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

# Lecture 3

September 29, 2022

*Instructor: Sepehr Assadi* *Scribe: Sepehr Assadi*

## Topics of this Lecture

## 1 Application I: Maximum Flow

In the max-flow problem, we have a directed graph $G = (V, E)$ with $n$ vertices and $m$ edges, source $s$, and sink $t$, and capacity $c_{u,v}$ on each edge $(u, v) \in E$. The goal is to route a maximum amount of flow from the source $s$ to sink $t$ while ensuring that no edge is responsible for routing more flow than its capacity, and that no flow is generated except out of source $s$ and no flow is consumed except by the sink $t$. This is a classical optimization problem that is often taught in undergrad algorithms' courses. See Figure 1 for an illustration.

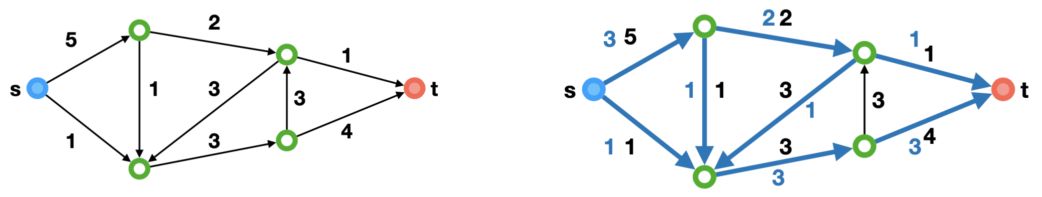

Figure 1: An input graph to the max-flow problem on the left with capacities denoted on the edges, and a maximum flow in this graph on the right.

We write the following LP for solving this problem:

$$\begin{aligned} \max_{f \in \mathbb{R}^{n^2}} \quad & \sum_{v \in V} f_{s,v} \\ \text{subject to} \quad & \sum_{u \in V} f_{u,v} = \sum_{w \in V} f_{v,w} \qquad \forall v \in V \setminus \{s,t\} && \text{(preservation of flow constraints)} \\ & 0 \leqslant f_{u,v} \leqslant c_{u,v} \qquad \forall (u,v) \in E. && \text{(capacity constraints)} \end{aligned}$$

It is straightforward to verify that this LP actually solves the original problem modulo a certain technicality: the flow $f \in \mathbb{R}^{n^2}$ may contain a *cycle*, meaning that the LP allows for some vertex $v \in V$ to "generate" some new flow, route it through a cycle back to $v$ and "consume" it again at $v$ (exercise: give a directed graph with a corresponding flow which is like this and is feasible and maximum). However, it is easy to see that given any flow $f \in \mathbb{R}^{n^2}$, we can repeatedly find a cycle in the support of the flow, reduce the flow on every edge of the cycle by the same amount until one of the edges no longer has any flow, thus "break" the cycle. This still preserves the total value of the flow, namely, the flow going out of $s$ (=going inside $t$) as we only reduced flow over a cycle. Repeatedly applying this then makes the flow cycle-free which is inline with the original definition of the problem.

## 2 Application II: Minimum Cut

In the min-cut problem, we have a directed graph $G = (V, E)$ with $n$ vertices and $m$ edges, source $s$, and sink $t$, and capacity $c_{u,v}$ on each edge $(u, v) \in E$. An $s$-$t$ cut in $G$ is any partition $S \sqcup T = V$ of vertices such that $s \in S$ and $t \in T$. We refer to edges going from $S$ to $T$ as *cut edges* of the cut $(S, T)$. The goal in this problem is to find a $s$-$t$ cut with the minimum total capacity on its cut edges. See Figure 2 below.

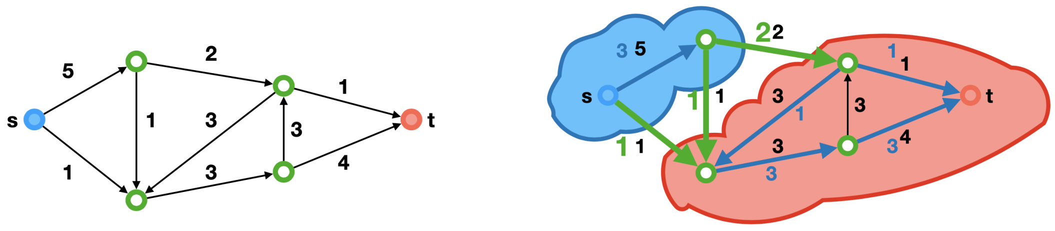

Figure 2: An input graph to the min-cut problem on the left (the same as the one for max-flow in Figure 1), and a minimum $s$-$t$ cut in this graph on the right (the blue set is $S$, the red set is $T$, and the green edges are cut edges).

The min-cut problem is intimately connected to the max-flow problem. For instance, it is easy to see that capacity of any $s$-$t$ cut $(S, T)$ is an *upper bound* on the maximum flow possible from $s$ to $t$; any flow going from $s$ to $t$ necessarily needs to use the cut edges of $(S, T)$ to "cross" the cut. This gives the following simple corollary:

$$\text{minimum } s\text{-}t \text{ cut capacity} \geqslant \text{maximum } s\text{-}t \text{ flow value.}$$

**Remark.** The connection between these problems however goes much deeper than this. For instance, the inequality above is actually always an equality, meaning that the minimum $s$-$t$ cut capacity is *equal* to the maximum $s$-$t$ flow in any graph. This is a topic related to the duality theory and will be covered in the next sections.

We write the following LP for solving this problem (strictly speaking, some of the constraints below are not necessarily and are added for simplification):

$$\begin{aligned}
\min_{y \in \mathbb{R}^n, z \in \mathbb{R}^m} \quad & \sum_{(u,v)\in E} c_{u,v} \cdot z_{u,v} \\
\text{subject to} \quad & y_v - y_u \leqslant z_{u,v} \qquad \forall (u,v) \in E \\
& y_s = 0 \quad \text{and} \quad y_t = 1 \\
& z_{u,v} \geqslant 0 \quad \text{and} \quad 0 \leqslant y_v \leqslant 1 \qquad \forall (u,v) \in E \text{ and } \forall v \in V.
\end{aligned}$$

The fact that this LP "solves" the min-cut problem is no longer trivial. Let us prove this in the following.

**Integral interpretation of the LP.** Firstly, as in Lecture 1, we start by seeing how this LP work *if* we instead solve this problem *integrally* instead of fractionally, i.e., when $y \in \mathbb{Z}^n, z \in \mathbb{Z}^m$ instead (note that this is no longer a linear program). Let $S := \{v \mid y_v = 0\}$ and $T := \{v \mid y_v = 1\}$, and since we assumed $y_v \in \{0, 1\}$ for all $v \in V$, and additionally $y_s = 0$ and $y_t = 1$, we get that $(S, T)$ is a $s$-$t$ cut. Now, consider the variables $z_{u,v}$ for $(u, v) \in E$:

- If $y_u = y_v$, then the second constraint of the LP becomes $z_{u,v} \geqslant 0$ and this is a minimization program and there are no additional constraints on $z_{u,v}$, we will have $z_{u,v} = 0$ in the optimal solution.
- If $y_u = 0$ and $y_v = 1$, then the second constraint of the LP becomes $z_{u,v} \geqslant 1$ and thus we will have $z_{u,v} = 1$ by the reasoning above.
- Finally, if $y_u = 1$ and $y_v = 0$, then the second constraint becomes $z_{u,v} \geqslant -1$ which is redundant because we also have the constraint $z_{u,v} \geqslant 0$.

The above implies that $z_{u,v} = 1$ iff $u \in S$ and $v \in T$ and otherwise it is 0; thus, $z$ is the characteristic vector of the cut edges of $(S, T)$. Thus, the value of the objective function is equal to the capacity of the cut $(S, T)$. As we are minimizing this in the program, this corresponds to solving the min-cut problem.

**Original LP and rounding.** We now consider working with the original LP (without the integrality constraints which were not linear). We design the following *randomized rounding* for turning an optimal solution to the LP into an integral one without decreasing the value in expectation; in other words, obtain an $s$-$t$ cut with the same value (in expectation) as that of the LP[1].

The rounding algorithm is as follows. Let $y \in \mathbb{R}^n, z \in \mathbb{R}^m$ be an optimal solution to the LP. Pick a random number $\theta \in (0, 1)$ uniformly at random and let $S := \{v \mid y_v \leqslant \theta\}$ and $T := \{v \mid y_v > \theta\}$. Given that $y_s = 0$, $y_t = 1$, and $\theta \in (0, 1)$, we get a valid $s$-$t$ cut $(S, T)$. For any edge $(u, v) \in E$, define an indicator random variable $X_{u,v} \in \{0, 1\}$ which is 1 iff $(u, v)$ is a cut edge of $(S, T)$, namely, $u \in S$ and $v \in T$. We have

$$\begin{aligned}
\mathop{\mathbb{E}}_{\theta}[\text{capacity of } (S,T)] &= \mathop{\mathbb{E}}_{\theta}\left[\sum_{(u,v)\in E} c_{u,v} \cdot X_{u,v}\right] = \sum_{(u,v)\in E} \mathop{\mathbb{E}}_{\theta}[X_{u,v}] && \text{(by the linearity of expectation)}\\
&= \sum_{(u,v)\in E} c_{u,v} \cdot \Pr\left(y_u \leqslant \theta < y_v\right) \\
& && \text{(as } (u,v) \text{ is a cut edge iff } u \in S\text{, i.e., } y_u \leqslant \theta\text{, and } v \in T\text{, i.e., } y_v > \theta)\\
&\leqslant \sum_{(u,v)\in E} c_{u,v} \cdot (y_v - y_u) && \text{(as } \theta \text{ is chosen uniformly from } (0,1))\\
&\leqslant \sum_{(u,v)\in E} c_{u,v} \cdot z_{u,v} && \text{(by the second constraint of the LP)}\\
&= \text{optimal LP value.}
\end{aligned}$$

This implies that the expected capacity of the cut output this way is at most equal to the minimum $s$-$t$ cut.

**Remark.** An important remark is in order here. While the above approach gives a randomized algorithm for rounding the value of the LP, a slightly more careful view of the same analysis implies that we can actually replace the random choice of $\theta \in (0, 1)$ with an *arbitrary* choice instead (for instance, simply take $\theta = 1/2$).

This is because in this algorithm, we always output an $s$-$t$ cut. Our analysis shows that the expected capacity of this cut is at most that of minimum $s$-$t$ cut. But then this implies that for *every* choice of $\theta$, the capacity of the returned cut should be equal to the minimum $s$-$t$ cut; otherwise, to "balance"

[1]Recall that since the LP is a *relaxation* of the min-cut problem, the optimum value of the LP is always at most as large as the optimum value to the min-cut problem; this rounding step implies that these values are equal to each other.

the expectation, some choice of $\theta$ should result in a value which is even less than the minimum $s$-$t$ cut which is not possible[a]

[a]if for a random variable $Y$, $\mathbb{E}[Y] = \min_{y \in \text{supp}(Y)} y$, then we should have that $\Pr(Y = \min_{y \in \text{supp}(Y)} y) = 1$.

## 3 The Simplex Algorithm

We now switch to designing our first "real" algorithm for LPs, called the *Simplex* algorithm designed by Dantzig. There is a vast literature on Simplex and many of its variants have been studied, especially because this algorithm for a very long term was the fastest known algorithm *in practice* (and perhaps in some cases still is). Yet, *in theory*, the **Simplex algorithm is <u>not</u> a polynomial time algorithm**, and thus does not have a particularly strong theoretical guarantee[2]. Given this (and other reasons), we are not going to spend too much time on the Simplex algorithm and will only review it at a high level in this lecture.

In the following, to provide further intuition about the algorithm, we are going to consider an example alongside a more general form of the algorithm. Moreover, there are several general exception handling steps in the Simplex algorithm but we are going to ignore all of them for now and present the most vanilla version, and then discuss those steps later in the lecture.

**Input.** The input to the problem is any LP.

*Example.* Our goal is to solve the following LP:

$$\begin{aligned} \max_{x_1, x_2 \in \mathbb{R}^2} \quad & x_1 + x_2 \\ \text{subject to} \quad & x_2 - x_1 \leqslant 1 \\ & x_1 \leqslant 3 \\ & x_2 \leqslant 2 \\ & x_1, x_2 \geqslant 0. \end{aligned}$$

*General form.* In general, we start with any LP:

$$\begin{aligned} \max_{x \in \mathbb{R}^n} \quad & c^T x \\ \text{subject to} \quad & Ax \leqslant b \\ & x \geqslant 0. \end{aligned}$$

(The last constraint $x \geqslant 0$ is added to make the exposition simpler and is not necessary so we can indeed have truly any LP here.)

**Step 1: Initial solution.** To run the Simplex algorithm, we need to start with a *feasible* solution to the input LP. This is in general a non-trivial step which is "as hard as solving LPs" but for now we are going to *assume* that $x = 0$ is a feasible solution to the LP.

*Example.* We start with solution $x_1 = x_2 = 0$ in the example.

*General form.* For now, we assume $x = 0$ is a feasible solution to the LP we like to solve.

[2]More on this later in this course.

**Step 2: Equational form.** We then write the LP in the equational form by the method of Lecture 2.

*Example.* Equational form:

$$\begin{aligned}\max_{x_1,x_2,s_1,s_2,s_3\in\mathbb{R}^2} \quad & x_1+x_2\\ \text{subject to} \quad & x_2-x_1+s_1=1\\ & x_1+s_2=3\\ & x_2+s_3=2\\ & x_1,x_2,s_1,s_2,s_3\geqslant 0.\end{aligned}$$

The linear system of this LP is the following:

$$\begin{bmatrix}-1 & 1 & 1 & 0 & 0\\ 1 & 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0 & 1\end{bmatrix}\cdot\begin{bmatrix}x_1\\x_2\\s_1\\s_2\\s_3\end{bmatrix}=\begin{bmatrix}1\\3\\2\end{bmatrix}.$$

*General form.* Equational form:

$$\begin{aligned}\max_{x\in\mathbb{R}^n,s\in\mathbb{R}^m} \quad & c^Tx\\ \text{subject to} \quad & [A \mid I_m]\cdot\begin{bmatrix}x\\s\end{bmatrix}=b\\ & x\geqslant 0.\end{aligned}$$

**Step 3: Starting with a basic feasible solution.** We then pick a basic feasible solution for the LP (recall the definition from Lecture 2). In case $x=0$ is a feasible solution, this step is simply to let the basis be the last $m$ columns of the constraint matrix, corresponding to the *slack* variables $s_1,\ldots,s_m$. This gives a basic feasible solution since the constraint matrix restricted to this part is $I_m$ which is full rank.

*Example.* $B=\{3,4,5\}$ (corresponding to $s_1,s_2,s_3$) is a basis for the following basic feasible solution:

$$\begin{bmatrix}x_1 & x_2 & s_1 & s_2 & s_3\end{bmatrix}=\begin{bmatrix}0 & 0 & 1 & 3 & 2\end{bmatrix}.$$

*General form.* Whenever $x=0$ is a feasible solution to the original LP (before the equational form), a basic feasible solution is then to set:

$$(x,s)=(0,b).$$

**Simplex tableau.** Each iteration of Simplex has a *Simplex tableau* that corresponds to some basic feasible solution. Each tableau is a different but equivalent way of writing the linear system of the LP so that: (*i*) the objective is expressed using only non-basic variables, (*ii*) each basic variable appears in the LHS of exactly one equation, and (*iii*) the RHS of the equations only consists of non-basic variables. For the first iteration,

*Example.* The starting Simplex tableau is:

$$\begin{aligned}s_1&=x_1-x_2+1\\ s_2&=3-x_1\\ s_3&=2-x_2\end{aligned}$$

---

$$o:=x_1+x_2.$$

*General form.* The starting Simplex tableau is:

$$s=b-A\cdot x$$

---

$$o:=c^Tx.$$

**Step 4: Pivot steps.** We then repeatedly find a non-basis variable with a positive coefficient in the objective value of the current tableau (the line with equation $o:=\cdots$); we increase this variable, called the *pivot*, as much as possible while checking that will not turn any of the basis variables in the given equations become negative. Whichever basis variable that becomes zero in this process is then swapped with pivot in the basis, namely, the pivot joins the basis and the other variable leaves it. This generates the next tableau which we then process again similarly.

*Example.* The pivot steps goes as follows:

- For the next iteration, we increase $x_2$ to 2, which brings $x_2$ to the basis and takes $s_3$ out:

$$\begin{aligned} s_1 &= x_1 + s_3 - 1 \\ s_2 &= 3 - x_1 \\ x_2 &= 2 - s_3 \end{aligned}$$

$$\overline{\phantom{xxxxxxxxxxxx}}$$

$$o := x_1 + 2 - s_3.$$

The corresponding basic feasible solution is:

$$\begin{bmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \end{bmatrix} = \begin{bmatrix} 0 & 2 & -1 & 3 & 0 \end{bmatrix}$$

- In the next iteration we increase $x_1$ to 3, which brings $x_1$ to the basis and takes $s_2$ out:

$$\begin{aligned} s_1 &= 2 - s_2 + s_3 \\ x_1 &= 3 - s_2 \\ x_2 &= 2 - s_3 \end{aligned}$$

$$\overline{\phantom{xxxxxxxxxxxx}}$$

$$o := 5 - s_2 - s_3$$

The corresponding basic feasible solution is:

$$\begin{bmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \end{bmatrix} = \begin{bmatrix} 3 & 2 & 2 & 0 & 0 \end{bmatrix}$$

*General form.* Each Simplex tableau looks like:

$$y = p + Q \cdot z$$

$$\overline{\phantom{xxxxxxxxxxxx}}$$

$$o := q + r^T z.$$

where $(y, z)$ form a permutation of variables $(x, s)$ (with $y$ being basis variables and $z$ being non-basis ones), $p \in \mathbb{R}^m$, $Q \in \mathbb{R}^{m \times n}$, $q \in \mathbb{R}$, and $r \in \mathbb{R}^n$. The rule for replacing the pivot is explained above and at this point we consider the choices to be arbitrary among all variables that satisfy the conditions.

**Step 5: Termination step.** We continue the above approach until we reach a situation wherein all the variables in the objective function have a *negative* coefficient.

*Example.* We are now at an optimal solution with value 5; the optimality can be seen as all the Simplex tableaus are corresponding to the same linear system (they are just different representation), and in any feasible solution of the LP, we need $s_2, s_3 \geqslant 0$, which implies that the objective value is always at most 5.

*General form.* The final Simplex tableau is:

$$y = p + Q \cdot z$$

$$\overline{\phantom{xxxxxxxxxxxx}}$$

$$o := q + r^T z,$$

where $r < 0$. We are now done as increasing any of $z$-variables can only reduce $o$ and since all the equations were always equivalent, and that we know $z \geqslant 0$ in any feasible solution, we have that the optimal solution has value $q$ and corresponds to the current basic feasible solution associated with this tableau.

The above description and example corresponds to the "cleanest" forms of running Simplex. In general, there are various exceptions and subtleties that need to be handled such as:

1. How to find a basic feasible solution at the first step? (a problem that we showed is as hard as solving a linear program in general)
2. What happens when objective value can become unbounded on the feasible region?
3. When the Simplex method end up not having an improving pivot step? (this is related to the notion of *degeneracy* and *cycling*)
4. What are some rules for picking which variable to leave the basis and which one to include, referred to as *Pivot rules*?

These parts are addressed in (possibly too much!) details in the course textbooks and for now we skip them.