Nearly Optimal Communication and Query Complexity of Bipartite Matching

Joakim Blikstad
KTH Royal Institute of Technology
blickstad@kth.se

Jan van den Brand
Georgia Institute of Technology
vdbrand@gatech.edu

Yuval Efron
Columbia University
ye2210@columbia.edu

Sagnik Mukhopadhyay
University of Sheffield
s.mukhopadhyay@sheffield.ac.uk

Danupon Nanongkai
MPI-INF, University of Copenhagen and KTH
danupon@gmail.com

Abstract—We settle the complexities of the maximum-cardinality bipartite matching problem (BMM) up to polylogarithmic 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 Turán STOC’88; Ivanyos, Klauck, Lee, Santha, and de Wolf STOC’14; Dobzinski, Nisan, and Oren 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.

Index Terms—F.1.1 Models of Computation, F.1.3 Complexity Measures and Classes, E.2 Analysis of Algorithms and Problem Complexity

I. 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 e.g. [11]–[27], 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. [7], [8], [14], [19], [28], [29]); see Table I for a summary of our results.

a) 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. [28], [30]–[32]). For BMM and BPM, their communication complexities have been extensively studied from several angles and perspectives, including exact solution protocols [8], [14], [28], [31], round restricted protocols [11], [20], [33]–[36], multiparty protocols [11], [37]–[41], approximate solution protocols [23], [39], [42], [43], matrix rank and polynomial representation [44], [45], and economics and combinatorial auctions [46]–[48].

In particular, Hajnal, Maass, and Turán [28] showed a lower bound of $\Omega(n \log n)$ for deterministic protocols. For randomized and quantum protocols, the lower bounds are $\Omega(n)$ [8], [31], [49]. For an upper bound, Ivanyos, Klauck, Lee, Santha, and de Wolf [8] implemented the Hopcroft-Karp algorithm to get an $O(n^{3/2} \log n)$-bit deterministic protocol (see also [14], [19]).

Closing the large gap between existing upper and lower bounds has been mentioned as an open problem in, e.g. [8], [14], [19], [28]. Beniamini and Nisan [45] recently showed that the rank of the communication matrix is $2^{\Omega(n \log n)}$, 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 [20], [21], [33], [34]. In this paper, we resolve this open problem with an $O(n \log^2 n)$ upper bound:

1 [28] 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. [41] has shown a $\Omega(\alpha^2 n k)$ lower bound for $k$-party point-to-point communication model for $\alpha$-approximation of BMM. [8] shows a $\Omega(n)$ quantum communication lower bound by a reduction from inner-product in $F_2$. 

Full version available at https://arxiv.org/abs/2208.02526.

Jan van den Brand was 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 and Mukhopadhyay were 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.
This 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 [18]).

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: Max-min cost bipartite perfect $b$-matching, Max-min 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).

b) 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 [51]–[56] for various graph problems. For BPM, Yao [57] showed that $n^2$ edge queries are necessary in the deterministic setting and Durr, Heiligman, Hoyer, and Mhalla [58] showed an $\Omega(n^2)$ lower bounds for the randomized setting 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 [7] showed a lower bound of $\Omega(n^{1.5})$ by using Ambainis’ adversary method [59] (see [44] for an alternative proof via approximate degree). The best upper bound is, however, at $O(n^{1.75})$ as shown by [50]. 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 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 [60]–[64]. 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 [29], [45], [65].

For AND-query or XOR-query complexity, a recent result of [45] showed that $\Omega(n^2)$ queries are necessary to compute BPM deterministically\(^4\). For OR-query, [45] also showed a deterministic lower bound of $\Omega(n \log n)$. The upper bound of $O(n^{1.5})$ for OR-queries (and, thereby, randomized XOR-queries, see 2.7) can be achieved by simulating the Hopcroft-Karp algorithm [19].

From the above results, it remained open to close the polynomial gaps for quantum and OR-queries (as mentioned in [19], [44]) 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 I. 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: nonsep

- The quantum edge query complexity is $O(n^{1.5} \log^2 n)$.
- The 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 [29], [66]–[68]. Beniamini [29] recently show UBPM lower bounds similar to those for BMM and BPM, i.e. $\Omega(n \log n)$ communication complexity, $\Omega(n^{1.5})$ 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 [29].

**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 Independent set (IS) query model [69]–[74]. 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., $E \cap (X \times Y) \neq \emptyset$).

c) Organization: In I-A, we provide a brief technical overview of our upper and lower bounds. In I-B, we list a few

\(^4\)For XOR-queries, [45] showed that BPM is evasive, i.e., requires $n^2$ queries.

\(^5\)[29] 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.
open problems that naturally arise from our work. II details our various upper bounds, starting with OR-query protocols. In II-C, we show the applications of the OR-query algorithm, namely two party communication complexity (II-C1), randomized XOR-query (II-C2), Independent set query (II-C3), OR-query (II-C4) and quantum edge query (II-C5). In the full version [75], we list different variants of the bipartite matching problem that our technique can solve as well. Furthermore, in the full version [75], we provide lower bounds for solving BPM in OR-, AND- and XOR-query settings.

A. Technical Overview

a) 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. [9], [10], [18], [27], [76]–[84]), 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. \( \Omega(n) \)) which 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 [85]) 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.1 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 [86]–[88], 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 \( O(n) \) 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 [88] instead.

b) 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 setting defined earlier; however, we have already shown in 1.1 that this is not possible. [45] 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 [45]) 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

\[ \begin{array}{|c|c|c|}
\hline
\text{Models} & \text{Previous papers} & \text{This paper} \\
\hline
\text{Two-party communication} & \Omega(n) \text{ Rand, } \Omega(n \log n) \text{ Det, Footnote 1 and 2} & O(n^{1.75}) [8, 14] \text{ Det, Thm 1.1} \\
\hline
\text{Quantum edge query} & \Omega(n^{1.5}) \text{ Det, [44]} & O(n^{1.75}) [50] \text{ Det, Thm 1.3} \\
\hline
\text{OR-query} & \Omega(n) \text{ Rand, } \Omega(n^{1.5}) \text{ Det, [45]} & O(n^{1.75}) \text{ Rand, 2.7 and [19]} \\
\hline
\text{XOR-query} & \Omega(n) \text{ Rand, } \Omega(n^2) \text{ Det [45]} & O(n^{1.75}) \text{ Rand, Thm 1.3} \\
\hline
\text{AND-query} & \Omega(n) \text{ Rand, } \Omega(n^2) \text{ Det [45]} & O(n^2) \text{ Trivial} \\
\hline
\end{array} \]

\[ \text{TABLE I} \]

\text{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.}
(or any other graph function) in the graph \( G' = (V, E_A \cap E_B) \). Our AND-query lower bound now follows from a reduction from the set disjointness problem.

Similarly, Beniamini and Nisan [45] 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, [44], [45] 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.

B. 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 [19], [44]. Like in many settings we consider, the best demand query upper and lower bounds for BMM and BPM are \( \Omega(n^{1.5}) \) 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 [44]) 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) [33]. 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 \( \tilde{\Omega}(n) \)) [90]. Unfortunately, such an efficient protocol for BPM does not exist even when we allow \( o(\sqrt{\log n}) \) rounds [20], [21]. More generally, \( r \)-round protocols are known to require \( n^{1+\Omega(1/r)} \) communication [11], [20]. 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 [91], [92] and matching [111]).

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. [93]–[106]). 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 [16], [107]–[110]. The best lower bound for this problem is \( \Omega(\sqrt{n} + D) \) [110], [111] (see also [112]). The best upper bound is \( O(n \log n) \) [110]. For sparse graphs, the upper bound can be improved to \( O(m^{\emph{1/3}}(\sqrt{n}D^{1/4} + D)) \) via continuous optimization [25]. (Better upper bounds via fast matrix multiplication also exist on the special case of congested clique [109].) 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 \( O(n) \) 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 [113] that was later extended to achieve an \( \tilde{O}(\sqrt{n} + D) \) upper bound in CONGEST [106].) 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. Setting 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

1177
serve as a step toward this goal. Particularly interesting goals are solving \((s, t)\)-max-flow/min-cut in the communication\(^3\), distributed, cut query, and streaming settings (Bounded round communication lower bounds in multi-party communication setting for \((s, t)\)-max-flow/min-cut have been studied in [114]). 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 [115]–[120]. Can these problems be solved as efficiently as max-flow in other settings, e.g., the communication, distributed, and streaming settings?

II. 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 \(O(n \log^3 n)\).
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 \(O(n^{1.5} \log^2 n)\).

a) 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 2.1 (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 \(F + 1\).

b) The vertex cover linear program: For a bipartite graph \(G = (V, E)\) with \(V = L \cup R, |L| = |R| = n\), the following linear program \((P^G)\) 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 \((P^G)\) is integral [121, Section 5], i.e. there exists an integer optimal solution to \((P^G)\).

\[
\begin{align*}
\text{minimize} & \quad \sum_{v \in V} x_v \\
\text{subject to} & \quad x_u + x_v \geq 1 \quad \forall (u, v) \in E \\
& \quad 0 \leq x_v \leq 1 \quad \forall v \in V
\end{align*}
\]

\((P^G)\)

c) 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 II-A), and later (in section II-B) 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 2.1), 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 \((P^G)\) has some feasible solution \(x\) with \(\sum x_v \leq F\). So we define another polytope \((P^G_F)\) as follows:

\[
\begin{align*}
\sum_{v \in V} x_v & \leq F + \frac{1}{3} \\
0 & \leq x_v \leq 1 \quad \forall v \in V
\end{align*}
\]

\((P^G_F)\)

Our decision algorithm either finds a feasible point for the above polytope, or it finds a witness of \(P^G_F\) 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 2.1). 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 \((P^G)\) is integral, it has an integral optimal objective value, which means that if a feasible solution \(x\) of \((P^G_F)\) exists, then there also exists a feasible solution \(x'\) which achieves \(\sum x'_v \leq F\).

Lemma 2.1: For any bipartite graph \(G = (V, E)\), if \(F\) is an integer such that \((P^G_F)\) is non-empty, then \(\text{vol}(P^G_F) \geq \left(\frac{1}{20n}\right)^2 2^n\).

Proof: Let \(x\) be an integral solution for \((P^G_F)\), of value \(F\). Indeed, if \((P^G_F)\) is feasible, then such an \(x\) must exist due to the integrality of \((P^G)\). 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 \([\frac{1}{20n}, 1][\frac{1}{20n}, 1]^n \times [1 - \frac{1}{20n}, 1]^n\) is completely contained in \((P^G_F)\). 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 any value in \([\frac{1}{20n}, 1]\); the point remains feasible for \((P^G_F)\). 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 \( \frac{1}{2} \)).

- 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 \( x_u + x_v \geq 2 - \frac{1}{20n} \); 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 \( \left( \frac{1}{20n} \right)^2 n \) is contained in \((P^G_F)\).

A. OR-query decision algorithm

In this section we describe our cutting planes based OR-query algorithm for solving the feasibility problem on \((P^G_F)\). 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 2.2:** 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 \((P^G_F)\), or else a witness, in the form of a matching of size \( F + 1 \), that \((P^G_F)\) is empty.

a) Center-of-gravity cutting planes method: We are now ready to introduce the cutting planes framework \([86],[87]\). 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 \((P^G_F)\)), 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 \((P^G_F)\), or have determined that no such feasible point exist.

We work in iterations, each iteration \( i \) is characterized by a polyhedron \( P_i \geq (P^G_F) \). We compute the center-of-gravity of \( P_i \), denoted by \( p_i = cg(P_i) \in P_i \), and defined to be \( cg(P_i) = \left( \int_{P_i} z \, dz \right) / \left( \int_{P_i} \, dz \right) \). Note that we know \( P_i \), so our algorithm can compute \( p_i = cg(P_i) \) without using any queries.

Either \( p_i \) is feasible for \((P^G_F)\), 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 \((P^G_F)\) 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 \( \text{FindViolatingEdge} \), which we explain how to do with OR-queries in Claim 2.2. 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 \((P^G_F)\) is feasible

1. \( P_0 \leftarrow \{ x \in [0,1]^{2n} \mid \sum_{v \in V} x_v \leq F + \frac{1}{7} \} \)
2. \( E' \leftarrow \emptyset \)
3. \( i \leftarrow 0 \)
4. \( \text{while } \text{vol}(P_i) \geq \left( \frac{1}{20n} \right)^2 n \text{ do} \)
5. \( p_i \leftarrow \text{cg}(P_i) \)
6. \( (u, v) \leftarrow \text{FindViolatingEdge}(E', p_i) \)
7. \( \text{if no edge was found then} \)
8. \( \text{return } \text{“Feasible”} = \text{/ } p_i \text{ is feasible for } (P^G_F) \)
9. \( E' \leftarrow E' \cup \{(u, v)\} \)
10. \( P_{i+1} \leftarrow P_i \cap \{ x \in \mathbb{R}^{2n} : x_u + x_v \geq 1 \} \)
11. \( i \leftarrow i + 1 \)
12. \( \text{return } \text{“Infeasible”} \)

**Claim 2.2:** For every \((P^G_F)\) containing a matching of size \( F + 1 \) the \text{FindViolatingEdge} algorithm can find a violating edge or else determine that none exist.

**Proof:** Given the center-of-gravity point \( p_i \), we let \( S = \{ (u, v) \in L \times R \mid \text{vol}(p_i^u + p_i^v < 1) \} \) 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 \( S \).

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

**Observation 2.2:** Let \( i \) be some iteration of the execution of algorithm 1, then \( P_i \geq \text{cg}(P_i) \).

**Proof:** For every \( i \), the set of constraints defining \( P_i \) is, by the behaviour of the algorithm, a subset of the constraints defining \( \text{cg}(P_i) \), thus the observation follows.

**Lemma 2.3:** 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 2.4:** 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:

\[
\frac{1}{e} \leq \frac{\text{vol}(P \cap H)}{\text{vol}(P)} \leq \left( 1 - \frac{1}{e} \right).
\]

This implies that, in our case, \( \text{vol}(P_{i+1}) \leq (1 - \frac{1}{e}) \text{vol}(P_i) \).

This means that in each iteration, we either find a feasible solution to \((P^G_F)\), or cut down the volume by a constant fraction as we have found a violating edge. Initially, \( \text{vol}(P_0) \leq 1 \), since it is contained in the unit-hypercube \([0,1]^{2n} \). By Lemma 2.1 we can terminate when \( P_i \) has volume less than \( \left( \frac{1}{20n} \right)^2 n \).
conclude that $(P_F^G)$ is empty in this case. This happens after at most $O \log((20n)^{2n}) = O(n \log n)$ iterations. ■

**Lemma 2.5:** Let $i_{\text{max}}$ denote the last iteration in the execution of the algorithm. Then either $P_{i_{\text{max}}} \in P_F^G$ which serves as a witness that a vertex cover of size $F$ exists, or $P_F^G = \emptyset$ 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 $(P_F^G)$, 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 $(P_F^G)$ is empty, which means we got to an iteration $i_{\text{max}}$ where $\text{vol}(P_i) < \left(\frac{1}{20n}\right)^{2n}$. 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 $P_{i_{\text{max}}}$ 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 2.1, 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 $P_{i_{\text{max}}}$.

We can deduce this is impossible, however, by simply noting that by the behaviour of the algorithm, it holds that $(P_F^G) = P_{i_{\text{max}}}$, and thus we can apply Lemma 2.1 which then says that $\text{vol}(P_i) \geq \left(\frac{1}{20n}\right)^{2n}$, which is a contradiction. ■

By Claim 2.2 and Lemma 2.3 we see that the algorithm makes a total of $O(n \log^2 n)$ OR-queries, and Lemma 2.5 argues its correctness. This concludes the proof of Lemma 2.2.

**B. OR-query optimization algorithm**

In this section we describe a standard modification (see e.g. [88, Section 4]) to our cutting planes **decision** algorithm, so that it solves the **optimization** version with the same query-complexity.

**Lemma 2.6:** 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 $\left\lceil \sum_{v \in V} p_i^v \right\rceil$ exists (since $(P_F^G)$ is integral). Hence we lower $F$ to $F' = \left\lceil \sum_{v \in V} p_i^v \right\rceil - 1$, by adding the constraint $\sum_{v \in V} x_v \leq F' + \frac{1}{3}$, and continue the cutting planes algorithm. Note that the constraint $\sum_{v \in V} x_v \leq F' + \frac{1}{3}$ forms a violating constraint for $p_i$ (and therefore cuts down the volume by a constant fraction, see Lemma 2.4, and counts as an iteration of the cutting planes algorithm).

At the end, the algorithm must terminate by determining that $(P_F^G)$ is empty (for the current value of $F$), in which case the found edges $E'$ contains a matching of size $F + 1$ (see Lemma 2.5). On the other hand, the last time we lowered $F$, we had a fractional vertex cover $p_i$ of size strictly less than $F + 2$. ■

**C. 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", and quantum edge query).

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 2.3:** 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 (where the query-settings are no longer compatible), which one can find in the full version of the paper [75].

2) **Randomized XOR-query:** Now we turn to the XOR-query setting. [45] 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 2.4:** There is a randomized algorithm which makes $O(n \log^2 n)$ XOR-queries and, w.h.p., solves BMM.

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

**Observation 2.3:** 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 $\Pr\left(\sum_{i=1}^k x_i r_i \text{ is odd} \right) = \frac{1}{2}$.

**Lemma 2.7:** 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 2.3, a constant probability of realizing this with

8 w.h.p. = with high probability; meaning with probability at least $1 - 1/n^c$ for an arbitrarily large constant $c$. 9
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 2.4: Just applying Lemma 2.7 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 $\text{XOR}(S') = 1$ (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 2.3 and 2.5, the entire algorithm requires $O(n \log^2 n)$ XOR-queries and is correct w.h.p.

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, [69]–[74]. 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 2.5: 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 $\mathbf{X}$, and we are asked to implement a separation oracle $\text{FindViolatingEdge}$. 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 2.2), 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 $\text{FindViolatingEdge}$ 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 $p' \in \mathbb{Z}^{L \times R}$ which we can use instead of $p$ when calling the separation oracle $\text{FindViolatingEdge}$. The integral point $p'$ will satisfy the following two properties: (i)
1) 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 $p$.
2) $\sum p'_v \leq \sum p_v$. 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 $\{(u, v) \in L \times R \mid p_u + p_v \geq 1\}$. In $H$, $p$ is a (fractional) vertex cover of size $\sum p_v$. This means that there exists an integral vertex cover of size $\lceil \sum p_v \rceil$ in $H$, since the minimum vertex cover linear program is integral for bipartite graphs. 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).

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 II-C5). Considering this model also helps to unveil the difficulty behind designing demand query algorithms (see open problems in section I-B for further discussion) for BPM, pointing to the fact that the barrier is not the size of the query, but rather its locality.

Claim 2.6: There is a deterministic algorithm which solves BMM with $O(n \log^2 n)$ OR$_k$-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 2.8: Any OR-query algorithm $A$ (for any graph problem) making $q$ queries can be converted to an OR$_k$-algorithm making $q + \lceil \frac{q}{k} \rceil$ 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 $A$ issues an OR-query $S \subseteq L \times R$ we do the following. Let $S_1, S_2, \ldots, S_t$ be a partition of $S \setminus E^c$ so that $|S_1| = |S_2| = \ldots = |S_{r-1}| = k$, and $|S_r| < k$. 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 $A$ (or else, after we have received “NO” from all the sets we return a “NO” answer to $A$). 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 $A$. In total, we thus charge at most $q$ cost to $A$: one per OR-query $A$ 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 $\lceil \frac{n^2}{k} \rceil$ times. So, other than the $q$ queries charged to $A$, we have made at most $\lceil \frac{n^2}{k} \rceil$ OR$_k$-queries to simulate the $q$ many OR-queries from $A$.

Plugging in our $O(n \log^2 n)$ OR-query algorithm to the above lemma yields Claim 2.6.

5) Quantum edge query (Q$_2$): In this section we consider the quantum edge-query model. See [123] for a formal definition and [124] for a more extensive background on quantum computing.

Claim 2.7: The quantum edge query complexity of solving BMM is $O(n \sqrt{n \log^2 n})$. 

\[1181\]
We use our ORₖ-query algorithm (Lemma 2.8) together with a well known quantum result (Lemma 2.9) regarding the quantum query complexity of the OR function.

Lemma 2.9 (Grover Search [125]): 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 2.7: Consider the instance of Lemma 2.8 where we put $k = \frac{n}{\log^{12} n}$, then we obtain an ORₖ-query algorithm for BMM with $O(n \log^2 n)$ queries. Each such ORₖ-query can be simulated, w.h.p., using $O(\sqrt{k} \log n) = O(\sqrt{n})$ quantum edge queries, by Lemma 2.9.

III. 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, which can be found in the full version of the paper [75], we also show how to solve unique bipartite perfect matching (UBPM). 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 1 in the two-party edge-partition communication setting, using $O(n \log^2 n)$ bits of communication. (i)

1) Maximum-cost bipartite perfect b-matching.
2) Maximum-cost bipartite b-matching.
3) Vertex-capacitated minimum-cost $(s, t)$-flow.
4) Transshipment (a.k.a. uncapacitated minimum-cost flow).
5) Negative-weight single source shortest path. 12
6) Minimum mean cycle.
7) Deterministic Markov Decision Process (MDP).

a) 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 [18] (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 III-A.

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

Definition 3.2 (b-matching): Given a graph $G = (V, E)$, a demand vector $b \in \mathbb{Z}^V_\geq 0$, and edge-costs $c \in \mathbb{Z}^E$, we call a vector $y \in \mathbb{Z}^E_0$ a b-matching (or a fractional b-matching if we allow $y \in \mathbb{R}^E_0$) 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 $\sum_{e \in E} c_e y_e$.

A. 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 $(G = (V, E), b \in \mathbb{Z}^V_+, c \in \mathbb{Z}^E)$ 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 $W \sum_{e \in E} b_e$.

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 $W = 1 + \sqrt{|V| \cdot max c_e}$), 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 $\ell$). 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 $\ell W + |V| max c_e \leq 2\ell W + |V| max c \leq \ell W + W = (\ell + 1)W$. That is we added more cost than we removed, hence contradicting that $M$ was of maximum cost.

B. 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 an appendix in the full version of the paper [75]. The main result of this section is the following:

Lemma 3.1: Max-cost bipartite b-matching can be solved using $O(n \log^2 (nW))$ communication, where $W := max\{max |c_e|, max b_e\}$ 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).

a) Dual linear program: Similarly as for the unweighted bipartite matching problem, we run a cutting planes algorithm on the dual linear program $(P(G,b,c))$ (refer to the constraints of Observation 3.2 for the primal linear program). We can think of $x \in (P(G,b,c))$ as a generalized version of a vertex cover, and an optimal solution would be one of minimum cost (w.r.t. costs $b_e$). 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.

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

Claim 3.1: 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 $(\mathcal{P}(G,b,c))$, which can be used to check if $(\mathcal{P}(G,b,c))$ has a solution with objective value at most $F$, for any integer $F \in \mathbb{Z}$.

$$\sum_{v \in V} b_v x_v \leq F + \frac{1}{2} \quad \text{s.t.} \quad \begin{align*}
  x_u + x_v & \geq c_{uv} \quad \forall (u, v) \in E \\
  0 \leq x_v & \leq W + 1 \quad \forall v \in V 
\end{align*} \quad (\mathcal{P}(G,b,c))$$

b) 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 3.2, whose proof is in an appendix in the full version of the paper [7]). 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 $P_0 = \{ x \in \mathbb{R}^V : \sum b_v x_v \leq F + \frac{1}{2} \}$.
- 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 $p_u^n + p_v^n < c_{uv}$, and the corresponding constraint we add is “$x_u + x_v \geq c_{uv}$”.
- We terminate when the volume is less than $\left(\frac{1}{200Wn}\right)^n$ (see Lemma 3.2).

Lemma 3.2 (Generalization of Lemma 2.1): If $(\mathcal{P}(G,b,c))$ is non-empty, then $\text{vol}(P_0) \geq \left(\frac{1}{200Wn}\right)^n$.

Observation 3.3 (Generalization of Claim 2.2): We can communicate a single violated constraint with $2 \log(n) + \log(W) + 1 = O(\log(nW))$ bits of communication.

c) Total communication: The generalized algorithm will start with initial polytope $P_0 \subseteq \{ x \in \mathbb{R}^V : \sum b_v x_v \leq F + \frac{1}{2} \}$, and terminate whenever $\text{vol}(P_f)$ becomes smaller than $\left(\frac{1}{200Wn}\right)^n$ (Lemma 3.2). In each iteration the volume is cut down by a constant fraction (Lemma 2.4). Hence we need $O \left( \log \left( \frac{W^n}{\left(\frac{1}{200Wn}\right)^n} \right) \right) = O(n \log(nW))$ iterations. Each iteration needs $O(\log(nW))$ bits of communication, for a total of $O(n \log^2(nW))$ bits of communication (Observation 3.3), proving Lemma 3.1. We also note that the same standard trick to convert the decision version to the optimization version (section II-B) works here as well.
C. Y. Lin and H. Lin, “Upper bounds on quantum query complexity of approximate maximum matching in the message-passing model,” in *STOC*. SIAM, 2016, pp. 156–164.

Z. Huang, B. Radunovic, M. Vojnovic, and Q. Zhang, “Communication complexity of approximate matching in distributed graphs,” in *STACS*, ser. LIPIcs, vol. 30. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015, pp. 460–473.

A. Assadi, S. Khanna, Y. Li, and G. Yaroslavtsev, “Maximum matchings in dynamic graph streams and the simultaneous communication model,” in *SODA*. SIAM, 2016, pp. 1345–1364.

M. Kapralov, “Space lower bounds for approximating maximum matching in the edge arrival model,” in *SODA*. SIAM, 2021, pp. 1874–1893.

M. Kapralov, G. Maystre, and J. Tardos, “Communication efficient coresets for maximum matching,” in *4th Symposium on Simplicity in Algorithms, SODA 2021, Virtual Conference, January 11–12, 2021*. SIAM, 2021, pp. 156–164.

Z. Huang, B. Radunovic, M. Vojnovic, and Q. Zhang, “Communication complexity of approximate maximum matching in the message-passing model,” in *Distributed Comput.*, vol. 33, no. 6, pp. 515–531, 2020.

M. Kapralov, S. Khanna, and M. Sudan, “Approximating matching size from random streams,” in *SODA*. SIAM, 2014, pp. 734–751.

M. Kapralov, S. Mitrovic, A. Norouzi-Fard, and J. Tardos, “Space efficient approximation to maximum matching size from uniform edge samples,” in *SODA*. SIAM, 2020, pp. 1753–1772.

G. Benamini, “The approximate degree of bipartite perfect matching,” *CoRR*, vol. abs/2004.14318, 2022.

G. Benamini and N. Nisan, “Bipartite perfect matching as a real polynomial,” in *STOC*. ACM, 2021, pp. 1118–1131.

A. E. Roth, “The economics of matching: Stability and incentives,” *Math. Oper. Res.*, vol. 7, no. 4, pp. 617–628, 1982.

M. Tennenholz, “Tractable combinatorial auctions and b-matching,” *Artif. Intell.*, vol. 140, no. 1/2, pp. 231–243, 2002.

D. P. Bertsekas, “Auction algorithms,” in *Encyclopedia of Optimization*. Springer, 2009, pp. 128–132.

A. A. Razborov, “On the distributional complexity of disjointness,” *Theor. Comput. Sci.*, vol. 106, no. 2, pp. 385–390, 1992.

C. Y. Lin and H. Lin, “Upper bounds on quantum query complexity inspired by the elitist-vaadin bomb tester,” in *Computational Complexity Conference*, ser. LIPIcs, vol. 33. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015, pp. 537–566.

A. L. Rosenberg, “On the time required to recognize properties of graphs: a problem,” *SIGACT News*, vol. 5, no. 4, pp. 15–16, 1973.

R. L. Rivest and J. Vuillemin, “A generalization and proof of the aanderaa-rozenberg conjecture,” in *STOC*. ACM, 1975, pp. 6–11.

J. Kahn, M. E. Saks, and D. Sturtevant, “A topological approach to evasiveness,” *Comb.*, vol. 4, no. 4, pp. 297–306, 1984.

P. Hajnal, “An omega(n^{4/3}) lower bound on the randomized complexity of graph properties,” *Comb.*, vol. 11, no. 2, pp. 131–143, 1991.

A. Chakrabarti and S. Khot, “Improved lower bounds on the randomized complexity of graph properties,” in *Random Struct. Algorithms*, vol. 30, no. 3, pp. 427–440, 2007.
[83] J. v. d. Brand, Y. T. Lee, A. Sidford, and Z. Song, “Solving tall dense linear programs in nearly linear time,” in STOC. ACM, 2020, pp. 775–788.

[84] J. v. d. Brand, Y. T. Lee, Y. P. Liu, T. Saranurak, A. Sidford, Z. Song, and D. Wang, “Minimum cost flows, MDPs, and $\ell_1$-regression in nearly linear time for dense instances,” in STOC. ACM, 2021, pp. 859–869.

[85] Y. T. Lee, A. Sidford, and S. C. Wong, “A faster cutting plane method and its implications for combinatorial and convex optimization,” in FOCS. IEEE Computer Society, 2015, pp. 1049–1065.

[86] A. Y. Levin, “On an algorithm for the minimization of convex functions over convex functions,” Soviet Mathematics Doklady, vol. 160, pp. 1244–1247, 1965.

[87] D. J. Newman, “Location of the maximum on unimodal surfaces,” Journal of the ACM (JACM), vol. 12, no. 3, pp. 395–398, 1965.

[88] P. M. Vaidya, “A new algorithm for minimizing convex functions over convex sets (extended abstract),” in FOCS. IEEE Computer Society, 1989, pp. 338–343.

[89] S. S. Vempala, R. Wang, and D. P. Woodruff, “The communication complexity of optimization,” in SODA. SIAM, 2020, pp. 1733–1752.

[90] S. Assadi and A. Dudeja, “A simple semi-streaming algorithm for global minimum cuts,” in SODA. SIAM, 2021, pp. 172–180.

[91] S. C. Liu, Z. Song, and H. Zhang, “Breaking the n-pass barrier: A streaming algorithm for maximum weight bipartite matching,” CoRR, vol. abs/2009.06106, 2020.

[92] S. Assadi, A. Jambulapati, Y. Jin, A. Sidford, and K. Tian, “Semi-streaming bipartite matching in fewer passes and optimal space,” in SODA. SIAM, 2022, pp. 627–669.

[93] R. G. Gallager, P. A. Humblet, and P. M. Spira, “A distributed algorithm for minimum-weight spanning trees,” ACM Trans. Program. Lang. Syst., vol. 5, no. 1, pp. 66–77, 1983.

[94] K. Kutten and D. Peleg, “Fast distributed construction of small $k$-dominating sets and applications,” J. Algorithms, vol. 28, no. 1, pp. 40–66, 1998.

[95] D. Nanongkai, “Distributed approximation algorithms for weighted shortest paths,” in STOC. ACM, 2014, pp. 565–573.

[96] M. Ghaffari and J. Li, “Improved distributed algorithms for exact shortest paths,” in STOC. ACM, 2018, pp. 431–444.

[97] S. Forster and D. Nanongkai, “A faster distributed single-source shortest paths algorithm,” in FOCS. IEEE Computer Society, 2018, pp. 660–677.

[98] M. Elkin, “Distributed exact shortest paths in sublinear time,” J. ACM, vol. 67, no. 3, pp. 15:1–15:36, 2020.

[99] A. Bernstein and D. Nanongkai, “Distributed exact weighted all-pairs shortest paths in near-linear time,” in STOC. ACM, 2019, pp. 334–342.

[100] U. Agarwal and V. Ramachandran, “Faster deterministic all pairs shortest paths in congest model,” in SPAA. ACM, 2020, pp. 11–21.

[101] M. Ghaffari, A. Karrenbauer, F. Kuhn, C. Lenzen, and B. Patt-Shamir, “Near-optimal distributed maximum flow,” SIAM J. Comput., vol. 47, no. 6, pp. 2078–2117, 2018.

[102] M. Henzinger, S. Krinninger, and D. Nanongkai, “A deterministic almost-tight distributed algorithm for approximating single-source shortest paths,” SIAM J. Comput., vol. 50, no. 3, 2021.

[103] S. Chechik and D. 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. ACM, 2020, pp. 464–473.

[104] M. Ghaffari, K. Nowicki, and M. Thorup, “Faster algorithms for edge connectivity via random 2-out contractions,” in SODA. SIAM, 2020, pp. 1260–1279.

[105] D. Nanongkai and H. Su, “Almost-tight distributed minimum cut algorithms,” in DISC, ser. Lecture Notes in Computer Science, vol. 8784. Springer, 2014, pp. 439–453.