**Polyhedral Computation** 2010/10/8

Lecture 2

*Professor: David Avis* *Scribe:Shohei Nishida*

## 1 Vertex Enumeration Problem

Given an $m \times n$ matrix $A = (a_{ij})$ and an $m$ dimensional vector $b$, a *convex polyhedron*, or simply *polyhedron*, $P$ is defined as:

$$P = \{x \in R^n : b + Ax \geq 0\}.$$

It is possible that $P = \emptyset$ or $P$ could be unbounded e.g. $P = \{x \in R^1 : x \geq 0\}$. A *polytope* is a bounded polyhedron. In this class we will often deal with polytopes to make things a bit simpler, but the extension to unbounded polyhedra is usually not so difficult. We will also normally assume that $P$ is *full dimensional*, i.e., there is an interior point $x$ that satisfies all inequalities of $P$ strictly.

A point $x \in P$ is a *vertex* of $P$ iff it is the unique solution to a subset of $n$ inequalities solved as equations. The *vertex enumeration problem* is to output all vertices of a polytope $P$.

$$A = \begin{bmatrix} -1 & 0 & 1 \\ 0 & -1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & -1 \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \qquad \begin{array}{l} 1 - x_1 + x_3 \geq 0 \\ 1 - x_2 + x_3 \geq 0 \\ 1 + x_1 + x_3 \geq 0 \\ 1 + x_2 + x_3 \geq 0 \\ -x_3 \geq 0 \end{array}$$

Figure 1: $A, b$ and its polyhedron $P$

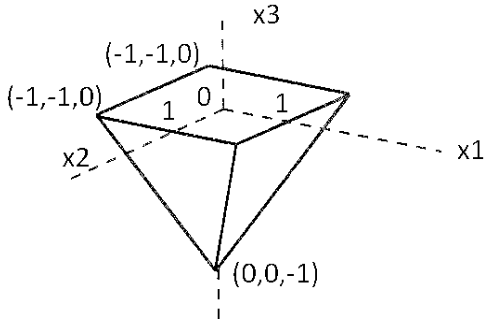

Figure 2: $P$ has 5 vertices

## 2 Brute force algorithm for vertex enumeration

From the definition of a vertex, we can get a brute force vertex enumeration algorithm.

1. Pick a subset of $n$ inequalities not yet examined. Let $B$ be the corresponding $n \times n$ sub-matrix of $A$, and let $\bar{b}$ be the corresponding sub-vector of $b$.
2. Try to solve $\bar{b} + Bx = 0$
   case a:no solution, go back to 1
   case b:no unique solution ($B$ linearly dependent), go back to 1
   case c:unique solution $\bar{x}$. If $\bar{x} \in P$, output as a vertex and go back to 1.

In case 2c, checking $\bar{x} \in P$ is very important because $\bar{x}$ can be outside of $P$. See Fig. 3 for an example. The subset of two inequalities, $x_2 \geq 0$ and $2 - x_1 - x_2 \geq 0$, have a unique solution $(0, 2)$ when solved as equations, but it lies outside of $P$.

Ex. Show the brute force algorithm can output the same vertex more than once.

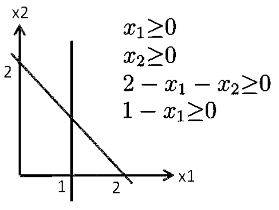

Figure 3: Case 2c: $x$=(0,2) is outside of $P$

## 3 Graph search vertex enumeration methods

### 3.1 Basic idea

The basic idea for an improved algorithm is to define a graph on the vertices and edges of the polytope. We then search the graph using some method such as DFS, BFS etc.
In order to this we need to define the edges of a polyhedron. Let $c$ be an $n$-vector and $d$ be a scalar. Then $c^T x = d$ is called a *hyperplane.* It is a *supporting hyperplane* if $P$ is completely contained in either $c^T x \leq d$ or in $c^T x \geq d$. A *face* $F$ is the intersection

$$F = P \cap \{x \in R^n : c^T x = d\}.$$

A second definition of vertex, equivalent to the first, is a 0-dimensional face. An *edge* is a 1-dimensional face, and a *facet* is an $(n-1)$-dimensional face. A second, more geometric definition of edge is as follows. Given vertices $v_1, v_2$ of $P$, the segment$[v_1, v_2]$ is an edge if and only if the unique convex combination of vertices of $P$ that yield its midpoint is $(v_1 + v_2)/2$.

### 3.2 DFS algorithm for vertex enumeration

1. Find any starting vertex $v$ of $P$.
2. Generate edges from $v$ sequentially
3. Move to any unexplored vertex adjacent to $v$. If none back track.

In our case, just getting a starting vertex $v$ for DFS is non-trivial. In fact it is polynomial-time equivalent to linear programming. Getting an adjacency oracle to define the edges is also tricky. We will base our computations on *dictionaries*, as is done for the simplex method of linear programming. To get a dictionary for $P = \{x \in R^n : b + Ax \geq 0\}$ we add one new nonnegative variable for each inequality:

$$x_{n+i} = b_i + \sum_{j=1}^{n} a_{ij} x_j, \quad x_{n+i} \geq 0 \quad i = 1, 2, ..., m.$$

These new variables are called *slack variables* and the original variables are called *decision variables.*

In order to have any vertex at all we must have $m \geq n$, and normally $m$ is significantly larger than $n$, allowing us to solve the equations for various sets of variables on the left hand side. The variables on the left hand side of a dictionary are called *basic*, and those on the right hand side are called *non-basic* or, equivalently, *co-basic.* We use the notation $B = \{i : x_i \text{ is basic}\}$ and $N = \{j : x_j \text{ is co-basic}\}$.

A *pivot* interchanges one index from $B$ and $N$ and solves the equations for the new basic variables, see Figure 4. A *basic solution* from a dictionary is obtained by setting $x_j = 0$ for all $j \in N$. In Figure 4 the basic solution is $x = (-1, -1, 0, 2, 2, 0, 0, 0)$ which is feasible since the slack variables, ie., the last 5 coordinates of the vector $x$, are all non-negative. The first three coordinates give the vertex, and these may be negative. It is a *basic feasible solution(BFS)* if $x_j \geq 0$ for every slack variable $x_j$. The

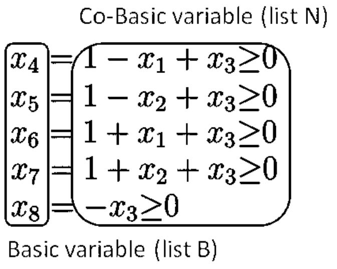
Co-Basic variable (list N)
$x_4 = 1 - x_1 + x_3 \geq 0$
$x_5 = 1 - x_2 + x_3 \geq 0$
$x_6 = 1 + x_1 + x_3 \geq 0$
$x_7 = 1 + x_2 + x_3 \geq 0$
$x_8 = -x_3 \geq 0$
Basic variable (list B)

Figure 4: Initial dictionary with $B = \{4, 5, 6, 7, 8\}$ and $N = \{1, 2, 3\}$

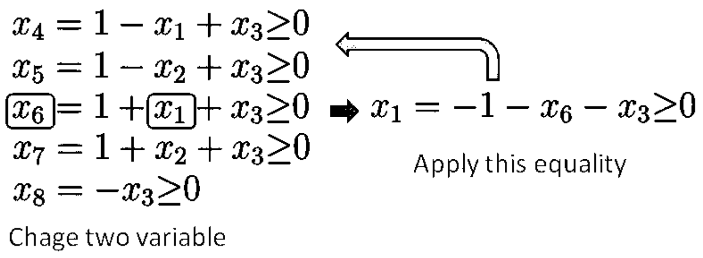
$x_4 = 1 - x_1 + x_3 \geq 0$
$x_5 = 1 - x_2 + x_3 \geq 0$
$\boxed{x_6} = 1 + \boxed{x_1} + x_3 \geq 0$
$x_1 = -1 - x_6 - x_3 \geq 0$
$x_7 = 1 + x_2 + x_3 \geq 0$
$x_8 = -x_3 \geq 0$
Apply this equality
Chage two variable

Figure 5: Pivoting from $N = \{1, 2, 3\}$ to $N = \{2, 3, 6\}$

key point is that a basic feasible solution is a vertex, and vice versa, for a dictionary where all the decision variables are basic. To see this, consider the definition of vertex in the first section. Let the $n$ co-basic variables define the inequalities in the submatrix $B$. Setting their values to zero in a basic solution means solving these inequalities as equations.

So to get a starting vertex for DFS, we need to find such a BFS. We start by pivoting decision variables to the LHS by choosing slack variables to leave the basis. There will usually be many ways to do this. If we are lucky, the resulting dictionary will have a BFS, ie. all slack variables in the basis have a non-negative $b$ value in the dictionary. If not, we need some further steps that we will discuss in the next lecture.

Next, we discuss how to generate edges in the DFS graph from this given BFS. We define an adjacency oracle for dictionaries via pivoting. Consider pivots involving only slack variables. Choose a slack row of the dictionary and some right hand side variable in this row and pivot. If the new dictionary is feasible, then this is an adjacent dictionary, otherwise the pivot is discarded. Formally we may define the adjacency oracle as follows. Let $B$ and $N$ be index sets for the current dictionary. For $i \in B$ and $j \in N$

$$Adj(N, i, j) = \begin{cases} N - j + i & \text{if this gives a feasible dictionary} \\ \emptyset & \text{otherwise} \end{cases}$$

(The notation $N - j + i$ is a convenient shorthand for $N \setminus \{j\} \cup \{i\}$.)

A dictionary is called *degenerate* if it has a slack basic variable $x_j = 0$. Pivoting on a row which contains a degenerate basic variable will give another feasible dictionary which has the same feasible solution, so we remain at the same vertex. This is called a *degenerate* pivot. Making a non-degenerate pivot leads to a different vertex. For the polytope in Figure 1 the graph of all feasible dictionaries is shown in Figure 7. Note that the middle four cobases all correspond to the vertex $x = (0, 0, -1)$. They are interconnected by degenerate pivots. Geometrically we can see what is happening by referring to Figure 2. There are four facets that contain $x = (0, 0, -1)$. Any three of these form a set of co-basic variables whose dictionary corresponds to this vertex. (Note a slack variable is zero if and only if the

Pivot (change to the LHS) decision variables

$$\boxed{x_1} = -1 + x_6 + x_8 \geq 0$$
$$\boxed{x_2} = -1 + x_7 + x_8 \geq 0$$
$$\boxed{x_3} = -x_8 \geq 0$$
$$x_4 = 2 - x_6 + x_8 \geq 0$$
$$x_5 = -2 - x_7 - 2x_8 \geq 0$$

Figure 6: Decision variables are all basic, $N = \{6, 7, 8\}$

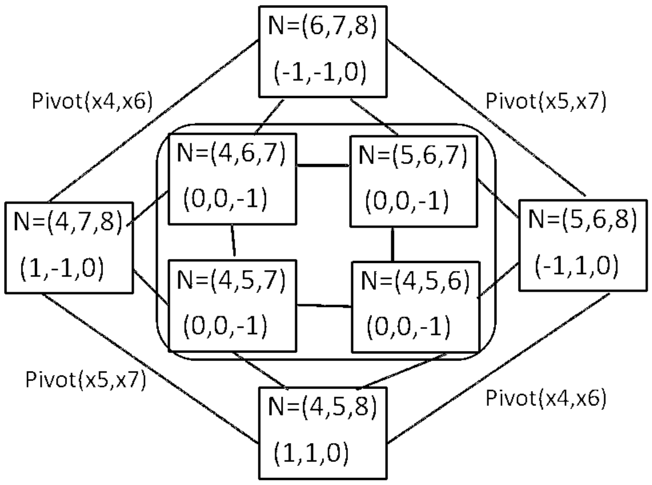

Figure 7: The graph of feasible dictionaries for Figure 1

corresponding basic solution satisfies the slack's inequality as an equation.) The other four vertices are *simple*, that is they lie on exactly $n$=3 facets. The graph search method is extremely good for polytopes which do not have highly degenerate vertices, and in particular for simple polytopes - those for which all vertices are simple.