CS 435, 2018
Lecture 8, Date: 19 April 2018
Instructor: Nisheeth Vishnoi

Variants of the Path Following Interior Point Method and Self-Concordance

In this lecture, we present various generalizations and extensions of the path following IPM which we have introduced in the previous lecture for the case of Linear Programming. As an application, we derive a fast algorithm for the minimum cost flow problem. Subsequently, we introduce the notion of Self-Concordance and present various barrier functions such as Vaidya's barrier function, Lee-Sidford's barrier function, and the Canonical barrier function.

Contents

1 An LP Formulation of the Min-Cost Flow problem

As an important application of the interior point method for linear programming developed in the previous lecture we consider the fundamental problem in combinatorial optimization – that of computing the minimum cost flow in directed graphs.

1.1 Problem Definition

In the minimum cost flow problem we are given a graph $G = (V, E)$ with n vertices and m edges (a flow network) such that through every (directed) edge $i \in E$ one can send between 0 and u_i units of flow. The goal is to find the cheapest way to send F units of flow from a vertex $s \in V$ to $t \in V$, such that there is no "leak" on any vertex in the graph. The cost of such a flow is calculated as the sum of costs over all edges and the cost of transporting f units through edge $i \in E$ incurs a cost of $c_i \cdot f$.

Below we provide a formal definition; as usually n denotes the number of vertices in G and m denotes the number of edges.[1]

Minimum Cost Flow on directed graphs

Input: A directed graph $G = (V, E)$ along with costs $c \in \mathbb{Z}^m$ and capacities $u \in \mathbb{Z}^m_{\geq 0}$ (on edges), two vertices $s, t \in V$ and a value $F \in \mathbb{N}$.

Goal: Find a flow $x^\star \in \mathbb{Q}^m_{\geq 0}$ – an optimal solution to the linear program

$$\begin{aligned} \min \ & c^\top x \\ \text{s.t.} \ & Bx = F\chi_{s,t}, \\ & 0 \leq x_i \leq u_i, \quad \forall \ i \in E, \end{aligned} \tag{1}$$

where B is the signed incidence matrix of G and $\chi_{s,t} = e_s - e_t \in \mathbb{R}^n$.

In the linear program (1) the equality constraints encode the flow conservation law: for every vertex $v \in V$ other than s, t, the incoming flow is equal to the outcoming flow, moreover F units of flow originate at s and end up in t. The constraint $0 \leq x_i \leq u_i$ for an edge $i = (w, v)$ says that the flow through i can go only in the direction from w to v and cannot exceed u_i units.

One can immediately see that an analogous problem for undirected graphs is a special case of the one we defined here, since one can simply put two directed edges (w, v) and (v, w) in place of every undirected edge wv. Moreover, this problem is more general then the maximum flow problem and the bipartite matching problem, studied in previous lectures. Indeed, the maximum flow problem boils down to finding the largest F for which (1) is feasible – and as such reduces to minimum cost flow for any cost function c.

[1] For convenience we often identify the set of edges E with $\{1, 2, \ldots, m\}$ and the set of vertices V with $\{1, 2, \ldots, n\}$. This is especially useful when indexing vectors or matrices by vertices and edges of G.

1.2 Linear Programming-based fast algorithms?

Most of the (long and rich) research history on the min-cost flow problem revolved around combinatorial algorithms – based on augmenting paths, push-relabel and cycle cancelling (see for instance the textbook [3]). The best known algorithm obtained using methods from this family has running time roughly $\widetilde{O}(mn)$ [5].

Since the min-cost problem (1) can be naturally stated as a linear program one can ask whether it is possible to derive a fast algorithm for min-cost flow using linear programming techniques. The result from the previous lecture suggests that obtaining an algorithm which performs roughly $\sqrt{m}$ iterations is possible.

While the worst case running time of one such iteration (using no structural properties of the linear program) is $O(m^3)$, in this case we might hope to do much better. Indeed, since the matrix B comes from a graph, one might hope to make the running time of one iteration small using fast Laplacian solvers by [10]. The best we can hope for is $\widetilde{O}(m)$ time per iteration – this would yield an algorithm with $\widetilde{O}(m^{3/2})$ running time. This already beats the best known combinatorial algorithm whenever the graph is not quadratically dense, i.e., whenever $m = o(n^2)$.

To derive such an algorithm we need to resolve several issues – the first of them being that the method we developed in the last lecture does not fit well to the form the min-cost linear program; we discuss it in Section 1.3. Moreover, we will need to show that one iteration of an appropriate variant of the path following method can be indeed reduced to solving a Laplacian system (to apply fast Laplacian solvers by [10]). Finally, the issue of finding a suitable point to initialize the interior point method also cannot be ignored, and we develop a fast algorithm for finding such.

1.3 The issue with the path following method from previous lecture

Let us now try to apply the primal path following method developed in the previous lecture to solve the problem (1). By inspecting the form of the linear program (1) we immediately see that it does not quite match what we have studied Lecture 7

$$\begin{aligned} \min_{x \in \mathbb{R}^m} \quad & c^\top x, \\ \text{s.t.} \quad & Ax \leq b. \end{aligned} \tag{2}$$

However, the linear programming framework is very flexible, and hence one can easily translate the program (1) to the form (2). This follows simply by turning equalities $Bx = F\chi_{s,t}$ into pairs of inequalities $Bx \leq F\chi_{s,t}$ and $-Bx \leq -F\chi_{s,t}$, and breaking $0 \leq x_i \leq u_i$ into $-x_i \leq 0$ and $x_i \leq u_i$.

By performing this transformations we can bring (1) into the form as above, however the issue becomes that the resulting polytope $P = \{x \in \mathbb{R}^m : Ax \leq b\}$ is not full-dimensional, which was a crucial assumption for our method in the previous lecture. Indeed, the running time of the algorithm depended on $\frac{1}{\delta}$ where δ is the distance

from the boundary of P of an initial point $x' \in P$. Since P is not full-dimensional, it is not possible to pick a point x' with positive δ.

One can try to fix this problem by modifying the feasible set and considering

$$P_\varepsilon = \{x \in \mathbb{R}^m : Ax \leq b + 1\varepsilon\},$$

for some tiny $\varepsilon > 0$, where 1 is the all-one vector. The P_ε is a slightly "inflated" version of P and, in particular, is full-dimensional. One can then minimize $c^\top x$ over P_ε, hoping that this gives a decent approximation of $\min\{c^\top x : x \in P\}$. However, there is at least one issues which arise when doing so. The $\varepsilon > 0$ needs to be exponentially small – roughly $\approx 2^{-L}$ (where L is the bit complexity of the linear program) in order to guarantee that we get a decent approximation of the original program. However, for such a small $\varepsilon > 0$, the polytope P_ε is very "thin" and thus we cannot provide a starting point x' which is far away from the boundary of P_ε. This adds at least an $\Omega(L) = \Omega(m)$ term in the number of iterations of the method and also forces us to use extended precision arithmetic (of roughly L bits) in order to execute the algorithm.

1.4 Resolving the issue

There are several different ways one can try to resolve these issues and obtain fast linear-programming based algorithms for the min-cost flow problem. One idea is based on the so-called primal-dual path interior point method (see for instance Chapter 5 in [12]), which is a method to solve the primal and the dual problems simultaneously, using Newton's method.

Another idea, which we are going to implement in this lecture is to abandon the constraint to work with full-dimensional polytopes and develop a method that directly solves linear programs with equality constraints. For this, the first step is to formulate the min-cost flow problem (1) in a form which involves only equality constraints and nonnegativity of variables. We introduce a new set of m variables $\{y_i\}_{i \in E}$ – the edge slacks and add equality constraints of the form

$$\forall i \in E \quad y_i = u_i - x_i.$$

Thus by denoting $b := F\chi_{s,t}$, our linear program becomes

$$\begin{aligned} &\min_{x \in \mathbb{R}^m} \; c^\top x, \\ &\text{s.t.} \quad \begin{pmatrix} B & 0 \\ I & I \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b \\ u \end{pmatrix}, \\ &\qquad x, y \geq 0. \end{aligned} \tag{3}$$

The number of variables in the new program is then $2m$ and the number of linear constraints is $n + m$.

By developing a Newton's method for equality-constrained problems and then deriving a new, corresponding path following interior point method for linear problems in the form as (3) we will be able to prove the following theorem.

Theorem 1 (Min-Cost Flow using IPM)**.** *There is an algorithm to solve the minimum cost flow problem* (3) *up to precision* $\varepsilon > 0$ *in* $\widetilde{O}(\sqrt{m}\log\frac{CU}{\varepsilon})$ *iterations, where* $U := \max_{i\in E}|u_i|$ *and* $C := \max_{i\in E}|c_i|$*. Each iteration involves solving one Laplacian linear system, i.e., of the form* $(BWB^\top)h = d$*, where* W *is a positive diagonal matrix,* $d \in \mathbb{R}^n$ *and* $h \in \mathbb{R}^n$ *is unknown.*

Since solving one such Laplacian system approximately takes time $\widetilde{O}(m)$ (by a result of Spielman and Teng [10]), we obtain an algorithm solving the min-cost flow problem in time roughly $\widetilde{O}(m^{3/2})$. To be precise, employing fast Laplacian solvers requires one to show that their approximate nature does not introduce any problems in the path following scheme – this was first demonstrated by Daitch and Spielman in [4].

Our proof of Theorem 1 is slightly different than the one by [4] (which was based on the primal-dual framework). It is divided into three parts. First, we derive a new variant of the path following method for linear programs of the form (3) (see Section 2) and show that it takes roughly $\sqrt{m}$ iterations to solve it. Next (in Section 2) we show that every iteration of this new method applied to (3) can be reduced (in $O(m)$ time) to solving one Laplacian system. Finally (in Section 3) we show how one can efficiently find a "good" starting point to initialize the interior point algorithm.

2 Path Following Method for the Standard Form of Linear Programming

The goal of this section is to derive a path following interior point method for linear programs of the form (3) (the reformulation of the min-cost flow problem). More generally, the method we are about to derive can deal with linear programs of the following form

$$\begin{aligned} \min_{x\in\mathbb{R}^m} \quad & c^\top x, \\ \text{s.t.} \quad & Ax = b, \\ & x \geq 0. \end{aligned} \tag{4}$$

where $A \in \mathbb{R}^{n\times m}$, $b \in \mathbb{R}^n$ and $c \in \mathbb{R}^m$. The optimization problem (4) is often referred to as the *standard form* of linear programming, while the form we studied in the last lecture: $\min\{c^\top x : Ax \leq b\}$ is called the *dual*[2] form (see [13]). Note that in this section we denote the number of variables by m and the number of (linear) constraints by n, i.e., the roles of n and m are swapped when compared to the previous lecture. This is done intentionally and is actually consistent with the previous lecture, since via duality the constraints become variables and vice versa.

[2]It is called the dual form, because it matches (up to renaming vectors and switching min to max) the Lagrange dual of (4).

For the sake of brevity, in this section we denote

$$E_b := \{x \in \mathbb{R}^m : Ax = b\}, \qquad E := \{x \in \mathbb{R}^m : Ax = 0\},$$

where the matrix A and the vector b are fixed throughout the discussion.

The idea of the algorithm for solving (4) is exactly the same as in the previous lecture: we will use the logarithmic barrier function to define a central path in the relative interior of the feasible set and then use Newton's method to progress along the path. To implement this idea, the first step is to derive a Newton's method for minimizing a function restricted to an affine subspace of $\mathbb{R}^m$.

2.1 Equality-constrained Newton's method

Consider a constrained optimization problem of the form

$$\begin{aligned} \min_{x \in \mathbb{R}^m} \quad & f(x) \\ \text{s.t.} \quad & Ax = b, \end{aligned} \tag{5}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function, $A \in \mathbb{R}^{n \times m}$ and $b \in \mathbb{R}^n$. We assume without loss of generality that A has rank n, i.e., the rows of A are linearly independent (otherwise there are redundant rows in A which can be removed). We denote the function f restricted to the domain E_b by $\widetilde{f} : E_b \to \mathbb{R}$, i.e. $\widetilde{f}(x) = f(x)$.

In the absence of equality constraints $Ax = b$ we know what does it mean to apply the Newton's method to f. The Newton's step at a point x is simply defined as

$$n(x) := -H(x)^{-1} g(x)$$

where $H(x) \in \mathbb{R}^{m \times m}$ is the Hessian of f at x and $g(x)$ is the gradient of f at x and the next iterate is computed as $x' := x + n(x)$.

When started at $x \in E_b$, the point $x + n(x)$ typically does not belong to E_b anymore and hence we need to adjust the Newton's method to move only in directions "tangent" to E_b. To this end, in the next section we define the notions of a gradient $\widetilde{g}(x)$, and Hessian $\widetilde{H}(x)$ so that the Newton step, defined by the formula

$$\widetilde{n}(x) := \widetilde{H}(x)^{-1} \widetilde{g}(x)$$

gives a well defined method for minimizing $\widetilde{f}$ (i.e., f restricted to E_b). Moreover, by defining an appropriate variant of the **NL** condition for $\widetilde{f}$ we will obtain the following theorem.

Theorem 2 (Quadratic Convergence of Constrained Newton's Method)**.** *Let $\widetilde{f} : E_b \to \mathbb{R}$ be a strictly convex function satisfying condition* **NL**, *$x_0 \in E_b$ be any point and $x_1 := x_0 + \widetilde{n}(x_0)$. If $\|\widetilde{n}(x_0)\|_{x_0} < \frac{1}{6}$ then*

$$\|\widetilde{n}(x_1)\|_{x_1} \leq 3 \|\widetilde{n}(x_0)\|_{x_0}^2 .$$

We note that the above is completely analogous to Theorem 4 in Lecture 6, where we dealt with the unconstrained setting – indeed, the above is a straightforward extension of this theorem.

2.2 Defining the Hessian and gradient on a subspace

To define an analogous Newton step $\widetilde{n}(x)$ for a point $x \in E_b$ with respect to $\widetilde{f}$, we need to understand first what the gradient $\widetilde{g}(x)$ and the Hessian $\widetilde{H}(x)$ are in this setting. To this end note that the crucial difference between working in $\mathbb{R}^n$ and working in E_b is that now, given $x \in E_b$ we cannot move in every possible direction from x, but only in "tangent" directions, i.e. to $x + h$ for some $h \in E$.

To define the gradient on a subspace let us first specify what are we looking for. The gradient $\widetilde{g}(x)$ should be a vector in E (the tangent space) such that the linear function $E \ni h \mapsto \langle h, \widetilde{g}(x)\rangle$ provides a good approximation of $f(x+h) - f(x)$ whenever $\|h\|$ is small.[3] More formally,

Definition 3 (Gradient restricted to subspace)**.** *The gradient of $\widetilde{f} : E_b \to \mathbb{R}$ at $x \in E_b$ is defined as the unique vector $\widetilde{g}(x) \in E$ such that*

$$\lim_{h \in E, \|h\| \to 0} \frac{\widetilde{f}(x+h) - \widetilde{f}(x) - \langle \widetilde{g}(x), h\rangle}{\|h\|} = 0. \tag{6}$$

Similarly, Hessian $\widetilde{H}(x)$ is a symmetric linear operator $E \to E$ which is a a linear approximation of the gradient $\widetilde{g}(x)$ (treated as a map $E_b \to E$).

Definition 4 (Hessian restricted to subspace)**.** *The Hessian of $\widetilde{f} : E_b \to \mathbb{R}$ at $x \in E_b$ is defined as the unique linear operator $\widetilde{H}(x) : E \to E$ such that*

$$\lim_{h \in E, \|h\| \to 0} \frac{\left\| \widetilde{g}(x+h) - \widetilde{g}(x) - \widetilde{H}(x)h \right\|}{\|h\|} = 0. \tag{7}$$

We note that in both definitions it is crucial that $h \in E$ only, and not $h \in \mathbb{R}^m$ – in fact, taking arbitrary $h \in \mathbb{R}^m$ does not make formal sense, since $\widetilde{f}$ is defined over the subspace E_b.

These definition look rather abstract, but in fact, the only way they differ from their "unconstrained" variants is the fact that we consider E as the "tangent space" instead of $\mathbb{R}^m$. As an example, consider the function $f(x_1, x_2) = 2x_1^2 + x_2^2$. When constrained to a "vertical" line $E_b = \{(x_1, x_2)^\top : x_1 \in \mathbb{R}, x_1 = b\}$ we obtain that the gradient is

$$\widetilde{g}(x_1, x_2) = \begin{pmatrix} 0 \\ 2x_2 \end{pmatrix}$$

and the Hessian $\widetilde{H}(x_1, x_2)$ is a linear operator $E \to E$ (where $E = \left\{ \begin{pmatrix} 0 \\ h \end{pmatrix} : h \in \mathbb{R} \right\}$ such that

$$\widetilde{H}(x_1, x_2) \begin{pmatrix} 0 \\ h \end{pmatrix} = \begin{pmatrix} 0 \\ 2h \end{pmatrix}.$$

[3] We use the standard inner product $\langle \cdot, \cdot \rangle$ on $\mathbb{R}^m$ and the Euclidean norm $\|\cdot\|$ to define these concepts, however this choice is arbitrary.

Indeed this is consistent with the intuition that by restricting the function to a vertical line, only the variable x_2 "matters".

More generally, as one would expect, $\widetilde{g}(x)$ and $\widetilde{H}(x)$ can be also recovered from $g(x)$ and $H(x)$. This observation is formalized in the lemma below.

Lemma 5. *Let $f : \mathbb{R}^m \to \mathbb{R}$ be a strictly convex and let $\widetilde{f} : E_b \to \mathbb{R}$ be its restriction to E_b. Then we have*

$$\widetilde{g}(x) = \Pi_E g(x),$$
$$\widetilde{H}(x) = \Pi_E H(x),$$

where $\Pi_E : \mathbb{R}^m \to E$ is the orthogonal projection operator onto E, i.e.

$$\Pi_E := I - A^\top (AA^\top)^{-1} A.$$

The above lemma follows simply just from the definition of $\widetilde{g}(x)$ and $\widetilde{H}(x)$.

2.3 Defining the Newton's step and NL condition on a subspace

Recall that the Newton step is defined to be

$$\widetilde{n}(x) := -\widetilde{H}(x)^{-1}\widetilde{g}(x).$$

Note that in the above, $\widetilde{H}(x)$ is an invertible operator $E \to E$, and hence $\widetilde{n}(x)$ is well defined. When trying to derive a derive a formula for the Newton step using Lemma 6 one might run into the following tempting, but **incorrect** calculation

$$\widetilde{n}(x) = \widetilde{H}(x)^{-1}\widetilde{g}(x) = (\Pi_E H(x))^{-1}\Pi_E g(x) = H(x)^{-1}\Pi_E^{-1}\Pi_E g(x) = H(x)^{-1} g(x).$$

The above is **wrong**, because the map Π_E is not invertible (if E is a proper subset of $\mathbb{R}^m$) and hence its inverse does not exist. The following lemma states the correct form of the Newton step restricted to the subspace.

Lemma 6. *Let $f : \mathbb{R}^m \to \mathbb{R}$ be a strictly convex function and let $\widetilde{f} : E_b \to \mathbb{R}$ be its restriction to E_b. Then, for every $x \in E_b$ we have*

$$\widetilde{n}(x) = H(x)^{-1}(A^\top (AH(x)A^\top)^{-1} AH(x)^{-1} - I)g(x).$$

Proof. The Newton step $\widetilde{n}(x)$ is defined as $-\widetilde{H}(x)^{-1}\widetilde{g}(x)$ and hence can be obtained as the unique solution h to the following linear system

$$\begin{cases} \widetilde{H}(x)h = -\widetilde{g}(x), \\ Ah = 0. \end{cases} \tag{8}$$

The first equation gives that

$$\Pi_E H(x)h = -\Pi_E g(x)$$

and hence $\Pi_E(H(x)h - g(x)) = 0$. In other words, this means that $H(x)h - g(x)$ belongs to the space orthogonal to E, i.e., $E^\top = \{A^\top \lambda : \lambda \in \mathbb{R}^n\}$. Consequently, (8) can be equivalently rewritten as a linear system of the form

$$\begin{pmatrix} H(x) & A^\top \\ A & 0 \end{pmatrix} \cdot \begin{pmatrix} h \\ \lambda \end{pmatrix} = \begin{pmatrix} -g(x) \\ 0 \end{pmatrix}$$

with unknowns $h \in \mathbb{R}^m$ and $\lambda \in \mathbb{R}^n$. From the above λ can be eliminated (for example using the Schur complement formula) to yield

$$\widetilde{n}(x) = H(x)^{-1}(A^\top (AH(x)A^\top)^{-1} AH(x)^{-1} - I)g(x).$$

□

The Newton's method for minimizing f over E_b then simply starts at any $x_0 \in E_b$ and iterates according to

$$x_{k+1} := x_k + \widetilde{n}(x_k) \qquad \text{for } k = 0, 1, 2, \ldots \tag{9}$$

As in the unconstrained setting we consider the local norm $\|\cdot\|_x$ defined on the tangent space at x by

$$\forall h \in E \qquad \|h\|_x^2 := \left\langle h, \widetilde{H}(x)h \right\rangle = h^\top H(x)h,$$

and define the **NL** condition to be

Definition 7 (Condition **NL**). *Let $\widetilde{f} : E_b \to \mathbb{R}$ be a function; we say that $\widetilde{f}$ satisfies the* **NL** *condition if*

$$\forall_{x,y} \|y - x\|_x = \delta < 1 \qquad (1 - 3\delta)\widetilde{H}(x) \preceq \widetilde{H}(y) \preceq (1 + 3\delta)\widetilde{H}(x).$$

In the above, the PSD ordering $\preceq$ is understood in the usual sense, yet restricted to the space E. More precisely

Definition 8. *For two self-adjoint operators $H_1, H_2 : E \to E$ we say that $H_1 \preceq H_2$ if $\langle h, H_1 h\rangle \leq \langle h, H_2 h\rangle$ for every $h \in E$.*

At this point, all components present in Theorem 2 have been formally defined. Note that all what has been done was redefining objects we already know, for them to make sense in the constrained setting. It is though not a suprise that the proof of Theorem 2 is also **exactly the same** as proof of Theorem 4 in Lecture 6 (in fact, one just has to replace f, g, H to $\widetilde{f}, \widetilde{g}$ and $\widetilde{H}$ for the proof to go through). For this reason we do not discuss it in detail.

2.4 Path Following for the standard form of Linear Programming

To develop an interior point method for solving linear programs in the standard form (4), one can work in the affine subspace $E_b := \{x \in \mathbb{R}^m : Ax = b\}$ and apply a similar path following scheme as in the previous lecture. More precisely, let us define the point $x^\star_\eta$ as

$$x^\star_\eta := \operatorname*{argmin}_{x \in E_b} \left(\eta c^\top x + F(x) \right),$$

where F is the logartihmic barrier function for the positive orthant, i.e.,

$$F(x) := -\sum_{i=1}^{m} \log x_i.$$

The idea is to use Newton's method to progress along the central path, i.e., the set $\Gamma_c := \{x^\star_\eta : \eta \geq 0\}$.

For clarity, let us now adjust the notation to this new setting. We define the function

$$f_\eta(x) := \eta c^\top x + F(x)$$

and $\widetilde{f}_\eta$ – its restriction to E_b. We let $\widetilde{n}_\eta(x)$ be the Newton step with respect to $\widetilde{f}_\eta$, i.e., (using Lemma 6)

$$\widetilde{n}_\eta(x) := H(x)^{-1}(A^\top (AH(x)A^\top)^{-1} AH(x)^{-1} - I)(\eta c + g(x)),$$

where $H(x)$ and $g(x)$ are the Hessian and gradient of $F(x)$ in $\mathbb{R}^m$. Note that in this case, the Hessian and the gradient of $F(x)$ have particularly simple forms:

$$H(x) = X^{-2}, \qquad g(x) = X^{-1}1,$$

where X is the diagonal matrix with the entries of x on the diagonal and 1 is the all-one vector. Thus

$$\widetilde{n}_\eta(x) := X^2(A^\top (AX^{-2}A^\top)^{-1} AX^2 - I)(\eta c + X^{-1}1). \tag{10}$$

The algorithm for this setting is analogous to what was presented in the previous lecture, with the only difference being the use of the constrained Newton's step $\widetilde{n}_\eta(x)$ instead of the unconstrained one $n_\eta(x)$.

Path Following IPM for Linear Programming in Standard Form:

1. Find an initial $\eta_0 > 0$ and $x_0 \in E_b \cap \mathbb{R}^m_{>0}$ with $\left\|\widetilde{n}_{\eta_0}(x_0)\right\|_{x_0} < \frac{1}{6}$.
2. Repeat for $k = 0, 1, 2, \ldots, T$ where T is s.t. $\eta_T > \frac{m}{\varepsilon}$
   - compute x_{k+1} according to the rule:
   $$x_{k+1} := x_k + \widetilde{n}_{\eta_k}(x_k)$$
   - set $\eta_{k+1} := \eta_k \left(1 + \frac{1}{20\sqrt{m}}\right)$.
3. Calculate $\hat{x}$ by performing a constant number of constrained Newton steps w.r.t. $f_{\eta_T}(x) := \eta_T c^\top x + F(x)$ (starting at x_T) and output $\hat{x}$.

The theorem which one can formulate in this setting is

Theorem 9 (Convergence of Path Following IPM)**.** *The Path Following IPM for Linear Programming in Standard Form, after $T = O\left(\sqrt{m}\log\frac{m}{\varepsilon\eta_0}\right)$ iterations, outputs a point $\hat{x} = x^\star_{\eta_T} \in E_b \cap \mathbb{R}^m_{>0}$ that satisfies*

$$c^\top x^\star_{\eta_T} \leq c^\top x^\star + \varepsilon.$$

Moreover, every iteration (step. 2) requires $O(m)$ time, plus solving one linear system of equations of the form $(AWA^\top)y = d$, where $W \succ 0$ is a diagonal matrix, $d \in \mathbb{R}^n$ is a given vector and y is the vector of variables.

For the application to min-cost flow it will be also important to show that finding the initial point on the central path is not a bottleneck. From the reasoning given in the Lecture 7, Section 5.3, one can derive the following lemma.

Lemma 10 (Starting Point)**.** *There is an algorithm which given a point $x' \in E_b \cap \mathbb{R}^m_{>0}$ such that $x'_i > \delta$ for every $i = 1, 2, \ldots, m$, finds a pair (η_0, x_0) such that*

- $x_0 \in E_b \cap \mathbb{R}^m_{>0}$,
- $\left\|\widetilde{n}_{\eta_0}(x_0)\right\|_{x_0} < \frac{1}{6}$,
- $\eta_0 \geq \Omega\left(\frac{1}{D} \cdot \frac{1}{\|c\|_2}\right)$, *where D is the (Euclidean) diameter of the feasible set $E_b \cap \mathbb{R}^m_{\geq 0}$.*

The algorithm preforms $\widetilde{O}(\sqrt{m}\log\frac{D}{\delta})$ iterations of the interior point method.

In particular, whenever we can show that the diameter of the feasible set is polynomially bounded and we can find a point which is at least an inverse-polynomial distance away from the boundary, then the total number of iterations of the Path Following IPM (to find the starting point and then find a point ε-close to optimum) is roughly $\widetilde{O}\left(\sqrt{m}\log\frac{\|c\|_2}{\varepsilon}\right)$.

3 A $O(\sqrt{m})$-iterations LP-based Algorithm for Minimum Cost Flow

We show how to conclude Theorem 1 using the path following method developed in the previous section.

3.1 Number of iterations

We first note that by Theorem 9 the linear program (3) can be solved in $O\left(\sqrt{m}\log\frac{m}{\varepsilon\eta_0}\right)$ iterations of the path following method. In order to match the number of iterations claimed in Theorem 1 we need to show that we can initialize the path following method at η_0 such that $\eta_0^{-1} = \text{poly}(m, C, U)$. We show this in Section 3.3.

3.2 Time per iteration

This part of the analysis is left as an exercise. More precisely, one has to show the following lemma.

Lemma 11 (Computing Newton's step by solving a Laplacian system)**.** *One iteration of the path following IPM applied to the linear program* (3) *can be implemented in time $O(m + T_{Lap})$ where T_{Lap} is the time required to solve a linear system of the form $(BWB^\top)h = d$, where W is a positive diagonal matrix, $d \in \mathbb{R}^n$ and $h \in \mathbb{R}^n$ is an unknown vector.*

The task of proving the above lemma is really a question of how to compute a Newton step in every iteration of the path following method. One can use the formula (10) as a starting point for proving the above. However, one still needs to determine how the task of solving a certain $2m \times 2m$ linear system reduces to just an $n \times n$ Laplacian system.

3.3 Finding a starting point efficiently

It remains to show how one can find a point close enough to the central path so that we can initialize the path following IPM. To this end we apply Lemma 10. It says that for that we just need to provide a strictly feasible solution (x, y) to (3), i.e., one which satisfies the equality constraints and such that $x_i, y_i > \delta > 0$ for some not-too-small δ (inverse-polynomial in m, C and U will do).

We start by showing how to find a positive flow of any value and then argue that by a certain *preconditioning* of the graph G we can also find a flow of a prescribed value F using this method.

Lemma 12 (Finding a strictly positive flow)**.** *There is an algorithm which given a directed graph $G = (V, E)$ and two vertices $s, t \in G$, outputs a vector $x \in [0, 1]^E$ such that:*

1. $Bx = \frac{1}{2}\chi_{s,t}$,
2. *if an edge $e \in E$ does not belong to any directed path from s to t, then $x_e = 0$,*

3. *if an edge $e \in E$ belongs to some directed path from s to t then*

$$\frac{1}{2m} \leq x_e \leq \frac{1}{2}.$$

The algorithm runs in $O(m)$ time.

Note that the edges that do not belong to any path from s to t can be completely removed from the graph, as they will not be part of any feasible flow. Below we prove that one can find such a flow in roughly $O(nm)$ time – the question on how to make this algorithm more efficient is left as an exercise.

Proof sketch of Lemma 12. We initialize $x \in \mathbb{R}^m$ to 0. Let us fix an edge $e = (v, w) \in E$. We start by checking, whether e belongs to *any* path connecting s and t in G. This can be done by checking (using DFS) whether v is reachable from s and whether t is reachable from w.

If there is no such path, then we ignore e. Otherwise, let $P_e \subseteq E$ be such a path. We send $\frac{1}{2m}$ units of flow through this path, i.e., update $x := x + \frac{1}{2m} \cdot \chi_{P_e}$, where χ_{P_e} is the indicator vector of the set P_e.

At the end of this procedure every edge which lies on a path from s to t has at $x_e \geq \frac{1}{2m}$ and $x_e \leq m \cdot \frac{1}{2m} \leq \frac{1}{2}$. The remaining edges e have $x_e = 0$. In every step we need to update x at $O(n)$ positions (the lentgh of P_e), hence the total running time is $O(nm)$. □

Flows constructed in the lemma above are strictly positive, yet they fail to satisfy the constraint $Bx = F\chi_{s,t}$, i.e., not enough flow is pushed from s to t. To get around this problem we need the final idea of preconditioning. We add one directed edge $\hat{e} = (s, t)$ to the graph with large enough capacity $u_{\hat{e}} := F$ and very large cost $c_{\hat{e}} := 2\sum_{i \in E} |c_i|$. The new graph $(V, E \cup \{\hat{e}\})$ we denote by $\hat{G}$ and call the preconditioned graph.

By Lemma 12 we can construct a flow of value $\frac{1}{2} < F$, then we can send the remaining $F - \frac{1}{2}$ units directly from s to t through $\hat{e}$. This allows us to construct a strictly positive feasible solution on the preconditioned graph.

Note importantly that this preconditioning does not change the optimal solution to the original instance of the problem, since sending even one unit of flow through e incurs a large cost – larger then taking any path from s to t in the original graph G. Therefore solving the minimum cost flow problem on $\hat{G}$ is equivalent to solving it on G. Given this observation, it is enough to run the path following algorithm on the preconditioned graph. Below we show how to find a suitable starting point for the path following method in this case.

Lemma 13 (Finding a point on the central path)**.** *There is an algorithm that given a preconditioned instance of min-cost flow problem* (3) *outputs a feasible starting point $z_0 = (x_0, y_0)^\top \in \mathbb{R}^{2m}_{>0}$ and $\eta_0 = \Omega\left(\frac{1}{m^3 CU}\right)$ with $U := \max_{i \in E} |u_i|$ and $C := \max_{i \in E} |c_i|$ such that*

$$\left\|\widetilde{n}_{\eta_0}(z_0)\right\|_{z_0} < \frac{1}{6}.$$

This algorithm performs $\widetilde{O}(\sqrt{m}\log(U))$ iterations of the path following IPM.

Proof. Note first that we may assume without loss of generality that all edges belong to some directed path from s to t in G. Next, we apply Lemma 12 to find a flow $x > 1 \cdot \frac{1}{2m}$ in G of value $f \leq \frac{1}{2}$ and we set $x_{\hat{e}} = F - f$, where $\hat{e}$ is the edge between s and t added during preconditioning. This implies that $Bx = F\chi_{s,t}$ and moreover for every edges $e \in E \cup \{\hat{e}\}$ we have

$$\frac{1}{2m} \leq x_e \leq u_e - \frac{1}{2},$$

since $u_e \in \mathbb{Z}$ for every e. Therefore, by setting the slack variables y_e to $y_e := u_e - x_e$ we obtain that

$$\frac{1}{2} \leq y_e.$$

We now apply Lemma 10 with $\delta := \frac{1}{2m}$.

It remains to find an appropriate bound on the diameter of the feasible set. Note that we have for every $e \in E \cup \{\hat{e}\}$

$$0 \leq x_e, y_e \leq u_e,$$

which implies that the Euclidean diameter of the feasible set is bounded by

$$\sqrt{2\sum_{i=1}^{m} u_i^2} \leq \sqrt{2m}U.$$

Thus, finally, by plugging these parameters into Lemma 10, the bound on η_0 and the bound on the number of iterations follow. □

4 Self-Concordance

In this section, by abstracting out the crucial aspects of the convergence proof of the Primal Path Following IPM from the previous lecture, we arrive at various its extensions. We develop path following algorithms for more general convex bodies (not only polytopes) and also discuss how these ideas lead to the fastest known algorithm for linear programming.

4.1 Revisiting properties of the logarithmic barrier

Let us now go back to the analysis of the primal path following algorithm from the previous lecture and try to extract what was essential to make it work. At the heart of the argument is Lemma 4 (see Lecture 7) which says that the invariant $\left\|n_{\eta_k}(x_k)\right\|_{x_k} < \frac{1}{6}$ is preserved. The proof of this Lemma reveals the following two important properties of the logarithmic barrier function. (Below we adapt the notation used in Lecture 7.)

Lemma 14 (Properties of the logarithmic barrier function)**.** *Let $F(x) := -\sum_{i=1}^{m} \log s_i(x)$ be the logarithmic barrier function, $g(x)$ be its gradient and $H(x)$ its Hessian. Then, we have*

1. *for every* $x \in \mathrm{int}(P)$ *it holds*

$$\left\| H(x)^{-1} g(x) \right\|_x^2 \leq m,$$

2. *for every* $x \in \mathrm{int}(P)$ *and every* $h \in \mathbb{R}^n$ *it holds*

$$\left| \nabla^3 F(x)[h,h,h] \right| \leq 2 \left(h^\top H(x) h \right)^{3/2}.$$

Recall that the first property has been used to show that one can progress along the central path by changing η multiplicatively by roughly $\left(1 + \frac{1}{\sqrt{m}}\right)$ (see Lemma 6 in the previous lecture). In the second condition $\nabla^3 F(x)[h_1, h_2, h_3]$ denotes the third derivative of F computed with respect to directions $h_1, h_2, h_3 \in \mathbb{R}^n$, i.e.,

$$\nabla^3 F(x)[h_1, h_2, h_3] := \frac{d}{dt_1} \frac{d}{dt_2} \frac{d}{dt_3} F(x + t_1 h_1 + t_2 h_2 + t_3 h_3).$$

This property was not specifically mentioned in the proof, however it is responsible for the **NL** condition being satisfied for F. Thus, it allows to keep recentering the point (bringing it closer to the central path) using Newton's method (see Lemma 5 in the previous lecture).

Proof of Lemma 14. Part 1. has been already proved in the previous lecture . We provide a slightly different proof here. Let $x \in \mathrm{int}(P)$ and let $S \in \mathbb{R}^{m \times m}$ be the diagonal matrix with slacks $s_i(x)$ on the diagonal. We have

$$H(x) = A^\top S^{-2} A \qquad \text{and} \qquad g(x) = A^\top S^{-1} 1,$$

where $1 \in \mathbb{R}^m$ is the all-one vector. We have

$$\begin{aligned} \left\| H(x)^{-1} g(x) \right\|_x &= g(x)^\top H(x)^{-1} g(x) \\ &= 1^\top S^{-1} A (A^\top S^{-2} A)^{-1} A^\top S^{-1} 1 \\ &= 1^\top \Pi 1, \end{aligned}$$

where $\Pi := S^{-1} A (A^\top S^{-2} A)^{-1} A^\top S^{-1}$. Note now that Π is an orthogonal projection matrix: indeed Π is symmetric and $\Pi^2 = \Pi$ (by a direct calculation), hence

$$\left\| H(x)^{-1} g(x) \right\|_x = 1^\top \Pi 1 = \|\Pi 1\|^2 \leq \|1\|^2 = m.$$

To prove part 2. note first that

$$\nabla^3 F(x)[h,h,h] = 2 \sum_{i=1}^m \frac{\langle a_i, h \rangle^3}{s_i(x)^3} \qquad \text{and} \qquad h^\top H(x) h = \sum_{i=1}^m \frac{\langle a_i, h \rangle^2}{s_i(x)^2}.$$

Therefore the claim follows from the inequality $\|z\|_3 \leq \|z\|_2$ applied to the vector $z \in \mathbb{R}^m$ defined by $z_i := \frac{\langle a_i, h \rangle}{s_i(x)}$ for all $i = 1, 2, \ldots, m$.

We remark that this inequality holds very generally: for every $1 \leq p_1 \leq p_2 \leq \infty$ we have $\|z\|_{p_2} \leq \|z\|_{p_1}$. This general fact can be proved by showing that the unit ball with respect to ℓ_{p_2} is contained in the unit ball with respect to ℓ_{p_1}. □

4.2 Self-concordant barrier functions

The two conditions stated in Lemma 14 have been first abstracted out by Nesterov and Nemirovskii [8] to understand the path following method more generally. Any convex function that satisfies these two properties and is a "barrier" for the polytope P (tends to infinity when approaching the boundary) is called a *Self-Concordant Barrier Function*. A formal definition follows.

Definition 15 (Self-Concordant Barrier Function)**.** *Let $F : \text{int}(K) \mapsto \mathbb{R}$ be a smooth enough function. We say that F is a self-concordant barrier function with parameter ν if:*

1. **(Barrier)** *$F(x) \to \infty$ as $x \to \partial K$.*
2. **(Convexity)** *F is strictly convex.*
3. **(Complexity parameter ν)** *For all $x \in \text{int}(K)$,*
$$\nabla f(x)^\top \left(\nabla^2 F(x)\right)^{-1} \nabla f(x) \leq \nu.$$
4. **(Self-Concordance)** *For all $x \in \text{int}(K)$,*
$$\left|\nabla^3 F(x)[h,h,h]\right| \leq 2\left(h^\top \nabla^2 F(x) h\right)^{3/2}.$$

Note that the logarithmic barrier on P satisfies the above conditions with the complexity parameter equal to $\nu = m$. Indeed, condition 3. above coincides with property 1. in Lemma 14 and condition 4. is the same as property 2. in Lemma 14.

5 Linear Programming using self-concordant barriers

Similarly as we have done for the logarithmic barrier function, now given any self-concordant barrier function $F(x)$ for the polytope P one can define a perturbed objective to be

$$f_\eta(x) = \eta c^\top x + F(x)$$

and the central path to consist of minimizers $x^\star_\eta$ of the above parametric family of convex functions. The *Primal Path Following IPM* presented in the previous lecture can be adapted with almost no changes. The only difference with respect to the scheme presented in the

previous lecture is that m is replaced by ν – the complexity parameter of the barrier F and thus now, in step 2. the value η is updated according to the rule

$$\eta_{k+1} := \eta_k \left(1 + \frac{1}{20\sqrt{\nu}}\right),$$

and the iteration terminates once $\eta_T > \frac{\nu}{\varepsilon}$.

By repeating the proof of Theorem 9 given in Lecture 7 one obtains the following

Theorem 16 (Path following for Linear Programming with general barriers)**.** *Let $F(x)$ be a self-concordant barrier function on $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ with complexity parameter ν. The Primal Path Following IPM, after $T = O\left(\sqrt{\nu} \log \frac{\nu}{\varepsilon \eta_0}\right)$ iterations outputs $\hat{x} \in P$ that satisfies*

$$c^\top \hat{x} \leq c^\top x^\star + \varepsilon.$$

Moreover, every iteration (step. 2) requires computing the gradient and Hessian of F at given point $x \in \mathrm{int}(P)$ and solving one linear system of the form $H(x)y = z$, where , $z \in \mathbb{R}^n$ is a given vector and y is unknown.

What the above says is that by coming up with a self-concordant barrier function with parameter $\nu < m$ we obtain an algorithm that performs roughly $\sqrt{\nu}$ iterations instead of $\sqrt{m}$ as for the logarithmic barrier.

A lower bound and a tight example

A natural question one can ask is: since the complexity parameter of the barrier function on P determines the number of iterations of the path following algorithm on K, can we get arbitrarily good barriers? Can we bring the complexity parameter down to, say $O(1)$? A negative answer to this question has been provided in [8]. In the simplest setting they prove

Theorem 17 (Lower bound on the complexity parameter for hypercube [8])**.** *Every self-concordant barrier function for the hypercube $K := [0,1]^n$ has complexity parameter at least n.*

For the proof we refer to [8]. Below we present the reasoning for a one-dimensional interval following [9].

Proof in the special case of $n = 1$. Let $F : [0,1] \to \mathbb{R}$ be a self-concordant barrier function for $K = [0,1]$. Self-concordance in the univariate setting translates to: for every $t \in (0,1)$

$$F'(t)^2 \leq \nu \cdot F''(t),$$
$$\left|F'''(t)\right| \leq 2F''(t)^{3/2}.$$

We prove that if F is a strictly convex barrier then the above imply that $\nu \geq 1$. For the sake of contradiction, assume $\nu < 1$. Let us denote $g(t) := \frac{F'(t)^2}{F''(t)}$ and consider the derivative of g.

$$g'(t) = 2F'(t) - \left(\frac{F'(t)}{F''(t)}\right)^2 F'''(t) = F'(t)\left(2 - \frac{F'(t)F'''(t)}{F''(t)^2}\right). \tag{11}$$

Now, using the self-concordance of F, we obtain

$$\frac{F'(t)F'''(t)}{F''(t)^2} \leq 2\sqrt{\nu}. \tag{12}$$

Moreover since F is a convex function, it holds that $F'(t) > 0$ for $t \in (\alpha, 1]$ for some $\alpha \in (0, 1)$. Hence, for $t > \alpha$, by combining (11) and (12) we have

$$g'(t) \leq 2F'(t)(1 - \sqrt{\nu}).$$

Hence

$$g(t) \geq g(\alpha) + 2(1 - \sqrt{\nu})(F(t) - F(\alpha)),$$

which gives a contradiction since on the one hand $g(t) \leq \nu$ (from self-concordance of F) and on the other hand $g(t)$ is unbounded, since $\lim_{t\to 1} F(t) = +\infty$. □

Interestingly, it has been also shown in [8] that every full-dimensional convex subset of $\mathbb{R}^n$ admits a self-concordant barrier (called the *universal barrier*) with parameter $O(n)$. An alternative proof of this result can be also obtained using the *entropic barrier* (see [1]) as defined below.

Definition 18 (Entropic Barrier)**.** *Let $K \subseteq \mathbb{R}^n$ be any convex, compact subset of $\mathbb{R}^n$. For any $\theta \in \mathbb{R}^n$ define a probability distribution p_θ on K with density:*

$$p_\theta(y) \propto e^{\theta^\top y} \qquad \text{for } y \in K.$$

Then, the entropic barrier $E : \mathrm{int}(K) \to \mathbb{R}$ is defined, for any point $x \in \mathrm{int}(K)$ as

$$E(x) := \int_K p_{\theta(x)}(y) \log p_{\theta(x)}(y) dy,$$

where $\theta(x)$ is the unique point $\theta \in \mathbb{R}^n$ such that $\int_K y p_\theta(y) dy = x$.

Note however that just the existence of a $O(n)$-self-concordant barrier does not imply that we can construct efficient algorithms for optimizing over P in $\sqrt{n}$ iterations. Indeed the universal barrier and the entropic barrier are mathematical constructions which are rather hard to deal with from the computational viewpoint – computing gradients and Hessians of these barriers are non-trivial computational problems. Coming up with "efficient" $O(n)$-self-concordant barriers for various convex sets K is still an important open problem, which was recently resolved for the special case of polytopes [7]. Subsequently we review two important examples of barrier function for linear programming.

5.1 Volumetric Barrier

Definition of the barrier

Definition 19 (Volumetric Barrier)**.** *Let $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ and let $F : \text{int}(P) \to \mathbb{R}$ be the logarithmic barrier for P. We define the Volumetric Barrier on P as*

$$V(x) := \frac{1}{2} \log \det \nabla^2 F(x).$$

Moreover, the Hybrid Barrier on P is defined as

$$G(x) := V(x) + \frac{n}{m} F(x).$$

Discussion

The barrier functions $V(x)$ and $G(x)$ were developed by Vaidya [11] in order to improve upon the $\sqrt{m}$ iterations bound obtained for linear programming using the logarithmic barrier. His idea was to consider the Dikin ellipsoid centered at x

$$\mathcal{E}_x := \{u \in \mathbb{R}^n : (u - x)^\top H(x)(u - x) \leq 1\},$$

and define a function which at a point x measures the volume of the Dikin ellipsoid $\mathcal{E}_x$. Indeed, up and additive constant, this is what $V(x)$ does

$$V(x) = -\log \text{vol}(\mathcal{E}_x) + \text{const.}$$

Intuitively, since for every $x \in \text{int}(P)$, the Dikin ellipsoid $\mathcal{E}_x$ is contained in P (see Exercise 1 in Problem Set 1), whenever x comes close to the boundary of P, $\mathcal{E}_x$ becomes very small and thus $V(x)$ tends to $+\infty$.

Such a barrier $V(x)$ can be alternatively thought of as a weighted logarithmic barrier function. Indeed, one can prove that the Hessian of $\nabla^2 V(x)$ is multiplicatively close[4] to the following matrix $Q(x)$

$$Q(x) := \sum_{i=1}^{m} \sigma_i(x) \frac{a_i a_i^\top}{s_i(x)^2},$$

where $\sigma(x) \in \mathbb{R}^m$ is the vector of statistical leverage scores at x, i.e., for $i = 1, 2, \ldots, m$ we have

$$\sigma_i(x) := \frac{a_i^\top H(x)^{-1} a_i}{s_i(x)^2}.$$

The vector $\sigma(x)$ estimates the relative "importance" of each of the constraints. In fact, it holds that

$$\sum_{i=1}^{m} \sigma_i(x) = n \qquad \text{and} \qquad \forall i \;\; 0 \leq \sigma_i(x) \leq 1.$$

[4]By that we mean that there is a constant $\delta > 0$ such that $\delta Q(x) \preceq \nabla^2 V(x) \preceq \delta^{-1} Q(x)$.

To provide some intuitions: if A was an incidence matrix of a graph then $\sigma_i(x)$ is the probability of the edge i appearing in a random spanning tree (according to a product distribution depending on the point x) and thus it says how essential the edge i to the graph is – in particular if i is a bridge, then $\sigma_i(x) = 1$. If some constraint is now repeated multiple times, its importance will not scale with the number of repetitions (as in the logarithmic barrier) but – in fact if several constraints are similar to each other, their relative importance becomes quite small. This is intuitively countering the problem with the logarithmic barrier: treating every constraint equally.

If for a momement we assumed that the leverage scores $\sigma(x)$ do not vary with x, i.e., $\sigma(x) \equiv \sigma$ then $Q(x)$ would correspond to the Hessian $\nabla^2 F_\sigma(x)$ of the following *weighted logarithmic barrier*

$$F_\sigma(x) = \sum_{i=1}^{m} \sigma_i \log s_i(x).$$

Recall that in the case of the logarithmic barrier, each $\sigma_i = 1$ and hence the total "mass" on constraints is m, while here $\sum_{i=1}^{m} \sigma_i = n$.

Complexity parameter

The above interpretation of the volumetric barrier $V(x)$ as a weighted logarithmic barrier can be then used to prove that it satisfies condition 3. from the Definition 15 with parameter $\nu = O(n)$. However, it does not satisfy the self-concordance condition. Instead, one can show the following (see for instance Chapter 4 in [12])

$$\frac{1}{2}\nabla^2 V(x) \preceq \nabla^2 V(y) \preceq 2\nabla^2 V(x) \quad \text{whenever } \|x - y\|_x \leq O\left(m^{-1/2}\right).$$

To make the above true not only for x, y with $\|x - y\|_x \leq \left(m^{-1/2}\right)$ but also for $\|x - y\|_x \leq O(1)$, Vaidya [11] introduced a slight adjustment of the volumetric barrier: the hybrid barrier $G(x) = V(x) + \frac{n}{m} F(x)$.

Lemma 20 (Self-Concordance of Hybrid barrier [11])**.** *The barrier function $G(x)$ is self-concordant with complexity parameter $\nu = O(\sqrt{nm})$.*

Note that this gives an improvement over the logarithmic barrier, whenever $n = o(m)$.

Computational Complexity

To construct fast path following algorithms, not only the complexity parameter of the barrier plays a role, but also how much time does one need to perform one iteration. The next lemma states what is the cost of one iteration when using the Hybrid Barrier

Lemma 21 (Computational Complexity of Hybrid barrier [11])**.** *The Newton's step in the Path Following Interior Point method with respect to the hybrid barrier $G(x)$ can be calculated in total $O(m^\omega)$ (matrix multiplication) time.*

Recall that one iteration using the logarithmic barrier reduces to solving one linear system. Thus, the above matches the cost per iteration for the logarithmic barrier in the worst case scenario. However, for certain special cases (as for the min-cost flow problem) the cost per iteration using the logarithmic barrier might be significantly smaller, even $\widetilde{O}(m)$ instead of $O(m^\omega)$.

5.2 Lee-Sidford barrier

The Convex set

We consider full-dimensional polytopes of the form $P = \{x \in \mathbb{R}^n : Ax \leq b\}$.

Definition of the barrier

The Lee-Sidford barrier (a modified version of the barrier studied in [7]) is defined as follows

Definition 22. *Let $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ and let $F : \text{int}(P) \to \mathbb{R}$ be the logarithmic barrier for P. We define the Lee-Sidford Barrier at a point $x \in \text{int}(P)$ as*

$$K'(x) = \max_{w \geq 0} \left[\log\det \left(\sum_{i=1}^m w_i \frac{a_i a_i^\top}{s_i(x)^2} \right) - \frac{n}{m} \sum_{i=1}^m w_i \log w_i \right] + \frac{n}{m} F(x). \tag{13}$$

Discussion

The work of Lee and Sidford [7] is an extension of the ideas of Vaidya which finally led to an algorithm for linear with $\sim \sqrt{n}$ iterations, each of which is solving a small number of linear systems.

Recall that the barrier function by Vaidya is the logarithm of the volume of the Dikin ellipsoid around the point x. The nice property of the Dikin ellipsoid which intuitively stands behind the $\sqrt{m}$ iteration bound is that whenever the feasible region P is symmetric and say $\mathcal{E}_0$ is the Dikin ellipsoid centered at 0 then

$$\mathcal{E}_0 \subseteq P \subseteq \sqrt{m}\mathcal{E}_0,$$

i.e., $\mathcal{E}_0$ is a subset of P and by inflating it by a factor of $\sqrt{m}$ it contains the polytope P (the first inclusion appears in Homework 1 and the second one in Problem Set 7). Thus if one would just make one step towards the negative cost $-c$ to the boundary of $\mathcal{E}_0$ then in one this step a multiplicative progress of $\left(1 + \frac{1}{\sqrt{m}}\right)$ is made. This intuition is not completely accurate as it holds only if we start at the analytic center of P, however one may think of intersecting P with a constraint "$c^\top x \leq C$" for some guess C which shifts the analytic center of P to the new iterate and continue this way.

This provides us with the intuition that Dikin ellipsoid being a $\sqrt{m}$-rounding of the polytope P is responsible for the $\sqrt{m}$ iterations bound. To improve upon that it is then reasonable to look for better "roundings" (or better ways to approximate P by an ellipsoid). The most natural one to consider here is the "optimal rounding", i.e., the John's

ellipsoid Given $x \in P$, J_x is defined as the ellipsoid of maximum volume centered at x and contained in P. One can prove that in a similar setting as considered above for the Dikin ellipsoid it holds

$$J_0 \subseteq P \subseteq \sqrt{n} J_0.$$

This looks promising, as indeed m turned into n when compared to the bound for Dikin ellipsoid. Inspired by this, one can consider the following barrier function

$$K(x) := -\log \mathrm{vol}(J_x),$$

similarly as in the volumetric barrier of Vaidya. By employing the dual view on $K(x)$ we also obtain

$$K(x) = \max_{w \geq 0, 1^\top w = n} \log\det \left(\sum_{i=1}^{m} w_i \frac{a_i a_i^\top}{s_i(x)^2} \right). \tag{14}$$

This gives an interpretation of $K(x)$ as the "optimal" volumetric barrier.

As Lee and Sidford prove, such a barrier $K(x)$ indeed has complexity parameter n, however the optimal weight vector $w(x) \in \mathbb{R}^m$ turns out to be non-smooth, which makes it impossible to use $K(x)$ as a barrier for optimization. To counter this issue, Lee and Sidford study a certain smoothening of the "John's" barrier. One possible smoothening one can consider here is the barrier $K'(x)$ as defined in (13). The entropy term in $K'(x)$ is making the dependency of w on x smoother and similarly to the hybrid barrier, this one has also the logarithmic barrier "mixed in" with coefficient $\frac{n}{m}$.

Complexity parameter

Lemma 23 (Self-Concordance of Lee-Sidford barrier)**.** *The barrier function $K'(x)$ is self-concordant with complexity parameter $O(n \cdot \mathrm{polylog}(m))$.*

Note in particular that this matches (up to logarithmic factors) the lower bound of $\Omega(n)$ on the complexity paramater of a barrier of an n-dimensional polytope. It is an open problem to completely get rid of the the dependency on m for the complexity parameter of such a barrier, i.e., to replace $\mathrm{polylog}(m)$ by $\mathrm{polylog}(n)$ in the lemma.

Computational Complexity

While the complexity parameter of the barrier $K'(x)$ is close to optimal, it is far from clear how one can maintain the Hessian of such a barrier and be able to perform every iteration of the Newton's method in time comparable to one iteration for the logarithmic barrier (solving one linear system). The way this algorithm is implemented in [7] is by keeping track of the current iterate x_k and the current vector of weights w_k. At every iteration, both the point x_k and the weights w_k are updated so as to advance on the central path (determined by the current w_k) towards the optimum solution. For the sake of computational efficiency, the weights w_k never really correspond to optimizers of (13) with respect to x_k, but are obtained as another Newton step with respect to the old weights w_{k-1}. This finally leads to the following result.

Theorem 24 (Informal, Lee-Sidford algorithm for LP [7])**.** *There is an algorithm for the Linear Programming problem based on the path following scheme using the Lee-Sidford barrier $K'(x)$ that performs $\widetilde{O}\sqrt{n}\log\frac{1}{\varepsilon}$ iterations to compute an ε-approximate solution. Every iteration of this algorithm requires solving a polylogarithmic number of $n \times n$ linear systems.*

We note that the above result gives the currently fastest known general algorithm for linear programming.

6 Semidefinite Programming using IPM

Recall that in Section 1 we have used the logarithmic barrier for the positive orthant to come up with a linear programming algorithm for linear programs in the standard form

$$\begin{aligned} \min_{x\in\mathbb{R}^m} \quad & c^\top x, \\ \text{s.t.} \quad & Ax = b, \\ & x \geq 0. \end{aligned}$$

A generalization of the above to matrices is the following problem of *Semidefinite Programming*

$$\begin{aligned} \min_{X} \quad & \langle C, X\rangle, \\ \text{s.t.} \quad & \langle A_i, X\rangle = b_i, \quad \text{for } i = 1, 2, \ldots, m \\ & X \succeq 0. \end{aligned} \tag{15}$$

Above, X denotes a variable being a symmetric $n \times n$ matrix and $C, A_1, A_2, \ldots, A_m$ are arbitrary $n \times n$ matrices. The inner product on the space of symmetric matrices we use in the above is

$$\langle M, N\rangle := \operatorname{tr}(MN).$$

To derive an algorithm for semidefinite programming (15) all what is needed (as demonstrated formally in the subsequent section) is a self-concordant barrier for the cone $C_n = \{X \in \mathbb{R}^{n\times n} : X \text{ is symmetric and PD}\}$ of positive definite matrices

6.1 A Barrier for the Positive Definite Cone

Definition of the barrier

We start by a definition

Definition 25 (Logarithmic barrier on the PD cone)**.** *The logarithmic barrier function $F : C_n \to \mathbb{R}$ on C_n is defined to be*

$$F(X) = -\log\det X.$$

At this point it is worth noting that $F(X)$ is indeed well defined as $\det X > 0$ for a positive definite matrix X.

Discussion

The logarithmic barrier for C_n is a generalization of the logarithmic barrier for the positive orthant. Indeed, if we restrict to C_n to the set of diagonal, positive definite matrices, we obtain

$$F(X) = -\sum_{i=1}^{n} \log X_{i,i}.$$

More generally it is not hard to see that

$$F(x) = -\sum_{i=1}^{n} \log \lambda_i(X),$$

where $\lambda_1(X), \ldots, \lambda_n(X)$ are the eigenvalues of X.

Complexity Parameter

Recall that the logarithmic barrier for the positive orthant is self-concordant with parameter n, in the below lemma we show that this also holds when we generalize it to C_n.

Lemma 26. *The logarithmic barrier function F on the cone of positive definite matrices C_n is self-concordant with parameter n.*

Proof. To start, note that to verify the conditions in Definition (15) we need to understand first what vectors "h" to consider. Since we are moving in the cone C_n we would like to consider any H of the form $X_1 - X_2$ for $X_1, X_2 \in C_n$, which coincides with the set of all symmetric matrices. Let $X \in C_n$ and $H \in \mathbb{R}^{n\times n}$ be a symmetric matrix. We have

$$\begin{aligned} \nabla F(X)[H] &= -\mathrm{tr}(X^{-1}H), \\ \nabla^2 F(X)[H,H] &= \mathrm{tr}(X^{-1}HX^{-1}H), \\ \nabla^3 F(X)[H,H,H] &= -2\mathrm{tr}(X^{-1}HX^{-1}HX^{-1}H). \end{aligned} \tag{16}$$

To demonstrate the idea, here we just derive a formula for $\nabla F(X)[H]$ (the remaining ones are obtained similarly). We would like to compute

$$\lim_{t\to 0} -\frac{\log\det(X+tH) - \log\det(X)}{t}.$$

We have

$$\log\det(X+tH) - \log\det(X) = \log\det\left(I + tX^{-1/2}HX^{-1/2}\right) = \mathrm{tr}\log\left(I + tX^{-1/2}HX^{-1/2}\right).$$

Now, using the Taylor expansion $\log(I+Y) = Y + O(\|Y\|^2)$ we obtain

$$\mathrm{tr}\log\left(I + tX^{-1/2}HX^{-1/2}\right) = \mathrm{tr}\left(tX^{-1/2}HX^{-1/2}\right) + O(t^2).$$

Hence

$$\lim_{t\to 0} -\frac{\log\det(X+tH)-\log\det(X)}{t} = -\mathrm{tr}\left(X^{-1/2}HX^{-1/2}\right) = -\mathrm{tr}(X^{-1}H),$$

because trace satisfies $\mathrm{tr}(AB) = \mathrm{tr}(BA)$.

We are now ready to verify self-concordance of $F(X)$. The fact that X is a barrier follows simply from the fact that if X tends to ∂C_n then $\det(X) \to 0$. Convexity of F also follows, since

$$\nabla^2 F(X)[H,H] = \mathrm{tr}(X^{-1}HX^{-1}H) = \mathrm{tr}\left(H_X^2\right) \geq 0,$$

where $H_X := X^{-1/2}HX^{-1/2}$. To prove the 4th condition, note that

$$\nabla^3 F(X)[H,H,H] = -2\mathrm{tr}\left(H_X^3\right),$$

and

$$2\left|\mathrm{tr}\left(H_X^3\right)\right| \leq \mathrm{tr}\left(H_X^2\right)^{3/2} = \nabla^2 F(X)[H,H]^{3/2}.$$

We are left with the task of determining the complexity parameter of F. This follows simply from the Cauchy-Schwarz inequality, as

$$\nabla F(X)[H] = -\mathrm{tr}(H_X) \leq \sqrt{n}\cdot \mathrm{tr}(H_X^2)^{1/2} = \sqrt{n}\nabla^2 F(X)[H,H]^{1/2}.$$

Thus indeed the complexity parameter of F is n. □

Computational complexity

To apply the logarithmic barrier for optimization over the positive definite cone C_n we also need to show that $F(X)$ is efficient, as the next lemma demonstrates.

Lemma 27. *The Hessian and the gradient of the logarithmic barrier F at a point $X \in C_n$ can be computed in time required to invert the matrix X.*

Proof. Follows directly from the formulas for the gradient and Hessian of F in (16). □

7 Optimizing over general convex sets through self-concordance

We note that when defining the notion of self-concordance, we have not assumed that the underlying set K is a polytope. We require only that K is a full-dimensional convex subset of $\mathbb{R}^n$. Consequently, nothing stops us from using the *Primal Path Following IPM* to optimize over arbitrary convex sets. In fact, it allows us to solve a problem of minimizing a linear function $c^\top x$ over a convex set K and the corresponding convergence guarantee is analogous to Theorem 16.

Theorem 28 (Path following for arbitrary convex sets)**.** *Let $F(x)$ be a self-concordant barrier function on $K \subseteq \mathbb{R}^n$ with complexity parameter ν. The Primal Path Following IPM, after $T = O\left(\sqrt{\nu}\log\frac{1}{\varepsilon\eta_0}\right)$ iterations outputs $\hat{x} \in K$ that satisfies*

$$c^\top \hat{x} \leq c^\top x^\star + \varepsilon.$$

Moreover, every iteration (step. 2) requires computing the gradient and Hessian of F at given point $x \in \mathrm{int}(K)$ and solving one linear system of the form $H(x)y = z$, where , $z \in \mathbb{R}^n$ is a given vector and y is unknown.

Reduction from Convex Optimization to optimizing linear functions over convex sets

We remark that even though the above talks about minimizing linear functions over convex sets, one can also use this framework to optimize general convex functions. This follows from the fact that every convex program

$$\min_{x\in K} f(x)$$

can be transformed into a task of minimizing a linear function over a convex set, indeed, the above is equivalent to

$$\min_{(x,t)\in K'} t$$

where $K' = \{(x,t) : x \in K, f(x) \leq t\}$. This means that we just need[5] to construct a self-concordant function F on the set K' to solve $\min_{x\in K} f(x)$.

Using this framework, the problem of optimizing over a given convex set K is reduced to finding a self-concordant barrier function for K with a small complexity parameter. In the next section we discuss an interesting example of a barrier function whose complexity is not yet understood.

7.1 Canonical barrier

Definition of the barrier

Definition 29. *For a convex set $K \subseteq \mathbb{R}^n$ the canonical barrier is defined as the function $U : \mathrm{int}(K) \to \mathbb{R}$ such that*

$$U(x) = \frac{1}{2}\log\det\nabla^2 U(x),$$

and satisfies the boundary condition $\lim_{x\to\partial K} U(x) = +\infty$.

We note that even the fact of existence of U is a non-trivial question that relies on a Theorem of Cheng-Yau [2] showing existence of the Einstein-Hessian metric on certain manifolds.

[5]Note that this reduction makes the convex set K' rather complicated, as the function f is built in it. Nevertheless, this reduction is still useful in many interesting cases.

Discussion

To arrive at the definition of the canonical barrier let us go back to the Volumetric barrier $V(x)$ and its variant $G(x)$ – the hybrid barrier. By abstracting out the derivation of the hybrid barrier from the logarithmic barrier, one might consider the following operation: we are given a barrier function F and construct a new one by the following formula

$$G(x) := \log\det \nabla^2 F(x) + \frac{n}{m} F(x).$$

When going from the log-barrier to the hybrid barrier, the barrier parameter decreases from m to $\sqrt{mn}$. Can we gain even more by iterating this construction several times? What if we do it infinitely many times? This motivates the question of studying the fixed points of the above and the canonical barrier $U(x)$ is simply formalizing this concept (by adding boundary conditions to this second order differential equation).

Complexity parameter

The complexity parameter of $U(x)$ has been studied in the case of regular (full-dimensional with no lines) convex cones by [6]. As it turns out, the canonical barrier achieves the optimal (up to constant) barrier parameter.

Lemma 30 ([6])**.** *If $K \subseteq \mathbb{R}^n$ is a regular convex cone then the canonical barrier $U(x)$ on K is self-concordant with parameter $O(n)$.*

Computational complexity

The question of efficiently evaluating the canonical barrier at a point x, as well as efficient computation of its gradients and Hessians is an open problem.

References

[1] Sébastien Bubeck and Ronen Eldan. The entropic barrier: a simple and optimal universal self-concordant barrier. In *Proceedings of The 28th Conference on Learning Theory, COLT 2015, Paris, France, July 3-6, 2015*, page 279, 2015.

[2] Shiu-Yuen Cheng and Shing-Tung Yau. On the existence of a complete kähler metric on non-compact complex manifolds and the regularity of fefferman's equation. *Communications on Pure and Applied Mathematics*, 33(4):507–544, 1980.

[3] Thomas H Cormen. *Introduction to algorithms*. 2009.

[4] Samuel I. Daitch and Daniel A. Spielman. Faster approximate lossy generalized flow via interior point algorithms. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 451–460, 2008.

[5] Andrew V. Goldberg and Robert Endre Tarjan. Solving minimum-cost flow problems by successive approximation. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 7–18, 1987.

[6] Roland Hildebrand. Canonical barriers on convex cones. *Mathematics of operations research*, 39(3):841–850, 2014.

[7] Yin Tat Lee and Aaron Sidford. Matching the universal barrier without paying the costs : Solving linear programs with õ(sqrt(rank)) linear system solves. *CoRR*, abs/1312.6677, 2013.

[8] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, 1994.

[9] Yurii Nesterov. *Introductory lectures on convex optimization*, volume 87. Springer Science & Business Media, 2004.

[10] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC'04: Proceedings of the 36th Annual ACM Symposium on the Theory of Computing*, pages 81–90, 2004.

[11] P. M. Vaidya. A new algorithm for minimizing convex functions over convex sets. In *30th Annual Symposium on Foundations of Computer Science*, pages 338–343, Oct 1989.

[12] Nisheeth K. Vishnoi. A mini-course on convex optimization, 2014.

[13] Steve Wright. *Primal-Dual Interior-Point Methods*. Society for Industrial and Applied Mathematics, 1997.