Optimization and Approximation ENS Lyon

Lecture 4

Lecturer: Alantha Newman *September 30, 2016*

1 Pivot Rules for the Simplex Algorithm

The simplex algorithm is really a *family* of algorithms where the actual algorithm is determined by the pivot rule. Let us look at some common pivot rules. For example, here are some natural choices for choosing the *entering* variable:

1. Choose the variable corresponding to $\bar{c}_e > 0$ with the smallest subscript e.
2. Choose the variable corresponding to $\max_e\{\bar{c}_e > 0\}$.
3. Choose the variable corresponding to $\max_{e,\ell}\{\bar{c}_e \frac{\bar{b}_\ell}{\bar{a}_{\ell e}}\}$, where (x_e, x_ℓ) is a pair of entering and leaving variables.
4. Other pivot rules in Vanderbei [Van01]: steepest edge, dual simplex.

1.1 Degeneracy

If a dictionary has the property that all basic variables are non-zero, then the dictionary and the associated basic feasible solution are called *non-degenerate*. This property ensures that we made progress (i.e. increased the objective value) in the pivot step leading to this dictionary. Moreover, if all dictionaries are non-degenerate, then we make progress at every iteration. In this case, the simplex algorithm is guaranteed to terminate in a finite number of steps.

However, if a polyhedron contains some *degenerate* basic feasible solutions, then we may not always be able to increase the objective value. Specifically, the problem is that when we can add the entering variable to the basis–as the entering variable transitions from non-basic to basic–*it can remain zero*. In this case, the objective value does not increase. In the new dictionary, we will have more than $n-m$ tight inequality constraints (i.e. more than $n-m$ zero variables), and the new dictionary will be degenerate.

Not increasing the objective value at some step of the algorithm is not ideal, but it may not be too problematic if we can reach some dictionary from which the objective function can be increased. The main problem occurs when we enter a cycle of degenerate dictionaries. In this case, the algorithm can get stuck in the cycle and will not terminate (assuming the pivot rule is fixed). Thus, it is necessary to address the issue of cycling. However, with respect to termination, it is the only issue we must address: the simplex algorithm either cycles or terminates.

1.2 Example of Cycling

For a given pivot rule, it can be difficult to find an instance that cycles. Vanderbei presents the following example in Chapter 3 [Van01]. Let us consider the following pivot rules:

- Choose entering variable with largest reduced cost.
- Choose leaving variable with smallest subscript.

$$\begin{aligned} \max \quad & 10x_1 - 57x_2 - 9x_3 - 24x_4 \\ \text{subject to:} \quad & 0.5x_1 - 5.5x_2 - 2.5x_3 + 9x_4 \le 0 \\ & 0.5x_1 - 1.5x_2 - 0.5x_3 + x_4 \le 0 \\ & x_1 \le 1 \\ & x_1, \quad x_2, \quad x_3, \quad x_4 \ge 0. \end{aligned} \tag{Q}$$

After six applications of the pivot rule, we arrive back at dictionary (Q). Moreover, both the objective value and the actual solution remain the same at each iteration.

2 Cycling

Suppose that in the course of running the simplex algorithm, we encounter the following sequence of dictionaries: $\{D_1, D_2, \ldots, D_k, D_1, \ldots\}$. This situation is called *cycling*, because this sequence of dictionaries will be repeated as long as the algorithm keeps running. (Here we are assuming that for any fixed dictionary, the pivot rule chooses the same entering and leaving variables.) Since each dictionary is uniquely associated with a basis, cycling means that we cycle through a sequence of bases.

When the simplex algorithm cycles, the value of z does not increase. (Otherwise, we would be at a new solution not seen in any previous iteration of the algorithm. This means that we have reached a new basis, which contradicts the assumption that we are currently in a cycle.) Moreover, the solution never changes!

Lemma 1. *Let D_i and D_{i+1} be dictionaries whose solutions have the same objective value. Then the two solutions are the same.*

Proof. We will show that the solution for D_i is the same as the solution for D_{i+1}. Consider dictionary D_i with x_e entering, x_ℓ leaving.

$$\begin{aligned} x_j &= b_j \cdots - a_{je}x_e \ldots \\ x_\ell &= b_\ell \cdots - a_{\ell e}x_e \ldots \\ z &= \gamma \cdots + c_e x_e \ldots \end{aligned} \tag{D_i}$$

If this is a valid pivot, then $c_e > 0, a_{\ell e} > 0$. So $b_\ell = 0 \Rightarrow x_\ell = 0$ in D_i. (Otherwise, we could increase the objective value as we move to (D_{i+1}).) Now we obtain dictionary D_{i+1}:

$$\begin{aligned} x_j &= b_j \cdots - (\ldots x_\ell) \\ x_e &= 0 \ldots \\ z &= \gamma \ldots \end{aligned} \tag{D_{i+1}}$$

The basic variable x_e can be expressed in terms of the non-basic variables, i.e. variables that do not belong to basis B_{i+1}:

$$x_e = 0 + \sum_{h \notin B_{i+1}} \bar{a}_{eh} x_h. \tag{1}$$

Consider a variable $x_j \in B_i$, $j \neq \ell$. It has value b_j in the solution associated with dictionary (D_i):

$$x_j = b_j - a_{je}x_e - \sum_{h \notin B_i, h \neq e} a_{jh} x_h.$$

To obtain the value of x_j in (D_{i+1}), we use the right-hand side of (1) to substitute for x_e. After this substitution, we have:

$$x_j = b_j - \sum_{h \notin B_{i+1}} \bar{a}_{jh} x_h.$$

Thus, the value of x_j does not change from (D_i) to (D_{i+1}), and so the solution is the same. □

Corollary 2. *Let $D_1, D_2, \ldots, D_k, D_1, \ldots$ be a sequence of cycling dictionaries. Then the solutions associated with the dictionaries are identical.*

Definition 3. *Let $D_1, D_2, \ldots, D_k, D_1, \ldots$ be a sequence of cycling dictionaries. We say a variable is* busy *if it is basic in some dictionary and non-basic in another.*

In other words, a *busy* variable enters some dictionary in the cycling sequence and leaves another.

Lemma 4. *Let $D_1, D_2, \ldots, D_k, D_1, \ldots$ be a sequence of cycling dictionaries. Each busy variables has value zero in all solutions associated with the dictionaries.*

Proof. Directly after a busy variable enters a basis, it must have value zero. (Otherwise, the objective value would increase.) By Lemma 1 it always maintains the same value, so it must always be zero. □

We now look at some methods to avoid cycling.

2.1 Random Perturbation

Recall that cycling only occurs when the polyhedron contains *degenerate* basic feasible solutions. One way to fix this is to *perturb* the linear program and thus the polyhedron that corresponds to the feasible region. The goal is to change the polyhedron so that each vertex is now the intersection of a *unique* set of n constraints, i.e. non-degenerate.

$$\begin{aligned} & & \max & \quad \sum_{i=1}^{n} c_i x_i \\ \text{s. t.} & \quad \sum_{i=1}^{n} a_{ji} x_i & \leq & \quad b_j, \quad j : 1 \leq j \leq m, \\ & \quad x_i & \geq & \quad 0, \quad i : 1 \leq i \leq n. \end{aligned}$$

After choosing small, random values $\epsilon_1, \epsilon_2, \ldots, \epsilon_m$ and perturbing, we obtain:

$$\begin{aligned} & & \max & \quad \sum_{i=1}^{n} c_i x_i \\ \text{s. t.} & \quad \sum_{i=1}^{n} a_{ji} x_i & \leq & \quad b_j + \epsilon_j, \quad j : 1 \leq j \leq m, \\ & \quad x_i & \geq & \quad 0, \quad i : 1 \leq i \leq n. \end{aligned}$$

It is now unlikely that more than n constraints are tight at some basic feasible solution. However, the new polyhedron may have more extreme points than the original.

2.2 Lexicographic Pivoting Rule

A practical way to implement random perturbation is to choose $\epsilon_1 \gg \epsilon_2 \gg \cdots \gg \epsilon_m$, where $\gg$ means "much greater than". Consider the linear program from Section 1.2:

$$\begin{aligned} \max \quad & 10x_1 - 57x_2 - 9x_3 - 24x_4 & & \\ \text{subject to:} \quad & 0.5x_1 - 5.5x_2 - 2.5x_3 + 9x_4 & \leq & \ 0 \\ & 0.5x_1 - 1.5x_2 - 0.5x_3 + x_4 & \leq & \ 0 \\ & x_1 & \leq & \ 1 \\ & x_1, \quad x_2, \quad x_3, \quad x_4 & \geq & \ 0. \end{aligned} \tag{Q}$$

In the intial dictionary, we have:

$$\begin{aligned} x_5 =& 0 - 0.5x_1 + 5.5x_2 + 2.5x_3 - 9x_4 \\ x_6 =& 0 - 0.5x_1 + 1.5x_2 + 0.5x_3 - x_4 \\ x_7 =& 1 - x_1 \\ z =& 0 + 10x_1 - 57x_2 - 9x_3 - 24x_4 \end{aligned} \tag{D_0}$$

Now let us consider the dictionary for the perturbed problem:

$$\begin{aligned} x_5 =& 0 + \epsilon_1 - 0.5x_1 + 5.5x_2 + 2.5x_3 - 9x_4 \\ x_6 =& 0 + \epsilon_2 - 0.5x_1 + 1.5x_2 + 0.5x_3 - x_4 \\ x_7 =& 1 + \epsilon_3 - x_1 \\ z =& 0 + 10x_1 - 57x_2 - 9x_3 - 24x_4 \end{aligned} \tag{D_0'}$$

Define the vector $\mathbf{b}_\epsilon$ as follows:

$$\mathbf{b}_\epsilon = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{S}$$

At some iteration, with basis B, we have:

$$\mathbf{x} = A_B^{-1}\mathbf{b}, \qquad \mathbf{x}_\epsilon = A_B^{-1}\mathbf{b}_\epsilon.$$

The solution associated with the resulting dictionary is then:

$$\mathbf{x}_{vec} = \mathbf{x} + \mathbf{x}_\epsilon.$$

The rows in (S) are linearly independent. When we say $\epsilon_1 \gg \epsilon_2$, we mean that ϵ_1 is so much greater than ϵ_2 that no coefficient of ϵ_2 encountered during any iteration of the simplex will have absolute value large enough to cancel out ϵ_1. The matrix A_B^{-1} is invertible (since the columns corresponding to basis B are linearly independent). The system of linear combinations $\mathbf{x}_\epsilon$ therefore has the same rank as $\mathbf{b}_\epsilon$, which is m.

If for our choice of leaving variable, all candidates have $b_j = 0$, we will choose the variable whose row in $\mathbf{x}_\epsilon$ is lexicographically greatest. Since the rank of $\mathbf{x}_\epsilon$ is m, there will be at least one non-zero entry in each row of $\mathbf{x}_\epsilon$, and the new solution reached will have objective value strictly greater than the previous one.

2.3 Bland's Pivot Rule

Bland's Rule: Choose the entering and leaving variables from the set of possible candidates according to minimum subscript.

Theorem 5. *Using Bland's Rule, the simplex algorithm does not cycle.*

Proof. Proof by contradiction. Assume that we do cycle on the dictionaries $\{D_1, D_2, \ldots, D_k\}$.

Among all busy variables (recall Definition 3), let x_t denote the variable with the maximum subscript. Let $D \in \{D_1, D_2, \ldots, D_k\}$ be a dictionary for which x_t is the *leaving* variable. Let x_s denote the entering variable in D.

$$\begin{aligned} x_j &= b_j - \sum_{i \notin B} a_{ji} x_i, \quad \forall j \in B, \\ z &= \gamma + \sum_{i \notin B} c_i x_i. \end{aligned} \tag{D}$$

Let D^* be a dictionary where x_t enters. The objective function for the dictionary D^* is:

$$z = \gamma + \sum_{i=1}^{n+m} c_i^* x_i, \quad \text{where } c_i^* = 0 \text{ for } i \in B^*. \tag{D^*}$$

The dictionaries D and D^* represent equivalent systems of equations. Therefore, the value of z is unchanged when we substitute solutions for D in D^*. For instance, consider the following solution in D:

$x_s = y$,

$x_i = 0$ for $i \notin B, i \neq s$,

$x_i = b_i - a_{is}y$ for $i \in B$.

Then we have:

$$\begin{aligned} z &= \gamma + c_s y, && \text{(from } D\text{)}, \\ z &= \gamma + c_s^* y + \sum_{i \in B} c_i^*(b_i - a_{is}y), && \text{(from } D^*\text{)}. \end{aligned}$$

Putting these together, we have:

$$c_s y \;=\; c_s^* y + \sum_{i \in B} c_i^*(b_i - a_{is}y),$$

and equivalently,

$$y(c_s - c_s^* + \sum_{i \in B} c_i^* a_{is}) \;=\; \sum_{i \in B} c_i^* b_i. \tag{$\star$}$$

The equation ($\star$) is true for every value of y. Thus, we have:

$$c_s - c_s^* + \sum_{i \in B} c_i^* a_{is} \;= 0. \tag{$\star\star$}$$

Now we observe the following facts:

(a) Since x_s is entering in D, $c_s > 0$.

(b) Since x_t is entering in D^*, no other busy variable was a candidate. This means that $c_s^* \leq 0$, because:

- either x_s is non-basic in D^*, and then $c_s^* \leq 0$, or
- x_s is basic in D^*, and then $c_s^* = 0$.

(c) Facts (a), (b) and ($\star\star$) imply that $\sum_{i \in B} c_i^* a_{is} < 0$. Then there exist an index $r \in B$ such that $c_r^* a_{rs} < 0$.

(d) Since $r \in B$, then x_r is basic in D. Since $c_r^* \neq 0$, then x_r is non-basic in D^*. Therefore, x_r is *busy* and $r < t$.

(e) Note that $x_r \neq x_t$ because $c_t^* > 0$ (x_t enters D^*), and $a_{ts} > 0$ (x_t leaving variable in D).

(f) Now $r < t$ and both x_r and x_t are non-basic in D^*. Since x_r was not chosen to enter, $c_r^* < 0$ so $a_{rs} > 0$.

(g) Since all busy variables have value zero in every dictionary $\{D_1, D_2, \ldots, D_k\}$, and x_r is busy, then $x_r = 0$.

In particular, consider D:

$$\begin{aligned} x_r &= 0 \cdots - a_{rs} x_s \\ z &= \gamma \cdots + c_s x_s. \end{aligned}$$

Therefore, x_r is a candidate for leaving in D. This is a contradiction to the fact that we used the smallest subscript rule. □

3 Complexity of the Simplex Algorithm

Pivot rules may differ in the complexity they require at each iteration. However, we are mainly concerned with the number of pivots required by the simplex algorithm. There is no known pivot rule that provably requires only a polynomial number of pivots in the worst case. The question of whether or not there exists such an efficient pivot rule has lead to interesting geometric questions about polytopes.

3.1 Hirsch Conjecture

The famous Hirsch Conjecture is related to proving lower bound on the number of pivots required by any pivot rule. Let P be a bounded polyhedron, and consider two vertices x and y in P. Let $d(x, y)$ denote the shortest path from x to y, i.e. the minimum number of vertices visited on any path from x to y, where a path connects adjacent vertices. Define the diameter of P as:

$$D(P) = \max_{x,y \in P} d(x, y)$$

Define:

$$\Delta(n, M) = \max_{P \in S} D(P)$$

where S is the set of all bounded polyhedran in $\mathbb{R}^n$ with M constraints.

Hirsch Conjecture: $\Delta(n, M) \leq M - n$. (Note that for a linear program in canonical form, the number of constraints is $M = m + n$, and therefore the Hirsch Conjecure posits: $\Delta(n, m) \leq m$.)

Note that if this conjecture were true, it does not immediately imply good upper bounds on the number of pivots for the simplex algorithm. However, if it is far from true, then the simplex algorithm cannot be efficient for any pivot rule. The Hirsch Conjecture was disproved in 2010 by Francisco Santos, who showed $\Delta(43, 86) > 43$. In terms of upper bound on the diameter of a polytope, Kleitman and Kalai showed that $D(P) \leq M^{\log_2 n + 2}$. However, no one has found a pivot rule that will provably find a path of at most this length.

3.2 Klee-Minty Cubes

In the 1970's, Klee and Minty found instances of linear programs that exhibit exponential behaviour for many pivot rules. Their examples have the following form:

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} 10^{n-j} x_j \\ \text{subject to:} \quad & 2 \sum_{j=1}^{i-1} 10^{i-j} x_j + x_i \leq 100^{i-1}, \quad \forall i = 1, 2, \ldots, n. \\ & x_j \geq 0, \qquad \forall j = 1, 2, \ldots, n. \end{aligned}$$

For example, when $n = 3$, the objective function is $100x_1 + 10x_2 + \cdots + x_3$, and the constraints are:

$$\begin{aligned} x_1 &\leq 1 \\ 20x_1 + x_2 &\leq 100 \\ 200x_1 + 20x_2 + x_3 &\leq 10000. \end{aligned}$$

In general, each variable x_i is essentially bounded above by 100^{i-1}, since the variables with smaller indices are negligible in comparison. This is the reason the feasible region is often referred to as a "cube". Using the largest coefficient rule, it can be shown that the simplex method will visit every vertex before reaching the optimal vertex. This establishes that the complexity of this pivot rule is 2^n.

References

[Van01] Robert J. Vanderbei. *Linear programming: Foundations and Extensions.* Kluwer Academic Publishers, Boston, MA, 2001.

These lecture notes are partly based on the following sources: lecture notes by Stéphan Thomassé from previous versions of the same course and Chapters 3 and 4 of [Van01].