Proceedings Paper:
Blikstad, J., Van Den Brand, J., Efron, Y. et al. (2 more authors) (2022) Nearly optimal communication and query complexity of bipartite matching. In: 2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS). 2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS), 31 Oct - 03 Nov 2022, Denver, CO, USA. Institute of Electrical and Electronics Engineers (IEEE) , pp. 1174-1185. ISBN 9781665455206

https://doi.org/10.1109/focs54457.2022.00113

Nearly Optimal Communication and Query Complexity of Bipartite Matching

Joakim Blikstad* Jan van den Brand† Yuval Efron‡ Sagnik Mukhopadhyay§ Danupon Nanongkai¶

Abstract

We settle the complexities of the maximum-cardinality bipartite matching problem (BMM) up to poly-logarithmic factors in five models of computation: the two-party communication, AND query, OR query, XOR query, and quantum edge query models. Our results answer open problems that have been raised repeatedly since at least three decades ago [Hajnal, Maass, and Turan STOC'88; Ivanyos, Klauck, Lee, Santha, and de Wolf FSTTCS'12; Dobzinski, Nisan, and Oren STOC'14; Nisan SODA'21] and tighten the lower bounds shown by Beniamini and Nisan [STOC'21] and Zhang [ICALP'04]. We also settle the communication complexity of the generalizations of BMM, such as maximum-cost bipartite b-matching and transshipment; and the query complexity of unique bipartite perfect matching (answering an open question by Beniamini [2022]). Our algorithms and lower bounds follow from simple applications of known techniques such as cutting planes methods and set disjointness.

*KTH Royal Institute of Technology, Sweden, `blikstad@kth.se`
†UC Berkeley & Simons Institute, USA, `vdbrand@berkeley.edu`
‡Columbia University, USA, `ye2210@columbia.edu`
§University of Sheffield, UK, `s.mukhopadhyay@sheffield.ac.uk`
¶University of Copenhagen, MPI-INF, and KTH, `danupon@gmail.com`

Contents

1 Introduction

In the *maximum-cardinality bipartite matching* problem (BMM), we are given a bipartite graph $G = (L \cup R, E)$ with n vertices on each side and m edges. The goal is to find a matching of maximum size in G. This problem, along with its special case of *bipartite perfect matching* (BPM), are central problems in graph theory, economics, and computer science. They have been studied in various computational models such as the sequential, two-party communication, query, and streaming settings [See [FF56, HK73, Lov79, KUW85, KVV90, MS04, Zha04, IKL+12, Mad13, Mad16, GO16, GG17, BHR19, DNO19, AV20, AK20, JST20, BLN+20, Nis21, AR20b, CKP+21, FGT21, AB21, ALT21, FGL+21, RSW22, CKL+22] and many more]. In this paper, we present simple algorithms and lower bound arguments that settle (up to polylog factors) the complexities of BMM and its generalizations (e.g. max-cost matching and transshipment) in at least five models of computation. Our results answer open problems that have been raised repeatedly since at least three decades ago (e.g. [HMT88, Zha04, IKL+12, DNO19, Nis21, Ben22a]); see Table 1 for a summary of our results.

Communication complexity. To be concrete, we start with the two-party communication model, where edges of the input graph G are partitioned between two players Alice and Bob. The goal is for Alice and Bob to compute the value of the BMM or to decide if a BPM exists in G by communicating as frugally as possible. Many fundamental graph problems have been studied in this model since the 80s (e.g. [PS82, BFS86, HMT88, DP89]). For BMM and BPM, their communication complexities have been extensively studied from several angles and perspectives, including exact solution protocols [BFS86, HMT88, IKL+12, DNO19], round restricted protocols [FKM+05, GKK12, GO16, AKL17, AB19b, AR20b], multiparty protocols [GO16, HRVZ15, AKLY16, Kap21, KMT21, HRVZ20], approximate solution protocols [Kap21, KKS14, KKS14, KMNT20, AB21], matrix rank and polynomial representation [Ben22b, BN21], and economics and combinatorial auctions [Rot82, Ten02, Ber09]. In particular, Hajnal, Maass, and Turán [HMT88] showed a lower bound of $\Omega(n \log n)$ for deterministic protocols[1]. For randomized and quantum protocols, the lower bounds are $\Omega(n)$ [BFS86, IKL+12, Raz92][2]. For an upper bound, Ivanyos, Klauck, Lee, Santha, and de Wolf [IKL+12] implemented the Hopcroft-Karp algorithm [HK73] to get an $O(n^{3/2} \log n)$-bit deterministic protocol (see also [DNO19, Nis21]).

Closing the large gap between existing upper and lower bounds has been mentioned as an open problem in, e.g., [HMT88, IKL+12, DNO19, Nis21]. Beniamini and Nisan [BN21] recently showed that the rank of the communication matrix is $2^{O(n \log n)}$, suggesting that a better upper bound might exist. On the other hand, $\Omega(n^2)$ lower bounds for $o(\sqrt{\log n})$-round communication may suggest that an $\Omega(n^{1+\Omega(1)})$ communication lower bound may exist [FKM+05, GKK12, AR20b, CKP+21]. In this paper, we resolve this open problem with an $O(n \log^2 n)$ upper bound:

Theorem 1.1. *The deterministic two-party communication complexity of* BMM *is* $O(n \log^2 n)$.

Note that our protocol can find the actual BMM (Alice and Bob know edges in the BMM in the end) and not just its value. We can in fact solve a more general problem of min-cost bipartite perfect b-matching which implies upper bounds for a large class of problems due to existing reductions (see [BLN+20]).

[1][HMT88] did, in fact, show this lower bound for st-connectivity, which, together with folklore reductions, imply the same bound for BPM.

[2] The $\Omega(n)$ lower bound follows by a simple reduction from set-disjointness. [HRVZ20] has shown a $\Omega(\alpha^2 nk)$ lower bound for k-party point-to-point communication model for α-approximation of BMM. [IKL+12] shows a $\Omega(n)$ quantum communication lower bound by a reduction from inner-product in $\mathbb{F}_2$.

Models	**Previous papers**		**This paper**
	Lower bounds	Upper bounds	
Two-party communication	$\Omega(n)$ Rand, $\Omega(n \log n)$ Det, Footnote 1 and 2	$\tilde{O}(n^{1.5})$ [DNO19, IKL+12]	$O(n \log^2 n)$, Det Theorem 1.1
Quantum edge query	$\Omega(n^{1.5})$ [Zha04, Ben22b]	$O(n^{1.75})$ [LL15]	$\tilde{O}(n^{1.5})$ Theorem 1.3
OR-query	$\Omega(n)$ Rand, $\Omega(n \log n)$ Det, [BN21]	$\tilde{O}(n^{1.5})$ Det, [Nis21]	$O(n \log^2 n)$, Det Theorem 1.3
XOR-query	$\Omega(n)$ Rand $\Omega(n^2)$ Det [BN21]	$\tilde{O}(n^{1.5})$ Rand Lemma 2.14 and [Nis21]	$O(n \log^2 n)$, Rand Theorem 1.3
AND-query	$\Omega(n)$ Rand, $\Omega(n^2)$ Det [BN21]	$O(n^2)$ Trivial	$\Omega(n^2)$, Rand Theorem 1.3

Table 1: The communication and query complexity bounds for BMM and BPM. All upper bounds are stated for BMM and all lower bounds are stated for BPM.

Theorem 1.2. *Given that all the weights/costs/capacities are integers polynomially large in n, we can solve the following problems in the two-party edge-partition communication setting, using $O(n \log^2 n)$ bits of communication: Maximum-cost bipartite perfect b-matching, Maximum-cost bipartite b-matching, Vertex-capacitated minimum-cost (s,t)-flow, Transshipment (a.k.a. uncapacitated minimum-cost flow), Negative-weight single source shortest path, Minimum mean cycle and Deterministic Markov Decision Process (MDP).*

Query complexity. Besides the communication complexity, we also settle the query complexity of BMM and BPM for several variants of the edge query model. In the standard edge query model, the querier can ask whether an edge in the input graph G is present or not. The goal is to solve the graph problem by making as few queries as possible. This query model, in both deterministic and randomized settings, has been studied for almost half a century [Ros73, RV75, RV76, KSS84, Haj91, CK07] for various graph problems. For BPM, Yao [Yao88] showed that n^2 edge queries are necessary in the deterministic setting and Dürr, Heiligman, Høyer, and Mhalla [DHHM06] showed an $\Omega(n^2)$ lower bounds for the randomized setting[3] thereby completely characterizing (up to constant factors) classical edge query complexity for BPM.

However, for several variants of the classical edge query complexity, there are known gaps between the best known upper and lower bounds for BPM. For example, in the case of *quantum* edge query protocols, Zhang [Zha04] showed a lower bound of $\Omega(n^{1.5})$ by using Ambainis' adversary method [Amb02] (see [Ben22b] for an alternative proof via approximate degree). The best upper bound is, however, at $O(n^{1.75})$ as shown by [LL15]. This upper bound is obtained by simulating the Hopcroft-Karp algorithm using *bomb queries* and relating it to the quantum edge queries.

Another well-studied variant of the classical query protocols is the XOR-query protocols (otherwise known as the *parity decision trees*) where the querier is allowed to ask the following question

[3][Yao88] showed a stronger result: Any non-trivial monotone graph property needs n^2 queries. [DHHM06] mentioned a $\Omega(n^2)$ randomized query complexity for Connectivity. A similar construction (which is essentially a reduction from the query complexity of OR_{n^2}) shows an $\Omega(n^2)$ lower bound for (s,t)-Reachability which reduces to BPM.

about the input graph $G = (V, E)$: Given a set S of potential edges of G, is $|S \cap E|$ odd or even? Similarly, AND-queries and OR-queries ask if $S \subseteq E$ or not and if $|S \cap E| \geq 1$ or not, respectively. Such query models have proven to be extremely important in the study of XOR-functions, the log-rank conjecture and lifting theorems [KM93, MO09, CKLM18, HHL18, MS20]. As usual, these query models can be studied in deterministic, randomized and quantum models as well. For graph problems, these query models have recently started to receive increasing attention [BN21, Ben22a, ACK21]. For AND-query or XOR-query complexity, a recent result of [BN21] showed that $\Omega(n^2)$ queries are necessary to compute BPM deterministically[4]. For OR-query, [BN21] also showed a deterministic lower bound of $\Omega(n \log n)$. The upper bound of $\tilde{O}(n^{1.5})$ for OR-queries (and, thereby, *randomized* XOR-queries, see Lemma 2.14) can be achieved by simulating the Hopcroft-Karp algorithm [Nis21].

From the above results, it remained open to close the polynomial gaps for quantum and OR-queries (as mentioned in [Nis21, Ben22b]) and whether randomization helps for XOR-queries and AND-queries. In this paper, we answer these questions: We provide upper bounds that are tight up to polylogarithmic factors for quantum and OR-queries. Our upper bound result also shows that randomization helps for XOR-queries. In contrast, for AND-queries we can show that an $\Omega(n^2)$ lower bound holds even for randomized algorithms. Our results are summarized below and in Table 1. Note that our lower bound argument also gives simplified proofs of the lower bounds for XOR-queries and OR-queries.

Theorem 1.3. *The following query bounds hold for* BMM*:*

- *The quantum edge query complexity is $O(n^{1.5} \log^2 n)$,*
- *The deterministic* OR*-query complexity is $O(n \log^2 n)$,*
- *The randomized* XOR*-query complexity is $O(n \log^2 n)$,*

Moreover, the randomized AND*-query complexity of* BPM *is $\Omega(n^2)$.*

Finally, our results also extend to the unique bipartite perfect matching problem (UBPM), which has been studied in, e.g., the sequential and parallel settings [KVV85, GKT01, HMT06, Ben22a]. Beniamini [Ben22a] recently show UBPM lower bounds similar to those for BMM and BPM, i.e. $\Omega(n \log n)$ communication complexity, $\tilde{\Omega}(n^{1.5})$ quantum edge query (under a believable conjecture[5]), $\Omega(n \log n)$ OR-queries, $\Omega(n^2)$ XOR-queries, and $\Omega(n^2)$ AND-queries. We complement these lower bounds with tight upper bounds, i.e. $O(n \log^2 n)$ deterministic communication protocol, $O(n^{1.5} \log^2 n)$ quantum edge query algorithm, $O(n \log^2 n)$ deterministic OR-query and randomized XOR-query algorithms, and $\Omega(n^2)$ randomized AND-query lower bound. Our upper bounds answer an open problem by Beniamini [Ben22a].

Update: After our paper was accepted in FOCS 2022, we observed that our technique also leads to a $O(n \log^2 n)$ deterministic protocol in the well-studied *Indenpendent set* (IS) query model [BHR+18, AL21, RWZ20, AA05, ABK+04, AB19a]. In this model, a query consists of two disjoint subsets of vertices X and Y, and the answer to the query is 1 iff there is an edge between X and Y (i.e., $E \cap (X \times Y) \neq \emptyset$).

Organization. In Section 1.1, we provide a brief technical overview of our upper and lower bounds. In Section 1.2, we list a few open problems that naturally arise from our work. Section 2 details our various upper bounds, starting with OR-query protocols. In Section 2.3, we show the applications of the OR-query algorithm, namely two party communication complexity (Section 2.3.1),

[4]For XOR-queries, [BN21] showed that BPM is *evasive*, i.e., requires n^2 queries.

[5][Ben22a] conjectured that the approximate degree of UBPM is $\Omega(n^{1.5})$ (see Conjecture 1) which would imply a similar lower bound for quantum edge query complexity.

randomized XOR-query (Section 2.3.2), Independent set query (Section 2.3.3), OR_k-query (Section 2.3.4) and quantum edge query (Section 2.3.5). We then list different variants of the bipartite matching problem (Section 3) that our technique can solve as well. Finally, in Section 4, we provide lower bounds for solving BPM in OR-, AND- and XOR-query settings.

1.1 Technical Overview

Upper bounds. Our algorithms follow an existing continuous optimization method. There are many such methods and the question is: *what is the right method?* An intuitive idea would be to implement some fast sequential algorithms for BMM and related problems (e.g. [DS08, Mad13, LS14, Mad16, CMSV17, CLS19, Bra20, LS20, AMV20, BLSS20, BLN+20, BLL+21, CKL+22]), which are based on *central path methods*. It is not clear, however, how to implement central path methods efficiently in query or communication settings. They require polynomially many iterations (e.g. $\Omega(\sqrt{n})$), each of which needs a large communication and query complexity (e.g. $\Omega(n)$ per iteration). Another option is to use one of the *cutting planes methods* (e.g. the Ellipsoid method). These methods are a framework for solving general convex optimization problems and thus are rather slow for BMM in the sequential setting (e.g. $\tilde{O}(mn)$ time [LSW15]) compared to more specialized alternatives based on central path methods. However, it turns out that cutting planes methods are the right framework for the communication and query settings! In particular, we can implement a cutting planes method with a low number of iterations, such as the *center-of-gravity* (CG) and *volumetric center* (VC) methods [Lev65, New65, Vai89], on the *dual linear program*, i.e. the minimum vertex cover linear program[6]. (We cannot use the Ellipsoid method due to its high number of iterations.) The CG and VC methods are not useful for solving BMM in the sequential setting due to their high running time (the CG method even requires exponential time); however, this high running time is hidden in the internal computation and thus does not affect the communication/query complexities.

Using the cutting planes methods above, our algorithm is simply the following: We start with an assignment $p : V \to \mathbb{R}^+$ on the vertices that is supposed to be a fractional vertex cover of value F, i.e. for every edge (u, v), $p(u) + p(v) \geq 1$ and $\sum_{v \in V(G)} p(v) \leq F$. In each iteration, we need to find a *violated constraint*, i.e. an edge (u, v) such that $p(u) + p(v) < 1$, or the value constraint if $\sum_{v \in V(G)} p(v) > F$. This violated constraint then allows us to compute a new assignment $p : V \to \mathbb{R}^+$ (which is the center of gravity of some polytope) to be used in the next iteration. It can be shown that this process needs to repeat only for $\tilde{O}(n)$ times to construct a fractional vertex cover of value at least F, or conclude no such cover exists.

This simple algorithm leads to efficient algorithms in many settings. For example, in the two-party communication setting, Alice and Bob only need to communicate one violated constraint in each iteration while they can compute the new assignment $p : V \to \mathbb{R}^+$ without any additional communication ($p : V \to \mathbb{R}^+$ depends only on the discovered violated constraints and not on the input graph). It is also not hard to implement this method in other settings. We note that in this paper we use the CG method for simplicity. This method leads to exponential internal computation. This can be made polynomial by using the VC method [Vai89] instead.

Lower bounds. For lower bounds, our goal is to prove a lower bound for BPM (which also implies a lower bound for BMM). Let us start with our randomized AND-query lower bound of $\Omega(n^2)$. A typical approach to show this is proving an $\Omega(n^2)$ communication complexity lower bound in the

[6]We thank an anonymous FOCS'22 reviewer for pointing out the result in [VWW20] that uses the cutting plane method to solve general linear program in the multiparty model of communication. Our result is independent and follows the same general cutting plane framework but we exploit that for our specific linear program, (i) cutting planes have short description and (ii) we have a better bound on the number of iterations.

setting defined earlier; however, we have already shown in Theorem 1.1 that this is not possible. [BN21] sidestepped this obstacle by considering the real polynomial associated with BPM. Known connections between the monomial complexity of this polynomial and AND-query complexity yield corresponding tight $\Omega(n^2)$ *deterministic* AND-query complexity for BPM.

It turns out that we can prove a *randomized* AND-query lower bound (and simplifying the lower bounds proofs of [BN21]) by revisiting the two-party communication lower bounds, but with a slightly different definition. Our main observation here is that AND-queries can be simulated cheaply by the following variant of the two-party communication model: Alice gets edge set $E_A \subseteq E$, Bob gets edge set $E_B \subseteq E$, and they solve BPM (or any other graph function) in the graph $G_\cap = (V, E_A \cap E_B)$. Our AND-query lower bound now follows from a reduction from the set disjointness problem.

Similarly, Beniamini and Nisan [BN21] use real polynomial techniques to prove deterministic lower bounds for XOR-queries and OR-queries. We provide simple alternative proofs via the communication complexity of BMM in the symmetric difference and union graphs $G = (V, E_A \oplus E_B)$ and $G = (V, E_A \cup E_B)$; such lower bounds can be proved via a reduction from the equality and st-reachability problems. Finally note that even though we simplify the query lower bounds proofs, [BN21, Ben22b] showed something stronger, i.e., a complete characterization of the unique multilinear polynomial over reals representing BPM which may have other interesting consequences beyond query complexity.

1.2 Open problems

The communication complexity of BMM and BPM has been a bottleneck for many tasks. The fact that it can be solved by a simple cutting planes method might be the gateway to solving many other problems. Below we list some of these problems.

1. **Demand query complexity of BMM.** The demand query setting is equivalent to when we can issue an OR-query only on the edges incident on a single left vertex (or, equivalently, an IS-query where set $X \subseteq L$ is singleton). Minimizing the number of demand queries used to solve BMM and BPM is motivated by economic questions [Nis21, Ben22b]. Like in many settings we consider, the best demand query upper and lower bounds for BMM and BPM are $O(n^{1.5})$ and $\Omega(n)$ respectively. Closing this gap remains open. Because of our efficient IS-query protocol, we believe that a possible direction is to extend our approach to get a better upper bound for demand query. For a better lower bound, our results suggest that one might need a technique specialized for the demand query lower bound: the two known approaches for proving a demand query lower bounds are via quantum and OR-queries (see, e.g., Figure 7 in [Ben22b]) and our quantum and OR-query upper bounds show that these approaches cannot be used.

2. **Bounded communication rounds and streaming passes.** Most graph problems, including BMM and BPM, admit an $\Omega(n^2)$ communication lower bound when only Alice can send a message (i.e. the one-way communication setting) [FKM+05]. If Bob gets to speak back once (the 2-round setting), some problems become much easier (e.g. the communication complexity of global edge connectivity reduces from $\Omega(n^2)$ to $\tilde{O}(n)$) [AD21]. Unfortunately, such an efficient protocol for BPM does not exist even when we allow $o(\sqrt{\log n})$ rounds [CKP+21, AR20b]. More generally, r-round protocols are known to require $n^{1+\Omega(1/r)}$ communication [AR20b, GO16]. An important question is to get tight r-round communication bounds for BMM and BPM. Our algorithm provides an $\tilde{O}(n)$ communication bound for the extreme case where $r = n$. One possible extension is to study *bounded-iteration* cutting planes

methods. For example, can we reduce the number of iterations if in each iteration we can identify more violating constraints? It will be exciting if a polylog(n)-round $\tilde{O}(n)$-communication protocol exists. It will be even more exciting if this can be extended to a polylog(n)-*passes streaming algorithm* (breaking [LSZ20, AJJ$^+$22] and matching [GO16]).

3. **Distributed Matching.** The distributed CONGEST model is an important model to study fundamental graph problems (e.g. minimum spanning tree, shortest paths, and minimum cut) on distributed networks (e.g. [GHS83, KP98, Nan14, GL18, FN18, Elk20, BN19, AR20a, GKK$^+$18, HKN21, CM20, GNT20, NS14, DEMN21]). Compared to other graph problems, computing BMM and BPM exactly in CONGEST is much less understood in this model. This is despite the studies of their variants since the 80s [Lub86, II86, Gal16, AKO18, AK20]. The best lower bound for this problem is $\tilde{\Omega}(\sqrt{n}+D)$[AKO18, DHK$^+$12] (see also [HWZ21]). The best upper bound is $O(n \log n)$ [AKO18]. For sparse graphs, the upper bound can be improved to $\tilde{O}(m^{3/7}(\sqrt{n}D^{1/4}+D))$ via continuous optimization [FGL$^+$21]. (Better upper bounds via fast matrix multiplication also exist on the special case of *congested clique* [Gal16].) A major open problem is to close the gap between upper and lower bounds. Our results may suggest a new approach for improving the known upper bounds for the problem. Past results seem to suggest that graph problems with $\tilde{O}(n)$ communication complexity usually admit an $\tilde{O}(\sqrt{n}+D)$ upper bound in CONGEST. (A recent example is the $\tilde{O}(n)$ communication complexity protocol of mincut [MN20] that was later extended to achieve an $\tilde{O}(\sqrt{n}+D)$ upper bound in CONGEST [DEMN21].) Proving that this is or is not the case for BMM and BPM will be an exciting result.

4. **General Matching.** The maximum matching problem on *general* (i.e. not-necessarily-bipartite) graphs is less understood than that on bipartite graphs. Unlike BMM, the linear programming formulations for general matching is rather unwieldy, making it difficult to apply the cutting planes method approach. Settling the communication and query complexity of general matching remain intriguing open problems. On one hand, there might be a hope to show truly super-linear (i.e., $\Omega(n^{1+\epsilon})$ for some constant $\epsilon > 0$) communication lower bounds in these models, thereby showing a gap between the bipartite and non-bipartite case. On the other hand, an $\tilde{O}(n)$ communication complexity upper bound for the general matching problem would hopefully shed some light on the interplay between matchings on bipartite versus general graphs.

5. **Maxflow/mincut and Related Problems.** Max (s,t)-flow, equivalently min (s,t)-cut, is a powerful tool that can be used to solve BMM, BPM, and many other fundamental graph problems. Efficiently solving this problem could only be a dream in the past in many computational models since even its special case of matching could not be solved efficiently. Our results serve as a step toward this goal. Particularly interesting goals are solving (s,t)-max-flow/min-cut in the communication[7], distributed, cut query, and streaming settings (Bounded round communication lower bounds in multiparty communication setting for (s,t)-max-flow/min-cut have been studied in [ACK19].). Also, there are problems that were recently shown to be solvable in max-flow time in the sequential setting such as Gomory-Hu tree, vertex connectivity, Steiner cut, hypergraph global min-cut, and edge connectivity augmentation [CGL$^+$20, LP20, LP21, LNP$^+$21, CQ21, MN21]. Can these problems be solved as efficiently as max-flow in other settings, e.g. the communication, distributed, and streaming settings?

Other problems include (i) showing $\Omega(n \log n)$ *randomized* communication lower bound for

[7]Here we expect Alice and Bob to know the flow values in their respective sets of edges. The decision version of this problem where we ask if the total flow is at least a threshold k is also interesting.

connectivity or even just for BMM and min-cost flow, (ii) closing the $\log n$ factor gap between OR-query upper and lower bounds, and (iii) settling the quantum OR-query and AND-query complexity of BMM and BPM.

2 Bipartite Matching Upper Bounds

Our goal in this section is to present a simple OR-query algorithm based on the cutting planes framework to find a maximum matching of a bipartite graph, i.e. to solve the BMM problem. From there we show how our OR-query algorithm can be translated to several other information theoretical models of computation. Formally, the following is the main theorem of the section.

Theorem 2.1. *Given n, there are algorithms solving* BMM *in the following models.*

1. *Deterministic two-party edge-partition communication, with communication complexity $O(n\log^2 n)$.*
2. *Deterministic* OR*-query, with query complexity $O(n\log^2 n)$.*
3. *Randomized* XOR*-query, with query complexity $O(n\log^2 n)$.*
4. *Quantum edge query, with query complexity $O(n^{1.5}\log^2 n)$.*

Overview. We employ a standard cutting planes framework to determine if a bipartite graph has a vertex cover of a given size F or not. We show that this cutting planes method can be implemented in $O(n\log n)$ iterations, where in each iteration we access the input graph a small number of times ($O(\log n)$) using OR-queries to find an edge that corresponds to a violated constraint (i.e. a cutting plane), if one exists. Throughout this work, we use the following well known characterization of the existence of a matching of a certain size in a bipartite graph.

Claim 2.2 (König's Theorem)**.** *A bipartite graph G has a minimum vertex cover of size F if and only if it does not have a matching of size $F+1$.*

The vertex cover linear program. For a bipartite graph $G=(V,E)$ with $V=L\cup R$, $|L|=|R|=n$, the following linear program ($\mathcal{P}^G$) over $x\in\mathbb{R}^V$ describes the fractional minimum vertex cover problem on G. Since G is bipartite, the constraint matrix is totally unimodular, and hence ($\mathcal{P}^G$) is integral [KVKV11, Section 5], i.e. there exists an integer optimal solution to ($\mathcal{P}^G$).

$$\begin{array}{lll} \text{minimize} & \sum_{v\in V} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \forall (u,v)\in E \\ & 0\leq x_v \leq 1 & \forall v\in V \end{array} \qquad (\mathcal{P}^G)$$

Decision version. We first consider a *decision version* of our problem, namely given an integer F we want to determine if G has a matching of size at least $F+1$. Note that if we can solve this decision version, then we can also—by binary-searching over F—solve the *optimization version* (i.e. finding the minimum size of a vertex cover / maximum size of a bipartite matching) with an overhead of $O(\log n)$. We start by focusing on solving the decision version (Section 2.1), and later (in Section 2.2) we show how to, via a simple modification of the algorithm, actually solve the optimization version *without* this extra $O(\log n)$ binary-search overhead.

By König's Theorem (Claim 2.2), determining whether G has a matching of size at least $F+1$ is equivalent to determining whether G (does not) have a vertex cover of size at most F. This is equivalent to determining if $(\mathcal{P}^G)$ has some feasible solution x with $\sum x_v \leq F$. So we define another polytope $(\mathcal{P}_F^G)$ as follows:

$$\begin{aligned} &\sum_{v\in V} x_v \leq F + \tfrac{1}{3} & \\ &x_u + x_v \geq 1 & \forall (u,v) \in E \\ &0 \leq x_v \leq 1 & \forall v \in V \end{aligned} \qquad (\mathcal{P}_F^G)$$

Our decision algorithm either finds a feasible point for the above polytope, or it finds a witness of $\mathcal{P}_F^G$ having no feasible points in the form of a set of edges that contains a matching of size $F+1$.

Note that we relax the constraint $\sum x_v \leq F$ a bit to $\sum x_v \leq F + \frac{1}{3}$. This ensures that our polytope has a significantly large volume if it is non-empty (see Lemma 2.3). Thus our cutting planes methods can terminate and conclude that the polytope is empty whenever the volume is too small. This relaxation does not impact the correctness of our algorithm: since $(\mathcal{P}^G)$ is integral, it has an integral optimal objective value, which means that if a feasible solution x of $(\mathcal{P}_F^G)$ exists, then there also exists a feasible solution x' which achieves $\sum x'_v \leq F$.

Lemma 2.3. *For any bipartite graph $G = (V, E)$, if F is an integer such that $(\mathcal{P}_F^G)$ is non-empty, then* $\mathrm{vol}(\mathcal{P}_F^G) \geq \left(\frac{1}{20n}\right)^{2n}$.

Proof. Let x be an integral solution for $(\mathcal{P}_F^G)$, of value F. Indeed, if $(\mathcal{P}_F^G)$ is feasible, then such an x must exist due to the integrality of $(\mathcal{P}^G)$. Let $I_0 = \{i \in [2n] \mid x_i = 0\}$ and $I_1 = \{i \in [2n] \mid x_i = 1\}$. We argue that the hypercube $[\frac{1}{20n}, \frac{1}{10n}]^{I_0} \times [1 - \frac{1}{20n}, 1]^{I_1}$ is completely contained in $(\mathcal{P}_F^G)$. That is, if, for each x_i with $x_i = 1$ we replace it with any value in $[1 - \frac{1}{20n}, 1]$; and for each x_i with $x_i = 0$ we replace it with with any value in $[\frac{1}{20n}, \frac{1}{10n}]$; the point remains feasible for $(\mathcal{P}_F^G)$. We verify this below.

- The $0 \leq x_v \leq 1$ constraints remain valid.
- Similarly, the $\sum x_v \leq F + \frac{1}{3}$ constraint remains valid, since we increase the value of x_i by at most $\frac{1}{10n}$ for each i, and there are $2n$ vertices in total (so we increase $\sum x_v$ by at most $\frac{1}{5}$).
- Lastly, the constraint $x_u + x_v \geq 1$ (for an edge $(u, v) \in E$) also remains valid, as either (i) both x_u and x_v were 1 before, in which case we now have $x_u + x_v \geq 2 - \frac{1}{10n}$; or (ii) exactly one of x_u or x_v was 1 before, in which case we increased the variable which was 0 by at least $\frac{1}{20n}$ and decreased the variable which was 1 by at most $\frac{1}{20n}$.

Thus we have argued that a hypercube of volume $\left(\frac{1}{20n}\right)^{2n}$ is contained in $(\mathcal{P}_F^G)$. ☐

2.1 OR-query decision algorithm

In this section we describe our cutting planes based OR-query algorithm for solving the feasibility problem on $(\mathcal{P}_F^G)$. We begin with a verbal overview of the algorithm, followed by pseudocode in Algorithm 1. The main lemma of this section is the following.

Lemma 2.4. *Given an integer F, there is a deterministic algorithm (Algorithm 1) using $O(n \log^2 n)$ OR-queries which on an input bipartite graph $G = (V, E)$ either finds a feasible point in $(\mathcal{P}_F^G)$, or else a witness, in the form of a matching of size $F + 1$, that $(\mathcal{P}_F^G)$ is empty.*

Center-of-gravity cutting planes method. We are now ready to introduce the cutting planes framework [Lev65, New65]. The idea is that we start with the polyhedra $P_0 = \{x \in [0,1]^V : \sum x_v \leq F + \frac{1}{3}\}$ (which contains $(\mathcal{P}_F^G)$), and repeatedly find "good" constraints "$x_u + x_v \geq 1$" (corresponding to edges $(u,v) \in E$) to add which reduce the volume sufficiently fast. Eventually, we either find a (fractional) feasible solution to $(\mathcal{P}_F^G)$, or have determined that no such feasible point exist.

We work in iterations, each iteration i is characterized by a polyhedron $P_i \supseteq (\mathcal{P}_F^G)$. We compute the *center-of-gravity* of P_i, denoted by $p_i = cg(P_i) \in P_i$, and defined to be $cg(P_i) = \left(\int_{P_i} z \, \mathrm{d}z\right) / \left(\int_{P_i} \mathrm{d}z\right)$. Note that we know P_i, so our algorithm can compute[8] $p_i = cg(P_i)$ without using any queries.

Either p_i is feasible for $(\mathcal{P}_F^G)$, in which case the cutting planes algorithm reports this and terminates. Otherwise there must exist some *violated constraint* "$x_u + x_v \geq 1$" in $(\mathcal{P}_F^G)$ but not in P_i (i.e. p_i does not satisfy this constraint, that is $p_i^u + p_i^v < 1$). In this case, we want to find such a violated constraint, and let $P_{i+1} = P_i \cap \{x \in \mathbb{R}^V : x_u + x_v \geq 1\}$, after which we continue with the next iteration of the cutting planes method on P_{i+1}. We say that an edge $(u,v) \in E$ is a *violating edge* for iteration i if $p_i^u + p_i^v < 1$. The process of finding a violating edge is the only part of the algorithm which requires access to the input graph, and hence the only place where OR-queries are being issued. Essentially, we need to implement a *separation oracle* FindViolatingEdge, which we explain how to do with OR-queries in Claim 2.5. The full algorithm can be found in Algorithm 1.

Algorithm 1: OR-query algorithm for BMM

```
Input: OR-query access to $G = (L \cup R, E)$, vertex set $L \cup R$, feasibility parameter $F$
Output: Whether $(\mathcal{P}_F^G)$ is feasible
1  $P_0 \leftarrow \left\{ x \in [0,1]^{2n} \;\middle|\; \sum_{v \in V} x_v \leq F + \frac{1}{3} \right\}$
2  $E' \leftarrow \emptyset$
3  $i \leftarrow 0$
4  while $vol(P_i) \geq \left(\frac{1}{20n}\right)^{2n}$ do
5      $p_i \leftarrow cg(P_i)$
6      $(u,v) \leftarrow$ FindViolatingEdge$(E', p_i)$
7      if no edge was found then
8          return "Feasible"                          // $p_i$ is feasible for $(\mathcal{P}_F^G)$
9      $E' \leftarrow E' \cup \{(u,v)\}$
10     $P_{i+1} \leftarrow P_i \cap \{x \in \mathbb{R}^{2n} \mid x_u + x_v \geq 1\}$
11     $i \leftarrow i + 1$
12 return "Infeasible"                                // $E'$ contains a matching of size $F + 1$
```

Claim 2.5 (OR-implementation of FindViolatingEdge)**.** *Using $O(\log n)$ OR-queries we can find a violating edge or else determine that none exist.*

Proof. Given the center-of-gravity point p_i, we let $S = \{(u,v) \in L \times R \mid p_i^u + p_i^v < 1\}$ be the set of pairs of vertices (u,v) which would be a violating edge if this pair was also an edge of the graph. Our task is thus to find some edge $e \in S \cap E$, or else determine that $S \cap E$ is empty. This can be done by a binary-search (with OR-queries) over S. ☐

[8]Finding the center-of-gravity in an n-dimensional polyhedron is NP-hard. However, all the considered models in Theorem 2.1 are query models, and in particular are purely information-theoretical, and we can thus disregard computational concerns. For ease of presentation, we work with center of gravity, but alternatively, one could use other variants of cutting plane using more computationally efficient notions of "center" such as volumetric centers [Vai89].

We now turn to prove several properties about our Algorithm 1.

Observation 2.6. *Let i be some iteration of the execution of Algorithm 1, then $P_i \supseteq \mathcal{P}_F^G$.*

Proof. For every i, the set of constraints defining P_i is, by the behaviour of the algorithm, a subset of the constraints defining $\mathcal{P}_F^G$, thus the observation follows. ☐

Lemma 2.7. *The algorithm terminates after $O(n \log n)$ iterations of the cutting planes method.*

Proof. We use the following well-known property of the center of gravity of a convex polytope.

Lemma 2.8 ([Grü60])**.** *For any convex polytope P with center of gravity c and any halfspace $H = \{x \mid \langle a, (x-c)\rangle \geq 0\}$ passing through c, it holds that:*

$$\frac{1}{e} \leq \frac{\mathrm{vol}(P \cap H)}{\mathrm{vol}(P)} \leq \left(1 - \frac{1}{e}\right).$$

This implies that, in our case, $\mathrm{vol}(P_{i+1}) \leq (1-\frac{1}{e})\mathrm{vol}(P_i)$. This means that in each iteration, we either find a feasible solution to $(\mathcal{P}_F^G)$, or cut down the volume by a constant fraction as we have found a violating edge. Initially, $\mathrm{vol}(P_0) \leq 1$, since it is contained in the unit-hypercube $[0,1]^{2n}$. By Lemma 2.3 we can terminate when P_i has volume less than $\left(\frac{1}{20n}\right)^{2n}$ and conclude that $(\mathcal{P}_F^G)$ is empty in this case. This happens after at most $O\left(\log((20n)^{2n})\right) = O(n \log n)$ iterations. ☐

Lemma 2.9. *Let i_{max} denote the last iteration in the execution of the algorithm. Then either $p_{i_{max}} \in \mathcal{P}_F^G$ which serves as a witness that a vertex cover of size F exists, or $\mathcal{P}_F^G = \emptyset$ and the set $E' \subseteq E$ (constructed by the algorithm) contains a matching of size $F+1$.*

Proof. In the case where we find a feasible point p in $(\mathcal{P}_F^G)$, this point is a fractional vertex cover of size at most $F + \frac{1}{3}$ for our graph (and hence a non-constructive witness that there exists an (integral) vertex cover of size F in the graph).

On the other hand, suppose we determined that $(\mathcal{P}_F^G)$ is empty, which means we got to an iteration i_{max} where $\mathrm{vol}(P_i) < \left(\frac{1}{20n}\right)^{2n}$. We argue that this actually means that the polyhedron P_i is empty. That is, we argue that we have found a set of edges $E' \subseteq E$ which contain a matching of size $F+1$ (E' is the set of edges whose constraints we added to $P_{i_{max}}$ during the cutting planes method). If this was not the case, that is if the maximum matching size in E' is at most F, then it must be the case, by Claim 2.2, that a vertex cover of size F exists in the subgraph $G' = (L \cup R, E')$, and hence that some integer point exists in our polyhedron $P_{i_{max}}$. We can deduce that this is impossible, however, by simply noting that by the behaviour of the algorithm, it holds that $(\mathcal{P}_F^{G'}) = P_{i_{max}}$, and thus we can apply Lemma 2.3 which then says that $\mathrm{vol}(P_i) \geq \left(\frac{1}{20n}\right)^{2n}$, which is a contradiction. ☐

By Claim 2.5 and Lemma 2.7 we see that the algorithm makes a total of $O(n \log^2 n)$ OR-queries, and Lemma 2.9 argues its correctness. This concludes the proof of Lemma 2.4.

2.2 OR-query optimization algorithm

In this section we describe a standard modification (see e.g. [Vai89, Section 4]) to our cutting planes *decision* algorithm, so that it solves the *optimization* version with the same query-complexity.

Lemma 2.10. *There is a deterministic algorithm using $O(n \log^2 n)$ OR-queries which solves the BMM problem. In particular, the algorithm finds a maximum matching M, together with a witness that M is maximum in the form of a fractional vertex cover of size strictly less than $|M| + 1$.*

Proof. The idea is to run Algorithm 1 starting with $F = 2n$. Whenever the algorithm finds a feasible point p_i, instead of terminating, we lower the value of F instead. The point p_i is a certificate that a vertex cover of size $\lfloor \sum_v p_i^v \rfloor$ exists (since $(\mathcal{P}^G)$ is integral). Hence we lower F to $F \leftarrow \lfloor \sum_v p_i^v \rfloor - 1$, by adding the constraint $\sum_v x_v \leq F + \frac{1}{3}$, and continue the cutting planes algorithm. Note that the constraint $\sum_v x_v \leq F + \frac{1}{3}$ forms a *violating constraint* for p_i (and therefore cuts down the volume by a constant fraction, see Lemma 2.8, and counts as an iteration of the cutting planes algorithm).

At the end, the algorithm must terminate by determining that $(\mathcal{P}_F^G)$ is empty (for the current value of F), in which case the found edges E' contains a matching of size $F + 1$ (see Lemma 2.9). On the other hand, the last time we lowered F, we had a fractional vertex cover p_i of size strictly less than $F + 2$. □

2.3 Applications

The goal of this section is to complete the proof of Theorem 2.1. We prove the theorem by showing how to simulate the OR-query cutting planes algorithm in the communication setting and the different query models (randomized XOR, IS, OR_k, and quantum edge query).

2.3.1 Communication complexity

We first consider the two-party edge-partition communication setting, where the edges E of the graph are partitioned into sets E_A and E_B given to Alice and Bob respectively.

Claim 2.11. *There is a communication protocol solving* BMM *in* $O(n \log^2 n)$ *bits of communication.*

A standard way of doing this is to simulate each OR-query $S \subseteq L \times R$ with 2 bits of communication: Alice and Bob check locally if $S \cap E_A$, respectively $S \cap E_B$, is non-empty and then share this information with each-other.

Alternatively, Alice and Bob can implement the "FindViolatingEdge"-subroutine of Algorithm 1 directly by checking locally for a violating edge and sharing it, if they find one, to the other party. This makes sure that E' is mutually known throughout the protocol. Sending an edge requires $O(\log n)$ bits of communication, and needs to be done $O(n \log n)$ times. So this alternative approach achieves the same final communication complexity (although in slightly fewer rounds of communication), and is also closer to our weighted matching algorithm in Section 3.2 (where the query-settings are no longer compatible).

2.3.2 Randomized XOR-query

Now we turn to the XOR-query setting. [BN21] showed that solving BPM is *evasive* for the XOR-query setting for any *deterministic* algorithm, meaning that any such algorithm needs to make n^2 queries (that is, the trivial algorithm for querying every potential edge individually is optimal)! Nevertheless, we show that *randomized* XOR-query algorithms are much more powerful, and can achieve almost linear number of queries instead.

Claim 2.12. *There is a randomized algorithm which makes* $O(n \log^2 n)$ XOR-*queries and, w.h.p.*[9], *solves* BMM.

In order to establish this result, we need the following folklore observation.

Observation 2.13. *For any k, let $x \in \{0,1\}^k$ be a binary string of length k, such that $x \neq 0^k$. If $r \in \{0,1\}^k$ is picked uniformly at random, then* $\Pr\left[(\sum_{i=1}^k x_i r_i) \text{ is odd}\right] = \frac{1}{2}$.

[9] *w.h.p. = with high probability*; meaning with probability at least $1 - 1/n^c$ for an arbitrarily large constant c.

Lemma 2.14. *A single* OR*-query can be simulated, w.h.p., by issuing* $O(\log n)$ *randomized* XOR*-queries.*

Proof. If we want to simulate an OR-query over a subset S, we can sample $S' \subseteq S$ randomly (independently keep every element with probability $\frac{1}{2}$) and issue an XOR-query over S'. If the answer to said OR-query was "YES", then we have, by Observation 2.13, a constant probability of realizing this with our XOR-query over S'. If we repeat $O(\log n)$ times, we can answer the OR-query correctly w.h.p. □

Proof of Claim 2.12. Just applying Lemma 2.14 to our OR-query algorithm would imply an $O(n \log^3 n)$ randomized XOR-query algorithm. An additional observation is required to bring the query complexity down to $O(n \log^2 n)$. We note that in each invocation of FindViolatingEdge, we need only simulate the first OR-query, after which we, w.h.p., have in hand a concrete set $S' \subseteq S$ for which $\mathsf{XOR}(S') = 1$ (or else determined that the answer to said OR-query should be "NO"). At this point we can binary-search *deterministically* using an additional $O(\log n)$ XOR-queries to find a violating edge in S'. Hence, each invocation of FindViolatingEdge can be simulated, w.h.p., via $O(\log n)$ XOR-queries; and thus by Lemmas 2.7 and 2.9, the entire algorithm requires $O(n \log^2 n)$ XOR-queries and is correct w.h.p. □

2.3.3 Independent set (IS) query

In this section we discuss a restricted version of the OR-query, namely the *Independent Set* (IS) query, as studied by, for example, [BHR+18, AL21, RWZ20, AA05, ABK+04, AB19a]. An IS-query consists of specifying two subsets $X \subseteq L$ and $Y \subseteq R$ and asking if there is any edge between some vertex in X and some vertex in Y (or, conversely if $X \cup Y$ forms an independent set)[10].

Claim 2.15. *There is a deterministic algorithm which solves* BMM *with* $O(n \log^2 n)$ IS*-queries.*

Proof. In each iteration, the cutting plane method finds some fractional point $p \in \mathbb{R}^{L \cup R}$, and we are asked to implement a separation oracle FindViolatingEdge for this point. That is we want to determine if any edge in the set $S = \{(u, v) \in L \times R \mid p_u + p_v < 1\}$ exists (and if so find it). With unrestricted OR-queries this is easy (see Claim 2.5), however it might not be the case that this set S is structured like an IS-query. In the case when p is integral, we can define $X = \{v \in L : p_v = 0\}$ and $Y = \{v \in R : p_v = 0\}$, and note that $S = X \times Y$. Hence, in the case of integral p, we can implement FindViolatingEdge using IS-queries: first we binary search on X, and then on Y, to find the violating edge if it exists.

We argue that there always exist an integral point $p' \in \mathbb{Z}^{L \cup R}$ which we can use instead of p when calling the separation oracle FindViolatingEdge. The integral point p' will satisfy the following two properties:

(i) For all pairs $(u, v) \in L \times R$, if $p_u + p_v \geq 1$ then $p'_u + p'_v \geq 1$ too. This means that if we found a violating edge for p', the same edge is also violating for p.

(ii) $\sum p'_v \leq \sum p_v$. This means that if there were no violating edges (i.e. p' formed a vertex cover), we have found a certificate that the maximum matching size is at most $\sum p'_v \leq \sum p_v$.

Indeed, consider the bipartite graph H with edge set $\{(u, v) \in L \times R : p_u + p_v \geq 1\}$. In H, p is a (fractional) vertex cover of size $\sum p_v$. This means that there exists an integral vertex cover of size $\lfloor \sum p_v \rfloor$ in H, since the minimum vertex cover linear program is integral for bipartite graphs.

[10]Generally, an Independent set query specifies only one subset of vertices whereas the *Bipartite* independent set query specifies two disjoint sets of vertices as defined here. However, for bipartite graphs, these two types of queries are equivalent.

Therefore, we pick p' to be an arbitrary such integral vertex cover, and we note that by definition it satisfies the above properties (i) and (ii). $\square$

2.3.4 OR_k-query

Here we discuss the OR-query of *limited width k*, i.e. the OR_k-query. That is, we are only allowed to ask OR-queries over sets $S \subseteq L \times R$ of size $|S| \leq k$. This model turns out to be useful as an intermediary step towards proving tight upper bounds for the quantum edge query model (see Section 2.3.5). Considering this model also helps to unveil the difficulty behind designing *demand query* algorithms (see open problems in Section 1.2 for further discussion) for BPM, pointing to the fact that the barrier is not the size of the query, but rather its locality.

Claim 2.16. *There is a deterministic algorithm which solves* BMM *with* $O(n \log^2 n)$ OR_n*-queries.*

In fact, we show, via an amortization argument, that *any* OR-query algorithm (for an arbitrary graph problem) can be simulated with OR_k-queries with an additive overhead dependent on k.

Lemma 2.17. *Any* OR*-query algorithm $\mathcal{A}$ (for any graph problem) making q queries can be converted to an* OR_k*-algorithm making $q + \lfloor \frac{n^2}{k} \rfloor$ queries.*

Proof. The main idea of the proof is the following: Every time an OR-query answers "NO", we know that none of the queried edges are present in the graph G. This is an important piece of information that helps us save queries in the future. More formally, we use the following amortization argument.

We keep track of a set $E^c \subseteq L \times R$ of pairs (u, v) which we know are **not** edges of G, that is $E \cap E^c = \emptyset$. Whenever $\mathcal{A}$ issues an OR-query $S \subseteq L \times R$ we do the following. Let $S_1, S_2, \ldots, S_r$ be a partition of $S \setminus E^c$ so that $|S_1| = |S_2| = \ldots |S_{r-1}| = k$, and $|S_r| \leq k$. We issue the OR_k-queries $S_1, S_2, \ldots, S_r$ sequentially, in order, until we get a "YES" answer which we return to $\mathcal{A}$ (or else, after we have received "NO" from all the sets we return a "NO" answer to $\mathcal{A}$). For the last query which we made (which was either a query to S_r, or a query which returned "YES"), we charge the cost to $\mathcal{A}$. In total, we thus charge at most q cost to $\mathcal{A}$: one per OR-query $\mathcal{A}$ issues.

For all the queries to S_i which got "NO" answers and for which $i < r$, we update $E^c \leftarrow E^c \cup S_i$. Hence, for each such "NO" answer we have increased the size of E^c by k. Note that this can happen at most $\lfloor \frac{n^2}{k} \rfloor$ times. So, other than the q queries charged to $\mathcal{A}$, we have made at most $\lfloor \frac{n^2}{k} \rfloor$ queries. Hence, in total, we made $q + \lfloor \frac{n^2}{k} \rfloor$ OR_k-queries to simulate the q many OR-queries from $\mathcal{A}$. $\square$

Plugging in our $O(n \log^2 n)$ OR-query algorithm to the above lemma yields Claim 2.16.

2.3.5 Quantum edge query (Q_2)

In this section we consider the quantum **edge**-query model. See [BdW02] for a formal definition and [NC16] for a more extensive background on quantum computing.

Claim 2.18. *The quantum edge query complexity of solving* BMM *is* $O(n\sqrt{n} \log^2 n)$.

We use our OR_k-query algorithm (Lemma 2.17) together with a well known quantum result (Lemma 2.19) regarding the quantum query complexity of the OR function.

Lemma 2.19 (Grover Search [Gro96])**.** *There is quantum query algorithm that computes w.h.p. the* OR *function over k bits with query complexity $O(\sqrt{k} \log k)$.*

Proof of Claim 2.18. Consider the instance of Lemma 2.17 where we put $k = \frac{n}{\log^2 n}$, then we obtain an OR_k-query algorithm for BMM with $O(n \log^2 n)$ queries. Each such OR_k-query can be simulated, w.h.p., using $O(\sqrt{k} \log n) = O(\sqrt{n})$ quantum edge queries, by Lemma 2.19. $\square$

3 Weighted and Vertex-Capacitated Variants

In this section we show that the cutting planes method is strong enough to be generalized to solve weighted and (vertex-)capacitated problems, for example *max-cost b-matching.* As an application, we also show how to solve *unique bipartite perfect matching* (UBPM) in Section 3.3. For the weighted problems, we focus on the two-party edge-partition communication setting, since there is no natural generalization of the OR-queries.

Theorem 3.1. *Given that all the weights/costs/capacities are integers polynomially large in n, we can solve the following problems*[11] *in the two-party edge-partition communication setting, using $O(n \log^2 n)$ bits of communication.*

(i) Maximum-cost bipartite perfect b-matching.

(ii) Maximum-cost bipartite b-matching.

(iii) Vertex-capacitated minimum-cost (s,t)-flow.

(iv) Transshipment (a.k.a. uncapacitated minimum-cost flow).

(v) Negative-weight single source shortest path.[12]

(vi) Minimum mean cycle.

(vii) Deterministic Markov Decision Process (MDP).

Reductions. Problems (i), (ii), (iii), (iv) are equivalent, and problems (v), (vi), (vii) can all be reduced to, for example, (i). All these reduction are shown in [BLN⁺20] (and can be verified as rectangular reductions, i.e., compatible with the two-party communication setting), *except* that (ii) (i.e. max-cost, not-necessarily-perfect, bipartite b-matching) can solve any of (actually really all of): (i), (iii), (iv); which we show in Section 3.1.

These reductions allow us to focus on a single one of these problems. We pick item (ii), that is *max-cost (not-necessarily-perfect)*[13] *bipartite b-matching*, which is the one which most closely resembles the unweighted bipartite matching problem. In Section 3.2 we show how the cutting planes framework can be generalized to work with the costs c and demand vector b.

Definition 3.2 (b-matching)**.** Given a graph $G = (V, E)$, a demand vector $b \in \mathbb{Z}_{\geq 0}^V$, and edge-costs $c \in \mathbb{Z}^E$, we call a vector $y \in \mathbb{Z}_{\geq 0}^E$ a *b-matching* (or a *fractional* b-matching if we allow $y \in \mathbb{R}_{\geq 0}^E$) if $\sum_{e \in \delta(v)} y_e \leq b_v$ for all $v \in V$ (where $\delta(v)$ is the set of edges incident to v). If $\sum_{e \in \delta(v)} y_e = b_v$ for all $v \in V$, then y is a *perfect* b-matching. The *cost* (or *weight*) of y is $\sum_{e \in E} c_e y_e$.

3.1 Max-cost perfect b-matching $\rightarrow$ Max-cost b-matching

We can reduce the perfect variant to the not-necessarily-perfect one. Suppose we are given an instance $(G = (V, E), b \in \mathbb{Z}_{\geq 0}^V, c \in \mathbb{Z}^E)$ of the perfect variant which we wish to solve. Firstly, we may assume that the costs are non-negative, since adding a constant W to all costs will increase the cost of a *perfect* b-matching by exactly $W \frac{\sum_{v \in V} b_v}{2}$.

[11] See [BLN⁺20, Section 8.6 (full version)] for a more extensive discussion of these variants.

[12] Although the reduction shown in [BLN⁺20] is randomized, we note that it can be made deterministic by noticing that both parties in the end will know all the edges of a shortest path tree.

[13] Some of the other problems, e.g. *perfect* b-matching, can have unbounded dual linear programs which make them trickier to work with in the cutting planes framework.

If we just solve max-cost b-matching, we in general do not obtain a perfect b-matching, since matchings of smaller cardinality might have higher cost. To encourage the max-cost b-matching to prioritize perfect matchings over non-perfect matchings, we simply add a large integer W to all the the costs. That is we use the cost function $c'_e = c_e + W$ instead (again, we can do this since we know exactly how this will affect the cost of a perfect b-matching). If W is sufficiently large (in particular set $W := 1 + |V| \cdot \max c_e$), any max-cost b-matching will also be a perfect b-matching (if one exist).

If it was not, suppose M is a non-perfect b-matching and of maximum cost for c', in a graph which allows a perfect b-matching. Then there must exist an augmenting path in M of length $2\ell + 1 \leq |V|$ (where we add $\ell + 1$ edges and remove ℓ). The total cost (w.r.t. c') of the added edges is now at least $(\ell+1)W$, while the cost of the removed edges is at most $\ell(W + \max c_e) \leq \ell W + |V| \max c_e < \ell W + W = (\ell+1)W$. That is we added more cost than we removed, hence contradicting that M was of maximum cost.

3.2 Cutting planes method for max-cost bipartite b-matching

In this section we briefly explain how the cutting planes algorithm can solve the *max-cost bipartite b-matching*, and hence prove Theorem 1.2. The details are postponed to Appendix A. The main result of this section is the following:

Lemma 3.3. *Max-cost bipartite b-matching can be solved using $O(n \log^2(nW))$ communication, where $W := \max\{\max |c_e|, \max b_v\}$ is the largest number in the input.*

Let $G = (V, E)$ (bipartite with $|V| = n$), $b \in \mathbb{Z}^V_{\geq 0}$ and $c \in \mathbb{Z}^E$ be an instance of the max-cost bipartite b-matching problem. We assume the edges (together with their costs) are partitioned between two parties Alice and Bob, say Alice owns E_A (together with c_e for $e \in E_A$) and Bob E_B (together with c_e for $e \in E_B$). We assume both players know the demands b (otherwise it can be communicated in $O(n \log W)$ bits).

Dual linear program. Similarly as for the unweighted bipartite matching problem, we run a cutting planes algorithm on the dual linear program ($\mathcal{P}^{(G,b,c)}$) (refer to the constraints of Definition 3.2 for the primal linear program). We can think of $x \in (\mathcal{P}^{(G,b,c)})$ as a generalized version of a vertex cover, and an optimal solution would be one of minimum cost (w.r.t. costs b_v). Similarly to the uncapacitated and unweighted case, since the graph is bipartite, ($\mathcal{P}^{(G,b,c)}$) is integral, and thus has an *integral* optimal solution.

$$\begin{array}{lll} \min & \sum_{v \in V} b_v x_v & \\ \text{s.t.} & x_u + x_v \geq c_{uv} & \forall (u,v) \in E \\ & x_v \geq 0 & \forall v \in V \end{array} \tag{$\mathcal{P}^{(G,b,c)}$}$$

Claim 3.4. *Any optimal solution x^* to ($\mathcal{P}^{(G,b,c)}$) has $x^*_v \leq W$ for all $v \in V$.*

Proof. If this is not the case, we can decrease x^*_v without violating any of the constraints, and the objective value $\sum b_v x_v$ becomes smaller. □

This motivates the following feasibility polytope ($\mathcal{P}^{(G,b,c)}_F$), which can be used to check if ($\mathcal{P}^{(G,b,c)}$) has a solution with objective value at most F, for any integer $F \in \mathbb{Z}$.

$$\begin{array}{ll} \sum_{v \in V} b_v x_v \leq F + \frac{1}{3} & \\ x_u + x_v \geq c_{uv} & \forall (u,v) \in E \\ 0 \leq x_v \leq W + 1 & \forall v \in V \end{array} \tag{$\mathcal{P}^{(G,b,c)}_F$}$$

Modifications to the algorithm. Like for the unweighted and uncapacitated case, we can show that if this polytope is non-empty, then it has significantly large volume (Lemma 3.5, whose proof is in Appendix A). This means that the cutting plane algorithm can terminate whenever the volume becomes too small. The only modifications we need to make to Algorithm 1 are thus the following:

- We start with a larger initial polytope $P_0 = [0, W+1]^V \cap \{x \in \mathbb{R}^V : \sum b_v x_v \leq F + \frac{1}{3}\}$.
- When we check for (and add) violating constraints we also use the edge-cost c_{uv}. That is an edge (u, v, c_{uv}) is violating if $p_i^u + p_i^v < c_{uv}$, and the corresponding constraint we add is "$x_u + x_v \geq c_{uv}$".
- We terminate when the volume is less than $(\frac{1}{20nW})^n$ (see Lemma 3.5).

Lemma 3.5 (Generalization of Lemma 2.3)**.** *If* $(\mathcal{P}_F^{(G,b,c)})$ *is non-empty, then* $\mathrm{vol}(\mathcal{P}_F^{(G,b,c)}) \geq \left(\frac{1}{20nW}\right)^n$*.*

Observation 3.6 (Generalization of Claim 2.5)**.** *We can communicate a single violated constraint with* $2\log(n) + \log(W) + 1 = O(\log(nW))$ *bits of communication.*

Total communication. The generalized algorithm will start with initial polytope $P_0 \subseteq [0, W+1]^n$, and terminate whenever $\mathrm{vol}(P_i)$ becomes smaller than $\left(\frac{1}{20nW}\right)^n$ (Lemma 3.5). In each iteration the volume is cut down by a constant fraction (Lemma 2.8). Hence we need $O\left(\log\left(W^n / \left(\frac{1}{20nW}\right)^n\right)\right) = O(n\log(nW))$ iterations. Each iteration needs $O(\log(nW))$ bits of communication, for a total of $O(n\log^2(nW))$ bits of communication (Observation 3.6), proving Lemma 3.3. We also note that the same standard trick to convert the *decision* version to the *optimization* version (Section 2.2) works here as well.

3.3 Application: Unique Bipartite Perfect Matching

In the *unique bipartite matching problem*, or UBPM for short, we are asked to determine if an (unweighted) bipartite graph has a *unique* perfect matching. The interplay between UBPM and BPM is quite subtle, by some measures, e.g. certificate complexity, the former is known to be strictly harder than the latter, in other settings, such as sequential, simple near linear time algorithms for UBPM have been known since the turn of the century [GKT01]. While for BPM, only a very recent line of work, employing heavy machinery from continuous optimization and dynamic data structures culminated in a near linear time algorithm for BPM [CKL+22]. In this section, we show that our upper bounds also hold for the UBPM problem, both for the communication and query models.

Theorem 3.7. *The* UBPM *problem can be solved in:*

- $O(n\log^2 n)$ *bits of communication in the deterministic two-party edge-partition communication model.*
- $O(n\log^2 n)$ *deterministic OR-queries.*
- $O(n\log^2 n)$ *randomized XOR-queries, w.h.p.*
- $\tilde{O}(n\sqrt{n})$ *quantum edge queries.*

Proof sketch. (Formal proof can in Appendix A).
Our main idea on how to solve UBPM can be summarized in two stages:

(1) First find a perfect matching M (see Theorem 2.1).

(2) Assign weights to the edges as follows: $c_e = 1$ if $e \in M$, and $c_e = 2$ otherwise. After this, we find a *max-cost* perfect matching M' (see Theorem 3.1).

If $M' \neq M$, we have proved that the perfect matching is not unique. Conversely, if $M' = M$, then M must be the *unique* perfect matching, since any other perfect matching (if they would exist) has higher cost. In the communication setting, this argument suffices. For the query models, however, one needs to be a bit more careful since we have not defined what, for example, an OR-query means in the *max-cost* setting. The formal proof of this—which is straightforward, although a bit technical—can be found in Appendix A. The other query-models follow from similar reductions as those in Section 2.3. □

Remark 3.8. We also note that there are alternative ways to solve UBPM after one has solved BPM. Instead of solving *max-cost matching* as the second step, one can instead determine if an alternating cycle (that is a cycle in G where every other edge is in M) exist. The perfect matching M is *unique* if and only if no such cycle exist. Note that finding such a cycle can easily be done in, for example, $O(n \log n)$ OR-queries by running depth-first-searches to detect a directed cycle in the directed residual graph (edges in M go from $R \to L$, other edges from $L \to R$).

4 Communication Lower Bounds

In this section we discuss three communication problems related to OR-queries, AND-queries, and XOR-queries respectively. We show simple lower bounds on the communication complexity of these three problems, and argue that this implies corresponding query lower bounds.

We summarize our obtained query lower bounds below in Theorem 4.1. Note that our lower bounds are asymptotically the same as those obtained in [BN21]. However, while [BN21] employs rather sophisticated mathematical machinery in order to obtain these lower bounds, we obtain them via simple communication complexity reductions from well known functions. In addition to the already known lower bounds of [BN21], our technique also shows one new result: namely that the $\Omega(n^2)$ AND-query lower bound even holds for **randomized** algorithms.

Theorem 4.1. *To solve the bipartite perfect matching (*BPM*) problem one needs to use:*

- $\Omega(n \log n)$ OR-*queries (deterministic).*
- $\Omega(n^2)$ AND-*queries (deterministic or randomized).*
- $\Omega(n^2)$ XOR-*queries (deterministic).*

In this section we give an overview of the main ideas—which are all relatively simple—but we postpone the formal proofs to Appendix B.

Problem setup. We consider a two-party communication setting between two players Alice and Bob. Alice is given a graph $G_A = (V, E_A)$ and Bob a graph $G_B = (V, E_B)$ on the same set of vertices $V = L \cup R$ (where $|L| = |R| = n$). They wish to solve BPM on an aggregate of their graphs. We consider three different types of aggregate graphs (and hence get three different communication problems), naturally corresponding to OR / AND / XOR:

- Union graph $G_\cup = (V, E_A \cup E_B)$.
- Intersection graph $G_\cap = (V, E_A \cap E_B)$.
- Symmetric difference graph $G_\oplus = (V, E_A \oplus E_B)$.

It is not difficult to see that any OR / AND / XOR query algorithms can be simulated in an communication protocol for $G_\cup$ / $G_\cap$ / $G_\oplus$ respectively. As an example, to answer an AND-query over $S \subseteq (L \times R)$ in $G_\cap$, Alice and Bob check locally if $S \subseteq E_A$, respectively if $S \subseteq E_B$, and exchange this information with each-other. This gives the following Lemma 4.2, which we formally prove in Appendix B.

Lemma 4.2. *If there is a query-algorithm $\mathcal{A}$ solving the bipartite-perfect-matching problems using q many* OR */* AND */* XOR *queries, then there is a communication protocol which solve the bipartite-perfect-matching problem on $G_\cup$ / $G_\cap$ / $G_\oplus$ respectively, using $2q$ bits of communication. If $\mathcal{A}$ is deterministic, then so is the communication protocol.*

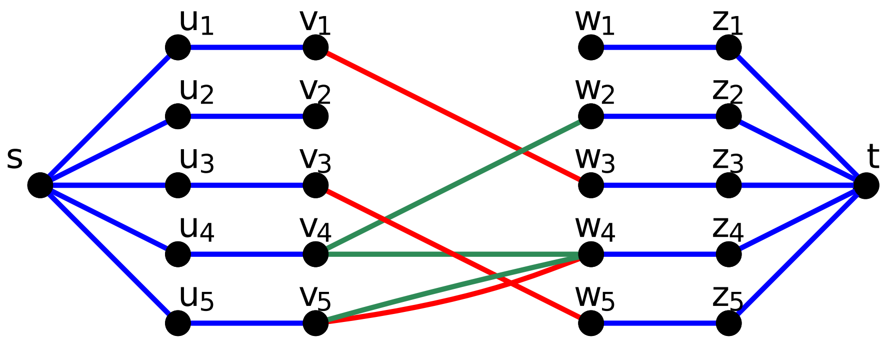

Figure 1: An example of our graph construction for $G_\cap$. The blue edges are known to both parties to be in the aggregate graph $G_\cap$. Between the v and w layers, Alice owns the red edges and Bob the green. The graph $G_\cap$ has a perfect matching if and only if it has an edge between some v_i and w_j.

Intersection Graph (AND). We construct a difficult instance for solving BPM on $G_\cap$ by reducing from Set-Disjointness on $\Theta(n^2)$ bits. Our construction can be seen in Figure 1, where determining if $G_\cap$ has a perfect matching boils down to determining if there exists any edge between some vertex v_i and some vertex w_j in $G_\cap$. This is exactly a Set-Disjointness problem of size $\Theta(n^2)$.

Since Set-Disjointness is known to require linear communication in the number of bits [KN97] (both for deterministic and randomized algorithms), we obtain the following Claim 4.3 whose formal proof can be found in Appendix B.

Claim 4.3. *Solving* BPM *(or* UBPM*) on $G_\cap$ requires $\Omega(n^2)$ bits of communication, even if public randomness is allowed.*

Symmetric Difference Graph (XOR). Our communication lower bound of $G_\oplus$ is very similar to the lower bound on $G_\cap$. We use the same graph-structure (Figure 1), but determining if there is any (v_i, w_j) edge in $G_\oplus$ now corresponds to solving the Equality problem (again on $\Theta(n^2)$ bits) instead of the Set-Disjointness problem. This is since an edge (v_i, w_j) exists if and only if exactly one of Alice or Bob have it, which is if and only if the set of Alice's edges does not equal the set of Bob's edges.

It is well known that Equality exhibits a large gap between its deterministic and randomized communication complexity. While the former is known to be $\Omega(n)$, the latter requires only $O(1)$ bits of communication in the presence of shared randomness (with error probability $< \frac{1}{3}$) [KN97]. This might also provide some intuition why we, in the case of XOR-queries, obtain the separation of $\tilde{O}(n)$ randomized upper bound (Claim 2.12) vs $\Omega(n^2)$ deterministic lower bound. The full proof of Claim 4.4 can be found in Appendix B.

Claim 4.4. *Solving* BPM *on $G_\oplus$ requires $\Omega(n^2)$ bits of communication for any deterministic protocol.*

Union Graph (OR). Our lower bound for $G_\cup$ follows a similar vein, but the graph construction is a bit different. By a standard reduction, the (s, t)-reachability problem on a (not-necessarily-bipartite) n-vertex graph can be solved by solving bipartite perfect-matching on a graph on $2n$ vertices. The (s, t)-reachability problem, in the edge-partition setting, is known to need $\Omega(n \log n)$ bits of communication for any deterministic protocol [HMT88]. Proving any super-linear *randomized* lower bound still remains open. The full proof of Claim 4.5 can be found in Appendix B.

Claim 4.5. *Solving* BPM *on $G_\cup$ requires $\Omega(n \log n)$ bits of communication for any deterministic protocol.*

Acknowledgement

Jan van den Brand is partially funded by ONR BRC grant N00014-18-1-2562 and by the Simons Institute for the Theory of Computing through a Simons-Berkeley Postdoctoral Fellowship.

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 715672. Nanongkai was also partially supported by the Swedish Research Council (Reg. No. 2019-05622).

The authors would like to thank Gal Beniamini for fruitful discussions and anonymous reviewers of FOCS 2022 for their useful suggestions.

References

[AA05] Noga Alon and Vera Asodi. Learning a hidden subgraph. *SIAM J. Discret. Math.*, 18(4):697–712, 2005.

[AB19a] Hasan Abasi and Nader H. Bshouty. On learning graphs with edge-detecting queries. In *ALT*, volume 98 of *Proceedings of Machine Learning Research*, pages 3–30. PMLR, 2019.

[AB19b] Sepehr Assadi and Aaron Bernstein. Towards a unified theory of sparsification for matching problems. In *SOSA*, volume 69 of *OASIcs*, pages 11:1–11:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[AB21] Sepehr Assadi and Soheil Behnezhad. On the robust communication complexity of bipartite matching. In *APPROX-RANDOM*, volume 207 of *LIPIcs*, pages 48:1–48:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[ABK+04] Noga Alon, Richard Beigel, Simon Kasif, Steven Rudich, and Benny Sudakov. Learning a hidden matching. *SIAM J. Comput.*, 33(2):487–501, 2004.

[ACK19] Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Polynomial pass lower bounds for graph streaming algorithms. In *STOC*, pages 265–276. ACM, 2019.

[ACK21] Sepehr Assadi, Deeparnab Chakrabarty, and Sanjeev Khanna. Graph connectivity and single element recovery via linear and OR queries. In *ESA*, volume 204 of *LIPIcs*, pages 7:1–7:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[AD21] Sepehr Assadi and Aditi Dudeja. A simple semi-streaming algorithm for global minimum cuts. In *SOSA*, pages 172–180. SIAM, 2021.

[AJJ$^+$22] Sepehr Assadi, Arun Jambulapati, Yujia Jin, Aaron Sidford, and Kevin Tian. Semi-streaming bipartite matching in fewer passes and optimal space. In *SODA*, pages 627–669. SIAM, 2022.

[AK20] Mohamad Ahmadi and Fabian Kuhn. Distributed maximum matching verification in CONGEST. In *DISC*, volume 179 of *LIPIcs*, pages 37:1–37:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[AKL17] Sepehr Assadi, Sanjeev Khanna, and Yang Li. On estimating maximum matching size in graph streams. In *SODA*, pages 1723–1742. SIAM, 2017.

[AKLY16] Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In *SODA*, pages 1345–1364. SIAM, 2016.

[AKO18] Mohamad Ahmadi, Fabian Kuhn, and Rotem Oshman. Distributed approximate maximum matching in the CONGEST model. In *DISC*, volume 121 of *LIPIcs*, pages 6:1–6:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[AL21] Arinta Auza and Troy Lee. On the query complexity of connectivity with global queries. *CoRR*, abs/2109.02115, 2021.

[ALT21] Sepehr Assadi, S. Cliff Liu, and Robert E. Tarjan. An auction algorithm for bipartite matching in streaming and massively parallel computation models. In *SOSA*, pages 165–171. SIAM, 2021.

[Amb02] Andris Ambainis. Quantum lower bounds by quantum arguments. *J. Comput. Syst. Sci.*, 64(4):750–767, 2002.

[AMV20] Kyriakos Axiotis, Aleksander Madry, and Adrian Vladu. Circulation control for faster minimum cost flow in unit-capacity graphs. In *FOCS*, pages 93–104. IEEE, 2020.

[AR20a] Udit Agarwal and Vijaya Ramachandran. Faster deterministic all pairs shortest paths in congest model. In *SPAA*, pages 11–21. ACM, 2020.

[AR20b] Sepehr Assadi and Ran Raz. Near-quadratic lower bounds for two-pass graph streaming algorithms. In *FOCS*, pages 342–353. IEEE, 2020.

[AV20] Nima Anari and Vijay V. Vazirani. Matching is as easy as the decision problem, in the NC model. In *ITCS*, volume 151 of *LIPIcs*, pages 54:1–54:25. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[BdW02] Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theor. Comput. Sci.*, 288(1):21–43, 2002.

[Ben22a] Gal Beniamini. Algebraic representations of unique bipartite perfect matching. *CoRR*, abs/2203.01071, 2022.

[Ben22b] Gal Beniamini. The approximate degree of bipartite perfect matching. *CoRR*, abs/2004:14318, 2022.

[Ber09] Dimitri P. Bertsekas. Auction algorithms. In *Encyclopedia of Optimization*, pages 128–132. Springer, 2009.

[BFS86] László Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory (preliminary version). In *FOCS*, pages 337–347. IEEE Computer Society, 1986.

[BHR$^+$18] Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. Edge estimation with independent set oracles. In *ITCS*, volume 94 of *LIPIcs*, pages 38:1–38:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[BHR19] Aaron Bernstein, Jacob Holm, and Eva Rotenberg. Online bipartite matching with amortized $O(\log^2 n)$ replacements. *J. ACM*, 66(5):37:1–37:23, 2019.

[BLL$^+$21] Jan van den Brand, Yin Tat Lee, Yang P. Liu, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Minimum cost flows, MDPs, and ℓ_1-regression in nearly linear time for dense instances. In *STOC*, pages 859–869. ACM, 2021.

[BLN$^+$20] 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 *FOCS*, pages 919–930. IEEE, 2020.

[BLSS20] Jan van den Brand, Yin Tat Lee, Aaron Sidford, and Zhao Song. Solving tall dense linear programs in nearly linear time. In *STOC*, pages 775–788. ACM, 2020.

[BN19] Aaron Bernstein and Danupon Nanongkai. Distributed exact weighted all-pairs shortest paths in near-linear time. In *STOC*, pages 334–342. ACM, 2019.

[BN21] Gal Beniamini and Noam Nisan. Bipartite perfect matching as a real polynomial. In *STOC*, pages 1118–1131. ACM, 2021.

[Bra20] Jan van den Brand. A deterministic linear program solver in current matrix multiplication time. In *SODA*, pages 259–278. SIAM, 2020.

[CGL$^+$20] Julia Chuzhoy, Yu Gao, Jason Li, Danupon Nanongkai, Richard Peng, and Thatchaphol Saranurak. A deterministic algorithm for balanced cut with applications to dynamic connectivity, flows, and beyond. In *FOCS*, pages 1158–1167. IEEE, 2020.

[CK07] Amit Chakrabarti and Subhash Khot. Improved lower bounds on the randomized complexity of graph properties. *Random Struct. Algorithms*, 30(3):427–440, 2007.

[CKL$^+$22] Li Chen, Rasmus Kyng, Yang P Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. *CoRR*, abs/2203.00671, 2022.

[CKLM18] Arkadev Chattopadhyay, Michal Koucký, Bruno Loff, and Sagnik Mukhopadhyay. Simulation beats richness: new data-structure lower bounds. In *STOC*, pages 1013–1020. ACM, 2018.

[CKP$^+$21] Lijie Chen, Gillat Kol, Dmitry Paramonov, Raghuvansh R. Saxena, Zhao Song, and Huacheng Yu. Almost optimal super-constant-pass streaming lower bounds for reachability. In *STOC*, pages 570–583. ACM, 2021.

[CLS19] Michael B. Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *STOC*, pages 938–942. ACM, 2019.

[CM20] Shiri Chechik and Doron Mukhtar. Single-source shortest paths in the CONGEST model with improved bound. In *PODC '20: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, August 3-7, 2020*, pages 464–473. ACM, 2020.

[CMSV17] Michael B Cohen, Aleksander Madry, Piotr Sankowski, and Adrian Vladu. Negative-weight shortest paths and unit capacity minimum cost flow in $O(m^{10/7} \log W)$ time. In *SODA*, pages 752–771. SIAM, 2017.

[CQ21] Chandra Chekuri and Kent Quanrud. Isolating cuts, (bi-)submodularity, and faster algorithms for connectivity. In *ICALP*, volume 198 of *LIPIcs*, pages 50:1–50:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[DEMN21] Michal Dory, Yuval Efron, Sagnik Mukhopadhyay, and Danupon Nanongkai. Distributed weighted min-cut in nearly-optimal time. In *STOC*, pages 1144–1153. ACM, 2021.

[DHHM06] Christoph Dürr, Mark Heiligman, Peter Høyer, and Mehdi Mhalla. Quantum query complexity of some graph problems. *SIAM J. Comput.*, 35(6):1310–1328, 2006.

[DHK$^+$12] Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of distributed approximation. *SIAM J. Comput.*, 41(5):1235–1265, 2012.

[DNO19] Shahar Dobzinski, Noam Nisan, and Sigal Oren. Economic efficiency requires interaction. *Games Econ. Behav.*, 118:589–608, 2019.

[DP89] Pavol Duris and Pavel Pudlák. On the communication complexity of planarity. In *FCT*, volume 380 of *Lecture Notes in Computer Science*, pages 145–147. Springer, 1989.

[DS08] Samuel I. Daitch and Daniel A. Spielman. Faster approximate lossy generalized flow via interior point algorithms. In *STOC*, pages 451–460. ACM, 2008.

[Elk20] Michael Elkin. Distributed exact shortest paths in sublinear time. *J. ACM*, 67(3):15:1–15:36, 2020.

[FF56] Lester Randolph Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8:399–404, 1956.

[FGL$^+$21] Sebastian Forster, Gramoz Goranci, Yang P. Liu, Richard Peng, Xiaorui Sun, and Mingquan Ye. Minor sparsifiers and the distributed laplacian paradigm. In *FOCS*, pages 989–999. IEEE, 2021.

[FGT21] Stephen A. Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite perfect matching is in quasi-nc. *SIAM J. Comput.*, 50(3), 2021.

[FKM$^+$05] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005.

[FN18] Sebastian Forster and Danupon Nanongkai. A faster distributed single-source shortest paths algorithm. In *FOCS*, pages 686–697. IEEE Computer Society, 2018.

[Gal16] François Le Gall. Further algebraic algorithms in the congested clique model and applications to graph-theoretic problems. In *DISC*, volume 9888 of *Lecture Notes in Computer Science*, pages 57–70. Springer, 2016.

[GG17] Shafi Goldwasser and Ofer Grossman. Bipartite perfect matching in pseudo-deterministic NC. In *ICALP*, volume 80 of *LIPIcs*, pages 87:1–87:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[GHS83] Robert G. Gallager, Pierre A. Humblet, and Philip M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Trans. Program. Lang. Syst.*, 5(1):66–77, 1983.

[GKK12] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *SODA*, pages 468–485. SIAM, 2012.

[GKK$^+$18] Mohsen Ghaffari, Andreas Karrenbauer, Fabian Kuhn, Christoph Lenzen, and Boaz Patt-Shamir. Near-optimal distributed maximum flow. *SIAM J. Comput.*, 47(6):2078–2117, 2018.

[GKT01] Harold N. Gabow, Haim Kaplan, and Robert Endre Tarjan. Unique maximum matching algorithms. *J. Algorithms*, 40(2):159–183, 2001.

[GL18] Mohsen Ghaffari and Jason Li. Improved distributed algorithms for exact shortest paths. In *STOC*, pages 431–444. ACM, 2018.

[GNT20] Mohsen Ghaffari, Krzysztof Nowicki, and Mikkel Thorup. Faster algorithms for edge connectivity via random 2-out contractions. In *SODA*, pages 1260–1279. SIAM, 2020.

[GO16] Venkatesan Guruswami and Krzysztof Onak. Superlinear lower bounds for multipass graph processing. *Algorithmica*, 76(3):654–683, 2016.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *STOC*, pages 212–219. ACM, 1996.

[Grü60] Branko Grünbaum. Partitions of mass-distributions and of convex bodies by hyperplanes. *Pacific Journal of Mathematics*, 10:1257–1261, 1960.

[Haj91] Péter Hajnal. An omega(n$^{4/3}$) lower bound on the randomized complexity of graph properties. *Comb.*, 11(2):131–143, 1991.

[HHL18] Hamed Hatami, Kaave Hosseini, and Shachar Lovett. Structure of protocols for XOR functions. *SIAM J. Comput.*, 47(1):208–217, 2018.

[HK73] John E. Hopcroft and Richard M. Karp. An n$^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.

[HKN21] Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. A deterministic almost-tight distributed algorithm for approximating single-source shortest paths. *SIAM J. Comput.*, 50(3), 2021.

[HMT88] András Hajnal, Wolfgang Maass, and György Turán. On the communication complexity of graph properties. In *STOC*, pages 186–191. ACM, 1988.

[HMT06] Thanh Minh Hoang, Meena Mahajan, and Thomas Thierauf. On the bipartite unique perfect matching problem. In *ICALP*, volume 4051 of *Lecture Notes in Computer Science*, pages 453–464. Springer, 2006.

[HRVZ15] Zengfeng Huang, Bozidar Radunovic, Milan Vojnovic, and Qin Zhang. Communication complexity of approximate matching in distributed graphs. In *STACS*, volume 30 of *LIPIcs*, pages 460–473. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[HRVZ20] Zengfeng Huang, Bozidar Radunovic, Milan Vojnovic, and Qin Zhang. Communication complexity of approximate maximum matching in the message-passing model. *Distributed Comput.*, 33(6):515–531, 2020.

[HWZ21] Bernhard Haeupler, David Wajc, and Goran Zuzic. Universally-optimal distributed algorithms for known topologies. In *STOC*, pages 1166–1179. ACM, 2021.

[II86] Amos Israeli and Alon Itai. A fast and simple randomized parallel algorithm for maximal matching. *Inf. Process. Lett.*, 22(2):77–80, 1986.

[IKL$^+$12] Gábor Ivanyos, Hartmut Klauck, Troy Lee, Miklos Santha, and Ronald de Wolf. New bounds on the classical and quantum communication complexity of some graph properties. In *FSTTCS*, volume 18 of *LIPIcs*, pages 148–159. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012.

[JST20] Yujia Jin, Aaron Sidford, and Kevin Tian. Semi-streaming bipartite matching in fewer passes and less space. *CoRR*, abs/2011.03495, 2020.

[Kap21] Michael Kapralov. Space lower bounds for approximating maximum matching in the edge arrival model. In *SODA*, pages 1874–1893. SIAM, 2021.

[KKS14] Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *SODA*, pages 734–751. SIAM, 2014.

[KM93] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. *SIAM J. Comput.*, 22(6):1331–1348, 1993.

[KMNT20] Michael Kapralov, Slobodan Mitrovic, Ashkan Norouzi-Fard, and Jakab Tardos. Space efficient approximation to maximum matching size from uniform edge samples. In *SODA*, pages 1753–1772. SIAM, 2020.

[KMT21] Michael Kapralov, Gilbert Maystre, and Jakab Tardos. Communication efficient coresets for maximum matching. In *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*, pages 156–164. SIAM, 2021.

[KN97] Eyal Kushilevitz and Noam Nisan. *Communication complexity.* Cambridge University Press, 1997.

[KP98] Shay Kutten and David Peleg. Fast distributed construction of small k-dominating sets and applications. *J. Algorithms*, 28(1):40–66, 1998.

[KSS84] Jeff Kahn, Michael E. Saks, and Dean Sturtevant. A topological approach to evasiveness. *Comb.*, 4(4):297–306, 1984.

[KUW85] Richard M. Karp, Eli Upfal, and Avi Wigderson. Constructing a perfect matching is in random NC. In *STOC*, pages 22–32. ACM, 1985.

[KVKV11] Bernhard H Korte, Jens Vygen, B Korte, and J Vygen. *Combinatorial optimization*, volume 1. Springer, 2011.

[KVV85] Dexter Kozen, Umesh V. Vazirani, and Vijay V. Vazirani. NC algorithms for comparability graphs, interval gaphs, and testing for unique perfect matching. In *FSTTCS*, volume 206 of *Lecture Notes in Computer Science*, pages 496–503. Springer, 1985.

[KVV90] Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In *STOC*, pages 352–358. ACM, 1990.

[Lev65] A. Y. Levin. On an algorithm for the minimization of convex functions over convex functions. *Soviet Mathematics Doklady*, 160:1244–1247, 1965.

[LL15] Cedric Yen-Yu Lin and Han-Hsuan Lin. Upper bounds on quantum query complexity inspired by the elitzur-vaidman bomb tester. In *Computational Complexity Conference*, volume 33 of *LIPIcs*, pages 537–566. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[LNP$^+$21] Jason Li, Danupon Nanongkai, Debmalya Panigrahi, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Vertex connectivity in poly-logarithmic max-flows. In *STOC*, pages 317–329. ACM, 2021.

[Lov79] László Lovász. On determinants, matchings, and random algorithms. In *FCT*, pages 565–574. Akademie-Verlag, Berlin, 1979.

[LP20] Jason Li and Debmalya Panigrahi. Deterministic min-cut in poly-logarithmic max-flows. In *FOCS*, pages 85–92. IEEE, 2020.

[LP21] Jason Li and Debmalya Panigrahi. Approximate gomory-hu tree is faster than n - 1 max-flows. In *STOC*, pages 1738–1748. ACM, 2021.

[LS14] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in õ(sqrt(rank)) iterations and faster algorithms for maximum flow. In *FOCS*, pages 424–433, 2014.

[LS20] Yang P. Liu and Aaron Sidford. Faster energy maximization for faster maximum flow. In *STOC*, pages 803–814. ACM, 2020.

[LSW15] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *FOCS*, pages 1049–1065. IEEE Computer Society, 2015.

[LSZ20] S. Cliff Liu, Zhao Song, and Hengjie Zhang. Breaking the n-pass barrier: A streaming algorithm for maximum weight bipartite matching. *CoRR*, abs/2009.06106, 2020.

[Lub86] Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15(4):1036–1053, 1986.

[Mad13] Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *FOCS*, pages 253–262. IEEE Computer Society, 2013.

[Mad16] Aleksander Madry. Computing maximum flow with augmenting electrical flows. In *FOCS*, pages 593–602. IEEE, 2016.

[MN20] Sagnik Mukhopadhyay and Danupon Nanongkai. Weighted min-cut: sequential, cut-query, and streaming algorithms. In *STOC*, pages 496–509. ACM, 2020.

[MN21] Sagnik Mukhopadhyay and Danupon Nanongkai. A note on isolating cut lemma for submodular function minimization. *CoRR*, abs/2103.15724, 2021.

[MO09] Ashley Montanaro and Tobias Osborne. On the communication complexity of XOR functions. *CoRR*, abs/0909.3392, 2009.

[MS04] Marcin Mucha and Piotr Sankowski. Maximum matchings via gaussian elimination. In *FOCS*, pages 248–255. IEEE Computer Society, 2004.

[MS20] Nikhil S. Mande and Swagato Sanyal. On parity decision trees for fourier-sparse boolean functions. In *FSTTCS*, volume 182 of *LIPIcs*, pages 29:1–29:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[Nan14] Danupon Nanongkai. Distributed approximation algorithms for weighted shortest paths. In *STOC*, pages 565–573. ACM, 2014.

[NC16] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information (10th Anniversary edition)*. Cambridge University Press, 2016.

[New65] Donald J Newman. Location of the maximum on unimodal surfaces. *Journal of the ACM (JACM)*, 12(3):395–398, 1965.

[Nis21] Noam Nisan. The demand query model for bipartite matching. In *SODA*, pages 592–599. SIAM, 2021.

[NS14] Danupon Nanongkai and Hsin-Hao Su. Almost-tight distributed minimum cut algorithms. In *DISC*, volume 8784 of *Lecture Notes in Computer Science*, pages 439–453. Springer, 2014.

[PS82] Christos H. Papadimitriou and Michael Sipser. Communication complexity. In *STOC*, pages 196–200. ACM, 1982.

[Raz92] Alexander A. Razborov. On the distributional complexity of disjointness. *Theor. Comput. Sci.*, 106(2):385–390, 1992.

[Ros73] Arnold L. Rosenberg. On the time required to recognize properties of graphs: a problem. *SIGACT News*, 5(4):15–16, 1973.

[Rot82] Alvin E. Roth. The economics of matching: Stability and incentives. *Math. Oper. Res.*, 7(4):617–628, 1982.

[RSW22] Mohammad Roghani, Amin Saberi, and David Wajc. Beating the folklore algorithm for dynamic matching. In *ITCS*, volume 215 of *LIPIcs*, pages 111:1–111:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[RV75] Ronald L. Rivest and Jean Vuillemin. A generalization and proof of the aanderaa-rosenberg conjecture. In *STOC*, pages 6–11. ACM, 1975.

[RV76] Ronald L. Rivest and Jean Vuillemin. On recognizing graph properties from adjacency matrices. *Theor. Comput. Sci.*, 3(3):371–384, 1976.

[RWZ20] Cyrus Rashtchian, David P. Woodruff, and Hanlin Zhu. Vector-matrix-vector queries for solving linear algebra, statistics, and graph problems. In *APPROX-RANDOM*, volume 176 of *LIPIcs*, pages 26:1–26:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[Ten02] Moshe Tennenholtz. Tractable combinatorial auctions and b-matching. *Artif. Intell.*, 140(1/2):231–243, 2002.

[Vai89] Pravin M. Vaidya. A new algorithm for minimizing convex functions over convex sets (extended abstract). In *FOCS*, pages 338–343. IEEE Computer Society, 1989.

[VWW20] Santosh S. Vempala, Ruosong Wang, and David P. Woodruff. The communication complexity of optimization. In *SODA*, pages 1733–1752. SIAM, 2020.

[Yao88] Andrew Chi-Chih Yao. Monotone bipartite graph properties are evasive. *SIAM J. Comput.*, 17(3):517–520, 1988.

[Zha04] Shengyu Zhang. On the power of ambainis's lower bounds. In *ICALP*, volume 3142 of *Lecture Notes in Computer Science*, pages 1238–1250. Springer, 2004.

A Omitted proofs from Section 3

Lemma 3.5 (Generalization of Lemma 2.3). *If* $(\mathcal{P}_F^{(G,b,c)})$ *is non-empty, then* $\operatorname{vol}(\mathcal{P}_F^{(G,b,c)}) \geq \left(\frac{1}{20nW}\right)^n$.

Proof. Similarly to Lemma 2.3, let x be some integral feasible solution to $(\mathcal{P}_F^{(G,b,c)})$, we know such a solution exists since $(\mathcal{P}^{(G,b,c)})$ has an optimal integer solution, and by assumption that $(\mathcal{P}_F^{(G,b,c)})$ is not empty, we can deduce that its value is at most $F+\frac{1}{3}$, which is at most F since it is integer. For the same reason, we can also, by Claim 3.4, assume w.l.o.g. that $x_v \leq W$ for all $v \in V$. Our goal is to prove that $(\mathcal{P}_F^{(G,b,c)})$ contains a cube of dimensions $\frac{1}{20nW}$, thus concluding the proof. Note that for each $v \in V$ we can independently increase the value of x_v by any value in the range $[0, \frac{1}{20nW}]$ while maintaining the feasibility of the resulting point w.r.t. $(\mathcal{P}_F^{(G,b,c)})$. This is due to the following observations.

- The constraint $\sum\limits_{v\in V} b_v x_v \leq F + \frac{1}{3}$ remains valid as as x has value F, and we increase each coordinate by at most $\frac{1}{20nW}$, as there are $2n$ vertices and $b_v \leq W$ for all $v \in V$, it means that we increase the value of the solution by at most $\frac{1}{10}$, which is less than $\frac{1}{3}$.
- All edge constrains $x_v + x_u \geq c_{uv}$ for all $(u,v) \in E$ clearly remain valid as we are only increasing the value of variables.
- As 3.4 allows us to assume that $x_v \leq W$ for all $v \in V$ for the original x, the constraints $0 \leq x_v \leq W+1$ for all $v \in V$ remain valid as well as each entry is increased by at most $\left(\frac{1}{20nW}\right)$. □

Theorem 3.7. *The* UBPM *problem can be solved in:*

- *$O(n\log^2 n)$ bits of communication in the deterministic two-party edge-partition communication model.*
- *$O(n\log^2 n)$ deterministic OR-queries.*

- *$O(n \log^2 n)$ randomized XOR-queries, w.h.p.*
- *$\tilde{O}(n\sqrt{n})$ quantum edge queries.*

Proof. Here we give the formal proof that we can determine if an unweighted bipartite graph $G = (V, E)$ (with $V = L \cup R$, $|L| = |R| = n$) has a *unique* bipartite matching in all our different models. We show that $O(n \log^2 n)$ OR-queries suffices, and the other models follows similarly as in Section 2.3.

We start by obtaining a maximum matching M of G, by Lemma 2.4. If M is not a perfect matching, G admits no perfect matching, and we are done. From now on assume that M is a perfect matching.

We now construct an instance of a weighted matching. Let $c_e = 10n$ if $e \in M$ and $c_e = 10n + 1$ otherwise. We wish to solve the max-cost matching problem with edge costs c. We will soon explain how to do this with OR-queries, but first we show how doing this completes the proof.

Let M' be a max-cost matching in G with respect to the costs c. Note that M has cost $10n^2$, so we know that M' has cost at least $10n^2$ too. First we argue that M' must be a perfect matching of G. This is because any non-perfect matching has weight at most $(n-1)(10n+1) < 10n^2$.

- If $M' \neq M$, we have found two perfect matchings of the graph, and are done (G does not have a *unique* perfect matching).
- If $M' = M$, we conclude that M is the *unique* perfect matching in the graph. This is since any other perfect matching of G, if they would exist, would have had strictly larger cost.

Now we return to explaining how to solve the max-cost (w.r.t. the cost c) matching using only OR-queries. Note that we already know the weights of all (potential) edges (either $10n$ if they are in M, which we know; or $10n+1$ otherwise), and this is not something which needs to be found out using OR-queries. By the above discussion, it suffices to check if any matching of weight strictly more than $10n^2$ exist or not. Hence we use a version of ($\mathcal{P}_F^{(G,b,c)}$) with c being our costs, b the all-ones vector, and $F = 10n^2$, to obtain the following polytope ($\mathcal{P}^{\mathsf{UBPM}}$):

$$\begin{aligned}
&\sum_{v\in V} x_v \le 10n^2 + \tfrac{1}{3} && \\
&x_u + x_v \ge 10n && \forall (u,v) \in M \\
&x_u + x_v \ge 10n + 1 && \forall (u,v) \in E \setminus M \\
&0 \le x_v \le 10n + 2 && \forall v \in V
\end{aligned} \qquad (\mathcal{P}^{\mathsf{UBPM}})$$

If this polytope is feasible, then M must be the unique perfect matching, and if it is feasible, another perfect matching M' of higher cost exist.

We note that in the above polytope (which is a special case of the ($\mathcal{P}_F^{(G,b,c)}$) polytope from Section 3.2), we know all the constraints initially except the "$x_u + x_v \ge 10n$ for $(u,v) \in E \setminus M$" constraints. Hence, when running the cutting planes algorithm we may start with all other constraints in our initial polytope. Now, to implement the FindViolatingEdge-subroutine (a.k.a. the separation oracle) given a point p_i, we binary search (with OR-queries) over the set of potentially violated edges, i.e. the set $S = \{(u,v) \in L \times R \mid p_i^u + p_i^v < 10n + 1, (u,v) \notin M\}$ (like in Claim 2.5, but now this set S is a bit different for our problem). Otherwise, the cutting planes algorithm is exactly the same as for the max-cost b-matching in Section 3.2. □

B Omitted proofs from Section 4

Lemma 4.2. *If there is a query-algorithm $\mathcal{A}$ solving the bipartite-perfect-matching problems using q many* OR / AND / XOR *queries, then there is a communication protocol which solve the bipartite-*

perfect-matching problem on $G_\cup$ / $G_\cap$ / $G_\oplus$ respectively, using $2q$ bits of communication. If $\mathcal{A}$ is deterministic, then so is the communication protocol.

Proof. Alice and Bob can simulate a query on the aggregate graph with a single bit sent in each direction as follows:

- An OR-query $S \subseteq L \times R$ can be simulated on $G_\cup$: Alice and Bob locally check if $|S \cap E_A| > 0$, respectively $|S \cap E_B| > 0$, and communicates this to the other party.
- An AND-query $S \subseteq L \times R$ can be simulated on $G_\cap$: Alice and Bob locally check if $|S \cap E_A| = |S|$, respectively $|S \cap E_B| = |S|$, and communicates this to the other party.
- An XOR-query $S \subseteq L \times R$ can be simulated on $G_\oplus$: Alice and Bob locally compute the parity of $|S \cap E_A|$, respectively $|S \cap E_B|$, and communicates this to the other party. If they have the same parity, then the parity of $|S \cap (E_B \oplus E_A)|$ is even, otherwise it is odd. $\square$

Claim 4.3. *Solving* BPM *(or* UBPM*) on $G_\cap$ requires $\Omega(n^2)$ bits of communication, even if public randomness is allowed.*

Proof. Let k be an integer and suppose Alice and Bob are tasked to solve a Set-Disjointness problem on k^2 bits. That is, suppose Alice is given a subset $A \subseteq [k] \times [k]$ and Bob is given a subset $B \subseteq [k] \times [k]$, and they want to determine if $A \cap B$ is empty or not. This is known to require $\Omega(k^2)$ bits of communication, even if public randomness is allowed [Raz92].

We now proceed by setting up two bipartite graphs $G_A = (L \cup R, E_A)$ and $G_B = (L \cup R, E_B)$ such that the graph $G_\cap = (L \cup R, E_A \cap E_B)$ has a perfect matching if and only if $A \cap B \neq \emptyset$. For an illustration, see Figure 1.

The graph will have $4k + 2$ vertices. Let $L = \{s,\ v_1, v_2, \ldots, v_k,\ z_1, z_2, \ldots, z_k\}$ and $R = \{t,\ u_1, u_2, \ldots, u_k,\ w_1, w_2, \ldots, w_k\}$. The edges (s, u_i), (u_i, v_i), (w_i, z_i) and (z_i, t) (for all $i \in [k]$) will be in both Alice's and Bob's graphs. Note that the edges (u_i, v_i) and (w_i, z_i) form an almost-perfect-matching in $G_\cap$: all vertices except s and t are matched. Hence, a perfect-matching exists in $G_\cap$ if and only if there is an augmenting path (with respect to the almost-perfect-matching) between s and t in the graph.

Additionally, for every $(i, j) \in A$ we add the edge (v_i, w_j) to Alice's edges, and similarly for every edge $(i, j) \in B$ we add the edge (v_i, w_j) to Bob's edges.

- If $(i, j) \in A \cap B$ exists, then (v_i, w_j) is an edge of $G_\cap$, and $G_\cap$ has a perfect matching: $(s, u_i, v_i, w_j, z_j, t)$ forms the desired (s, t)-augmenting path.
- On the other hand, if $A \cap B = \emptyset$, then $G_\cap$ has no perfect matching since s and t are in different components in the graph and hence no augmenting path between them exists.

Note that the version of Set-Disjointness where we are promised that $|A \cap B| \leq 1$ is still difficult (i.e. also has an $\Omega(n^2)$ communication lower bound) [Raz92]. This means that our lower bound also holds for the *unique* bipartite matching problem (UBPM), as there can never be more than one perfect matching in this promise version. $\square$

Claim 4.4. *Solving* BPM *on $G_\oplus$ requires $\Omega(n^2)$ bits of communication for any deterministic protocol.*

Proof. Let k be an integer and suppose Alice and Bob are tasked to solve a Equality problem on k^2 bits. That is, suppose Alice is given a subset $A \subseteq [k] \times [k]$ and Bob is given a subset $B \subseteq [k] \times [k]$, and they want to determine if $A = B$ (this is equivalent to determine if $A \oplus B = \emptyset$). This is known to require $\Omega(k^2)$ bits of communication for any deterministic algorithm.

We use a similar graph construction on $4k+2$ vertices as in the $G_\cap$-query setting (again, for an illustration, see Figure 1). The edges (s, u_i), (u_i, v_i), (w_i, z_i) and (z_i, t) (for all $i \in [k]$) will be in be in Alice's graph, but not Bob's (so that all these edges will be $G_\oplus$). Again, note that a perfect-matching exists in $G_\oplus$ if and only if there is an augmenting path (with respect to the almost-perfect-matching $\{(u_i, v_i), (w_i, z_i) : i \in [k]\}$) between s and t in the graph.

Additionally, we add the edge (v_i, w_j) to Alice's graph if $(i, j) \in A$, and similarly add the edge (v_i, w_j) to Bob's graph if $(i, j) \in B$. That is the edge (v_i, w_j) is in $G_\oplus$ if and only if $(i, j) \in A \oplus B$.

- If $(i, j) \in A \oplus B$ exist, then then (v_i, w_j) is an edge of $G_\oplus$, and $G_\oplus$ has a perfect matching: $(s, u_i, v_i, w_j, z_j, t)$ forms the desired (s, t)-augmenting path.
- On the other hand, if $A \oplus B = \emptyset$, then $G_\oplus$ has no perfect matching since s and t are in different components in the graph and hence no augmenting path between them exists. □

Claim 4.5. *Solving* BPM *on $G_\cup$ requires $\Omega(n \log n)$ bits of communication for any deterministic protocol.*

Proof. Suppose the two parties are given an instance of (s, t)-connectivity. That is Alice is given edges F_A and Bob F_B in a k-vertex (not-necessarily-bipartite) graph $H = (V, F_A \cup F_B)$. They are also both given two vertices $s, t \in V$ and want to determine if s and t are in the same connected component in H. This is known to require $\Omega(n \log n)$ bits of communication.

We proceed by constructing bipartite graphs $G_A = (L \cup R, E_A)$ and $G_B = (L \cup R, E_B)$ for Alice and Bob, such that $G_\cup = (L \cup R, E_A \cup E_B)$ has a perfect matching if and only if s and t are connected in H.

We let $L = \{v : v \in V \setminus \{s\}\}$ and $R = \{v' : v \in V \setminus \{t\}\}$. Note that $|L| = |R| = k - 1$. Let the edge (v, v') be in both E_A and E_B (and thus also an edge of $G_\cup$) for all $v \in V \setminus \{s, t\}$. These edges form an almost-perfect matching of size $k - 2$: all vertices except s and t' are matched. Again, $G_\cup$ has a perfect matching if and only if there is an augmenting path (with respect to this partial almost-perfect matching) between s and t in the graph.

For each edge $(v, u) \in F_A$, we add (v, u') (unless $v = t$) and (v', u) (unless $v = s$) to Alice's edges E_A. We do the same for Bob.

Now there is an (s, t)-path in H if and only if there is an augmenting path (w.r.t. the almost-perfect matching $\{(v, v') : v \in V \setminus \{s, t\}\}$) between s and t in $G_\cup$. Indeed, a path $(s, v_1, v_2, \ldots, v_r, t)$ in H corresponds to the augmenting path $(s, v_1', v_1, v_2', v_2, \ldots, v_r', v_r, t)$ in $G_\cup$. □