CSC2411 - Linear Programming and Combinatorial Optimization[*]
Lecture 4: Simplex Algorithm (Contd.)

Notes taken by Chuan Wu

February 6, 2005

Summary: We continue our discussion of the simplex algorithm. We give a high-level description of the algorithm and discuss the different variations of the algorithm determined by the exact pivot rules they use. We end by showing an exponential time lower bound for the simplex algorithm that is due to Klee and Minty (1972).

1 Review

Let x_0 be a BFS for a linear program and $B(1), \cdots, B(m)$ be the corresponding basis. Let j be the entering column when we move from one BFS to another. The new BFS is calculated as:

$$x'_{i0} = \begin{cases} x_{i0} - \theta_0 x_{ij} & i \neq l \\ \theta_0 & i = l \end{cases},$$

where θ_0 is defined as:

$$\theta_0 \overset{def}{=} \min_{\{i|x_{ij}>0\}} \frac{x_{i0}}{x_{ij}} = \frac{x_{l0}}{x_{lj}},$$

and the new basis is:

$$B'(i) = \begin{cases} B(i) & i \neq l \\ \text{J} & i = l \end{cases}.$$

2 How do We Choose the New Basic Variable

When we move from one BFS to another (Figure 1), column j is inserted into the basis. Let $\vec{v}$ be the unit "*Change Vector*" from the original BFS $\vec{x}$ to the new BFS $\vec{x'}$. We have

$$v_i = \begin{cases} -x_{ij} & i = B(i) \\ 1 & i = j \\ 0 & otherwise \end{cases}$$

[*] Lecture Notes for a course given by Avner Magen, Dept. of Computer Sciecne, University of Toronto.

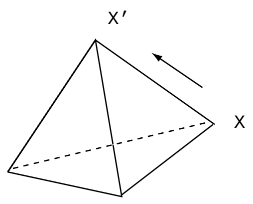

Figure 1: A move step in simplex algorithm

Then the new BFS is:

$$\vec{x'} = \vec{x} + \theta_0 \vec{v}.$$

The new objective function is

$$\begin{aligned} z' &= \langle \vec{c}, \vec{x'} \rangle \\ &= \langle \vec{c}, \vec{x} + \theta_0 \vec{v} \rangle \\ &= \langle \vec{c}, \vec{x} \rangle + \theta_0 \langle \vec{c}, \vec{v} \rangle \\ &= z + \theta_0 (c_j - \sum_{i=1}^{m} x_{ij} c_{B(i)}) \end{aligned}$$

If we have $\theta_0 > 0$ and $c_j - \sum_{i=1}^{m} x_{ij} c_{B(i)} < 0$, we can improve the objective function by entering j to the basis. So when we are considering whether to let column j enter the basis, we calculate

$$\bar{c_j} = c_j - \sum_{i=1}^{m} x_{ij} c_{B(i)}.$$

If $\bar{c_j} < 0$ and $\theta_0 > 0$, we know this move is profitable in that we get a decrease in the cost of objective function. The next claim assures that when there is no j for which $\bar{c_j} < 0$, we are in fact done.

Claim 2.1. *Let X_0 be a BFS x_0, with*

$$\bar{c_j} \geq 0 \qquad \forall j \notin B(1), B(2), \ldots, B(m)$$

. Then x_0 is optimal.

Proof. Let B be the set of indices of the columns in the basis of x_0. Let N be the set of indices of the rest of the columns. In matrix form, we have

$$Ax = A_B x_B + A_N x_N = b,$$

and so

$$x_B = A_B^{-1}(b - A_N x_N).$$

Our goal now is to express the objective function as a function of x_N only.

$$\begin{aligned} z &= \langle c, x\rangle \\ &= c_B x_B + c_N x_N \\ &= c_B A_B^{-1}(b - A_N x_N) + c_N x_N \\ &= c_B A_B^{-1} b + (c_N - c_B A_B^{-1} A_N) x_N \end{aligned}$$

For the current BFS x_0, we have $x_N = \begin{bmatrix} 0 \\ \cdots \\ 0 \end{bmatrix}$. So z_0, the current value of the objective function is $c_B A_B^{-1} b$.

Further we notice that

$$A_B^{-1} A_N = (x_{ij}),$$

and so

$$(c_N - c_B A_B^{-1} A_N) x_N = \sum_{j \notin B} \bar{c_j} x_j.$$

For every feasible solution x, we have $x_j \geq 0$ and as $\vec{c_j} \geq 0$ for $j \notin B$, we get that for every feasible solution

$$\langle c, x\rangle \geq c_B A_B^{-1} b = z_0.$$

In other words, z_0 is the minimum and x_0 is an optimal solution. □

A this point, we recall the case in which $x_{ij} \leq 0$ for all i shows that P is unbounded. We can now say that the objective function is unbounded below if $x_{ij} \leq 0$ for all i and $\bar{c_j} < 0$. There is also another case in which $\bar{c_j} \geq 0$ and $\theta_0 = \infty$. Then feasible solution set P of the linear program is unbounded but the objective function might still be bounded. However, the simplex algorithm will never encounter this second case since when $\bar{c_j} \geq 0$, j will never enter the basis.

3 Putting it all Together

3.1 Pseudocode for Simplex Algorithm

Algorithm 3.1 (Simplex Algorithm).

```
1. Start with a BFS $x_0$ with basis $B(1), \ldots, B(m)$ by found by the Start Phase
   algorithm (sometimes called 'Phase I'). (Refer to section 3.1 in lecture note 2.)

2. Repeat:
   For column $j$ which is not in current basis, calculate $x_{ij}, i = 1, \ldots, m$, and $\bar{c_j}$.

   (a) If $\bar{c_j} \geq 0$ , $\forall j \notin B$, then stop and report current BFS is optimal.
   (b) Choose **any** j for which $\bar{c_j} < 0$:
```

   i. If $x_{ij} \leq 0$ for any i, stop and report the objective function is unbounded.
   ii. Calculate $\theta_0 = \min_{\{i|x_{ij}>0\}} \frac{x_{i0}}{x_{ij}}$. Let l be **any** index with $\theta_0 = \frac{x_{l0}}{x_{lj}}$. Pivot on j, l and move to the new BFS on the basis B' which replaces $B(l)$ with j.

3.2 Questions about Simplex Algorithm

1. Is simplex algorithm a polynomial-time algorithm? In fact, does it ever terminate?
2. What is the rule we should apply to make the decision of the two "**any**"s in simplex algorithm?
3. Can we make the calculation of x_{ij} and $\bar{c_j}$ simple?

Discussion of the above questions:

1. There are a finite number of different vertices that the algorithm can visit. So if it doesn't visit the same vertex twice, it must terminate. If it does, then it will repeat the sequence of vertices between the first and second visit to the repeated vertex over and over again. In other words, the algorithm terminates iff it does not *cycle*. We further notice that whenever $\theta_0 > 0$, we strictly decrease the objective function; therefore $\theta_0 = 0$ throughout the cycle.

   $\theta_0 = 0$ can only happen when the current BFS is degenerated and in other words it has more than $n - m$ zeros. Although we move from one basis to another, the BFS we get is still the same. In the geometry of the linear program, the corresponding vertex of the polytope lies on more than $n-m$ facets. For example in Figure 2, let A,B,C,D be the four facets corresponding to four constraints of a linear program and each three of them decide a basis. If the move of basis in simplex algorithm is $ABC \rightarrow ABD \rightarrow BCD \rightarrow ABC$, we are moving in a cycle in which the vertex (so the BFS) is still the same.

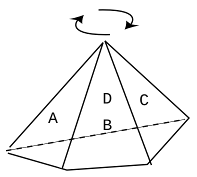

Figure 2: A vertex on the intersection of 4 facets in 3 dimension.
Cycling may occur on such a vertex for some pivot rules.

2. The pivot rules for simplex algorithm are discussed in the following section.

3. In each step, we can use the ”old” x_{ij} and $\bar{c_j}$ calculated in the former step to get the new x'_{ij} and $\bar{c_j}'$. For column k in the former iteration, we have

$$A_k = \sum_{i=1}^{m} x_{ik} A_{B(i)}.$$

Let j be the column entering the basis in this iteration and $B(l)$ be the leaving column. From last lecture, we know

$$A_{B(l)} = \frac{A_j - \sum_{i=1,i\neq l}^{m} x_{ij} A_{B(i)}}{x_{lj}}.$$

So in the next iteration, we have

$$\begin{aligned} A_k &= \sum_{i=1}^{m} x_{ik} A_{B(i)} \\ &= \sum_{i\neq l} x_{ik} A_{B(i)} + x_{lk} A_{B(l)} \\ &= \sum_{i\neq l} x_{ik} A_{B(i)} + x_{lk} \frac{A_j - \sum_{i\neq l} x_{ij} A_{B(i)}}{x_{lj}} \\ &= \sum_{i\neq l} (x_{ik} - \frac{x_{lk} x_{ij}}{x_{lj}}) A_{B(i)} + \frac{x_{lk}}{x_{lj}} A_j \end{aligned}$$

Thus we have in the new iteration

$$x'_{ik} = \begin{cases} x_{ik} - \frac{x_{lk} x_{ij}}{x_{lj}} & i \neq l \\ \frac{x_{lk}}{x_{lj}} & i = l \end{cases} .$$

Similarly, we can get $\vec{c_k}'$ from $\vec{c_k}$. We mention that often the coefficients X_{ij} and c_i are arranged in a structure called “Tablaux”.

3.3 Pivoting rules

1. (Largest coefficient rule) Choose the column j to enter the basis which has the smallest $\bar{c_j}$:

$$j = \arg \min_{j:\bar{c_j}<0} \bar{c_j}$$

This pivoting rule may lead to cycling.

2. (Greatest increase rule) Choose the column j to enter the basis which generates the biggest gain toward optimality of the objective function.

$$j = \arg \max_{j:\bar{c_j}<0} (z' - z)$$

This pivoting rule may lead to cycling.

3. (Bland's rule)Choose the lowest numbered column to enter the basis:

$$j = \min\{j : \bar{c_j} < 0\}$$

and the lowest numbered column to leave the basis in case of tie

$$B(l) = \min\{B(l) : \frac{x_{l0}}{x_{lj}} = \min_{\{i|x_{ij}>0\}} \frac{x_{i0}}{x_{ij}}\}$$

This rule guarantees no cycling.

3.4 Simplex is not a Polynomial-Time Algorithm

As we have introduced simplex algorithm as the most commonly used algorithm to solve linear programs and discussed the uncertainty in the algorithm, it is natural to come to the question of telling the "goodness" of simplex algorithm based on mathematical criterion and ask if simplex is a polynomial-time algorithm. We hope to show simplex algorithm can come to the optimal solution in polynomial time by one of the pivoting rules.

First of all, we may ask whether there are a polynomial number of vertices in the polytope P of the feasible solutions. Unfortunately, the answer is "NO". There are polytopes that have exponentially many vertices. An simple example of such a polytope is the n-dimension $\{0, 1\}$ cube defined by:

$$0 \leq x_j \leq 1 \quad j = 1, 2, \ldots, n$$

The n-cube has $2n$ facets, one for each inequality, and 2^n vertices, one for setting each subset of $\{x_1, x_2, \ldots, x_n\}$ to 1 and the rest to 0. The example of a 3-cube is shown in Figure 3.

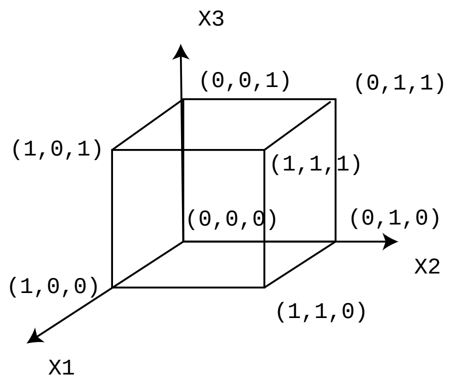

Figure 3: A 3-dimension {0,1} cube

Having an exponential number of vertices is not necessarily a negative evidence to the running time of simplex algorithm, since in general it does not traverse all vertcies.

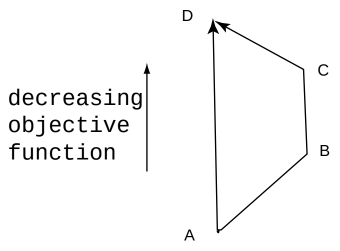

Figure 4: Path for simplex algorithm

For example in Figure 4, it may take the path $A \to D$ instead of $A \to B \to C \to D$ based on different pivot rules. To claim that the algorithm is indeed "in trouble" one must show an example where the algorithm follows a long sequence of vertices, so that the objective function decreases along this sequence. Such a sequence will essentially show that if the algorithm take the most unfortunate pivot rule in each vertex it will require many iterations. For this purpose, we have to come up with a polyhedron and an objective function so that there is a long decreasing path. A first possible attempt might be to take the n-cobe as the polyhedron. This, however, can easily be seen to be fruitless effort : for any choice of objective function the longest decreasing path is of size n.

We still wish to utilize the nice combinatorial structure of the n-cube. To this end we *perturb* the cube in a special way that gives rise to such a long path. Here is a precise way of how to do it that was suggested by Klee and Minty (1972). Let ϵ be a constant in $(0, 1/2)$, and consider the following set of inequalities.

$$\begin{array}{ccccc} 0 & \leq & x_1 & \leq & 1 \\ \epsilon x_1 & \leq & x_2 & \leq & 1 - \epsilon x_1 \\ & & \vdots & & \\ \epsilon x_{i-1} & \leq & x_i & \leq & 1 - \epsilon x_{i-1} \\ & & \vdots & & \\ \epsilon x_{n-1} & \leq & x_n & \leq & 1 - \epsilon x_{n-1} \end{array}$$

The example of a perturbed 3-cube is shown in Figure 5.

We define a linear program on these constraints as:

$$min - x_n \tag{1}$$

$$\begin{array}{lcccl} s.t. & x_1 & \geq & 0 & \\ & x_1 & \leq & 1 & \\ & x_i & \geq & \epsilon x_{i-1} & i = 2, 3, \cdots, n \\ & x_i & \leq & 1 - \epsilon x_{i-1} & i = 2, 3, \cdots, n \end{array}$$

When we solve this linear program with simplex algorithm, we have to make decision on the entering and leaving columns at each step. We will show that we can find a most unfortunate sequence of choices at each selection step, so that the simplex algorithm uses an exponential number of pivots before coming to the optimal

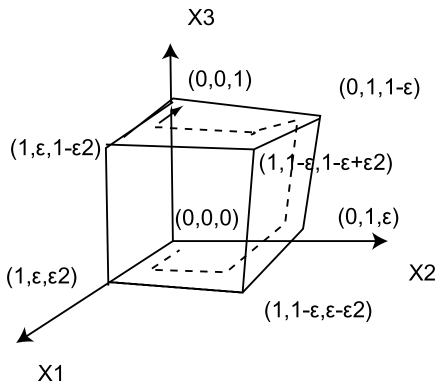

Figure 5: A perturbed 3-dimension cube

solution. We do this by exhibiting an exponential sequence of basic feasible solutions $x_1, x_2, \ldots, x_{2^n}$ to the linear program defined in (1), such that $cx_{i+1} < cx_i$ for $i = 1, \ldots, 2^n$.

First we represent each vertex of the n-dimension $\{0,1\}$ cube with an encoding in $\{0,1\}^n$. For example, in the 3-dimension cube shown in Figure 3, vertices are encoded as 000, 100, 110, 010, 011, 111, 101, 001. We encode each vertex in the perturbed n-cube in $\{0,1\}^n$ too by mapping ϵ to 0. For example, in the perturbed 3-dimension cube shown in Figure 5, the vertices and their corresponding encoding are:

$Vertex$	$Encoding$
$(0,0,0)$	000
$(1,\epsilon,\epsilon^2)$	100
$(1,1-\epsilon,\epsilon-\epsilon^2)$	110
$(0,1,\epsilon)$	010
$(0,1,1-\epsilon)$	011
$(1,1-\epsilon,1-\epsilon+\epsilon^2)$	111
$(1,\epsilon,1-\epsilon^2)$	101
$(0,0,1)$	001

So when ϵ is mapped to 0, we have that one vertex in the n-cube maps to one vertex in the perturbed cube and the edge between two vertices in the n-cube maps to the edge between the two corresponding vertices in the perturbed cube.

Now we want to find a Hamiltonian path in $\{0,1\}^n$ and show that corresponding vertices in the perturbed cube have increasing x_n values. A gray code is a Hamiltonian path in the cube and binary reflected gray code is a special type of gray code that is defined recursively as in Figure 6.

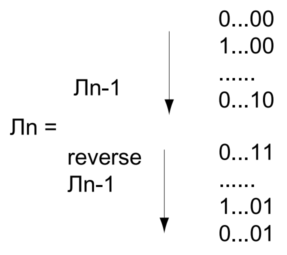

Figure 6: Hamiltonian path Π_n

For example, the Hamiltonian path defined on the perturbed 3-dimension cube is:

$$\Pi_3 = 000 \to 100 \to 110 \to 010 \to 011 \to 111 \to 101 \to 001.$$

Claim 3.2. *The objective function of the linear program defined in (1) is decreasing if we move from BFS to BFS by following the Hamiltonian path defined on binary reflected gray code.*

We first look at how we can calculate the BFS of the linear program defined in (1). This linear program has $2n$ constraints. Recalling from Question 2 of Assignment 1, BFS is associated with n linear independent inequalities satisfied as equalities. From each pair of inequalities $x_i \geq \epsilon x_{i-1}$ and $x_i \leq 1 - \epsilon x_{i-1}$, we can only choose one to satisfy as equality. So we have 2^n choices in total. It's easy to show that in each of the 2^n selections, the selected n inequalities are linear independent, for $det(A_N) \neq 0$, A_N is the $n \times n$ matrix composed of coefficients from the n selected inequalities. Thus we can get 2^n basic feasible solutions, which correspond to the 2^n vertices in the perturbed n-cube.

Then we go on to prove the claim.

Proof. We prove by induction.

Basis

If $n = 2$, the Hamiltonian path is $\Pi_2 = 00 \to 10 \to 11 \to 01$. The linear program is

$$\begin{array}{lrcl} & min - x_2 & & \\ s.t. & x_1 & \geq & 0 \\ & x_1 & \leq & 1 \\ & x_2 & \geq & \epsilon x_1 \\ & x_2 & \leq & 1 - \epsilon x_1 \end{array} \tag{2}$$

It is trivial to show the objective function is decreasing on the Hamiltonian path Π_2.

If $n = 3$, the linear program is

$$\begin{array}{rrcl} & & min - x_3 & \\ s.t. & x_1 & \geq & 0 \\ & x_1 & \leq & 1 \\ & x_2 & \geq & \epsilon x_1 \\ & x_2 & \leq & 1 - \epsilon x_1 \\ & x_3 & \geq & \epsilon x_2 \\ & x_3 & \leq & 1 - \epsilon x_2 \end{array} \qquad (3)$$

We first consider the first part of Π_3: $000 \to 100 \to 110 \to 010$. On this path, the last coordinate is always 0, which means we always take the equality

$$x_3 = \epsilon x_2. \qquad (4)$$

If we only look at the first two bits, this path is the same as Π_2. Further we notice x_3 doesn't appear in the first 4 inequalities in linear program (3) and these 4 inequalities are the same as those in linear program (2). As we have shown the objective function $min - x_2$ is decreasing by following Π_2, which means x_2 increases, we know x_3 increases too based on Equation (4). So the objective function $min - x_3$ in linear program (3) decreases. Then we move from 010 to 011. x_3 changes from ϵx_2 to $1 - \epsilon x_2$. Since $\epsilon < \frac{1}{2}$, we have $\epsilon x_2 < 1 - \epsilon x_2$. So this is an improving step. Next we look at the last part of Π_3: $011 \to 111 \to 101 \to 001$. On this path, the last coordinate is always 1, which means now we always take the equality

$$x_3 = 1 - \epsilon x_2. \qquad (5)$$

If we only look at the first two bits on the path $011 \to 111 \to 101 \to 001$, it is the reverse of Π_2. So by following this path, x_2 decreases, then x_3 increases based on Equation (5). Thus the objective function of linear program (3) continues to decrease.

Induction step

We assume the claim holds for $n-1$-cube and try to prove it for n-cube. The proof is similar as what we did to show the claim holds for the case of $n = 3$ based on the result of $n = 2$.

We start by moving from vertex $0 \cdots 0$ to $0 \cdots 10$ by following the first part of Π_n. By induction, this is an decreasing path for $n-1$-cube. Since the last coordinate of the vertices on this part of path is always 0, we know we should always choose

$$x_n = \epsilon x_{n-1}.$$

Thus x_n is increasing and the objective function of linear program (1) is decreasing.

Next we move from $0 \cdots 10$ to $0 \cdots 11$. x_n changes from ϵx_{n-1} to $1 - \epsilon x_{n-1}$. Since $\epsilon < \frac{1}{2}$, we have $\epsilon x_{n-1} < 1 - \epsilon x_{n-1}$. So this is a decreasing step.

Then we follow the last part of Π_n to move from $0 \cdots 11$ to $0 \cdots 01$. The first $n-1$ bits on this part of path is the reverse of Π_{n-1}. So x_{n-1} is decreasing on the path. The last coordinate is always 1 here and so

$$x_n = 1 - \epsilon x_{n-1}$$

for the corresponding vertices. Thus x_n continues to increase, which leads to continuous decreasing of the objective function. Then we can conclude our proof. $\square$

From the Klee Minty cube example, we have shown that by starting from some vertex and choosing certain pivot rules in simplex algorithm, it takes exponential time to achieve optimality. We state (without proof) that the *largest coefficient* pivot rule will lead to the above long path. Noticing that there's no guarantee which vertex to start with when using simplex algorithm, we are able to assume that we start from this very vertex which leads to an exponential length path to the optimal solution.