Chapter 10

Linear Programming V: Center of Gravity Algorithm

As we stated earlier, going over algorithms that solve LPs in polynomial time (Theorem 6.3) is beyond the scope of this book. But, in this chapter, we review some basic concepts behind these algorithms in order to shed some light on them.

10.1 Basics of Convexity in Linear Programming

We use this chapter to review some elementary definitions from convexity and their connection to linear programming. We start with the definition of the convex set.

Definition 10.1. A set $S \subseteq \mathbb{R}^n$ is **convex** iff for all points $x, y \in S$ and any $t \in [0, 1]$,

$$t \cdot x + (1 - t) \cdot y \in S.$$

Geometrically, this means that in any convex set, any line segment connecting two points in the set also belongs to the convex set. Figure 10.1 gives some examples.

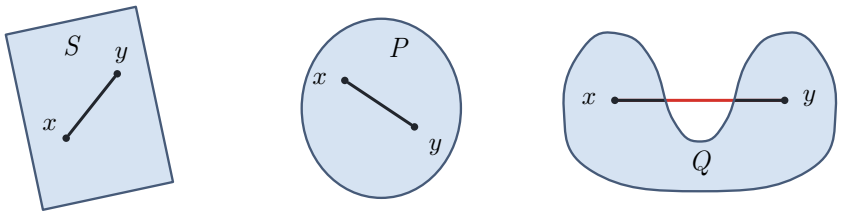


Figure 10.1: Sets S and P are convex, but set Q is not, because segment xy is not contained in Q.

Two of the simplest convex sets are the following.

Definition 10.2. For $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$, a **hyperplane** is a set $\{x \in \mathbb{R}^n \mid a^\top x = b\}$, and a **halfspace** is a set $\{x \in \mathbb{R}^n \mid a^\top x \geqslant b\}$.

Geometrically, a hyperplane divides $\mathbb{R}^n$ into two halfspaces (that intersect on the hyperplane). Algebraically, a hyperplane is defined by an equality constraint, and a halfspace by an inequality constraint. It is straightforward to verify that both hyperplanes and halfspaces are convex sets. But the main convex sets of interest when it comes to LPs are the following.

Definition 10.3. A **polyhedron** is the intersection of a finite number of halfspaces. A **polytope** is a bounded polyhedron, namely, one that can fit in a finite-radius ball.

Again, it is easy to verify that polyhedra[1] and polytopes are convex sets. There are however convex sets that cannot be a polyhedron, e.g., a ball (we do not prove this formally, but intuitively, a ball can only be the intersection of *infinitely* many halfspaces).

Example 10.4. A box $[-1,1]^n$ is a polytope. See Figure 10.2. We can easily check that a box B is a polyhedron as it can be written as the intersection of $2n$ halfspaces:

$$\text{for all } i \in [n]: \quad \{x \in \mathbb{R}^n \mid x_i \leqslant 1\} \cap \{x \in \mathbb{R}^n \mid x_i \geqslant -1\}.$$

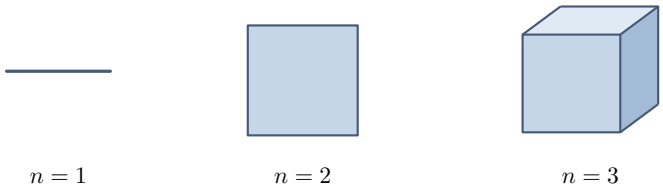


Figure 10.2: The n-dimensional box $[-1,1]^n$ for $n \in \{1,2,3\}$.

Example 10.5. A crosspolytope $\{x \in \mathbb{R}^n \mid \sum_{i=1}^n |x_i| \leqslant 1\}$ is a polytope. See Figure 10.3. This example is less obvious. We can describe it as the intersection of 2^n halfspaces:

$$\text{for all } S \subseteq [n]: \quad \left\{ x \in \mathbb{R}^n \mid \sum_{i \in S} x_i - \sum_{i \notin S} x_i \leqslant 1 \right\}.$$

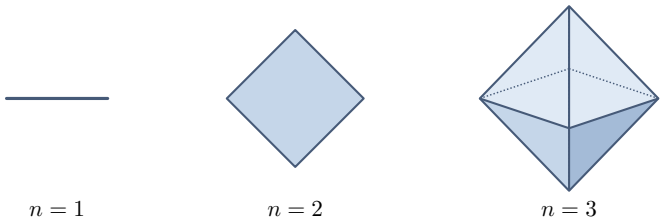


Figure 10.3: The n-dimensional crosspolytope $\{x \in \mathbb{R}^n \mid \sum_{i=1}^n |x_i| \leqslant 1\}$.

Relationship to Linear Programming. Building on the above definitions, we now revisit LPs from a convexity point of view. Let P be an LP defined as follows:

$$\max_{x \in \mathbb{R}^n} \quad c^\top x \quad \text{subject to} \quad Ax \leqslant b,$$

with $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, and $b \in \mathbb{R}^m$. Any inequality constraint $a_i^\top x \leqslant b_i$ in the LP is a halfspace, which defines a convex set. The set of all feasible solutions of P is the intersection of m halfspaces, which is a polyhedron. Thus, in linear programming we want to optimize a linear objective function (a convex function) over a polyhedron (a convex set)[2].

[1] The plural form of polyhedron is *polyhedra*.

[2] It is worth mentioning that there is also the more general problem of *convex optimization* wherein the goal is to optimize a general convex function over a general convex set.

10.1.1 Volume of Convex Sets and Center of Gravity

In order to present the main algorithm in this chapter, we need to review two more key definitions for convex sets, their *volume* and their *center of gravity*.

Definition 10.6. For any convex set $K \subseteq \mathbb{R}^n$, we define the **volume** of K as:

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

It is worth noting that in $\mathbb{R}^3$, namely, "our familiar three dimensional space", this definition matches what we normally define as a volume.

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 10.7. *For any convex set $K \subseteq \mathbb{R}^n$, a parameter $\gamma > 0$, and $K_\gamma = \{\gamma \cdot x \mid x \in K\}$,*

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

The center of gravity of a convex set is 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 10.8. 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, for $\text{Center}(K) = (c_1, \ldots, c_n)$ and any $i \in [n]$:

$$c_i := \frac{\int_{x \in K} x^\top \cdot e_i \, dx}{\text{Volume}(K)},$$

where $e_i = (0, \ldots, 0, 1_i, 0, \ldots, 0)$ is the i-th vector in the standard basis.

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

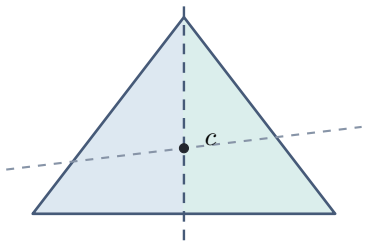
$$\begin{aligned}\text{Center}(B) &= b^{-n} \cdot \int_{x \in B} x \, dx = b^{-n} \cdot \int_{x_1=0}^{b} \int_{x_2=0}^{b} \cdots \int_{x_n=0}^{b} [x_1, \ldots, x_n] \, dx \\ &= b^{-n} \cdot [b^{n-1} \cdot \frac{b^2}{2}, \ldots, b^{n-1} \cdot \frac{b^2}{2}] = [\frac{b}{2}, \ldots, \frac{b}{2}].\end{aligned}$$

We use the following important result from convex geometry due to [Grü60]. In words, this result states that no matter how we "cut" a convex set through its center of gravity via a hyperplane, the remaining pieces have a constant fraction of the volume of the original set.

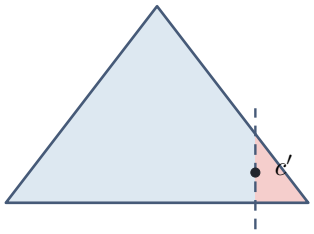
Proposition 10.9 ([Grü60])**.** *Let $K \subseteq \mathbb{R}^n$ be any convex set and $c := \text{Center}(K)$ be its center of gravity. For any vector $h \in \mathbb{R}^n$ and hyperplane $H := \left\{x \in \mathbb{R}^n \mid h^\top \cdot x \leqslant h^\top \cdot c\right\}$,*

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

The proof of this result is beyond the scope of this text. But see Figure 10.4 for an illustration.



(a) Any hyperplane through the center of gravity splits the set into almost-equal parts.



(b) For a point far from the center of gravity, some hyperplanes give highly unbalanced parts (though others may not).

Figure 10.4: An illustration of Proposition 10.9.

10.2 Center of Gravity Algorithm

We now present a simple strategy—somewhat in spirit of binary search—for solving linear programs, referred to as the **center of gravity** method. This algorithm appears to have been discovered independently by [Lev65] and [New65] around the same time. It can be used to solve the more general problem of convex optimization although in this chapter, we stick to its application to linear programming.

Our focus will be on solving the **feasibility problem** in LPs: Given a polytope (P) defined as $\{x \in \mathbb{R}^n \mid Ax \leqslant b\}$ for m constraints defined by $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, determine whether or not (P) is empty (recall from Section 7.3.1 the feasibility problem is as general as solving the linear programs completely).

For the center of gravity algorithm to work, we need the following assumption.

Assumption 10.10. 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 10.10 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 stronger, which ensures that the polytope—whenever non-empty—is actually full-dimensional with a "non-trivial" volume. These assumptions are standard and in the case of LPs are, to some extent, without loss of generality; we will see how to ensure this guarantee in the context of an example later in this chapter. We are now ready to present the algorithm.

Algorithm 10.11.

(*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., $a_i^\top \cdot c_t > b_i$.
- (c) Let $H_t := \left\{x \in \mathbb{R}^n \mid {a_i}^\top \cdot x \leqslant {a_i}^\top \cdot c_t\right\}$ 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.

The following theorem establishes the correctness of Algorithm 10.11.

Theorem 10.12. *The center of gravity algorithm (Algorithm 10.11), given any polytope (P) defined as* $\{x \in \mathbb{R}^n \mid Ax \leqslant b\}$ *under Assumption 10.10, 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 10.10, 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 satisfy:

$$a_i{}^\top \cdot x > a_i{}^\top \cdot c_i > b_i,$$

which also violated the i-th constraint of (P). This implies that in every iteration $t \in [T]$ 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 10.10 and since volume of n-dimensional box $[-r, r]^n$ is $(2r)^n$ as calculated earlier. On the other hand, by Proposition 10.9, we also have,

$$\text{Volume}(P_t) \leqslant (1 - 1/e) \cdot \text{Volume}(P_{t-1}),$$

for any $t \geqslant 2$; this in turn implies that

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

For $T = 3n \cdot \ln(R/r)$, this implies that

$$\text{Volume}(P_T) \leqslant (1-1/e)^{T-1} \cdot (2R)^n \leqslant \exp\left(-\frac{T-1}{e}\right) \cdot (2R)^n \leqslant \exp\left(-1.1n \ln(R/r)\right) \cdot (2R)^n < (2r)^n,$$

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

Figure 10.5 gives an illustration of the algorithm.

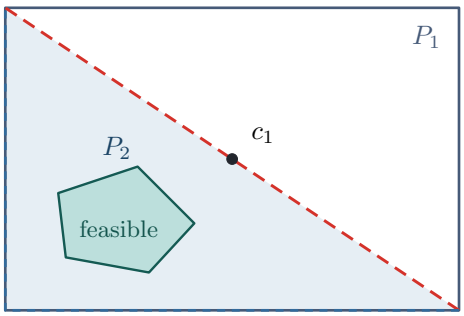


Figure 10.5: The first iteration of the center of gravity method for LP feasibility. The second polytope chosen by the algorithm is specified by the dashed region.

Remark. It is worth examining the interaction of Algorithm 10.11 with the input LP: the only step of the algorithm that needs access to the polytope (P) is in Line ((*ii*)b). Even this step does not necessarily require 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., ${a_i}^\top \cdot y > b_i$.

While a separation oracle can be obtained by plugging in the vector y in every constraint of (P), in many cases, one can do this more efficiently (and solve LPs with exponentially-many constraints). See Theorem 6.3 for a similar guarantee for polynomial time algorithms.

Remark. In order to be able to run the center of gravity algorithm—in addition to the separation oracle mentioned earlier—we need to be able to compute the center of gravity of a convex set. This requirement is quite strong since computing center of gravity of polytopes is a #**P**-hard problem[a]. As a result, this algorithm, despite its simplicity and (relatively) small number of iteration, rarely leads to *time*-efficient algorithms for linear programming 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 next part of this chapter).

[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 [BV04] for more on this topic.

Remark. The center of gravity algorithm belongs to the broader family of *cutting plane* methods. In these methods, one maintains a convex region guaranteed to contain a feasible solution and repeatedly "cut" this region via a separation oracle similar to the center of gravity algorithm. The most canonical algorithm in this family is the *Ellipsoid algorithm* of [Kha79] (see also [GLS81]) which gives a polynomial time algorithm for solving LPs (Theorem 6.3). The rough idea of the Ellipsoid algorithm is to maintain an enclosing ellipsoid on the feasible region whose volume shrinks by a $(1 - 1/n)$ factor after each cut, leading to a larger number of iterations in the algorithm compared to Algorithm 10.11; however, the benefit is that each iteration is now implementable in polynomial time.

10.3 Communication Complexity of Bipartite Matching

We now see a surprising application of the center of gravity method.

Problem 10.13. 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?

An elegant result of [BvdBE+22] shows that the lower bound above is much closer to the right answer than the trivial upper bound. Formally, [BvdBE+22] prove the following result.

Theorem 10.14 ([BvdBE+22]). *There exists a deterministic protocol for Problem 10.13 that needs 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 quite some time and raised frequently in the literature [IKL+12, DNO14], without much progress. This result thus acts as a perfect example of the power of even quite simple continuous optimization methods in solving classical combinatorial problems.

We are now going to prove Theorem 10.14. Recall the following primal dual pair for bipartite maximum matching and minimum vertex cover:

Bipartite matching LP:

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

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 Propositions 6.5 and 6.6, 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 work with a LP feasibility problem, not an optimization one, and also guarantee Assumption 10.10. We consider this relaxation:

(D1): Feasibility vertex cover LP:

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

(D2): Relaxed vertex cover LP:

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

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 10.10 by (D2).

Claim 10.15. *(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 Proposition 6.6, 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 belongs to (D1) proving that (D1) is also feasible. □

Claim 10.16. *When (D2) is feasible, it contains a $2n$-dimensional box of width $\frac{1}{4n}$ in $[0,2]^{2n}$.*

Proof. By Claim 10.15, if (D2) is feasible, so is (D1). Let y be a feasible point in (D1) and we can assume without loss of generality that $y \in [0,1]^{2n}$. Define:

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

Notice that $Z \subseteq [0,2]^{2n}$. 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). Since remaining constraints of $(D1)$ and $(D2)$ are the same, this implies that $Z \subseteq (D2)$ which implies the claim. □

Given the claims above, we have that $(D2)$ satisfies all properties of Assumption 10.10. At this point, we have all we need to run the center of gravity method. Before that however, we should note $(D2)$ being feasible, implies $(D1)$ being feasible, which implies G admits a vertex cover of size at most $n-1$; thus G cannot have a perfect matching by weak duality. Conversely, if G has a perfect matching, then, again by weak duality, $(D1)$ cannot be feasible, which means $(D2)$ cannot be feasible. Thus, our task is effectively only checking feasibility of $(D2)$.

The protocol will be as follows.

Algorithm 10.17.

(*i*) Alice and Bob let $P_1 = [0,2]^{2n}$.

(*ii*) For $t = 1$ to $T = 3n \cdot \ln(R/r)$ iterations for $R = 2$ and $r = 1/4n$:

(a) Each player computes the center of gravity of the polytope P_t as $c_t := \text{Center}(P_t)$.

(b) Both players check if c_t violates any of the constraints in their inputs. The first and last constraints of (D2) are known to both players and they can do this without any communication. The second-type 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 no constraint is violated, the players can verify this with $O(1)$ communication between themselves, and both have c_t as the answer.

(c) Let $H_t := \{y \in \mathbb{R}^V \mid y_u + y_v \geqslant (c_t)_u + (c_t)_v\}$ and $P_{t+1} = P_t \cap H_t$ be the polytope for next iteration.

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

We can now prove Theorem 10.14 using Algorithm 10.17.

Proof of Theorem 10.14. By construction, this algorithm is running the center of gravity method of Algorithm 10.11 for (D2) $\cap\ [0,2]^{2n}$. Thus, by Theorem 10.12 (and Claim 10.16 that ensures Assumption 10.10 holds for (D2) $\cap\ [0,2]^{2n}$), 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 10.15, it implies that (D1) is also feasible, which, by the weak duality (Theorem 7.3), implies the maximum matching in G has size at most $n-1$, 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 (think of running the algorithm only on E_H instead of all of G; the behavior of the algorithm remains exactly the same). Now, by strong duality (Theorem 7.6), 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 $O(n \log^2 n)$ communication protocol for bipartite perfect matching as desired. This concludes the proof of Theorem 10.14. □

10.4 Another Application: Submodular Minimization

[To be prepared later?]

Exercises

Exercise 10.1. This exercise shows that the constant in Grünbaum's theorem (Proposition 10.9) cannot be improved. Let $\Delta_n = \{x \in \mathbb{R}^n \mid x_i \geqslant 0 \text{ for all } i \in [n],\ \sum_{i=1}^n x_i \leqslant 1\}$ be the standard n-dimensional simplex.

1. Show that the center of gravity of Δ_n is $c = \left(\frac{1}{n+1}, \ldots, \frac{1}{n+1}\right)$, so that $\sum_{i=1}^n c_i = \frac{n}{n+1}$.

2. Consider the halfspace $H = \left\{x \in \mathbb{R}^n \mid \sum_{i=1}^n x_i \leqslant \frac{n}{n+1}\right\}$, whose bounding hyperplane passes through c. Show that

$$\frac{\text{Volume}(\Delta_n \cap H)}{\text{Volume}(\Delta_n)} = \left(\frac{n}{n+1}\right)^n \xrightarrow{n\to\infty} \frac{1}{e}.$$

   Conclude that some halfspace through the center of gravity cuts off a piece of relative volume arbitrarily close to $1/e$, so the factor in Proposition 10.9 is best possible.

Exercise 10.2. The guarantee of Theorem 10.12 relies on Assumption 10.10: whenever the feasible region is non-empty, it contains a ball of radius $r > 0$. Give an example of a non-empty polytope $P \subseteq \mathbb{R}^2$ with $\text{Volume}(P) = 0$, and argue that, without Assumption 10.10, the center of gravity method may never place a query point inside P, and hence can fail to certify that P is non- empty.