Lecture 11: The Simplex Algorithm

Lecturer: *Huacheng Yu*

Today we will discuss the *simplex algorithm* for solving LPs. Interestingly, the simplex algorithm is known to have worst-case exponential running time, but it is very efficient in practice. We will work with a maximization LP, i.e.,

$$\begin{aligned} \max \quad & \sum_{i\in[n]} c_i x_i \\ s.t. \quad & \sum_{i\in[n]} a_{ji} x_i \le b_j \qquad \forall j \in [m] \\ & x_i \ge 0 \qquad\qquad\quad \forall i \in [n] \end{aligned}$$

1 Basic Feasible Solutions

Definition 1 (Basic Feasible Solution)**.** *Let x be a feasible solution to an LP in $\mathbb{R}^n$. x is a* basic feasible solution *(BFS) if there are n linearly independent constraints in the LP which are tight (equality holds).*

Lemma 1. *Every LP with bounded value has an optimal solution which is a BFS.*

Proof sketch. If x satisfies fewer than n linearly independent constraints with equality, then the intersection of these constraints is a subspace of nonzero dimension. Then we can move inside this subspace until we hit another linearly independent constraint. Such a constraint is guaranteed to exist because of the n linearly independent constraints $x_i \ge 0$. □

We can think of the constraints of the LP as the faces of a polytope, and a BFS as a vertex of the polytope. The idea behind the simplex algorithm is to start from a vertex of the polytope, and locally improve the solution by moving along an edge to a neighboring vertex. When no more improvements are possible, we output the vertex we ended at.

Finding an initial BFS to start from is actually nontrivial: we need to find n constraints that can be satisfied with equality while also satisfying every other constraint. It turns out that we can find a BFS using a different LP, but for this lecture, we will focus on the local improvement step.

2 Pivoting

We will now discuss the local improvement step of the simplex algorithm, also known as the *pivoting* step. The description of a pivot is simple: we start at a BFS, which satisfies n constraints with equality. We relax one of these constraints, which allows us to move along a 1-dimensional subspace (an edge). Then we move as much as possible until

another constraint becomes tight (we hit another vertex). The cleverness comes in doing this efficiently.

To simplify the analysis, we will introduce *slack variables* $y_j \geq 0$, which represent the difference between the two sides of each LP constraint. Then we can equivalently rewrite each constraint j as an equality like so:

$$\sum_{i \in [n]} a_{ji} x_i \leq b_j \qquad \longrightarrow \qquad \sum_{i \in [n]} a_{ji} x_i + y_j = b_j \ .$$

What now becomes nice is that every constraint is an equality constraint, except the non-negativity constraints on each variable. This "standardization" of the inequality constraints allows for a much cleaner analysis.

Observe that in this new LP, we have the n original variables x_i, plus the m new slack variables y_j, for a total of $n + m$ variables. We also have the m original constraints (which have been transformed into equalities), as well as the $n+m$ non-negativity constraints, one for each of our $n + m$ variables. In a BFS, the number of constraints satisfying equality will be equal to the number of variables. Therefore, in a BFS for the new LP, the $n + m$ constraints that take equality should include n variables among $x_1, \ldots, x_n, y_1, \ldots, y_m$ taking value 0.

Recall that we can write the LP in matrix form as $\max c^\top x$ subject to $Ax = b$, where we abuse notation so that $x = (x_1, \ldots, x_{m+n})$ includes the newly introduced slack variables $y_1, \ldots, y_m$, and c, A include the appropriate coefficients for the new entries of x.

WLOG assume A has rank m, otherwise, we could simply remove redundant constraints. We can use row additions to force m independent columns to form the identity matrix. Of the $n + m$ variables, n of them are 0. We call these *non-basic* variables, and the other m variables *basic* variables. Use row addition and column swapping to force A into the following (illustrated for $n = 3, m = 5$)

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & ? & ? & ? \\ 0 & 1 & 0 & 0 & 0 & ? & ? & ? \\ 0 & 0 & 1 & 0 & 0 & ? & ? & ? \\ 0 & 0 & 0 & 1 & 0 & ? & ? & ? \\ 0 & 0 & 0 & 0 & 1 & ? & ? & ? \end{pmatrix} ,$$

where the first m columns correspond to the basic variables, and the question marks denote arbitrary values. This corresponds to using Gaussian Elimination to calculate the values of basic variables, given the values of non-basic variables (which are 0). It is possible to transform A into this form using elementary row operations, because we assumed in the definition of BFS that the constraints taking equality are linearly independent, and for each non-basic variable x_i, there is a constraint $x_i \geq 0$ taking equality.

We can then use row addition to zero out the entries of c corresponding to the basic

variables, yielding

$$c' = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & ? & ? & ? \end{pmatrix}$$

$$A' = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & ? & ? & ? \\ 0 & 1 & 0 & 0 & 0 & ? & ? & ? \\ 0 & 0 & 1 & 0 & 0 & ? & ? & ? \\ 0 & 0 & 0 & 1 & 0 & ? & ? & ? \\ 0 & 0 & 0 & 0 & 1 & ? & ? & ? \end{pmatrix} \begin{pmatrix} ? \\ ? \\ ? \\ ? \\ ? \end{pmatrix} = b'$$

Also note that since the current BFS is a feasible solution, b' must be nonnegative in every coordinate. Then to pivot, we do the following:

Step 1. Choose a non-basic variable x'_i such that $c'_i > 0$. Recall that non-basic variables are 0 in the current BFS, so the condition means that increasing the variable improves the value of the LP.

Claim 2. *If all non-basic $c'_i \leq 0$, then the current BFS is optimal.*

Proof. If all non-basic $c'_i \leq 0$, then the transformed objective (which is maximized when the original objective is maximized) satisfies $c'^{\top} x' = \sum_{\text{non-basic } x'_i} c'_i x'_i \leq 0$. Since the current BFS is feasible and sets all non-basic $x'_i = 0$, it attains the optimal value. □

Step 2. Choose a basic variable x'_k such that $a'_{ki} > 0$ and minimizes b'_k / a'_{ki}. Recall that for the constraint represented by row k, the non-basic variables are set to 0 and the basic variables have coefficients of 0, except for x'_k. Therefore, when we increase the value of x'_i, we need to maintain the equality constraints $x'_k + a'_{ki} x'_i = b'_k$ for all $k \in [m]$ by increasing/decreasing x'_k accordingly, while also ensuring that $x'_k \geq 0$ for all $k \in [m]$. It holds that the x'_k is the first variable that will hit 0 in this process.

If x'_k does not exist (i.e., $a'_{ji} \leq 0$ for all j), then the LP is unbounded because we can indefinitely increase x'_i without hitting a non-negativity constraint for another variable.

Step 3. Make x'_i basic and x'_k non-basic: increase x'_i until $x'_k = 0$, then maintain the format of c', A' by dividing row k by a'_{ki}, using row addition to zero out the other entries in column i, and swapping columns i and k.

Analysis. By construction, the value of the BFS never decreases. To avoid revisiting the same BFS multiple times, we can choose the smallest possible indices in steps 1 and 3 (proof omitted). Therefore, the simplex algorithm terminates after at most $\binom{m+n}{n}$ (the number of BFSs) iterations. It turns out that there exist LPs that force roughly this worst-case behavior, but most LPs in practice take few iterations.

3 Finding an initial BFS

Given a linear program, it is nontrivial to find a BFS to start the above algorithm, since the unique interesection of the boundary of n constraints may not satisfy the other constraints. It turns out that this can be done by solving another linear program.

Consider an LP with constraints $\sum_i a_{ji} x_i = b_j$ and $x_i \geq 0$ (the objective is not important when finding any BFS). We introduce another variable z_j for each constraint, and write a new LP.

$$\begin{aligned}
\max \quad & -\sum_j z_j \\
s.t. \quad & \forall j, z_j + \sum_i a_{ji} x_i = b_j, \qquad (\text{if } b_j \geq 0) \\
& \text{or } -z_j + \sum_i a_{ji} x_i = b_j, \qquad (\text{if } b_j < 0) \\
& \forall i, x_i \geq 0, \\
& \forall j, z_j \geq 0.
\end{aligned}$$

Note this new LP has a trivial feasible solution: for all i, $x_i = 0$, and for all j, $z_j = |b_j|$. It is easy to verify that this solution is also basic. Hence, we can repeatedly applying the above pivoting operations to compute the optimal solution for this LP.

If the optimal value is 0, this can directly be translated to a BFS for the original LP. The optimal solution must have all $z_j = 0$. Hence, it suffices to drop the new variables z_j, as it will remove m variables and m tight constraints. One may continue the algorithm from this BFS for the original LP.

If the optimal value is nonzero, the original LP is infeasible. This is because the optimal value must be negative, while any feasible solution for the original LP yields a solution with objective value 0 (by setting all $z_j = 0$).