On the Communication Complexity of Maximum Matching and Negative-Weight Shortest Paths

Yu Cheng
Department of Computer Science, Brown University, Providence, RI, USA

Tianle Jiang
Department of Computer Science, Duke University, Durham, NC, USA

Pachara Sawettamalya
Department of Computer Science, Princeton University, Princeton, NJ, USA

Huacheng Yu
Department of Computer Science, Princeton University, Princeton, NJ, USA

Abstract

We revisit several fundamental graph problems in the deterministic two-party communication model. Our main contributions include:

- We give a new $\widetilde{O}(n^{3/2})$-bit protocol for computing a maximum matching in general graphs. While the same upper bound can be obtained by simulating the classic algorithms of Micali-Vazirani [17] and Gabow [9], our protocol is conceptually simple and avoids the intricacies of finding a maximal set of shortest augmenting paths.
- We give a new $\widetilde{O}(n)$-bit protocol for negative-cycle detection and negative-weight single-source shortest paths. Our protocol simplifies that of Blikstad et al. [6] by replacing a long chain of reductions with a more direct approach based on vertex potentials.
- We give a combinatorial $\widetilde{O}(n)$-bit protocol for computing a maximum matching in bipartite graphs, obtained by reinterpreting the near-linear communication protocol of Blikstad et al. [6] through a discretized analysis.

Together, these results provide simpler protocols for several basic graph problems. We hope they will inspire further advances on the communication complexity of a wide range of graph problems.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Communication complexity

Keywords and phrases Graph algorithms, communication complexity, maximum matching, negative-weight shortest paths

Digital Object Identifier 10.4230/LIPIcs.ESA.2026.72

Related Version *Full Version*: https://arxiv.org/abs/2607.05751

Funding *Yu Cheng*: Supported in part by NSF Award CCF-2307106.
Tianle Jiang: Supported in part by NSF Award IIS-2402823. Part of the work was done while visiting Brown University.
Pachara Sawettamalya: Supported in part by NSF CAREER Award CCF-2339942.
Huacheng Yu: Supported in part by NSF CAREER Award CCF-2339942.

1 Introduction

Graph theory is central to computer science, both in theory and in practice. The field studies fundamental problems such as connectivity, shortest paths, matching, spanning forests, and cycle detection. Efficient solutions for these problems form the algorithmic backbone of many computational systems. While the complexity of these problems is well understood in centralized models, much less is known when the input is distributed across players with only partial views of the graph.

In this work, we study the *communication complexity of graph problems* in the standard two-party model [23]. The edge set of a graph is partitioned between two players, Alice and Bob, who must collaboratively solve a global graph problem while minimizing the amount of communication between them. Although this model is closely related to query, streaming, and distributed models of computation, many fundamental graph problems remain poorly understood from the perspective of communication complexity.

A recent breakthrough of Blikstad et al. [6] showed that *maximum bipartite matching* and several problems reducible to it admit deterministic protocols using only $\widetilde{O}(n)$ bits of communication.[1] At the core of their approach is a cutting-plane framework testing whether a polytope (or feasible region of a linear program (LP)) is empty or has non-trivial volume. This framework is particularly well-suited for the two-party edge-partition model: Given a graph problem $\mathcal{T}$, one formulates an LP whose feasibility gives the answer to $\mathcal{T}$, with constraints that are locally checkable and succinctly encodable for communication. The cutting-plane framework is then used to determine the feasibility of this LP, and the result is translated back into an answer for $\mathcal{T}$.

As a concrete example, Blikstad et al. [6] obtained a near-linear communication protocol for bipartite matching via the dual LP. Specifically, they reduce maximum bipartite matching to testing the feasibility of a vertex cover LP, which can be done using $O(n \log^2 n)$ bits of communication via the cutting-plane framework.

1.1 Our Contributions

1.1.1 Maximum Matching in General Graphs

Given the success of LP-based methods for bipartite graphs, a natural question is whether the same framework can be extended to *general (non-bipartite) matching*. However, general graphs do not admit a polynomial-size LP formulation with the same clean matching/vertex-cover duality as in bipartite graphs. The standard (primal) LP for non-bipartite matching has exponentially many constraints, so its dual LP has exponentially many variables. This makes it difficult to generalize the volume-based analysis in [6], which is highly sensitive to the dimension of the underlying polytope.

While achieving near-linear communication for general matching appears to require new ideas, we make partial progress by designing a simple protocol for general matching using $\widetilde{O}(n^{3/2})$ bits of communication.

▶ **Theorem 1** (Informal). *There is a deterministic two-party communication protocol for maximum matching in general graphs using $\widetilde{O}(n^{3/2})$ bits.*

This $\widetilde{O}(n^{3/2})$-bit upper bound is not entirely new: it can also be obtained by simulating the classic $O(m\sqrt{n})$-time algorithms such as Micali-Vazirani [17] and Gabow [9]. However, these algorithms are quite intricate even in the sequential setting. Although local computation is free in the communication model (which makes sophisticated data structures less of a bottleneck), turning them into explicit communication-efficient protocols still requires a careful implementation of the many steps in these algorithms.

Our main contribution lies in the *simplicity* of both the protocol and its correctness proof. At a high level, our protocol combines a greedy approach based on the Tutte-Berge formula with a careful simulation of Edmonds' Blossom algorithm [7]. For the protocol description, it

[1] Throughout this paper, we use $\widetilde{O}(\cdot)$ to hide polylog(n) factors.

suffices to focus on the problem of deciding whether the input graph has a perfect matching. In the first phase, we maintain the set of low-value Tutte-Berge covers (Definition 6) that cover the edges revealed so far, and greedily reveal edges that invalidate as many such covers as possible. The first phase uses $\widetilde{O}(n^{3/2})$ bits of communication, and either correctly decides that no perfect matching exists, or returns a matching of size $\frac{n}{2} - \widetilde{O}(\sqrt{n})$. In the second phase, we simulate $\widetilde{O}(\sqrt{n})$ rounds of the Blossom algorithm to augment this partial matching to a perfect matching, using another $\widetilde{O}(n^{3/2})$ bits of communication. It is worth noting that our greedy first phase is conceptually simple and more communication-efficient than directly simulating the Blossom algorithm from scratch (which would require $\widetilde{O}(n^2)$ bits).

1.1.2 Beyond Bipartite Matching

We revisit the cutting-plane approach of [6]. By abstracting away the problem-specific aspects of its application to bipartite matching, we show that it provides a general framework for designing communication-efficient protocols for graph problems.

We consider the following four tasks:

- Cycle: Given an unweighted directed graph, decide whether it contains a directed cycle.
- NegativeCycle: Given a directed graph with integer weights in $[-W, W]$, decide whether it contains a negative-weight cycle.
- SSSP: Given a directed graph with integer weights in $[0, W]$ and a source vertex s, compute the shortest-path distances from s to every vertex v.
- NegativeSSSP: Given a directed graph with integer weights in $[-W, W]$ and a source vertex s, either detect a negative-weight cycle or, if none exists, compute the shortest-path distances from s to every vertex v.

▶ **Theorem 2** (Informal). *There are deterministic two-party communication protocols using* $O(n \cdot \mathrm{polylog}(n, W))$ *bits of communication for* Cycle, NegativeCycle, SSSP, *and* NegativeSSSP.

Two of these problems admit straightforward near-linear communication protocols via simulations of classical algorithms: Cycle can be solved using Kosaraju's algorithm, and SSSP can be solved using Dijkstra's algorithm. We discuss these simulations in Appendix B and Appendix C.

The remaining two problems, NegativeCycle and NegativeSSSP, are more challenging due to the presence of negative edge weights. In the sequential setting, both can be solved using the Bellman-Ford algorithm. However, a straightforward simulation of Bellman-Ford requires $\widetilde{\Omega}(n^2)$ bits in the communication model. To avoid this cost, we apply the cutting-plane framework of [6] to a vertex-potential polytope (which encodes the triangle inequalities for shortest-path distances). This gives a near-linear communication protocol for NegativeCycle, as we discuss in Section 4.1. Moreover, a feasible point of this polytope is a key ingredient in our protocol for NegativeSSSP.

The near-linear communication upper bound for NegativeSSSP is not new: it was established in [6]. Their approach, however, proceeds through a chain of reductions, ultimately reducing NegativeSSSP to bipartite min-cost b-matching, which they solve using $\widetilde{O}(n)$ bits of communication (see Figure 1). These layers of reductions make it difficult to extract a self-contained protocol for NegativeSSSP from their work.

Our contribution is a conceptually simple and direct protocol for NegativeSSSP that avoids these reductions. At a high level, the protocol computes a feasible point of the vertex-potential polytope used for NegativeCycle. We then use this point as a proxy for the vertex potentials that would arise in Johnson's algorithm for all-pairs shortest paths. Reweighting the edges using these potentials essentially yields a communication-efficient reduction from

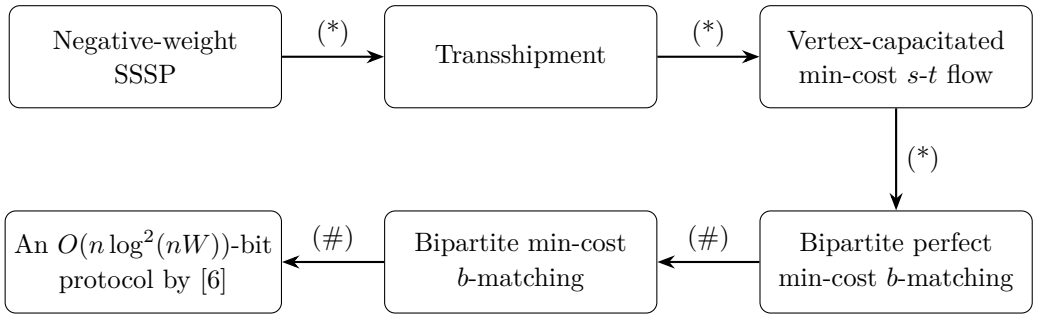

Figure 1 A sequence of reductions that computes negative-weight SSSP in $O(n\log^2(nW))$ bits of communication. The reductions (*) are given in [21] and (#) are given in [6].

NegativeSSSP to SSSP. Combined with an efficient simulation of Dijkstra's algorithm, this allows us to recover the exact shortest-path distances using near-linear communication. We describe our approach in detail in Section 4.2.

1.1.3 Combinatorial Protocol for Bipartite Matching

We present a *combinatorial* variant of the near-linear communication protocol for bipartite matching in [6]. Instead of tracking the continuous volume of the vertex-cover polytope, our protocol tracks a discrete analogue: the number of feasible grid points, corresponding to $\frac{1}{\text{poly}(n)}$-integral vertex covers. We show that this discrete "volume" is a sufficiently accurate proxy for continuous volume in the analysis. This gives a near-linear communication protocol for bipartite matching whose implementation is purely combinatorial. We discuss this in Section 5.

1.2 Two-Party Communication Complexity of Graph Problems

Table 1 Two-party communication complexities of various graph problems.

Problem	**Communication Complexity**	**Remarks**
BFS, DFS, spanning forest, connectivity, reachability	$\Theta(n\log n)$	Direct simulation of classic algorithms; lower bound by [12].
k-edge connectivity	$O(kn\log n)$	Nagamochi-Ibaraki algorithm [18] using k spanning forests.
s-t vertex connectivity	$\widetilde{O}(n)$	A reduction to vertex-capacitated max flow [6].
Global vertex connectivity	$\Theta(n^2)$ (det.) $\widetilde{\Theta}(n^{3/2})$ (rand.)	Combining local connectivity (det.); [5] (rand.).
Bipartite maximum matching	$O(n\log^2 n)$	Continuous LP-based protocol [6]. **We give a combinatorial protocol in Section 5.2.**
General maximum matching	$\widetilde{O}(n^{3/2})$	Simulation of [17, 9]. **We give a simple protocol in Section 2.**

Table 1 Continued

Problem	Communication Complexity	Remarks
Strongly connected components	$O(n \log n)$	Kosaraju's algorithm (see Appendix B).
Directed cycle detection	$O(n \log n)$	Kosaraju's algorithm and DFS (see Appendix B).
Negative cycle detection	$O(n \log^2(nW))$	A chain of reductions [6]. **We give a simple protocol in Section 4.1.**
Single-source shortest paths (SSSP)	$O(n \log(nW))$	Dijkstra's algorithm (see Appendix C).
Negative-weight SSSP	$O(n \log^2(nW))$	Reducing to bipartite min-cost b-matching [6]. **We give a simple protocol in Section 4.2.**
Global min-cut	$\widetilde{O}(n)$	Derandomized reduction to cut-query algorithms of [19, 1, 15].
Exact min s-t cut and undirected uncapacitated max flow	$\widetilde{O}(n^{11/7})$ (det.) $\widetilde{O}(n^{3/2})$ (rand.)	[13] (det.); [10, 16, 22] (rand.).
Cut sparsification	$\widetilde{O}(n/\varepsilon^2)$	Combining local cut sparsifiers of size $\widetilde{O}(n/\varepsilon^2)$ from both players.
ε-approximate min s-t cut	$\widetilde{O}(n/\varepsilon^2)$	Taking a min s-t cut of a cut sparsifier.
Minimum spanning tree	$\widetilde{O}(n)$	Simulating Kruskal's algorithm.
$(\Delta+1)$-coloring	$\widetilde{O}(n)$ (det.) $\Theta(n)$ (rand.)	[2] (det.); [8] (rand.).

2 Maximum Matching in General Graphs

The problem of *perfect matching in general graphs* (PM) is defined in the communication model as follows: Alice and Bob are given a disjoint edge partition of a graph $G = (V, E)$ where $n = |V|$ is even. Their goal is to determine whether G has a perfect matching and, if so, to output one.

In this section, we present a deterministic $\widetilde{O}(n^{3/2})$-bit communication protocol for PM.

▶ **Theorem 3.** *There is a deterministic protocol $\mathcal{P}^{\mathrm{PM}}$ that solves* PM *using $O(n^{3/2} \log^{3/2} n)$ bits of communication.*

As a corollary, the protocol can be extended to compute a *maximum matching* with an additional $O(\log n)$ factor in communication.

▶ **Corollary 4.** *There is a deterministic protocol $\mathcal{P}^{\mathrm{MM}}$ that computes a maximum matching in G using $O(n^{3/2} \log^{5/2} n)$ bits of communication.*

Proof. For $0 \le r \le \frac{n}{2}$, let G_r be the graph obtained by adding $n-2r$ new auxiliary vertices to G, connecting each auxiliary vertex to every original vertex (and no edges between auxiliary vertices). Observe that G_r has a perfect matching if and only if G has a matching of size at

least r.[2] Thus, Alice and Bob can binary search over r to find the size of a maximum matching in G using $O(\log n)$ calls to $\mathcal{P}^{\mathrm{PM}}$ on graphs G_r with at most $2n$ vertices. For the final value of r, the original-original edges in a perfect matching of G_r form a maximum matching of G. By Theorem 3, the total communication is $O(\log n) \cdot O(n^{3/2} \log^{3/2} n) = O(n^{3/2} \log^{5/2} n)$ bits. ◀

A Two-Phase Protocol for Perfect Matching

We now describe the protocol for Theorem 3. Fix an integer parameter $1 \leq k < \frac{n}{2}$ to be optimized later. The protocol proceeds in two phases.

- In Phase 1, Alice and Bob either find a matching of size $\frac{n}{2} - k$, or correctly conclude that G has no perfect matching. Phase 1 requires $O\left(\frac{n^2 \log^2 n}{k}\right)$ bits of communication.
- In Phase 2, starting from a matching of size $\frac{n}{2} - k$, Alice and Bob repeatedly find augmenting paths. This requires k iterations to reach a perfect matching if one exists. By simulating the blossom algorithm [7], each augmenting path can be found using $O(n \log n)$ bits of communication. Phase 2 requires $O(kn \log n)$ bits of communication.

Choosing $k = \Theta(n^{1/2} \log^{1/2} n)$ gives total communication

$$O\left(\frac{n^2 \log^2 n}{k} + kn \log n\right) = O(n^{3/2} \log^{3/2} n),$$

which proves Theorem 3.

We next describe Phase 1 and analyze its correctness and communication cost in Section 2.1. The implementation and analysis of Phase 2 are provided in the full version of the paper.

2.1 Phase One: Eliminating Tutte-Berge Covers

In this section, we show how to implement Phase 1 of the protocol.

▶ **Lemma 5.** *Let $G = (V, E)$ be a general graph where $n = |V|$ is even. Let $1 \leq k < \frac{n}{2}$. There is a deterministic protocol that, using $O\left(\frac{n^2 \log^2 n}{k}\right)$ bits of communication, either finds a matching of G of size $\frac{n}{2} - k$, or correctly concludes that G has no perfect matching.*

The key idea is based on the Tutte-Berge formula [20, 3] (Theorem 7), which gives a min-max characterization of the maximum matching size in general graphs. Motivated by this, we define the notion of Tutte-Berge covers (Definition 6), which play a similar role as vertex covers in bipartite graphs: a low-value Tutte-Berge cover of G certifies that G cannot have a large matching.

Phase 1 maintains a public graph $H \subseteq G$ and the set $\mathcal{C}$ of low-value Tutte-Berge covers that cover H. Alice and Bob repeatedly reveal edges to invalidate covers in $\mathcal{C}$. If $\mathcal{C}$ becomes empty, then the Tutte-Berge formula implies that H has a matching of the target size. If some cover in $\mathcal{C}$ survives all edges of G, then it certifies that G has no perfect matching. The main point of the analysis is that, when G has a perfect matching M, every remaining cover is invalidated by many edges of M, so the players can eliminate covers quickly.

[2] Given a size-r matching in G, the remaining $n - 2r$ original vertices can be matched to the auxiliary vertices. Conversely, a perfect matching of G_r matches each auxiliary vertex to a distinct original vertex, leaving exactly $2r$ original vertices to be matched among themselves.

▶ **Definition 6** (Tutte-Berge Cover). *Let V be a vertex set. A Tutte-Berge cover is a tuple $C = (A, S_1, \ldots, S_t)$ such that $A, S_1, \ldots, S_t$ form a partition of V, and each S_i has odd size. The value of C is defined as*

$$\delta(C) := |A| + \sum_{i=1}^{t} \frac{|S_i| - 1}{2}.$$

We say that C covers an edge $e = (u, v)$ if either $u \in A$ or $v \in A$, or if both u and v are in S_i for some $i \in [t]$. For a graph H, we say that C covers H if it covers every edge in $E(H)$. We say an edge e invalidates C if C does not cover e.

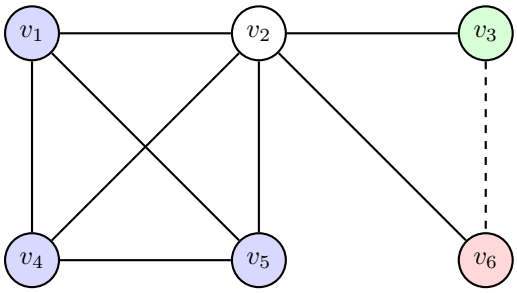

Figure 2 An example of a Tutte-Berge cover $C = (A = \{v_2\}, S_1 = \{v_1, v_4, v_5\}, S_2 = \{v_3\}, S_3 = \{v_6\})$, whose value is $\delta(C) = |A| + \sum_i \frac{|S_i|-1}{2} = 2$. The solid edges are exactly the edges covered by C. An edge with endpoints in two different odd sets invalidates C, e.g., the dashed edge (v_3, v_6).

We use the following classical result as a black box.

▶ **Theorem 7** (Tutte-Berge Formula [20, 3]). *For every graph G,*

$$\max_{\text{matching } M \text{ of } G} |M| = \min_{\text{Tutte-Berge cover } C \text{ of } G} \delta(C).$$

We now describe Phase 1 of the protocol (Algorithm 1).

Algorithm 1 Phase 1 of $\mathcal{P}^{\mathsf{PM}}$.

Input: A graph $G = (V, E)$ where $n = |V|$ is even, and an integer $1 \le k < \frac{n}{2}$.
Output: A matching of G of size $\frac{n}{2} - k$, or conclude that G has no perfect matching.

$H \leftarrow (V, \varnothing)$;
$\mathcal{C} \leftarrow \left\{C : C \text{ is a Tutte-Berge cover with } \delta(C) = \frac{n}{2} - k - 1\right\}$;
$T \leftarrow \left\lceil \frac{n^2 \ln n}{k} \right\rceil$;
for $i = 0, \ldots, T - 1$ **do**
  Find an edge $e_i \in (G \setminus H)$ that invalidates the largest number of covers in $\mathcal{C}$;
  if *no edge invalidates any cover in* $\mathcal{C}$ **then**
    return that G has no perfect matching;
  $H = H \cup \{e_i\}$;
  Remove from $\mathcal{C}$ every cover invalidated by e_i;
  if $\mathcal{C} = \varnothing$ **then**
    return a matching of H of size $\frac{n}{2} - k$;
return that G has no perfect matching;

The following lemma will be useful for showing that the players can make good progress.

▶ **Lemma 8.** *Let C and M be a Tutte-Berge cover and a matching on the same vertex set V, respectively. Then C covers at most $\delta(C)$ edges of M.*

In particular, if $|V| = n$ is even, M is a perfect matching, and $\delta(C) = \frac{n}{2} - k - 1$, then at least $k + 1$ edges of M invalidate C.

Proof. Let $C = (A, S_1, \ldots, S_t)$. By definition, C covers an edge only if it is incident to A or lies inside some S_i. Because M is a matching, at most $|A|$ edges of M are incident to A, and at most $\frac{|S_i|-1}{2}$ edges lie inside each S_i as $|S_i|$ is odd. Therefore, C can cover at most $|A| + \sum_{i=1}^{t} \frac{|S_i|-1}{2} = \delta(C)$ edges of M.

For the second claim, a perfect matching has $\frac{n}{2}$ edges, of which at most $\delta(C) = \frac{n}{2} - k - 1$ are covered. The remaining $\frac{n}{2} - \left(\frac{n}{2} - k - 1\right) = k + 1$ edges are uncovered by, and each invalidates C. ◀

We are now ready to prove the correctness of Algorithm 1 and analyze the communication cost of simulating it.

Proof of Lemma 5. Throughout Lemma 5, $\mathcal{C}$ is maintained to be exactly the set of Tutte-Berge covers of value $\frac{n}{2} - k - 1$ that cover the current graph H. We consider each of the algorithm's return statements.

Case 1. The algorithm returns at Line 7 in iteration i. Then $\mathcal{C} \neq \varnothing$, otherwise the algorithm would have returned at Line 11 in the previous iteration. Fix any $C \in \mathcal{C}$. We know C covers H, and since no edge in $G \setminus H$ invalidates C, C covers G. By Lemma 8, every matching of G has size at most $\delta(C) = \frac{n}{2} - k - 1 < \frac{n}{2}$. The algorithm correctly decides that G has no perfect matching.

Case 2. The algorithm returns at Line 11 in iteration i. Then H has no Tutte-Berge covers of value at most $\frac{n}{2} - k - 1$.[3] Consequently, by Theorem 7, the maximum matching size of $H \subseteq G$ is at least $\frac{n}{2} - k$. The algorithm correctly returns such a matching.

Case 3. The algorithm returns at Line 12 after T iterations. Suppose for contradiction that G has a perfect matching M.

Let $\mathcal{C}_i$ denote $\mathcal{C}$ at the start of iteration i. Then $\mathcal{C}_i \neq \varnothing$ for all $i = 0, \ldots, T-1$. Fix any i. By Lemma 8, each $C \in \mathcal{C}_i$ is invalidated by at least $k + 1$ edges of M. By double counting and averaging, some edge $e \in M$ invalidates at least a $\frac{2k}{n}$-fraction of the covers in $\mathcal{C}_i$.

The algorithm greedily chooses the next edge, which eliminates at least this many covers, so $|\mathcal{C}_{i+1}| \leq \left(1 - \frac{2k}{n}\right) |\mathcal{C}_i|$. Since $|\mathcal{C}_0| \leq (n+1)^n$, after T iterations

$$|\mathcal{C}_T| \leq \left(1 - \frac{2k}{n}\right)^T (n+1)^n \leq \exp\left(-\frac{2kT}{n}\right)(n+1)^n < 1,$$

using $1 - x \leq e^{-x}$ and our choice of $T = \left\lceil \frac{n^2 \ln n}{k} \right\rceil$. This contradicts $\mathcal{C}_T \neq \varnothing$. Therefore, the algorithm correctly concludes that G has no perfect matching.

Finally, we analyze the communication cost. Since H and $\mathcal{C}$ are public, each player can locally compute how many covers in $\mathcal{C}$ each of their own edge invalidates. In each iteration, Alice sends her best edge and Bob sends his, and they take whichever invalidates

[3] For a Tutte-Berge cover C, we can increase $\delta(C)$ by 1 without reducing its edge coverage, e.g., by moving a size-1 odd set into A or moving two vertices from a larger odd set into A.

more covers (breaking ties by taking Alice's edge). Each iteration requires $O(\log n)$ bits of communication. There are $T = O\left(\frac{n^2 \log n}{k}\right)$ iterations, so the total communication is $O(T \log n) = O\left(\frac{n^2 \log^2 n}{k}\right)$ bits. ◀

3 A Cutting-Plane Framework for Polytope Emptiness Testing

In this section, we describe a cutting-plane framework for testing whether a polytope is empty. The core idea is closely related to classical optimization techniques such as the center-of-gravity and ellipsoid methods. We do not claim novelty for this framework, as we distill it from [6], where it was used to obtain a near-linear communication protocol for bipartite matching. By abstracting away the problem-specific aspects of bipartite matching, we present it as a general framework that can be applied to other communication problems.

Consider the following "emptiness-testing" problem: given a convex polytope $P \subseteq \mathbb{R}^n$ via a separation oracle, we are promised that either $P = \varnothing$ or $\mathrm{vol}(P) \geq \tau$ for some known volume threshold $\tau > 0$. The goal is to determine which of the two cases holds using a small number of queries to the separation oracle of P.

Algorithm 2 A cutting-plane algorithm for polytope emptiness testing.

Input: Access to a separation oracle for a convex polytope $P \subseteq \mathbb{R}^n$, a convex polytope $P_0 \supseteq P$ with known volume $\mathrm{vol}(P_0) > 0$, and a volume threshold $0 < \tau < \mathrm{vol}(P_0)$. It is promised that either $P = \varnothing$ or $\mathrm{vol}(P) \geq \tau$.

Output: Decide whether $P = \varnothing$ or $\mathrm{vol}(P) \geq \tau$. In the latter case, output a point $p \in P$.

$T \leftarrow \lceil 3 \ln(\mathrm{vol}(P_0)/\tau) \rceil$;
for $i = 0, \ldots, T-1$ **do**
  $p_i \leftarrow$ center of mass of P_i;
  Query the separation oracle for P at p_i: it either asserts that $p_i \in P$, or returns a linear constraint $\langle a^{(i)}, x \rangle \leq b^{(i)}$ that is satisfied by every point of P but violated by p_i;
  if *the oracle asserts that* $p_i \in P$ **then**
    return that $\mathrm{vol}(P) \geq \tau$ and $p_i \in P$;
  $P_{i+1} \leftarrow P_i \cap \{x : \langle a^{(i)}, x \rangle \leq b^{(i)}\}$;
  if $P_{i+1} = \varnothing$ **then**
    return that $P = \varnothing$;
return that $P = \varnothing$;

▶ **Lemma 9.** *Let $P_0 \subseteq \mathbb{R}^n$ be a convex polytope with volume $\mathrm{vol}(P_0) > 0$. Let $0 < \tau < \mathrm{vol}(P_0)$. Let $P \subseteq P_0$ be a convex polytope with the promise that either $P = \varnothing$ or $\mathrm{vol}(P) \geq \tau$. Then Algorithm 2 correctly distinguishes between the two cases after making at most $O(\log(\mathrm{vol}(P_0)/\tau))$ queries to the separation oracle for P. Moreover, in the case $\mathrm{vol}(P) \geq \tau$, Algorithm 2 also returns a point $p \in P$.*

We defer the proof of Lemma 9 to Appendix A. Lemma 9 suggests the following general recipe for solving a graph problem $\mathcal{T}$ via polytope emptiness testing:

1. Reformulate a relevant (sub)problem as testing whether a convex polytope P is empty or has volume at least τ, for appropriate choices of P and τ.

2. Show that a separation oracle for P can be efficiently simulated in the communication model.
3. Apply Algorithm 2 to distinguish between $P = \varnothing$ and $\mathrm{vol}(P) \geq \tau$.
4. Translate the outcome back to the original problem $\mathcal{T}$.

This recipe is particularly well-suited for the two-party model. Primal LP formulations of many graph problems (e.g., bipartite matching or shortest paths) have variables corresponding to edges, which are not jointly known by the players. The key observation is that the dual LPs often have one *constraint* per edge. In many cases, each player can locally check the constraints corresponding to their own edges, and a violated constraint can be encoded by a short message.

As concrete examples, [6] applied this framework to a fractional vertex-cover polytope (the dual of fractional matching). By Kőnig's theorem, this polytope is empty if and only if the input graph has a perfect matching (see Section 5.1). In Section 4, we apply the same framework to a vertex-potential polytope (the dual of shortest paths) to obtain new communication protocols for negative cycle detection and negative-weight SSSP.

4 Negative Cycle Detection and Negative-Weight Shortest Paths

4.1 Negative Cycle Detection

The problem of *negative cycle detection* (NegativeCycle) is defined as follows: Alice and Bob are given a disjoint edge partition of a directed weighted graph $G = (V, E, w)$ with integer edge weights in $[-W, W]$. Their goal is to decide whether G has a negative cycle.

Let $n = |V|$. We consider the following vertex-potential polytope, whose constraints encode approximate triangle inequalities.

$$P^{\triangle}(G) := \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} x_v - x_u \leq w(u,v) + \frac{1}{n^2} & \forall (u,v) \in E \\ 0 \leq x_v \leq nW & \forall v \in V \end{array} \right\}. \tag{1}$$

The additive $\frac{1}{n^2}$ slack will allow us to lower bound the volume of $P^{\triangle}(G)$ when it is nonempty. At the same time, for integer edge weights, this slack does not affect negative-cycle detection: we show that $P^{\triangle}(G)$ is empty if and only if G has a negative cycle.

▶ **Lemma 10.** *Let $G = (V, E, w)$ be a directed weighted graph on $n \geq 2$ vertices with integer edge weights in $[-W, W]$. Let $P^{\triangle}(G)$ be the polytope defined in Equation* (1)*.*

1. *If G has a negative cycle, then $P^{\triangle}(G) = \varnothing$.*
2. *If G has no negative cycles, then $\mathrm{vol}\big(P^{\triangle}(G)\big) \geq n^{-2n}$.*

Proof. We first prove (1). Suppose for contradiction that $P^{\triangle}(G) \neq \varnothing$, and let $x \in P^{\triangle}(G)$. Let $(v_1, \ldots, v_k)$ be a negative cycle in G where $k \leq n$. Then for each $i \in [k]$ (with $v_{k+1} = v_1$), we have $x_{v_{i+1}} - x_{v_i} \leq w(v_i, v_{i+1}) + \frac{1}{n^2}$. Since the cycle is negative and all edge weights are integers, $\sum_{i \in [k]} w(v_i, v_{i+1}) \leq -1$. Summing over $i \in [k]$, we have $0 = \sum_{i \in [k]} (x_{v_{i+1}} - x_{v_i}) \leq \sum_{i \in [k]} w(v_i, v_{i+1}) + \frac{k}{n^2} \leq -1 + \frac{1}{n} < 0$, which is a contradiction.

We now prove (2). When G has no negative cycles, we construct a polytope $Q \subseteq P^{\triangle}(G)$ with volume n^{-2n}. Let G' be the graph obtained from G by adding a new vertex s' and directed edges (s', v) of weight $(n-1)W$ for all $v \in V$. Since G' also has no negative cycles, shortest-path distances from s' are well-defined. Let $d(v) := d_{G'}(s', v)$ for each $v \in V$.

We note two properties of $d(\cdot)$. First, $d(\cdot)$ satisfies the triangle inequality: $d(v) \leq d(u) + w(u, v)$ for all $(u, v) \in E$. Second, $d(v) \in [0, (n-1)W]$ for all $v \in V$. The upper bound follows from the direct edge (s', v), and for the lower bound, there is a shortest path $(s', v_1, \ldots, v_\ell = v)$ from s' to v with $\ell \leq n$, so $d(v) = (n-1)W + \sum_{i=1}^{\ell-1} w(v_i, v_{i+1}) \geq (n-1)W + (\ell-1)(-W) \geq 0$.

Consider the n-dimensional cube $Q := \prod_{v \in V} \left[d(v), d(v) + \frac{1}{n^2}\right]$. We can verify that $Q \subseteq P^{\triangle}(G)$. Fix any $q = (q_v)_{v \in V} \in Q$. We have $0 \le d(v) \le q_v \le d(v) + \frac{1}{n^2} \le (n-1)W + \frac{1}{n^2} \le nW$. Moreover, for all $(u, v) \in E$, $q_v - q_u \le d(v) + \frac{1}{n^2} - d(u) \le w(u, v) + \frac{1}{n^2}$ by the triangle inequality for $d(\cdot)$. Therefore, $\mathrm{vol}\big(P^{\triangle}(G)\big) \ge \mathrm{vol}(Q) = n^{-2n}$. ◀

We now present our protocol for NegativeCycle. Note that when G has no negative cycles, the protocol outputs a vector $p \in P^{\triangle}(G)$, which we will use in Section 4.2 to solve NegativeSSSP.

▶ **Theorem 11.** *There is a deterministic protocol $\mathcal{P}^{\mathsf{NegativeCycle}}$ that solves* NegativeCycle *using $O(n \log^2(nW))$ bits of communication. Moreover, if the input graph G has no negative cycles, the protocol outputs a vector $p \in P^{\triangle}(G)$, where $P^{\triangle}(G)$ is defined in Equation* (1)*.*

Proof. By Lemma 10, it suffices to distinguish whether $P^{\triangle}(G) = \varnothing$ or $\mathrm{vol}\big(P^{\triangle}(G)\big) \ge n^{-2n}$. Alice and Bob can do this by simulating Algorithm 2 with $P = P^{\triangle}(G)$, $P_0 = [0, nW]^n$, and $\tau = n^{-2n}$.

Note that P_0 is public and independent of G. Since P_i is entirely determined by P_0 and previously communicated constraints, both players know P_i and can locally compute the center of mass p_i without communication. The constraints $0 \le x_v \le nW$ are already captured by P_0. To simulate the separation oracle for P at p_i, Alice and Bob only need to check the edge constraints $x_v - x_u \le w(u, v) + \frac{1}{n^2}$, each of which can be checked locally by the player holding the edge (u, v). Both players send a violated constraint if they have one (or a *null* message otherwise). If both players send a violated constraint, they break ties by using Alice's. If neither player sends a violated constraint, the oracle asserts $p_i \in P$.

By Lemma 9, this protocol correctly distinguishes between the two promised cases after $O(\log(\mathrm{vol}(P_0)/\tau)) = O(\log(n^{3n} W^n)) = O(n \log(nW))$ queries to the separation oracle, and outputs a point $p \in P^{\triangle}(G)$ when $P^{\triangle}(G) \ne \varnothing$. Each query to the separation oracle requires each player to communicate at most one constraint, and each constraint can be encoded using $O(\log(nW))$ bits (i.e., an edge (u, v) and its weight). Therefore, the total communication is $O(n \log^2(nW))$ bits. ◀

4.2 Negative-Weight Single-Source Shortest Paths

The problem of *negative-weight single-source shortest paths* (NegativeSSSP) is defined as follows: Alice and Bob are given a disjoint edge partition of a directed weighted graph $G = (V, E, w)$ with integer edge weights in $[-W, W]$, and a source vertex $s \in V$. Their goal is to either decide that G has a negative cycle or compute the single-source shortest-path distances $d_G(s, v)$ for all $v \in V$.

We use a vertex-potential reweighting idea, familiar from Johnson's algorithm for all-pairs shortest paths [14]. For a function $\phi : V \to \mathbb{R}$, let

$$w_\phi(u, v) := w(u, v) + \phi(u) - \phi(v), \qquad \forall (u, v) \in E,$$

and let $G_\phi := (V, E, w_\phi)$. Along any path from s to v, the $\phi(\cdot)$ terms telescope. Taking the minimum over all s-v paths shows

$$d_{G_\phi}(s, v) = d_G(s, v) + \phi(s) - \phi(v) \qquad \forall v \in V.$$

In particular, if $w_\phi(e) \ge 0$ for every edge $e \in E$, one can compute $d_{G_\phi}(s, \cdot)$ using any nonnegative-weight shortest-path algorithm and recover $d_G(s, v)$.

In our application, setting ϕ to the vector $p \in P^{\triangle}(G)$ returned by Theorem 11 only guarantees $w_\phi(u, v) \ge -\frac{1}{n^2}$. To obtain nonnegative edge weights, we round $w_\phi(u, v)$ up to the nearest multiple of $\frac{1}{n}$. This rounding has only a mild effect on shortest-path distances:

it increases the weight of any single edge by at most $\frac{1}{n}$ and the weight of any simple path by at most $\frac{n-1}{n} < 1$. Since $d_G(s,v)$ is an integer, we can then run a nonnegative-weight shortest-path algorithm and recover the original distances by taking floors at the end.[4]

▶ **Lemma 12** (Reweighting with approximate potentials). *Let $G = (V, E, w)$ be a directed graph on $n \geq 2$ vertices with integer edge weights and no negative cycles. Suppose $\phi : V \to \mathbb{R}$ satisfies $w(u,v) + \phi(u) - \phi(v) \geq -\frac{1}{n^2}$ for all $(u,v) \in E$. Let $\widetilde{G} = (V, E, \widetilde{w})$, where*

$$\widetilde{w}(u,v) := \frac{1}{n}\left\lceil n\big(w(u,v) + \phi(u) - \phi(v)\big)\right\rceil \qquad \forall (u,v) \in E.$$

Then, $\widetilde{w}(u,v) \geq 0$ for every $(u,v) \in E$. Moreover, G and $\widetilde{G}$ have the same reachability relation, and for every vertex $v \in V$ reachable from s,

$$d_G(s,v) = \left\lfloor d_{\widetilde{G}}(s,v) + \phi(v) - \phi(s)\right\rfloor.$$

Proof. Let $w_\phi(u,v) = w(u,v) + \phi(u) - \phi(v)$ and $G_\phi = (V, E, w_\phi)$. Since $w_\phi(u,v) \geq -\frac{1}{n^2}$, we have $nw_\phi(u,v) \geq -\frac{1}{n} > -1$. Therefore, $\lceil nw_\phi(u,v)\rceil \geq 0$ and $\widetilde{w}(u,v) \geq 0$.

Because $\widetilde{G}$, G_ϕ, and G all share the same edge set E, they share the same reachability relation.

Fix any $v \in V$ reachable from s. For every $e \in E$, $\widetilde{w}(e)$ rounds up $w_\phi(e)$ to the nearest multiple of $\frac{1}{n}$, so

$$w_\phi(e) \leq \widetilde{w}(e) < w_\phi(e) + \frac{1}{n}. \tag{2}$$

Applying the left inequality of Equation (2) to a shortest s-v path in $\widetilde{G}$ gives

$$d_{G_\phi}(s,v) \leq d_{\widetilde{G}}(s,v).$$

Since reweighting preserves cycle weights and G has no negative cycles, G_ϕ also has no negative cycles. Therefore, there is a shortest simple s-v path in G_ϕ with at most $n-1$ edges. Applying the right inequality of Equation (2) to this path gives

$$d_{\widetilde{G}}(s,v) < d_{G_\phi}(s,v) + \frac{n-1}{n} < d_{G_\phi}(s,v) + 1.$$

Combining these inequalities with $d_{G_\phi}(s,v) = d_G(s,v) + \phi(s) - \phi(v)$, we have

$$d_G(s,v) \leq d_{\widetilde{G}}(s,v) + \phi(v) - \phi(s) < d_G(s,v) + 1.$$

Since $d_G(s,v)$ is an integer, taking the floor proves the lemma. ◀

We conclude this section with a deterministic protocol for NegativeSSSP.

▶ **Theorem 13.** *There is a deterministic protocol $\mathcal{P}^{\mathsf{NegativeSSSP}}$ that solves* NegativeSSSP *using $O(n \log^2(nW))$ bits of communication.*

Proof. The protocol proceeds as follows.

1. Run $\mathcal{P}^{\mathsf{NegativeCycle}}$ (Theorem 11) on G. If a negative cycle is detected, terminate. Otherwise, both players obtain the same point $p \in P^{\triangle}(G)$.

[4] Analogous rounding arguments appear in the near-linear-time algorithm of [4] for negative-weight SSSP.

2. Let $\phi(v) := p_v$ for all $v \in V$.
   Let $\widetilde{G} = (V, E, \widetilde{w})$ where $\widetilde{w}(u,v) := \frac{1}{n}\lceil n\,(w(u,v) + \phi(u) - \phi(v))\rceil$, as in Lemma 12. Since both players know p, the edge weight $\widetilde{w}(u,v)$ can be computed locally by the player holding the edge (u,v).
3. Run $\mathcal{P}^{\mathsf{SSSP}}$ (Theorem 19) to compute $d_{\widetilde{G}}(s,v)$ for all $v \in V$. Formally, $\mathcal{P}^{\mathsf{SSSP}}$ requires integer weights, so the players scale up $\widetilde{w}$ by a factor of n, run $\mathcal{P}^{\mathsf{SSSP}}$ on $(V, E, n\widetilde{w})$ to compute $n \cdot d_{\widetilde{G}}(s,v)$ for all $v \in V$, and then divide by n to get $d_{\widetilde{G}}(s,v)$.
4. Output $d_G(s,v) := \lfloor d_{\widetilde{G}}(s,v) + \phi(v) - \phi(s)\rfloor$ for every v reachable from s, and $d_G(s,v) := \infty$ otherwise.

For correctness, the constraints of $P^{\triangle}(G)$ as defined in Equation (1) guarantee that $p_v - p_u \le w(u,v) + \frac{1}{n^2}$, or equivalently, $w_\phi(u,v) \ge -\frac{1}{n^2}$, so ϕ satisfies the prerequisite of Lemma 12. By Lemma 12, the edge weights $\widetilde{w}(u,v)$ are nonnegative and multiples of $\frac{1}{n}$, which become nonnegative integers after scaling up by a factor of n. By Theorem 19, the players correctly compute $d_{\widetilde{G}}(s,v)$ via $\mathcal{P}^{\mathsf{SSSP}}$. The correctness of $\mathcal{P}^{\mathsf{NegativeSSSP}}$ then follows from Lemma 12.

It remains to analyze the communication cost of the protocol. In Step 1, $\mathcal{P}^{\mathsf{NegativeCycle}}$ requires $O(n\log^2(nW))$ bits of communication by Theorem 11. In Step 3, since the constraints of $P^{\triangle}(G)$ guarantee that $0 \le p_v \le nW$ for all $v \in V$, we have $w_\phi(u,v) = w(u,v) + p_u - p_v \le (n+1)W$, and hence the scaled weights $n\widetilde{w}(u,v)$ are nonnegative integers that are $O(n^2W)$, so $\mathcal{P}^{\mathsf{SSSP}}$ requires $O(n\log(nW))$ bits by Theorem 19. The total communication cost is $O(n\log^2(nW))$ bits. ◀

5 Maximum Matching in Bipartite Graphs

In this section, we first restate the $O(n\log^2 n)$-bit protocol in [6] for bipartite matching, and then we present our combinatorial protocol (resulted from discretizing the said work) in Section 5.2.

5.1 A Near-Linear Communication Protocol for Bipartite Matching

In the problem of *bipartite perfect matching* (BPM), Alice and Bob are given a disjoint edge partition of an undirected bipartite graph $G = (V, E)$. Their goal is to decide whether G admits a perfect matching and output one if it does. We assume without loss of generality that $n = |V|$ is even and that each side of G has exactly $\frac{n}{2}$ vertices. The breakthrough result of [6] gave a protocol for this problem with near-linear communication complexity.

▶ **Theorem 14** ([6]). *There is a deterministic protocol $\mathcal{P}^{\mathsf{BPM}}$ which solves* BPM *using $O(n\log^2 n)$ bits of communication.*

We briefly describe the protocol from [6], before sketching a proof for Theorem 14. Consider the following vertex-cover polytope of G:

$$P^{\mathrm{VC}}(G) := \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} x_u + x_v \ge 1 & \forall (u,v) \in E, \\ \sum_{v\in V} x_v \le \frac{n}{2} - \frac{1}{2}, & \\ 0 \le x_v \le 1 & \forall v \in V \end{array} \right\}. \tag{3}$$

The following lemma is from [6]. For completeness, we provide its proof in Appendix D.

▶ **Lemma 15** ([6]). *Let G be a bipartite graph with $\frac{n}{2}$ vertices on each side. Let $P^{\mathrm{VC}}(G)$ be the polytope defined in Equation* (3)*.*

1. *If G has a perfect matching, then $P^{\mathrm{VC}}(G) = \varnothing$.*
2. *If G has no perfect matching, then $\mathrm{vol}(P^{\mathrm{VC}}(G)) \geq (4n)^{-n}$.*

Proof sketch of Theorem 14. By Lemma 15, it is sufficient to decide whether $P^{\mathrm{VC}}(G) = \varnothing$ or $\mathrm{vol}(P^{\mathrm{VC}}(G)) \geq (4n)^{-n}$. Alice and Bob can do this by simulating Algorithm 2 with $P = P^{\mathrm{VC}}(G)$, $P_0 = [0,1]^n \cap \{x : \sum_v x_v \leq \frac{n}{2} - \frac{1}{2}\}$, and $\tau = (4n)^{-n}$.

Both players know P_0 and can compute the center of mass of each P_i without communication. To simulate the separation oracle for P, Alice and Bob only need to check the constraints $x_u + x_v \geq 1$, each of which can be checked locally by the player holding the edge (u, v). By Lemma 9, this protocol correctly distinguishes the two promised cases after $O(\log(\mathrm{vol}(P_0)/\tau)) = O(n \log n)$ queries to the separation oracle. Each query requires each player to communicate at most one constraint, and each constraint can be encoded using $O(\log n)$ bits (i.e., an edge (u, v)). Therefore, the total communication is $O(n \log^2 n)$ bits.

Let H be the graph formed by the communicated edges. If Algorithm 2 returns $P^{\mathrm{VC}}(G) = \varnothing$ because $\mathrm{vol}\big(P^{\mathrm{VC}}(H)\big) < (4n)^{-n}$, then by Lemma 15, H must have a perfect matching. ◀

5.2 A Combinatorial Protocol for Bipartite Perfect Matching

The cutting-plane framework (Algorithm 2) is inherently continuous: it operates over a polytope, tracks progress using its volume, and computes the center of mass to find a violated constraint that eliminates a constant fraction of the volume. In this section, we present a combinatorial protocol for BPM that avoids these continuous operations.

▶ **Theorem 16.** *There is a deterministic protocol $\mathcal{P}^{\mathsf{BPM}}$ that does not rely on the cutting-plane framework (Algorithm 2) and solves* BPM *using $O(n \log^2 n)$ bits of communication.*

Our idea is to discretize the vertex-cover polytope $P^{\mathrm{VC}}(G)$. Consider a set of fractional vertex covers $Q^{\mathrm{VC}}(G)$ defined as follows:

$$Q^{\mathrm{VC}}(G) := \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} x_u + x_v \geq 1 & \forall (u,v) \in E, \\ \sum_{v \in V} x_v \leq \frac{n}{2} - \frac{1}{2}, & \\ x_v \in \left\{0, \frac{1}{d}, \ldots, 1\right\} & \forall v \in V \end{array} \right\}. \tag{4}$$

We say that a nonnegative vector in $\mathbb{R}^n$ is *d-uniform* if each of its coordinates is a multiple of $\frac{1}{d}$. With this notation, $Q^{\mathrm{VC}}(G)$ is the set of d-uniform fractional vertex covers of G of size at most $\frac{n}{2} - \frac{1}{2}$. For sufficiently large $d = \mathrm{poly}(n)$, the discrete quantity $|Q^{\mathrm{VC}}(H)|$ is essentially proportional to $\mathrm{vol}(P^{\mathrm{VC}}(H))$ throughout the protocol. As the players communicate edges and add them to a shared public graph, we can use this discrete quantity as a proxy for volume to measure progress.

We formally state our protocol for BPM in Algorithm 3.

Algorithm 3 A combinatorial protocol for bipartite matching.

Input: A bipartite graph $G = (V, E)$ with $n \geq 2$ vertices.
Output: Decide whether G has a perfect matching, and output one if it does.

1 $d \leftarrow 8n^7$;
2 $H_0 \leftarrow (V, \varnothing)$;
3 $T \leftarrow \lceil 10\, n \ln(d+1) \rceil$;
4 **for** $i = 0, \ldots, T-1$ **do**
5 Find an edge $e_i \in (G \setminus H_i)$ such that $|Q^{\mathrm{VC}}(H_i \cup \{e_i\})| \leq 0.9 \cdot |Q^{\mathrm{VC}}(H_i)|$, where $Q^{\mathrm{VC}}(H)$ is the set of d-uniform fractional vertex covers of H as defined in Equation (4);
6 **if** *there is no such* e_i **then**
7 **return** that G has no perfect matching;
8 $H_{i+1} \leftarrow H_i \cup \{e_i\}$;
9 **if** $Q^{\mathrm{VC}}(H_{i+1}) = \varnothing$ **then**
10 **return** that G has a perfect matching, together with a perfect matching of H_{i+1};

▶ **Lemma 17** (Existence of a progress-making edge). *Let G be a bipartite graph with a perfect matching and $\frac{n}{2}$ vertices on each side. Let H be any subgraph of G on the same vertices with $Q^{\mathrm{VC}}(H) \neq \varnothing$, where $Q^{\mathrm{VC}}(H)$ is defined in Equation (4) for $d = 8n^7$. Then, there is an edge $e \in (G \setminus H)$ such that*

$$\left|Q^{\mathrm{VC}}(H \cup \{e\})\right| \leq 0.9 \cdot \left|Q^{\mathrm{VC}}(H)\right|.$$

We defer the proof of Lemma 17 to Appendix D.1 and first use it to prove Theorem 16.

Proof of Theorem 16. We first show that Algorithm 3 always returns an answer. Suppose for contradiction that it finishes all T iterations without reaching Line 7 or Line 10. Then

$$\left|Q^{\mathrm{VC}}(H_T)\right| \leq (0.9)^T \cdot \left|Q^{\mathrm{VC}}(H_0)\right| \leq (0.9)^T \cdot (d+1)^n < 1,$$

where the last inequality follows from our choice of $T = \lceil 10\, n \ln(d+1) \rceil$. This implies $\left|Q^{\mathrm{VC}}(H_T)\right| = 0$, but then the algorithm would have returned in Line 10 of the final iteration, which is a contradiction. We proceed by verifying correctness at the algorithm's return statements.

Case 1. Algorithm 3 returns at Line 7 in iteration i. Then $Q^{\mathrm{VC}}(H_i) \neq \varnothing$: this is true for $i = 0$, and for $i > 0$ the algorithm would have returned at Line 10 in the previous iteration. If G had a perfect matching, Lemma 17 would guarantee the existence of an edge e_i satisfying the required condition. Hence G has no perfect matching, and the algorithm correctly decides this.

Case 2. Algorithm 3 returns at Line 10 in iteration i. Then $Q^{\mathrm{VC}}(H_{i+1}) = \varnothing$. Observe that any integral vertex cover of H_{i+1} of size $\frac{n}{2} - 1$ is a feasible point in $Q^{\mathrm{VC}}(H_{i+1})$. Therefore, H_{i+1} has no vertex cover of size $\frac{n}{2} - 1$. By Kőnig's theorem, H_{i+1} has a perfect matching. Because $H_{i+1} \subseteq G$, this is also a perfect matching of G, which the algorithm outputs.

We now analyze the communication cost of simulating Algorithm 3 in the two-party setting. Note that each H_i is public. In each iteration i, both players locally check if they have an edge $e \notin H_i$ satisfying $\left|Q^{\mathrm{VC}}(H_i \cup \{e\})\right| \leq 0.9 \cdot \left|Q^{\mathrm{VC}}(H_i)\right|$, and send such an edge if they have one. If both players send an edge, they break ties by using Alice's. If neither

player sends an edge, the algorithm returns at Line 7. The agreed-upon edge is then added to form H_{i+1}, and both players can locally check whether $Q^{\mathrm{VC}}(H_{i+1}) = \varnothing$. Therefore, each iteration requires each player to communicate at most one edge, which can be encoded using $O(\log n)$ bits. Since $d = 8n^7$ and $T = \lceil 10\, n \ln(d+1) \rceil = O(n \log n)$, the total communication is $O(n \log^2 n)$ bits. ◀

References

1 Simon Apers, Yuval Efron, Pawel Gawrychowski, Troy Lee, Sagnik Mukhopadhyay, and Danupon Nanongkai. Cut query algorithms with star contraction. In *Proceedings of the 63rd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 507–518, 2022. doi:10.1109/FOCS54457.2022.00055.

2 Sepehr Assadi, Amit Chakrabarti, Prantar Ghosh, and Manuel Stoeckl. Coloring in graph streams via deterministic and adversarially robust algorithms. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*, pages 141–153, 2023. doi:10.1145/3584372.3588681.

3 Claude Berge. Sur le couplage maximum d'un graphe. *Comptes Rendus de l'Académie des Sciences*, 247:258–259, 1958.

4 Aaron Bernstein, Danupon Nanongkai, and Christian Wulff-Nilsen. Negative-weight single-source shortest paths in near-linear time. In *Proceedings of the 63rd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 600–611, 2022. doi:10.1109/FOCS54457.2022.00063.

5 Joakim Blikstad, Yonggang Jiang, Sagnik Mukhopadhyay, and Sorrachai Yingchareonthawornchai. Global vs. s-t vertex connectivity beyond sequential: Almost-perfect reductions and near-optimal separations. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing (STOC)*, pages 2305–2316, 2025. doi:10.1145/3717823.3718316.

6 Joakim Blikstad, Jan van den Brand, Yuval Efron, Sagnik Mukhopadhyay, and Danupon Nanongkai. Nearly optimal communication and query complexity of bipartite matching. In *Proceedings of the 63rd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1174–1185, 2022. doi:10.1109/FOCS54457.2022.00113.

7 Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

8 Maxime Flin and Parth Mittal. $(\Delta + 1)$ vertex coloring in $O(n)$ communication. *Distributed Computing*, 38(1):19–29, 2025. doi:10.1007/S00446-024-00475-3.

9 Harold N. Gabow. The weighted matching approach to maximum cardinality matching. *Fundamenta Informaticae*, 154(1–4):109–130, 2017. doi:10.3233/FI-2017-1555.

10 Hossein Gholizadeh and Yonggang Jiang. A subquadratic two-party communication protocol for minimum cost flow. *CoRR*, abs/2510.03427, 2025. doi:10.48550/arXiv.2510.03427.

11 Branko Grünbaum. Partitions of mass-distributions and convex bodies by hyperplanes. *Pacific Journal of Mathematics*, 10(4):1257–1261, 1960.

12 András Hajnal, Wolfgang Maass, and György Turán. On the communication complexity of graph properties. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 186–191, 1988. doi:10.1145/62212.62228.

13 Yonggang Jiang, Danupon Nanongkai, and Pachara Sawettamalya. Minimum s-t cuts with fewer cut queries. In *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA)*, pages 258–296, 2026. doi:10.1137/1.9781611978971.12.

14 Donald B. Johnson. Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM*, 24(1):1–13, 1977. doi:10.1145/321992.321993.

15 Yotam Kenneth-Mordoch and Robert Krauthgamer. Cut-query algorithms with few rounds. In *Proceedings of the 33rd Annual European Symposium on Algorithms (ESA)*, pages 100:1–100:14, 2025. doi:10.4230/LIPIcs.ESA.2025.100.

16 Yotam Kenneth-Mordoch and Robert Krauthgamer. Faster all-pairs minimum cut: Bypassing exact max-flow. In *Proceedings of the 58th Annual ACM Symposium on Theory of Computing, (STOC)*, pages 1254–1265, 2026. doi:10.1145/3798129.3800836.

17 Silvio Micali and Vijay V. Vazirani. An $O(\sqrt{|V|}\,|E|)$ algorithm for finding maximum matching in general graphs. In *Proceedings of the 21st IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 17–27, 1980.

18 Hiroshi Nagamochi and Toshihide Ibaraki. A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica*, 7(5–6):583–596, 1992. doi:10.1007/BF01758778.

19 Aviad Rubinstein, Tselil Schramm, and Matthew S. Weinberg. Computing exact minimum cuts without knowing the graph. In *Proceedings of the 9th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 39:1–39:16, 2018. doi:10.4230/LIPIcs.ITCS.2018.39.

20 W. T. Tutte. The factorization of linear graphs. *Journal of the London Mathematical Society*, 22(2):107–111, 1947.

21 Jan van den Brand, Yin Tat Lee, Danupon Nanongkai, Richard Peng, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Bipartite matching in nearly-linear time on moderately dense graphs. In *Proceedings of the 61st IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 919–930, 2020. doi:10.1109/FOCS46700.2020.00090.

22 Jan van den Brand, Zhao Song, and Albert Weng. Computing flows in subquadratic space. *CoRR*, abs/2605.09547, 2026. doi:10.48550/arXiv.2605.09547.

23 Andrew Chi-Chih Yao. Some complexity questions related to distributed computing. In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing (STOC)*, pages 209–213, 1979.

A Omitted Proofs from Section 3

Proof of Lemma 9. Algorithm 2 runs for at most T iterations and queries the separation oracle exactly once per iteration, so it makes at most $T = \lceil 3\ln(\mathrm{vol}(P_0)/\tau)\rceil = O(\log(\mathrm{vol}(P_0)/\tau))$ queries in total. It remains to show that its output is always correct.

Note that $P \subseteq P_i$ for every $0 \le i \le T$. This holds initially by the assumption $P \subseteq P_0$, and continues to hold because P_{i+1} is formed by intersecting P_i with a halfspace that contains every point of P. We proceed by verifying correctness at each of the algorithm's three return statements.

Case 1. Algorithm 2 returns at Line 6 when the separation oracle asserts that $p_i \in P$. Then $P \neq \varnothing$, and by the promise on P, the algorithm correctly decides $\mathrm{vol}(P) \ge \tau$ and returns a point $p_i \in P$.

Case 2. Algorithm 2 returns at Line 9 in iteration i. Then $P \subseteq P_{i+1} = \varnothing$, and the algorithm correctly decides $P = \varnothing$.

Case 3. Algorithm 2 returns at Line 10 after T iterations. Then $P_T \neq \varnothing$. Because

$$P_{i+1} = P_i \cap \{x : \langle a^{(i)}, x\rangle \le b^{(i)}\}$$

and the center of mass p_i of P_i violates this constraint, by Grünbaum's Theorem [11][5], we have

$$\mathrm{vol}(P_{i+1}) \le \left(1 - \frac{1}{e}\right)\mathrm{vol}(P_i).$$

[5] Specifically, Grünbaum [11] showed that for any convex body K and any halfspace H that does not contain the center of mass of K, $\mathrm{vol}(K \cap H) \le \left(1 - \frac{1}{e}\right)\mathrm{vol}(K)$.

By iteratively applying this inequality, we get

$$\mathrm{vol}(P_T) \leq \left(1 - \frac{1}{e}\right)^T \mathrm{vol}(P_0) \leq e^{-T/e} \cdot \mathrm{vol}(P_0) < \tau,$$

where we used $1 - x \leq e^{-x}$, $\mathrm{vol}(P_0)/\tau > 1$, and $T \geq 3\ln(\mathrm{vol}(P_0)/\tau) > e\ln(\mathrm{vol}(P_0)/\tau)$. Because $P \subseteq P_T$, we have $\mathrm{vol}(P) \leq \mathrm{vol}(P_T) < \tau$. By the promise on P, we must have $P = \varnothing$, and the algorithm correctly decides that. ◀

B Directed Cycle Detection

The problem of *directed cycle detection* (Cycle) is defined as follows: the players are given an edge partition of a directed graph $G = (V, E)$ without self-loops and with $n = |V|$. Their goal is to determine whether G contains a directed cycle.

▶ **Theorem 18.** *There is a deterministic protocol $\mathcal{P}^{\mathsf{Cycle}}$ which solves* Cycle *using $O(n \log n)$ bits of communication.*

We simulate Kosaraju's algorithm to compute all strongly connected components (SCCs) of G. Recall that Kosaraju's algorithm performs two depth-first searches (DFS). We first discuss how to simulate DFS on a directed graph with a specific vertex ordering (Algorithm 4).

Algorithm 4 Depth-First Search (DFS).

Input: A directed graph $G = (V, E)$ and an ordering $L = (v_1, \ldots, v_n)$ of V.
Output: Runs a DFS on G starting from vertices in the order L.

```
1  Function Search(v)
2      Visited[v] ← true;
3      for each edge (v, u) ∈ E do
4          if Visited[u] = false then
5              Search(u);
6      return;
7  Visited[v] ← false for all v ∈ V;
8  for i = 1, ..., n do
9      if Visited[v_i] = false then
10         Search(v_i);
```

In Line 3, we simulate the loop over v's edges as follows: Alice repeatedly checks her remaining (unchecked) edges (v, u) for one with u unvisited; each time she finds one, she sends it to Bob, and both proceed to the recursive `Search` call in Line 5 before returning to check her remaining edges. Once she has exhausted all her edges leaving v, she sends a single "done" message, and then it is Bob's turn to do the same with his edges. Once Bob exhausts his edges and sends "done", they backtrack at Line 6. Both players maintain a correct copy of the Visited array. For each vertex, each player sends exactly one "done" message. Each communicated edge leads to an unvisited vertex, which is immediately marked as visited, so at most $n - 1$ edges are communicated in total. Each edge requires $O(\log n)$ bits to encode, so the total communication for one DFS is $O(n \log n)$ bits.

We now build on this to discuss how to simulate Kosaraju's algorithm.

Proof of Theorem 18. We simulate Kosaraju's algorithm: Perform a first DFS on G with an arbitrary vertex ordering, and push a vertex v onto a stack immediately before `Search`(v) returns at Line 6. Then perform a second DFS on the transpose graph $G^\top$ (where the direction of each edge is reversed, and each player can locally reverse their edges) using the stack order (last-in-first-out, i.e., in decreasing order of their finish times from the first DFS) as the vertex ordering L. All vertices visited within a single outer call to `Search` in Line 10 in the second DFS form one SCC.

Note that G has a directed cycle if and only if there is at least one SCC of size at least 2. Therefore, after computing all SCCs, Alice and Bob can decide Cycle without further communication. In total, the protocol simulates two DFS traversals and uses $O(n \log n)$ bits of communication. ◀

C Nonnegative-Weight Single-Source Shortest Paths

The problem of *single-source shortest paths* (SSSP) is as follows: the players are given a disjoint edge partition of a directed weighted graph $G = (V, E, w)$ whose weights are integers in $[0, W]$, and a source vertex $s \in V$. Their goal is to compute the shortest path distance $d_G(s, v)$ from s to every vertex $v \in V$.

▶ **Theorem 19.** *There is a deterministic protocol $\mathcal{P}^{\mathsf{SSSP}}$ that computes* SSSP *using $O(n \log(nW))$ bits of communication.*

We show that Dijkstra's algorithm can be simulated using $O(n \log(nW))$ bits of communication. For simplicity, we first recall a basic version of Dijkstra's algorithm without a priority queue.

Algorithm 5 Dijkstra's Algorithm (without a priority queue).

Input: A directed weighted graph $G = (V, E, w)$ and a source vertex s.
Output: The shortest-path distance $d(v)$ from s to every $v \in V$.

1 $S \leftarrow \{s\}$;
2 $d(s) \leftarrow 0$;
3 $d(v) \leftarrow +\infty$ for each $v \in V \setminus \{s\}$;
4 **for** $i = 1, \ldots, n-1$ **do**
5 **for** $v \in (V \setminus S)$ **do**
6 Compute $\widetilde{d}(v) := \min_{u \in S, (u,v) \in E} \{d(u) + w(u, v)\}$ (or $\widetilde{d}(v) := +\infty$ if no such u exists);
7 $v_i \leftarrow \arg\min_{v \in V \setminus S} \widetilde{d}(v)$ (breaking ties lexicographically);
8 $d(v_i) \leftarrow \widetilde{d}(v_i)$;
9 $S \leftarrow S \cup \{v_i\}$;
10 **return** $d(v)$ for all $v \in V$;

Proof of Theorem 19. Let E_A and E_B denote the sets of edges held by Alice and Bob, respectively. They maintain a consistent copy of the set S and the array d, and simulate Algorithm 5 by replacing Lines 5–8 with the following process: In iteration i,

1. Alice locally computes $\widetilde{d}_A(v) := \min_{u \in S, (u,v) \in E_A} \{d(u) + w(u, v)\}$ for each vertex $v \in V \setminus S$, where $\widetilde{d}_A(v) := +\infty$ if no such u exists. Alice sends the vertex v_i^A minimizing $\widetilde{d}_A(v)$ (breaking ties lexicographically) and the corresponding $\widetilde{d}_A(v_i^A)$ to Bob. Bob does the same and sends v_i^B and $\widetilde{d}_B(v_i^B)$ to Alice.

2. Alice and Bob then agree on $v_i \in \{v_i^A, v_i^B\}$ based on which of $\widetilde{d}_A(v_i^A)$ and $\widetilde{d}_B(v_i^B)$ is smaller (again breaking ties lexicographically).
3. Then they set $d(v_i) := \min(\widetilde{d}_A(v_i^A), \widetilde{d}_B(v_i^B))$. (Note that it is possible that $v_i^A = v_i^B$ but $\widetilde{d}_A(v_i^A) \neq \widetilde{d}_B(v_i^B)$.)

This produces the same choice of v_i and the same value $d(v_i)$ as Lines 5–8, because the minimum candidate over all edges from S to $V \setminus S$ is the minimum of Alice's best candidate and Bob's best candidate. Since edge weights lie in $[0, W]$, all finite distances are at most $(n-1)W$, so each distance value can be encoded using $O(\log(nW))$ bits, where $+\infty$ can be represented using a special symbol. Each iteration therefore uses $O(\log n + \log(nW)) = O(\log(nW))$ bits of communication, and there are $(n-1)$ iterations. Hence, the total communication is $O(n \log(nW))$ bits. ◀

D Omitted Proofs from Section 5

▶ **Lemma 15** ([6]). *Let G be a bipartite graph with $\frac{n}{2}$ vertices on each side. Let $P^{\mathrm{VC}}(G)$ be the polytope defined in Equation (3).*

1. *If G has a perfect matching, then $P^{\mathrm{VC}}(G) = \varnothing$.*
2. *If G has no perfect matching, then $\mathrm{vol}(P^{\mathrm{VC}}(G)) \geq (4n)^{-n}$.*

Proof. Recall that in a bipartite graph, the sizes of the (integral) maximum matching and minimum vertex cover are the same. Moreover, the integrality gaps of the maximum matching and minimum vertex cover LPs are both 1.

If G has a perfect matching, then the minimum vertex cover of G has size $\frac{n}{2}$. Because any feasible point in $P^{\mathrm{VC}}(G)$ is a fractional vertex cover of size at most $\frac{n}{2} - \frac{1}{2}$, $P^{\mathrm{VC}}(G)$ must be empty.

If G has no perfect matching, then G has an integral vertex cover $y \in \{0,1\}^n$ of size $\frac{n}{2} - 1$. We have $\sum y_v \leq \frac{n}{2} - 1$ and $y_u + y_v \geq 1$ for all $(u, v) \in E$. We show that $B \subseteq P^{\mathrm{VC}}(G)$ where

$$B = \prod_{v \in V} \begin{cases} \left[\frac{1}{4n}, \frac{1}{2n}\right] & \text{if } y_v = 0, \\ \left[1 - \frac{1}{4n}, 1\right] & \text{if } y_v = 1. \end{cases}$$

Fix any $q \in B \subseteq [0,1]^n$. For any edge $(u, v) \in E$, at least one of y_u or y_v is 1, so $q_u + q_v \geq \frac{1}{4n} + \left(1 - \frac{1}{4n}\right) = 1$. Moreover, $\sum_v q_v \leq \sum_v \left(y_v + \frac{1}{2n}\right) = \left(\frac{n}{2} - 1\right) + \frac{1}{2} = \frac{n}{2} - \frac{1}{2}$. This shows that $q \in P^{\mathrm{VC}}(G)$. Therefore, $B \subseteq P^{\mathrm{VC}}(G)$ and $\mathrm{vol}(P^{\mathrm{VC}}(G)) \geq \mathrm{vol}(B) = (4n)^{-n}$ as claimed. ◀

D.1 Proof of Lemma 17

We first introduce the notion of a *balanced polytope*.

▶ **Definition 20** (β-balanced polytope). *Let $P \subseteq \mathbb{R}^n$ be a polytope. We say that P is β-balanced if there exists a point $p \in P$ such that $d(p, F) \geq \beta$ for all facets F of P, where $d(p, F)$ is the distance from p to the supporting hyperplane of F.*

Given a bipartite graph $G = (V, E)$, recall that we define $P^{\mathrm{VC}}(G)$ and $Q^{\mathrm{VC}}(G)$ as follows:

$$P^{\mathrm{VC}}(G) := \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} x_u + x_v \geq 1 & \forall (u,v) \in E, \\ \sum_{v \in V} x_v \leq \frac{n}{2} - \frac{1}{2}, & \\ 0 \leq x_v \leq 1 & \forall v \in V \end{array} \right\}$$

$$Q^{\mathrm{VC}}(G) := \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} x_u + x_v \geq 1 & \forall (u,v) \in E, \\ \sum_{v \in V} x_v \leq \frac{n}{2} - \frac{1}{2}, & \\ x_v \in \left\{0, \frac{1}{d}, \ldots, 1\right\} & \forall v \in V \end{array} \right\}.$$

A key observation is that every non-empty vertex-cover polytope is $\frac{1}{8n}$-balanced. Corollary 21 follows directly from the proof of Lemma 15.

▶ **Corollary 21.** *Let G be a bipartite graph with no perfect matching and $\frac{n}{2}$ vertices on each side. Then, $P^{\mathrm{VC}}(G)$ is $\frac{1}{8n}$-balanced.*

The following lemma states that if the grid has sufficiently fine resolution, the volume of a balanced polytope can be well approximated by the (scaled) number of grid points it contains. We prove Lemma 22 in the full version of the paper.

▶ **Lemma 22** (Volume approximation for β-balanced polytopes)**.** *Let $P \subseteq \mathbb{R}^n$ be a β-balanced polytope. Let $d = \lceil \beta^{-1} n^6 \rceil$. Let $Q = P \cap (\mathbb{Z}/d)^n$ be the set of discretized points in P. Then*

$$1 - n^{-3} \leq \frac{d^{-n} \cdot |Q|}{\mathrm{vol}(P)} \leq 1 + n^{-3}.$$

The following fact is immediate from the definitions of $P^{\mathrm{VC}}(G)$ and $Q^{\mathrm{VC}}(G)$.

▶ **Fact 23.** $Q^{\mathrm{VC}}(G) = P^{\mathrm{VC}}(G) \cap (\mathbb{Z}/d)^n$.

We are now ready to prove Lemma 17.

▶ **Lemma 17** (Existence of a progress-making edge)**.** *Let G be a bipartite graph with a perfect matching and $\frac{n}{2}$ vertices on each side. Let H be any subgraph of G on the same vertices with $Q^{\mathrm{VC}}(H) \neq \varnothing$, where $Q^{\mathrm{VC}}(H)$ is defined in Equation (4) for $d = 8n^7$. Then, there is an edge $e \in (G \setminus H)$ such that*

$$\left|Q^{\mathrm{VC}}(H \cup \{e\})\right| \leq 0.9 \cdot \left|Q^{\mathrm{VC}}(H)\right|.$$

Proof. Since $Q^{\mathrm{VC}}(H) \neq \varnothing$, by Fact 23, $P^{\mathrm{VC}}(H)$ is also non-empty. By Lemma 15, this means H has no perfect matching, and then by Corollary 21, $P^{\mathrm{VC}}(H)$ is $\frac{1}{8n}$-balanced. By Lemma 22,

$$\mathrm{vol}\big(P^{\mathrm{VC}}(H)\big) \leq \frac{d^{-n}}{1 - n^{-3}} \cdot \left|Q^{\mathrm{VC}}(H)\right|.$$

Let z be the center of mass of $P^{\mathrm{VC}}(H)$. Then z is a fractional vertex cover of H of size $\sum_{v \in V} z_v \leq \frac{n}{2} - \frac{1}{2}$. Let M be a perfect matching of G. There exists an edge $f = (a, b) \in M$ that is not covered by z, i.e., $z_a + z_b < 1$. Since z is a fractional vertex cover of H, $f \in (G \setminus H)$.

Let $H' := H \cup \{f\}$. Because z violates the constraint of edge f, Grünbaum's Theorem [11] gives

$$\mathrm{vol}\big(P^{\mathrm{VC}}(H')\big) \leq \left(1 - \frac{1}{e}\right) \cdot \mathrm{vol}\big(P^{\mathrm{VC}}(H)\big).$$

We consider two cases. If $P^{\mathrm{VC}}(H') = \varnothing$, then $Q^{\mathrm{VC}}(H') = \varnothing$ by Fact 23, and the lemma holds.

If $P^{\mathrm{VC}}(H') \neq \varnothing$, then by Lemma 15, H' has no perfect matching. By Corollary 21, $P^{\mathrm{VC}}(H')$ is $\frac{1}{8n}$-balanced. By Lemma 22, we have

$$\mathrm{vol}\big(P^{\mathrm{VC}}(H')\big) \geq \frac{d^{-n}}{1+n^{-3}} \cdot \big|Q^{\mathrm{VC}}(H')\big|,$$

and consequently,

$$\frac{\big|Q^{\mathrm{VC}}(H')\big|}{|Q^{\mathrm{VC}}(H)|} \leq \frac{1+n^{-3}}{1-n^{-3}} \cdot \frac{\mathrm{vol}\big(P^{\mathrm{VC}}(H')\big)}{\mathrm{vol}(P^{\mathrm{VC}}(H))} \leq \frac{1+n^{-3}}{1-n^{-3}} \cdot \left(1-\frac{1}{e}\right) \leq 0.9,$$

where the last inequality holds for all $n \geq 2$. Thus f satisfies $\big|Q^{\mathrm{VC}}(H \cup \{f\})\big| \leq 0.9 \cdot \big|Q^{\mathrm{VC}}(H)\big|$, as claimed. ◀