Applications of Linear Programming: Part 2

Saladi Rahul, 3rd October 2023

1 Quick recap of the previous lecture

In the previous lecture on linear programming (LP), we discussed the "format" of an LP: a bunch of unknown variables, a linear objective function and a family of linear constraints based on the unknown variables. Then we saw a toy example with two variables. In this case, we could geometrically plot all the constraints, the feasible region and construct "level sets" of the objective function to find the last time it intersects the feasible region. Ignoring corner cases, the last time the level set of an objective function intersects the feasible region will be a *vertex* point. For the sake of these lectures, believe that linear programming can be solved in polynomial time. We will not have time to discuss the algorithms for solving an LP.

Then we saw four examples which were solved by mapping them to the linear programming format. The mapping for the maximum flow problem was quite straightforward. The latter three problems (motivated by machine learning) required a few simple tricks before they could be mapped to the linear programming format.

In this lecture, we will solve few more problems by using linear programming. For most students, this paradigm of mapping concrete problems into an abstract linear program is new and this typically requires some time and practice to get comfortable. This is the reason the instructor has decided to spend two full lectures where some popular problems in theoretical computer science and machine learning are solved using linear programming.

2 Shortest path problem

In this section we will study the (s, t) shortest path problem. The input is a directed graph $G = (V, E)$ with a source vertex s and a sink vertex t. Each edge $(u, v) \in E$ has a weight $w_{uv} > 0$. The goal is to report the *distance* of the shortest path from s to t.

Let $dist(v)$ be the distance of the shortest path from s to v, for all $v \in V$. As a base case, we have $dist(s) = 0$. Recall the following equation about shortest path distances,

$$dist(v) = \min_{(u,v)\in E} \{dist(u) + w_{uv}\}, \quad \text{for all } v \in V \setminus \{s\}.$$

However, as we saw in the previous lecture the "min" function cannot be directly encoded in a linear program! Lets start by introducing our variables: $d(v)$, for all $v \in V$. We will set $d(s) = 0$. The constraints of the linear program will be the following:

$$d(v) \leq d(u) + w_{uv}, \text{ for all } (u, v) \in E.$$

Attempt 1. Now we need to decide whether the objective function will be a min or a max. Lets start with min and see if it works. The LP will look as follows:

$$\begin{aligned}(*)\ \min \quad & d(t) \\ s.t. \quad & d(s) = 0 \\ & d(v) \leq d(u) + w_{uv}, \text{ for all } (u,v) \in E\end{aligned}$$

The claim is that the solution to the above LP will be $-\infty$. Why? So, min objective function will not work.

Attempt 2. This time let us use the max objective function. The LP will look as follows:

$$\begin{aligned}(**)\ \max \quad & d(t) \\ s.t. \quad & d(s) = 0 \\ & d(v) \leq d(u) + w_{uv}, \text{ for all } (u,v) \in E\end{aligned}$$

The above LP (**) might look counterintuitive since the objective function is max whereas we are studying the "shortest" path problem. However, we will now see that LP (**) will indeed return the correct answer.

Lemma 1. *For all $v \in V$, LP (**) returns $d(v) \leq dist(v)$.*

Proof. Let Π be the sequence of vertices in V in increasing order of their $dist(v)$ values. We will prove the lemma via induction. As a base case, the first element in Π is s. We have $d(s) = 0$ and $dist(s) = 0$, and hence $d(s) \leq dist(s)$. In general, let $v \neq s$ be an arbitrary vertex in Π and assume that $d(u) \leq dist(u)$ for all $d(u) < d(v)$. Let the last edge on the shortest path from s to v be (u, v). Then,

$$\begin{aligned}dist(v) &= dist(u) + w_{uv} \\ &\geq d(u) + w_{uv} \quad \text{(via induction hypothesis and } w_{uv} > 0) \\ &\geq d(v) \quad \text{(via constraint in LP (**))}\end{aligned}$$

□

In fact, now we will claim that LP (**) returns a more stronger result.

Lemma 2. *LP (**) returns $d(t) = dist(t)$.*

Proof. A feasible solution to LP (**) is obtained by setting $d(v) = dist(v)$, for all $v \in V$. In particular, this implies that $d(t) = dist(t)$ is a feasible solution. By Lemma 1, we know that $d(t) > dist(t)$ is not possible. Since LP (**) has a max objective function, it will therefore return $d(t) = dist(t)$. □

Exercise. What is the value returned by LP (**) if there is no path from s to t?

In the next iteration, add what happens (a) if the weight of the edges are negative and (b) if there are negative weight cycles.

3 Maximum weight bipartite matching

Motivation. Google ads is a huge revenue generating machine for Google. Consider the following simple setting. There are n slots which Google has opened on their website for costumers to place their ads. We have m customers each wanting to place one ad on Google website. A customer i has interest in a subset of the slots and is willing to pay an amount p_{ij} for slot j. Naturally, the goal from the perspective of Google will be to maximize the profit earned by matching costumers to the ad slots.

Problem statement. We are given a bi-partite graph $G = (X \cup Y, E)$. Each edge $e = (u, v) \in E$ with $u \in X$ and $v \in Y$ has a weight w_e. A *perfect matching* is a set $M \subseteq E$ such that each vertex in X and Y is incident to exactly one edge in M. The goal is to report that perfect matching M which maximizes the total weight of the edges in M.

See Figure 1 for an example. We have three customers $A, B,$ and C, and three ad slots $1, 2,$ and 3. It is easy to see that applying any simple greedy strategy can lead to a sub-optimal solution. For example, if we start with A and lets say it picks slot 3. Then lets say B picks slot 2. Eventually C is only left with the option of taking slot 1. This leads to a profit of 400. However, the optimal solution is obtained by the following perfect matching: $A->1, B->3, C->2$. Then a profit of 600 is obtained.

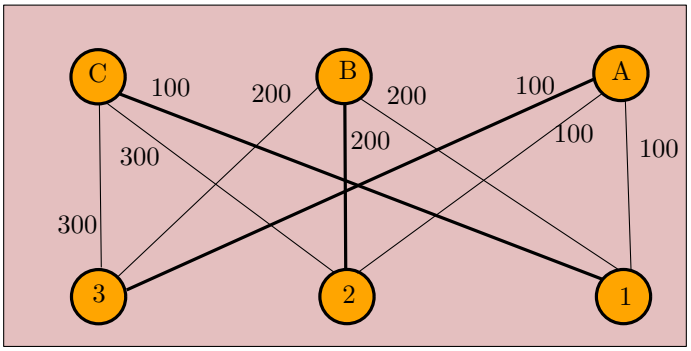


Figure 1: A bi-partite graph with three nodes on each side.

Lets start writing the LP for the maximum weight matching problem. What are the decision variables? Each edge is either picked or not picked. Let x_e be the decision variable for each edge $e \in E$. Consider the following program IP

$$
\begin{aligned}
(IP) \quad \max \quad & \sum_{e \in E} w_e x_e \\
s.t. \quad & \sum_{e=(u,v) \in E} x_e = 1, \forall v \in (X \cup Y) \\
& x_e \in \{0, 1\}
\end{aligned}
$$

Convince yourself that any feasible solution to the integer program will correspond to a feasible matching. However, is IP a linear program? No, because $x_e \in \{0, 1\}$ is not a linear

equation. How do we convert the above IP into an LP? By relaxing the condition that x_e has to be integral to allowing x_e being fractional.

$$
\begin{aligned}
(LP) \quad \max \quad & \sum_{e \in E} w_e x_e \\
s.t. \quad & \sum_{e=(u,v) \in E} x_e = 1, \quad \forall v \in (X \cup Y) \\
& 0 \leq x_e \leq 1, \quad \forall e \in E
\end{aligned}
$$

- Let OPT_{IP} and OPT_{LP} be the optimal value of the program IP and the linear program LP. What is the relation between OPT_{IP} and OPT_{LP}? Clearly, $OPT_{IP} \leq OPT_{LP}$ since any feasible solution to IP is also a feasible solution for LP.
- The maximum weight matching will have value equal to OPT_{IP}. So, what good is it to solve LP if $OPT_{LP} \gg OPT_{IP}$? Luckily, for this linear program it turns out that $OPT_{LP} = OPT_{IP}$! (Actually, it follows from the property of *unimodular matrices*).
- Finally, is it enough if $OPT_{LP} = OPT_{IP}$? The answer is NO, since the x_e values returned by the LP can be fractional. See Figure 2.

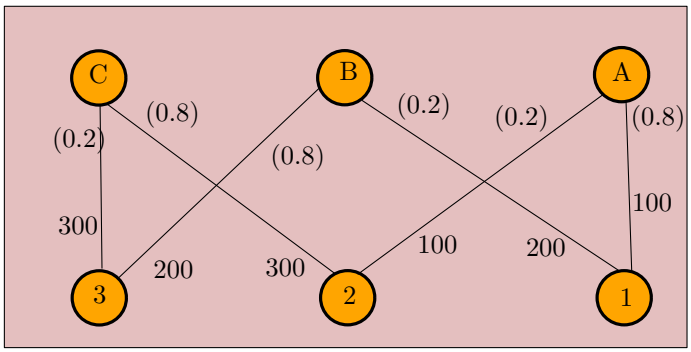


Figure 2: Fractional x_e values assigned to edges. The edges not shown in the figure have $x_e = 0$. In this case, $OPT_{LP} = 0.2*300+0.8*300+0.2*200+0.8*200+0.2*100+0.8*100 = 600$.

However, there is a nice trick to convert any optimal solution with fractional values into an optimal solution with all integral values (either 0 or 1). We state the lemma below. The proof of this lemma is taken from Matousek and Gartner's book on Linear Programming.

Lemma 3. *If there is a feasible solution for (LP), then there exists an optimal solution in which all the variables are integral.*

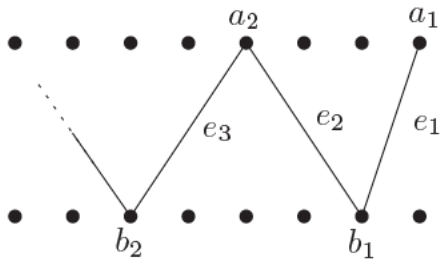
Proof of Theorem 3.2.1. Let $\mathbf{x}^*$ be an optimal solution of the LP relaxation, and let $w(\mathbf{x}^*) = \sum_{e \in E} w_e x_e^*$ be the value of the objective function at $\mathbf{x}^*$. Let us denote the number of nonintegral components of the vector $\mathbf{x}^*$ by $k(\mathbf{x}^*)$.

If $k(\mathbf{x}^*) = 0$, then we are done. For $k(\mathbf{x}^*) > 0$ we describe a procedure that yields another optimal solution $\tilde{\mathbf{x}}$ with $k(\tilde{\mathbf{x}}) < k(\mathbf{x}^*)$. We reach an integral optimal solution by finitely many repetitions of this procedure.

Let $x_{e_1}^*$ be a nonintegral component of the vector $\mathbf{x}^*$, corresponding to some edge $e_1 = \{a_1, b_1\}$. Since $0 < x_{e_1}^* < 1$ and

$$\sum_{e \in E : b_1 \in e} x_e^* = 1,$$

there exists another edge $e_2 = \{a_2, b_1\}$, $a_2 \neq a_1$, with $x_{e_2}^*$ nonintegral. For a similar reason we can also find a third edge $e_3 = \{a_2, b_2\}$ with $0 < x_{e_3}^* < 1$. We continue in this manner and look for nonintegral components along a longer and longer path $(a_1, b_1, a_2, b_2, \ldots)$:

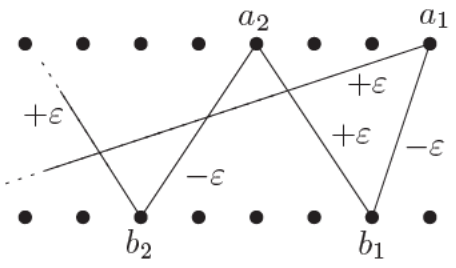


Since the graph G has finitely many vertices, eventually we reach a vertex that we have already visited before. This means that we have found a *cycle* $C \subseteq E$ in which $0 < x_e^* < 1$ for all edges. Since the graph is bipartite, the length t of the cycle C is even. For simplicity of notation let us assume that

the edges of C are $e_1, e_2, \ldots, e_t$, although in reality the cycle found as above need not begin with the edge e_1.

Now for a small real number ε we define a vector $\tilde{\mathbf{x}}$ by

$$\tilde{x}_e = \begin{cases} x^*_e - \varepsilon & \text{for } e \in \{e_1, e_3, \ldots, e_{t-1}\} \\ x^*_e + \varepsilon & \text{for } e \in \{e_2, e_4, \ldots, e_t\} \\ x^*_e & \text{otherwise.} \end{cases}$$



It is easy to see that this $\tilde{\mathbf{x}}$ satisfies all the conditions

$$\sum_{e \in E: v \in e} \tilde{x}_e = 1, \quad v \in V,$$

since at the vertices of the cycle C we have added ε once and subtracted it once, while for all other vertices the variables of the incident edges haven't changed their values at all. For ε sufficiently small the conditions $0 \le \tilde{x}_e \le 1$ are satisfied too, since all components $x^*_{e_i}$ are strictly between 0 and 1. Hence $\tilde{\mathbf{x}}$ is again a feasible solution of the LP relaxation for all sufficiently small ε (positive or negative).

What happens with the value of the objective function? We have

$$w(\tilde{\mathbf{x}}) = \sum_{e \in E} w_e \tilde{x}_e = w(\mathbf{x}^*) + \varepsilon \sum_{i=1}^{t} (-1)^i w_{e_i} = w(\mathbf{x}^*) + \varepsilon\Delta,$$

where we have set $\Delta = \sum_{i=1}^{t} (-1)^i w_{e_i}$. Since $\mathbf{x}^*$ is optimal, necessarily $\Delta = 0$, for otherwise, we could achieve $w(\tilde{\mathbf{x}}) > w(\mathbf{x}^*)$ either by choosing $\varepsilon > 0$ (for $\Delta > 0$) or by choosing $\varepsilon < 0$ (for $\Delta < 0$). This means that $\tilde{\mathbf{x}}$ is an optimal solution whenever it is feasible, i.e., for all ε with a sufficiently small absolute value.

Let us now choose the largest $\varepsilon > 0$ such that $\tilde{\mathbf{x}}$ is still feasible. Then there has to exist $e \in \{e_1, e_2, \ldots, e_t\}$ with $\tilde{x}_e \in \{0, 1\}$, and $\tilde{\mathbf{x}}$ has fewer nonintegral components than $\mathbf{x}^*$. □

Let us now consider another situation, in which we have more employees than positions and we want to fill all positions optimally, i.e., so that the sum of scores is maximized. Then we have $|X| < |Y|$