\log n$ factor from the binary search in Algorithm 3. Combined with the $\tilde{O}(n + m)$ work and $\tilde{O}(1)$ depth algorithm of [3], we obtain a pseudodeterministic $\tilde{O}(m)$ work and $\tilde{O}(1)$ depth parallel algorithm, thereby proving Theorem 4.

5.3 Streaming

Both Algorithm 3 and Algorithm 2 involve only the following non-trivial steps:

1. Adding the edges E^s_{star} of a star graph to the graph G. This can be done by simply suffixing the edges $E^s_{\text{star}} \setminus E$ to the input stream with weights 0. In order to do this, one needs to track the edges $E \cap E^s_{\text{star}}$ in the input stream. This is easy to do using $O(n \log n)$ space since $|E^s_{\text{star}}| = n$ – one can simply maintain a list of all edges of E^s_{star} and whether they have already appeared in the stream.
2. Stitching a weight assignment w' with a weight assignment w. Our algorithm only does this when n is an upper bound on $\sum_{e \in E} w'(e)$. Moreover, our algorithm can always compute $w'(e)$ for any edge e given only the endpoints of e. Thus, every time we encounter an edge e in the stream with edge $w(e)$, we can simply change its weight to $w(e) \cdot (n+1) + w'(e)$. This part implicitly requires space to store the information to compute w', which is just s, if $w' = w_s$, or x and s, if $w' = w_s^{<x}$, or the cut-set S along with vertices s and t if w' is w_1 or w_2. Storing the cut-set S is the most expensive and only requires $O(n \log n)$ space.
3. Testing whether two cut-sets S and S' are equal. Since we have no time restriction in the streaming model, once we have the sets S and S' stored using $O(n \log n)$ space, this is completely trivial to do using no additional space.
4. Computing the weight of a cut (S, T) with respect to a weight assignment w'. In our algorithm, we only need to do this when $w' = w_s^{<x}$, and the algorithm already knows x. Thus, this is easy to do: we maintain W, initialised to 0. Every time an edge e crossing the cut (S, T) is added to the graph in the stream, we modify $W = W + w'(e)$, and every time an edge e is deleted from the graph, we modify $W = W - w'(e)$. The only space required here is to store x and W. This requires $O(\log n)$ space.

Let $\mathcal{A}$ be a streaming algorithm that uses $S(n, m)$ space and makes $P(n, m)$ passes over the stream. With the aforementioned observations, Algorithm 2 and Algorithm 3 can be implemented with space $\tilde{O}((S(n, m) + n)$ and $O(P(n, m) \cdot \log(n))$ passes over the stream. Combined with the $\tilde{O}(n)$ space and $O(\log n)$ passes algorithm of [31], we obtain a pseudodeterministic $\tilde{O}(n)$ space and $O(\log^2 n)$ passes streaming algorithm, thereby proving Theorem 2.

5.4 Cut Query

For cut query algorithms, the argument is largely very simple, since the only restriction in the model is on queries to the graph and not on any computational aspect. The only non-trivial implementation question is about how to stitch weights $[w, w']$ in the cut query model. Formally, given a cut oracle $\mathcal{O}$, where $\mathcal{O}(S) = w(\text{cut}_G(S))$, we want to design an oracle $\mathcal{O}'$ where $\mathcal{O}'(S) = [w, w'](\text{cut}_G(S))$. This is easy to do because of our choices of stitched weights and graphs. In both Algorithm 2 and Algorithm 3, $w'(\text{cut}_G(S))$, for any $\emptyset \subsetneq S \subsetneq V$, can be computed given only S, and the parameters of the weight function w'. Assume $s \in S$:

1. $w_s(\text{cut}_G(S))$ is equal to $n - |S|$.
2. $w_s^{<x}(\text{cut}_G(S))$ is equal to $|\{v \in V \mid v < x\} \cap (V \setminus S)|$.
3. Let $\hat{S}$ be the cut-set with respect to which w_1 is defined, such that $s \in \hat{S}$. Then $w_1(\hat{S}) = |(V \setminus S) \cap (V \setminus \hat{S})|$. The same argument holds for w_2 and t.

Thus, the oracle $\mathcal{O}'$ can simply return as its answer $\mathcal{O}'(S) = \mathcal{O}(S) \cdot (n+1) + w'(\text{cut}_G(S))$.

Let $\mathcal{A}$ be a cut-query algorithm that makes $C(n, m)$ many cut queries. With the aforementioned observation, it's easy to see that Algorithm 3 and Algorithm 2 can be implemented such that they require $O(C(n, m) \cdot \log n)$ many queries. Thus, combined with the algorithm of [31], we obtain an algorithm which requires $\tilde{O}(n)$ many cut queries, thereby proving Theorem 3.

References

1. Aryan Agarwala and Ian Mertz. Bipartite matching is in catalytic logspace. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2025.
2. Nima Anari and Vijay V. Vazirani. Matching is as easy as the decision problem, in the NC model. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, Seattle, Washington, USA, January 12-14, 2020*, volume 151 of *LIPIcs*, pages 54:1–54:25. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ITCS.2020.54.
3. Daniel Anderson and Guy E. Blelloch. Parallel minimum cuts in $O(m \log^2 n)$ work and low depth. *ACM Trans. Parallel Comput.*, 10(4):18:1–18:28, 2023. doi:10.1145/3565557.
4. Vladimir Braverman, Robert Krauthgamer, Aditya Krishnan, and Shay Sapir. Lower bounds for pseudo-deterministic counting in a stream. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, Paderborn, Germany, July 10-14, 2023*, volume 261 of *LIPIcs*, pages 30:1–30:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.30.
5. Suresh Chari, Pankaj Rohatgi, and Aravind Srinivasan. Randomness-optimal unique element isolation, with applications to perfect matching and related problems. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of Computing*, pages 458–467, 1993. doi:10.1145/167088.167213.
6. Arkadev Chattopadhyay, Yogesh Dahiya, and Meena Mahajan. Query Complexity of Search Problems. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023)*, volume 272 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 34:1–34:15, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2023.34.
7. Lijie Chen, Zhenjian Lu, Igor C. Oliveira, Hanlin Ren, and Rahul Santhanam. Polynomial-time pseudodeterministic construction of primes. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 1261–1270. IEEE, 2023. doi:10.1109/FOCS57990.2023.00074.
8. Samir Datta, Raghav Kulkarni, and Sambuddha Roy. Deterministically isolating a perfect matching in bipartite planar graphs. *Theory of Computing Systems*, 47(3):737–757, 2010. doi:10.1007/s00224-009-9204-8.
9. Stephen Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite perfect matching is in quasi-NC. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 754–763, 2016. doi:10.1145/2897518.2897564.
10. Eran Gat and Shafi Goldwasser. Probabilistic search algorithms with unique answers and their cryptographic applications. *Electron. Colloquium Comput. Complex.*, TR11-136(18:136), 2011. URL: https://eccc.weizmann.ac.il/report/2011/136.
11. Pawel Gawrychowski, Shay Mozes, and Oren Weimann. Minimum cut in $O(m \log^2 n)$ time. *Theory Comput. Syst.*, 68(4):814–834, 2024. doi:10.1007/s00224-024-10179-7.
12. Sumanta Ghosh and Rohit Gurjar. Matroid intersection: A pseudo-deterministic parallel reduction from search to weighted-decision. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*, volume 207 of *LIPIcs*, pages 41:1–41:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.APPROX/RANDOM.2021.41.

13 Sumanta Ghosh, Rohit Gurjar, and Roshan Raj. A deterministic parallel reduction from weighted matroid intersection search to decision. *Algorithmica*, 86(4):1057–1079, 2024. doi:10.1007/s00453-023-01184-2.

14 Oded Goldreich, Shafi Goldwasser, and Dana Ron. On the possibilities and limitations of pseudodeterministic algorithms. In Robert D. Kleinberg, editor, *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, pages 127–138. ACM, 2013. doi:10.1145/2422436.2422453.

15 Shafi Goldwasser and Ofer Grossman. Bipartite perfect matching in pseudo-deterministic NC. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, Warsaw, Poland, July 10-14, 2017*, volume 80 of *LIPIcs*, pages 87:1–87:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.87.

16 Shafi Goldwasser, Ofer Grossman, Sidhanth Mohanty, and David P. Woodruff. Pseudo-deterministic streaming. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, Seattle, Washington, USA, January 12-14, 2020*, volume 151 of *LIPIcs*, pages 79:1–79:25. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ITCS.2020.79.

17 Shafi Goldwasser, Russell Impagliazzo, Toniann Pitassi, and Rahul Santhanam. On the pseudo-deterministic query complexity of NP search problems. In *36th Computational Complexity Conference (CCC 2021)*, volume 200 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 36:1–36:22, 2021. doi:10.4230/LIPIcs.CCC.2021.36.

18 R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961. doi:10.1137/0109047.

19 Vladimir Grebinski and Gregory Kucherov. Optimal reconstruction of graphs under the additive model. *Algorithmica*, 28(1):104–124, 2000. doi:10.1007/s004530010033.

20 Ofer Grossman. *Pseudo-determinism*. PhD thesis, MIT, USA, 2023. URL: https://hdl.handle.net/1721.1/151209.

21 Ofer Grossman, Meghal Gupta, and Mark Sellke. Tight space lower bound for pseudo-deterministic approximate counting. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 1496–1504. IEEE, 2023. doi:10.1109/FOCS57990.2023.00091.

22 Ofer Grossman and Yang P. Liu. Reproducibility and pseudo-determinism in log-space. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 606–620, 2019. doi:10.1137/1.9781611975482.38.

23 Jianxiu Hao and James B. Orlin. A faster algorithm for finding the minimum cut in a directed graph. *J. Algorithms*, 17(3):424–446, 1994. doi:10.1006/jagm.1994.1043.

24 Monika Henzinger, Jason Li, Satish Rao, and Di Wang. Deterministic near-linear time minimum cut in weighted graphs. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 3089–3139. SIAM, 2024. doi:10.1137/1.9781611977912.111.

25 David R. Karger. Random sampling in cut, flow, and network design problems. In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, STOC '94, pages 648–657, New York, NY, USA, 1994. ACM. doi:10.1145/195058.195422.

26 David R. Karger. Minimum cuts in near-linear time. *J. ACM*, 47(1):46–76, 2000. doi:10.1145/331605.331608.

27 David R. Karger and Rajeev Motwani. An NC algorithm for minimum cuts. *SIAM J. Comput.*, 26(1):255–272, 1997. doi:10.1137/S0097539794273083.

28 David R. Karger and Clifford Stein. A new approach to the minimum cut problem. *J. ACM*, 43(4):601–640, 1996. doi:10.1145/234533.234534.

29 Jiatu Li, Edward Pyne, and Roei Tell. Distinguishing, predicting, and certifying: On the long reach of partial notions of pseudorandomness. In *65th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2024, Chicago, IL, USA, October 27-30, 2024*, pages 1–13. IEEE, 2024. doi:10.1109/FOCS61266.2024.00095.

30 Zhenjian Lu, Igor C. Oliveira, and Rahul Santhanam. Pseudodeterministic algorithms and the structure of probabilistic time. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 303–316. ACM, 2021. doi:10.1145/3406325.3451085.

31 Sagnik Mukhopadhyay and Danupon Nanongkai. Weighted min-cut: sequential, cut-query, and streaming algorithms. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, STOC 2020, pages 496–509, New York, NY, USA, 2020. ACM. doi:10.1145/3357713.3384334.

32 Ketan Mulmuley, Umesh V Vazirani, and Vijay V Vazirani. Matching is as easy as matrix inversion. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 345–354, 1987. doi:10.1145/28395.383347.

33 Hiroshi Nagamochi and Toshihide Ibaraki. Computing edge-connectivity in multigraphs and capacitated graphs. *SIAM J. Discret. Math.*, 5(1):54–66, 1992. doi:10.1137/0405004.

34 Igor C. Oliveira. Polynomial-time pseudodeterministic constructions (invited talk). In Olaf Beyersdorff, Mamadou Moustapha Kanté, Orna Kupferman, and Daniel Lokshtanov, editors, *41st International Symposium on Theoretical Aspects of Computer Science, STACS 2024, March 12-14, 2024, Clermont-Ferrand, France*, volume 289 of *LIPIcs*, pages 1:1–1:1. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.STACS.2024.1.

35 Igor C. Oliveira and Rahul Santhanam. Pseudodeterministic constructions in subexponential time. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 665–677. ACM, 2017. doi:10.1145/3055399.3055500.

36 Klaus Reinhardt and Eric Allender. Making nondeterminism unambiguous. *SIAM Journal on Computing (SICOMP)*, 29(4):1118–1131, 2000. doi:10.1137/S0097539798339041.

37 Ola Svensson and Jakub Tarnawski. The matching problem in general graphs is in Quasi-NC. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 696–707. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.70.

38 Dieter van Melkebeek and Gautam Prakriya. Derandomizing isolation in space-bounded settings. *SIAM Journal on Computing (SICOMP)*, 48(3):979–1021, 2019. doi:10.1137/17M1130538.