Simplike: A Simplex-like Algorithm

By: Jordan Long

We are designing an algorithm to solve a linear program (LP) of the form:

$$\begin{aligned} max \quad & c^T x \\ where \quad & Ax \leq b \end{aligned}$$

where A is an m by n matrix, $c \in \mathbb{R}^n$, $x \in \mathbb{R}^n$, and $b \in \mathbb{R}^m$.

For simplicity of the algorithm, we assume the following:

- The polyhedron $P = \{x : Ax \leq b\}$ is pointed, as in it contains no lines. This means that given any $x, d \in \mathbb{R}^n$, the set $\{x + \lambda d : \lambda \in \mathbb{R}\}$ is not a subset of our polyhedron. It can be shown that this guarantees the existence of basic feasible solutions (BFS), which can be thought of visually as corners of our polyhedron. The existence of BFS implies that $rank(A) = n$, so $m \geq n$. Also, if the problem has an optimal solution, it can be shown that it has an optimal solution that is a BFS.
- All BFS are non-degenerate, meaning that there are exactly n linearly independent tight constraints at a BFS.
- We are given a BFS $\bar{x}$.

A general description of the algorithm is as follows:

1. Check if the current BFS is optimal.
2. If it is not optimal, move to a better BFS or say the problem is unbounded.

Definition: For any m by n matrix M and $J \subseteq \{1, ..., m\}$, we define M_J as the matrix obtained from the rows indexed by J.

Definition: $B \subseteq \{1, ..., m\}$ is a basis if $|B| = n$ and $rank(A_B) = n$. Given a basis B, let $N = \{1, ..., m\} \setminus B$. We call the constraints indexed by B the *basic constraints* and those indexed by N the *nonbasic constraints*.

Let $\bar{x}$ be a BFS to our LP. By the non-degeneracy of P, there are n linearly independent tight constraints at $\bar{x}$ so let B be the basis formed by

these constraints. Note that $A_B\bar{x} = b_B$ and A_B is invertible since it has rank n.

We want to find some way of characterizing the optimality of our BFS, and of moving in the direction of a better BFS if the current one is not optimal. If we are going to travel away from $\bar{x}$, it intuitively makes sense to travel along an 'edge' of the polyhedron P in a direction that increases the value of $c^T x$. How can we characterize this?

Let $d \in \mathbb{R}^n$ be our direction of travel, and $\theta \in \mathbb{R}$ with $\theta > 0$. For feasibility, we require $A(\bar{x} + \theta d) \leq b$. Since $A_B\bar{x} = b_B$, to ensure that $A_B(\bar{x} + \theta d) \leq b_B$ we must choose d so that $\theta A_B d \leq 0$. Since $A_N\bar{x} < b_N$, to ensure that $A_N(\bar{x} + \theta d) \leq b_N$ we can choose any d, and then restrict θ as necessary.

To capture the idea of moving along an 'edge' to an adjacent BFS, we maintain all but one of the tight constraints at $\bar{x}$. Thus we want

$$A_B d = \begin{bmatrix} 0 \\ \vdots \\ -1 \, in \, row \, j \\ \vdots \\ 0 \end{bmatrix} \iff d = -A_B^{-1} e_j, \; j \in \{1, ..., n\}$$

If we choose d in this way, we preserve feasibility since

$$A_B(\bar{x} + \theta d) = A_B\bar{x} + \theta A_B(-A_B^{-1} e_j) = b_B - \theta e_j \leq b_B$$

Note that $A_B^{-1} e_j$ is precisely column j of A_B^{-1}, so we want to move in precisely that direction for some j. Which j do we choose? We want to travel in a direction of increasing cost. That is, we want $c^T x < c^T(\bar{x} + \theta d)$. Thus we need

$$c^T d > 0 \iff -c^T A_B^{-1} e_j > 0 \iff 0 > c^T A_B^{-1} e_j$$

So if we can find a j such that $0 > c^T A_B^{-1} e_j$, we know that traveling along the direction $A_B^{-1} e_j$ increases the value of $c^T x$. If $\forall j \in \{1, ..., n\}$ we have $0 \leq c^T A_B^{-1} e_j$, or equivalently $(A_B^{-1})^T c \geq 0$, then going to any adjacent BFS would decrease the cost. Since we are optimizing what turns out to be a convex function on a convex set, our local optimum is a global optimum and the current BFS is the optimal solution.

Result: If $(A_B^{-1})^T c \geq 0$, then the current BFS is optimal and our algorithm can terminate.

If the current BFS is not optimal, then there exists j such that $c^T A_B^{-1} e_j < 0$, and so we let $d = -A_B^{-1} e_j$ and move along that direction until one of the other constraints in $A_N((x) + \theta d) \leq b_N$ is tight. When does this occur?

First note that if $A_N d \leq 0$, then the problem is unbounded since $\forall \theta \in \mathbb{R}$ with $\theta > 0$, $\bar{x} + \theta d$ is feasible and moving in that direction increases the cost, as previously noted.

Result: If $A_N d \leq 0$, then the LP is unbounded and our algorithm can terminate.

Let us denote row i of A_N as a_{Ni}. If the problem is not unbounded, then there exists $i \in \{1, ..., m-n\}$ such that $a_{Ni} d > 0$ so we can choose θ such that $a_{Ni}(\bar{x} + \theta d) = b_{Ni}$ by setting $\theta = \frac{b_{Ni} - a_{Ni}\bar{x}}{a_{Ni} d}$. We introduct a new tight constraint and maintain feasibility if we set

$$\theta = min_{i\, with\, a_{Ni} d > 0} \left\{ \frac{b_{Ni} - a_{Ni}\bar{x}}{a_{Ni} d} \right\}$$

Thus index j leaves the basis and index i enters the basis. But how do we know that $\bar{x} + \theta d$ is actually a BFS? We know that there are at least n constraints tight at $\bar{x} + \theta d$ since we left $n-1$ tight in $A_B(\bar{x} + \theta d) \leq b_B$ and introduced at least one more in $A_N(\bar{x} + \theta d) \leq b_N$. We just need to show that the new tight constraint is linearly independent from the other $n-1$ constraints and it will follow that $\bar{x} + \theta d$ is a BFS.

Let $\bar{B} = B \cup \{i\} \setminus \{j\}$. Suppose for a contradiction that the rows in $A_{\bar{B}}$ were linearly dependent. So there would exist scalars $\lambda_1, \lambda_2, \ldots \lambda_{n-1}$ not all 0 such that

$$a_{\bar{B}i} d = (\lambda_1 a_{\bar{B}1} + \lambda_2 a_{\bar{B}2} + \cdots + \lambda_{n-1} a_{\bar{B}n-1}) d = 0$$

with the last equality resulting from the fact that $a_{B \setminus j} d = 0$. Since we chose i such that $a_{Bi} d > 0$, we have our contradiction. Thus all of the constraints are linearly independent and we have a new BFS.

In every iteration of our algorithm, we check if the current BFS is optimal, and if not we move to a better BFS. We continue until we reach our optimal

solution, or discover that the LP is unbounded. Since the polyhedron P has finitely many basic feasible solutions, and we always move to a BFS with a higher value of $c^T x$, we never visit the same BFS twice, and so the algorithm will require only finitely many iterations to find the optimal BFS or declare unboundedness.