CS 521: Linear Programming **Rutgers: Fall 2022**

Lecture 8

Oct 28, 2022

Instructor: Sepehr Assadi *Scribes: Xi Chen, Xiaoxiao He, Siwei Mai, Hengyi Wang*

Topics of this Lecture

1 More Background on Convexity **1**

1.1 Convex Optimization 1

1.2 Gradients 2

1.3 Volume of Convex Sets and Center of Gravity 3

2 Center of Gravity Method **4**

2.1 Center of Gravity Method for (General) Convex Optimization 4

2.2 Center of Gravity Method for Linear Programs 6

3 Communication Complexity of Bipartite Matching **8**

1 More Background on Convexity

We introduced basics of convexity in the previous lecture, and saw how it plays an important role in linear programming. In this lecture, we start our study of *cutting plane* methods that form a general family of algorithms for linear programming and more generally convex optimization. We start by reviewing basic definitions from convex optimization first and then introduce a very simple cutting plane method, the *center of gravity* algorithm.

1.1 Convex Optimization

Recall the definition of convex functions and convex sets from the previous lecture. A convex optimization problem is defined as follows.

Definition 1. Let $f : \mathbb{R}^n \to \mathbb{R}$ a convex function and K be a convex set. Return:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{subject to} \quad & x \in K. \end{aligned}$$

Since any (convex) polyhedra is a convex set and a linear objective function is a convex function, linear programming is a special case of convex optimization. In this lecture, we will see a cutting plane method for solving this general optimization problem. But before that, we need to review *gradients* and in particular their properties for convex functions briefly.

1.2 Gradients

The gradient of a function at a point $x \in \mathbb{R}^n$ is a vector that, roughly speaking, captures the "speed" or "direction and rate of increase" of the function on that point. In other words, moving in the direction of the gradient vector at this point results in the largest increase in the value of the function.

Formally, the gradient is defined as follows.

Definition 2. The gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$ at a point $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ is a vector $\nabla f(x) \in \mathbb{R}^n$ defined as:

$$\nabla f(x) = [\frac{\partial f}{\partial x_1}(x)\,,\ \frac{\partial f}{\partial x_2}(x)\,,\ \cdots\,,\ \frac{\partial f}{\partial x_n}(x)],$$

where $\frac{\partial f}{\partial x_i}(x)$ is the partial derivative of $f(x)$ at point x with respect to x_i for $i \in [n]$.

As a quick reminder, the partial derivative of $f(x)$ at a point x with respect to x_i is:

$$\frac{\partial f}{\partial x_i}(x) = \lim_{t \to 0} \frac{f(x + t \cdot e_i) - f(x)}{t},$$

where $e_i = (\underbrace{0, \cdots, 0}_{i-1}, 1, 0, \cdots 0)$ is the standard vector. As such, we have the following fact.

Fact 3. *For any $f : \mathbb{R}^n \to \mathbb{R}$, a point $x \in \mathbb{R}^n$ and a vector $v \in \mathbb{R}^n$, we have,*

$$\lim_{t \to 0} \frac{f(x + t \cdot v) - f(x)}{t} = \langle \nabla f(x)\,,\, v \rangle.$$

For example, consider the function $f(x) = x_1 x_2^2 + 2x_2$. Then, its gradient at any point x is the vector

$$\nabla f(x) = [x_2^2\,,\, 2x_1 x_2 + 2].$$

Gradients play a crucial role in convex optimization. In this lecture however, we only work with the following simple property of gradients for convex functions.

Proposition 4. *For any convex function $f : \mathbb{R}^n \to \mathbb{R}$ and two points $x, y \in \mathbb{R}$, we have,*

$$f(y) \geqslant f(x) + \langle \nabla f(x)\,,\, y - x \rangle.$$

Proof. For any $\lambda \in [0, 1]$, by the convexity of f, we have,

$$\lambda f(y) + (1 - \lambda) f(x) \geqslant f(\lambda y + (1 - \lambda)x) = f(x + \lambda(y - x)).$$

By re-ordering the terms in the above equation, we have that

$$f(y) - f(x) \geqslant \frac{f(x + \lambda(y - x)) - f(x)}{\lambda}.$$

Now, taking the limit of $\lambda \to 0$ in the above equation and using Fact 3, we have that

$$f(y) - f(x) \geqslant \lim_{\lambda \to 0} \frac{f(x + \lambda(y - x)) - f(x)}{\lambda} = \langle \nabla f(x)\,,\, y - x \rangle,$$

concluding the proof. □

It is worth mentioning that the inverse of the above proposition is also true: any function that satisfies the given property for all $x, y \in \mathbb{R}^n$ is a convex function (we leave the proof as a simple exercise for the reader). Goemeterically, Proposition 4 states that the tangent line on any point x of a convex function always remain below the function. See Figure 1 for an illustration.

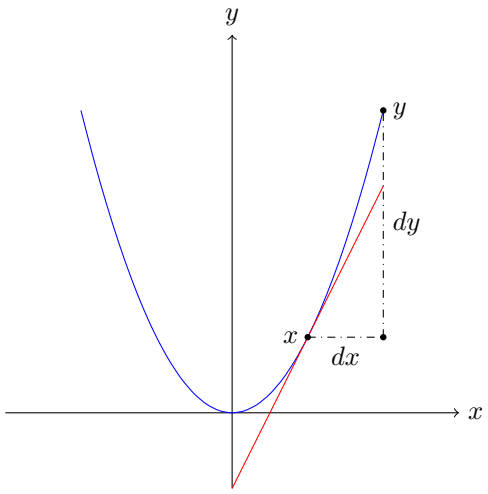


Figure 1: An illustration of Proposition 4: tangent lines of convex function remain below the function.

1.3 Volume of Convex Sets and Center of Gravity

One can define the volume of a convex set in any dimension similar to what we mean by volume in the three-dimensional space.

Definition 5. For any convex set $K \subseteq \mathbb{R}^n$, we define the **volume** of K as:

$$\text{Volume}(K) := \int_{x \in k} dx.$$

As an example, the volume of the n-dimensional box $B = [0, b]^n$ is

$$\text{Volume}(B) = \int_{x \in B} dx = \int_{x_1=0}^{b} \int_{x_2=0}^{b} \cdots \int_{x_n=0}^{b} dx = b^n.$$

Fact 6. *Let K be any convex set in $\mathbb{R}^n$, $\gamma > 0$ be a parameter, and define $K_\gamma = \{\gamma \cdot x \mid x \in K\}$. Then,*

$$\text{Volume}(K_\gamma) = \gamma^n \cdot \text{Volume}(K).$$

The center of gravity of a convex set is then defined as an "average" of the points in the set. Intuitively, this is the unique point in the convex set where the weighted relative position of the distributed mass of the points in the set sums to zero. Formally,

Definition 7. The **center of gravity** of a convex set $K \subseteq \mathbb{R}^n$ is a point in $\mathbb{R}^n$ defined as:

$$\text{Center}(K) := \frac{\int_{x \in K} x \, dx}{\text{Volume}(K)};$$

specifically, the i-th coordinate of $\text{Center}(K) = (c_1, \ldots, c_n)$ is obtained as:

$$c_i := \frac{\int_{x \in K} \langle x \, , \, e_i \rangle \, dx}{\text{Volume}(K)}$$

As an example, the center of gravity of the n-dimensional box $B = [0, b]^n$ is:

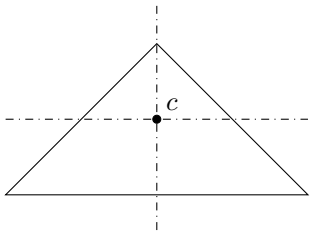
$$\begin{aligned}\text{Center}(B) &= b^{-n} \cdot \int_{x \in B} x \, dx = b^{-n} \cdot \int_{x_1=0}^{b} \int_{x_2=0}^{b} \cdots \int_{x_n=0}^{b} [x_1, \ldots, x_n] \, dx \\ &= b^{-n} \cdot [b^{n-1} \cdot \frac{b^2}{2}, \ldots, b^{n-1} \cdot \frac{b^2}{2}] = [\frac{b}{2}, \ldots, \frac{b}{2}].\end{aligned}$$

We use the following important result from convex geometry due to Grünbaum [3] in this lecture. In words, this proposition states that no matter how we "cut" a convex set through its center of gravity via a hyperplane, the remaining piece has a constant fraction of the volume of the original set. Formally,

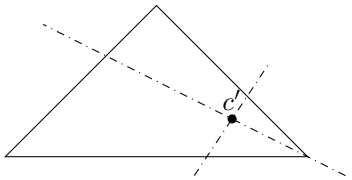
Proposition 8 (Grünbaum's theorem [3])**.** *Let $K \subseteq \mathbb{R}^n$ be any convex set and $c := \text{Center}(K)$ be its center of gravity. For any vector $h \in \mathbb{R}^n$ and hyperplane $H := \{x \in \mathbb{R}^n \mid \langle h \, , \, x \rangle \leqslant \langle h \, , \, c \rangle\}$, we have,*

$$\frac{1}{e} \cdot \text{Volume}(K) \leqslant \text{Volume}(K \cap H) \leqslant \left(1 - \frac{1}{e}\right) \cdot \text{Volume}(K).$$

The proof of this result is beyond the scope of this course and thus we do not provide it here. But see Figure 2 below for an illustration.



(a) Any hyperplane cutting the center of gravity partitions the triangle into almost-same size subsets.



(b) However, for a point far from the center of gravity, it is possible that some hyperplanes lead to highly unbalanced subsets, although some others may not.

Figure 2: Illustration of Grünbaum's theorem in Proposition 8.

We now have all the tools needed to present our algorithms in this lecture.

2 Center of Gravity Method

In this lecture, we present a simple strategy—somewhat in spirit of binary search—for solving convex optimization problems, referred to as the *center of gravity* method. This method was discovered independently on both sides of the Iron Curtain by Levin [4] and Newman [5].

We start by presenting a version of the algorithm for solving the convex optimization problem in Definition 1 and then show a specialization of the algorithm for solving LPs separately.

2.1 Center of Gravity Method for (General) Convex Optimization

Consider the following iterative algorithm for the convex optimization problem of Definition 1. We start by picking the center of gravity of K. We then compute the gradient of f at this point, which gives us a direction to minimize the value of f over K at the highest rate. Thus, we can now "cut" the parts of K that are above this direction and focus on the remaining part (which by Proposition 8 has become sufficiently smaller). We then continue like this the remaining part of K by computing its center of gravity and do as follows. Eventually, we have narrowed down K to the extent that only (near) optimal points remain and we can return those. We now formalize this algorithm (and slightly deviate from the given strategy for technical reasons that will become clear later).

Center of Gravity Method: An algorithm for minimizing a convex function f over a convex set K.

(*i*) Let $K_1 = K$ be the original convex set.

(*ii*) For $t = 1$ to T iterations:

(a) Let $c_t := \text{Center}(K_t)$ be the center of gravity of K_t.

(b) Compute $\nabla f(c_t)$ and let $H_t := \{x \in \mathbb{R}^n \mid \langle \nabla f(c_t)\,,\, x\rangle \leqslant \langle \nabla f(c_t)\,,\, c_t\rangle\}$.

(c) Let $K_{t+1} = K_t \cap H_t$.

(*iii*) Return $\underset{t\in[T]}{\text{argmin}}\, f(c_t)$ as the final answer.

Theorem 9. *For any $\varepsilon > 0$, the center of gravity method on a function $f : \mathbb{R}^n \to [-U, U]$ over a convex set $K \subseteq \mathbb{R}^n$, in $T = O(n \log(U/\varepsilon))$ iterations returns a point $\tilde{x}$ satisfying*

$$f(\tilde{x}) \leqslant f(x^*) + \varepsilon,$$

where x^* is the minimum of f over K.

Proof. By Proposition 8, for any iteration $t \in [T]$, we have,

$$\text{Volume}(K_{t+1}) \leqslant (1 - \frac{1}{e}) \cdot \text{Volume}(K_t) \leqslant (1 - \frac{1}{e})^t \cdot \text{Volume}(K), \tag{1}$$

where the second inequality is by the repeated application of first one.

For any $\gamma \in (0, 1)$, define

$$K_\gamma := \{(1 - \gamma) \cdot x^* + \gamma \cdot x \mid x \in K\}.$$

By Fact 6, we have that $\text{Volume}(K_\gamma) = \gamma^n \cdot \text{Volume}(K)$.

Consider the first iteration $t \in [T]$ such that $K_\gamma \subseteq K_t$ but $K_\gamma \not\subseteq K_{t+1}$. Such an iteration would happen as long as $T > e \cdot n \ln(1/\gamma)$ because $K_\gamma \subseteq K = K_1$ by convexity and by Eq (1), we have,

$$\text{Volume}(K_T) \leqslant (1 - 1/e)^{T-1} \cdot \text{Volume}(K) = (1 - 1/e)^{e \cdot n \ln(1/\gamma)} \cdot (\frac{1}{\gamma})^n \cdot \text{Volume}(K_\gamma) < \text{Volume}(K_\gamma);$$

thus, K_T cannot contain K_γ and so in some iteration before that we should have "cut" some part of K_γ.

Now, let c_t be the center of K_t computed in this particular iteration t. We claim that $f(c_t) \leqslant f(x^*) + 2\varepsilon \cdot U$, which we prove as follows.

Let y be any point in $K_\gamma \setminus K_{t+1}$. By Proposition 4,

$$f(y) \geqslant f(c_t) + \langle \nabla f(c_t)\,,\, y - c_t\rangle = f(c_t) + (\langle \nabla f(c_t)\,,\, y\rangle - \langle \nabla f(c_t)\,,\, c_t\rangle) > f(c_t),$$

where the last inequality is because $y \notin H_t$.

On the other hand, $y \in K_\gamma$ can be written as $y = (1 - \gamma) \cdot x^* + \gamma \cdot x$ for some $x \in K$. Thus,

$$\begin{aligned} f(y) = f((1 - \gamma) \cdot x^* + \gamma \cdot x) &\leqslant (1 - \gamma) \cdot f(x^*) + \gamma \cdot f(x) && \text{(by the convexity of } f\text{)} \\ &\leqslant f(x^*) + \gamma \cdot |f(x^*)| + \gamma \cdot |f(x)| \leqslant f(x^*) + 2\gamma \cdot U. \end{aligned}$$

Setting $\gamma = \varepsilon/2U$ now concludes the proof as in this case $T = O(n \log(U/\varepsilon))$ also holds. □

Remark. In order to be able to run the center of gravity method, we need to be able to (i) compute the gradient of f at a given point, and (ii) compute the center of gravity of a convex set. Among these two, the second requirement is quite strong since computing center of gravity (even for polytopes) is a #P-hard problem[a]. As a result, this method, despite its simplicity and (relatively) small number of iterations it needs, rarely leads to *time*-efficient algorithms for convex optimization due to its requirement of computing the center of gravity[b] (although it may lead to efficiency in other metrics as we shall see in the last part of this lecture).

[a] This means that this problem is at least as hard as counting the number of satisfying assignments to a SAT instance (recall that even finding one satisfying assignment to SAT is NP-hard already).

[b] There are algorithms known for finding an approximate center of gravity that can be made to work in this framework; see [1] for more on this topic.

2.2 Center of Gravity Method for Linear Programs

We now show a specific instantiation of the center of gravity method for solving linear programs. As is typical in this course, our focus will be on solving the feasibility problem: Given a polytope (P) defined as $\{x \in \mathbb{R}^n \mid Ax \leqslant b\}$ for m constraints defined by $a_i \in \mathbb{R}^n$, $b_i \in \mathbb{R}$, determine whether or not (P) is empty. In fact, for the center of gravity method to work, we need an extra assumption on the problem:

Assumption 10. *The polytope (P) is a subset of the n-dimensional box $[-R, R]^n$ for some $R \geqslant 1$. Moreover, if (P) is non-empty, then it fully contains an n-dimensional box $z + [-r, r]^n$ for some $z \in \mathbb{R}^n$ and $r > 0$.*

The first part of Assumption 10 is rather benign because whenever we have a polytope (P), it by definition belongs to some finite n-dimensional box; we are only assuming we know a "good" bound on the dimension of this box. The second part of the assumption is more strong which ensures that the polytope, whenever non-empty, is actually full-dimensional with a "non-trivial" volume. These assumptions are standard when using cutting plane methods and in the case of LPs are, to some extent, without loss of generality – we will see this in the context of an example later in this lecture, and then in full details later in the course.

We are now ready to present the algorithm.

Center of Gravity Method for LPs: An algorithm for testing LP feasibility under Assumption 10.

(i) Let $P_1 = [-R, R]^n$ be the box that is promised to contain (P) if it is non-empty.

(ii) For $t = 1$ to $T = 3n \cdot \ln(R/r)$ iterations:

- (a) Let $c_t := \text{Center}(P_t)$ be the center of gravity of the polytope P_t.
- (b) Check if c_t belongs to (P); if so, return c_t and terminate. Otherwise, let $a_i \in A$, $b_i \in b$ for $i \in [m]$ be a violated constraint, i.e., $\langle a_i , c_t \rangle > b_i$.
- (c) Let $H_t := \{x \in \mathbb{R}^n \mid \langle a_i , x \rangle \leqslant \langle a_i , c_t \rangle\}$ and $P_{t+1} = P_t \cap H_t$ be the polytope for next iteration.

(iii) If the algorithm never terminated so far, return (P) is empty.

Theorem 11. *The center of gravity method for LPs, given any polytope (P) defined as $\{x \in \mathbb{R}^n \mid Ax \leqslant b\}$ under Assumption 10, correctly decides whether (P) is empty and if not outputs a point $x \in P$.*

Proof. Firstly, the algorithm only outputs a point in (P) if it finds a feasible point and thus never makes an error on this part. In other words, whenever (P) is empty, the algorithm correctly outputs that.

We now argue that if (P) is non-empty and has a non-trivial volume under Assumption 10, the algorithm outputs a point in (P). By induction, we have that in every iteration $t \in [T]$ of the algorithm, $P \subseteq P_t$ since

the only points $x \in \mathbb{R}^n$ discarded from P_{t+1} compared to P_t all have

$$\langle a_i \, , \, x \rangle > \langle a_i \, , \, c_i \rangle > b_i,$$

which also violated the i-th constraint of (P). This implies that in every iteration of the algorithm before termination, we have

$$\text{Volume}(P_t) \geqslant \text{Volume}(P) \geqslant (2r)^n,$$

where the second inequality is by Assumption 10 and since volume of n-dimensional box $[-r, r]^n$ is $(2r)^n$ as calculated earlier. On the other hand, by Proposition 8, we also have,

$$\text{Volume}(P_t) \leqslant (1 - 1/e)^{t-1} \cdot \text{Volume}(P_1) = (1 - 1/e)^{t-1} \cdot (2R)^n.$$

For $t = 3n \cdot \ln(R/r)$, this become

$$\text{Volume}(P_t) \leqslant \exp(-1.1n \cdot \ln(R/r)) \cdot (2R)^n < r^n,$$

which means that in this case, the algorithm should have terminated before the given iteration. This concludes the proof. □

The following figure gives an illustration of the algorithm.

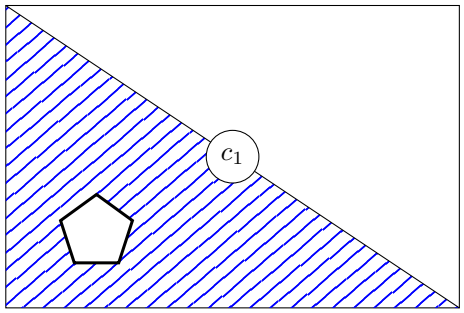


Figure 3: The first iteration of the center of gravity method for LP feasibility. The second polytope chosen by the algorithm is specified by the dashed region.

Remark. It is worth examining the interaction of this algorithm with the input linear program more carefully. The only step of the algorithm that needs access to the polytope (P) is in Line ((*ii*)b). Even this step does not necessarily requires having full access to (P). Instead we need the following:

- **Separation Oracle**: Given a point $y \in \mathbb{R}^n$, either correctly output y belongs to (P) or return a constraint $a_i \in A$, $b_i \in b$ that is violated by y, i.e., $\langle a_i \, , \, y \rangle > b_i$.

While a separation oracle can be trivially obtained by plugging in the vector y in every single constraint of (P), in many cases, one can do this more efficiently (which even allows for solving LPs with exponentially-many constraints). This is a general property of (most) cutting plane methods and we will revisit this in more details later in the course.

We conclude this section by recalling that computing the center of gravity of polytopes is a "hard" computational task and thus the center of gravity method typically does not lead to an efficient algorithm for solving LPs. We will see a (somewhat) remedy for this using another cutting plane algorithm, the *Ellipsoid algorithm* in the next lecture.

3 Communication Complexity of Bipartite Matching

Before concluding this lecture, we show a perhaps surprising application of the center of gravity method, when the focus is on another notion of efficiency beside time complexity. Consider the following problem.

Problem 1. *We have a bipartite graph $G = (V, E)$ with n vertices on each side whose edges are partitioned between two players Alice and Bob, who receive subgraphs G_A and G_B, respectively. The players want to compute a* perfect matching *in G, namely, a matching that matches all the vertices, or output that G does not have a perfect matching.*

Since neither player has the entire graph, the players need to communicate with each other, by sending messages to each other (which are arbitrary strings). The goal is to design a protocol for this problem with minimal *communication, namely, the worst-case total length of messages between the players.*

This problem admits a trivial protocol by players communicating all their inputs with each other; but this requires $O(n^2)$ bits of communication which is considered too much. On the other hand, there is also a trivial lower bound of $\Omega(n \log n)$ bits for this problem because if Alice has one of the $n!$ possible perfect matchings and Bob has no edges, Alice still needs to communicate $\log(n!) = \Theta(n \log n)$ bits to Bob for him to know the matching also. The question now is which of these two bounds is closer to the right answer?

A very recent elegant result by [2] shows that the lower bound above is much closer to the right answer than the trivial upper bound. Formally, they prove the following result.

Theorem 12 ([2])**.** *There is a deterministic protocol for Problem 1 that requires players to communicate $O(n \log^2 n)$ bits in total.*

As we shall see, the proof of this theorem turns out to be quite elegant and simple if one looks at the problem the "right way" and use the center of gravity method. However, this problem was open for about three decades and asked frequently in the literature, without much progress. This result thus acts as a perfect example of the power of these (quite simple) continuous optimization methods in solving classical combinatorial problems.

Proof of Theorem 12. Recall the following primal dual pair for the bipartite maximum matching and minimum vertex cover problems:

Bipartite matching LP:

$$\begin{aligned} \max_{e \in \mathbb{R}^E} \quad & \sum_{e \in E} x_e & & \\ \text{subject to} \quad & \sum_{e \ni v} x_e \leqslant 1 & & \forall v \in V \\ & x_e \geqslant 0 & & \forall e \in E. \end{aligned}$$

Bipartite vertex cover LP:

$$\begin{aligned} \min_{y \in \mathbb{R}^V} \quad & \sum_{v \in V} y_v & & \\ \text{subject to} \quad & y_u + y_v \geqslant 1 & & \forall (u, v) \in E \\ & y_v \geqslant 0 & & \forall v \in V. \end{aligned}$$

As we proved in earlier lectures, the integrality gap of both these LPs is one, or in other words, the optimal solutions to these LPs can be rounded without any loss in the value to integral matchings and vertex covers, respectively (even though the LPs allow fractional values).

For reasons that will become clear shortly, we will focus on solving the vertex cover LP instead (a brief intuition is that in this LP, each constraint is given fully to either Alice or Bob). In particular, we will run the center of gravity method for the vertex cover LP to check whether it has a vertex cover of size $n - 1$ or not (by strong duality, this is equivalent to checking if G has a perfect matching or not). However, to run the center of gravity method, we need to satisfy Assumption 10 which may not happen for the LP above. We consider the following relaxation:

(D1): Bipartite vertex cover LP:

$$\sum_{v \in V} y_v \leqslant n - 1$$
$$y_u + y_v \geqslant 1 \qquad \forall (u,v) \in E$$
$$y_v \geqslant 0 \qquad \forall v \in V.$$

(D2): Relaxed vertex cover LP:

$$\sum_{v \in V} y_v \leqslant n - \frac{1}{2}$$
$$y_u + y_v \geqslant 1 \qquad \forall (u,v) \in E$$
$$y_v \geqslant 0 \qquad \forall v \in V.$$

Notice that the only difference between the two LPs is the first constraint. The following claims show that this change does not affect the feasibility of these LPs, while establishing the required properties for satisfying Assumption 10 by (D2).

Claim 13. *(D1) is feasible iff (D2) is feasible.*

Proof. Since (D1) is a subset of (D2), if (D1) is feasible, then so is (D2). We now prove the converse. Let y be a point in (D2). By the randomized rounding method of Lecture 4, we obtain that there exists an integral $z \in \{0,1\}^V$ such that z is also in (D2). But since z is integral, we have that $\sum_v z_v \leqslant n - 1/2 < n$ implies that $\sum_v z_v \leqslant n - 1$. Thus, z also belong to (D1) proving that (D1) is also feasible. □

Claim 14. *If (D2) is feasible, then it contains an n-dimensional box with width $r = 1/8n$.*

Proof. By Claim 13, if (D2) is feasible, so is (D1). Let y be a feasible point in (D1) and define:

$$Z := \left\{ z \in \mathbb{R}^V \mid y_v \leqslant z_v \leqslant y_v + \frac{1}{4n} \text{ for all } v \in V \right\}.$$

For any point $z \in Z$, we have that

$$\sum_v z_v \leqslant \sum_v (y_v + \frac{1}{4n}) \leqslant 1/2 + \sum_v y_v \leqslant 1/2 + (n-1) = n - 1/2,$$

where the second inequality is because G has $2n$ vertices in total, and the next one is because y is in (D1). This implies that $Z \subseteq (D2)$ which implies the claim. □

Claim 15. *(D2) is always a subset of the n-dimensional box $[0,n]^n$.*

Proof. The variables y_v for $v \in V$ cannot receive a value $> n - 1/2$ without violating the first constraint. □

At this point, we have all we need to run the center of gravity method. The protocol will be as follows:

A protocol for bipartite perfect matching:

(*i*) Alice and Bob let $P_1 = [0,n]^n$. For $t = 1$ to $T = 3n \cdot \ln(R/r)$ iterations for $R = n$ and $r = 1/8n$:

- (a) The players on their own compute the center of gravity of the polytope P_t, i.e., $c_t := \text{Center}(P_t)$.
- (b) Both players check if c_t violates any of the constraints in their inputs. The first and last constraint of (D2) is known to both players and they can do this without any communication. The second constraints are partitioned between Alice and Bob: if there is a violated constraint, i.e., an edge $(u,v) \in G$, one of the players can communicate it to the other using $O(\log n)$ bits. If there is no violating constraints, the players can verify this with $O(1)$ communication between themselves, and both have c_t as the answer.
- (c) Let $H_t := \left\{ y \in \mathbb{R}^V \mid y_u + y_v \geqslant c_{tu} + c_{tv} \right\}$ and $P_{t+1} = P_t \cap H_t$ be the polytope for next iteration.

(*ii*) If the algorithm terminated, return G has no perfect matching. Otherwise, the players return a perfect matching among the set of communicated edges (which is known to both of them).

By construction, this algorithm is running the center of the gravity method described in Section 2.2. Thus, by Theorem 11 (and by Claim 14 and Claim 15 that ensure Assumption 10 is satisfied), whenever (D2) is feasible, it returns a point in (D2) and otherwise, correctly outputs (D2) is infeasible.

First, suppose (D2) is feasible. Then, by Claim 13, it implies that (D1) is also feasible, which, by the strong duality, implies the maximum matching in G has size $n-1$ – in other words, G does not have a perfect matching.

Now, suppose (D2) is infeasible. Consider the graph $H = (V, E_H)$ where E_H is the set of all communicated edges between Alice and Bob. The infeasibility of (D2) implies that E_H does not have a vertex cover of size less than n. Again, by the strong duality, this implies that E_H has a matching of size n, i.e., a perfect matching. Thus, the players can indeed return a perfect matching of H in the last step of the protocol, implying the correctness of the protocol.

Finally, the number of iterations of the algorithm is $O(n \log(R/r)) = O(n \log n)$ and each iteration involves communicating $O(\log n)$ bits. This gives an $O(n \log^2 n)$ bit protocol for bipartite perfect matching as desired. □

References

[1] D. Bertsimas and S. Vempala. Solving convex programs by random walks. *Journal of the ACM (JACM)*, 51(4):540–556, 2004. 6

[2] J. Blikstad, J. v. d. Brand, Y. Efron, S. Mukhopadhyay, and D. Nanongkai. Nearly optimal communication and query complexity of bipartite matching. *arXiv preprint arXiv:2208.02526*, 2022. 8

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

[4] A. Y. Levin. An algorithm for minimizing convex functions. In *Doklady Akademii Nauk*, volume 160, pages 1244–1247. Russian Academy of Sciences, 1965. 4

[5] D. J. Newman. Location of the maximum on unimodal surfaces. *Journal of the ACM (JACM)*, 12(3):395–398, 1965. 4