Algorithms in the Real World

Linear and Integer Programming I
- Introduction
- Geometric Interpretation
- Simplex Method
- Dual Formulation
Linear and Integer Programming

Linear or Integer programming (one formulation)

find $x$ to minimize $z = c^T x$  \textbf{cost or objective function}

subject to $Ax \leq b$  \textbf{inequalities}

$x \geq 0$

$c \in \mathbb{R}^n$,  $b \in \mathbb{R}^m$,  $A \in \mathbb{R}^{n \times m}$

**Linear programming:**

$x \in \mathbb{R}^n$  (polynomial time)

**Integer programming:**

$x \in \mathbb{Z}^n$  (NP-complete)

Extremely general framework, especially IP
Related Optimization Problems

Unconstrained optimization

\[ \min \{ f(x) : x \in \mathbb{R}^n \} \]

Constrained optimization

\[ \min \{ f(x) : c_i(x) \leq 0, \ i \in I, \ c_j(x) = 0, \ j \in E \} \]

Quadratic programming

\[ \min \{ (x^TQx)/2 + c^T x : a_i^T x \leq b_i, \ i \in I, \ a_i^T x = b_j, \ j \in E \} \]

Zero-One programming

\[ \min \{ c^T x : Ax = b, \ x \in \{0,1\}^n, \ c \in \mathbb{R}^n, \ b \in \mathbb{R}^m \} \]

Mixed Integer Programming

\[ \min \{ c^T x : Ax \leq b, \ x \geq 0, \ x_i \in \mathbb{Z}^n, \ i \in \text{Int\_Variables}, \ x_r \in \mathbb{R}^n, \ r \in \text{Real\_Variables} \} \]
How important is optimization?

• 50+ packages available
• 1300+ papers just on interior-point methods
• 100+ books in the library
• 10+ courses at most universities
• 100s of companies
• All major airlines, delivery companies, trucking companies, manufacturers, etc... make serious use of optimization.
Linear+Integer Programming

Outline

Linear Programming
- General formulation and geometric interpretation
- Simplex method
- Ellipsoid method
- Interior point methods

Integer Programming
- Various reductions from NP hard problems
- Linear programming approximations
- Branch-and-bound + cutting-plane techniques
- Case study from Delta Airlines
Applications of Linear Programming

1. A substep in most integer and mixed-integer linear programming (MIP) methods
2. Selecting a mix: oil mixtures, portfolio selection
3. Distribution: how much of a commodity should be distributed to different locations.
4. Allocation: how much of a resource should be allocated to different tasks
5. Network Flows
Create two variables per edge: $x_1, x_1'$

Create one equality per vertex:

$x_1 + x_2 + x_3' = x_1' + x_2' + x_3$

and two inequalities per edge:

$x_1 \leq 3, \ x_1' \leq 3$

add edge $x_0$ from out to in

**maximize** $x_0$
In Practice

In the “real world” most problems involve at least some integral constraints.

• Many resources are integral
• Can be used to model yes/no decisions (0-1 variables)

Therefore “1. A substep in integer or MIP programming” is the most common use in practice
Algorithms for Linear Programming

- **Simplex** (Dantzig 1947)
- **Ellipsoid** (Kachian 1979)
  first algorithm known to be polynomial time
- **Interior Point**
  first practical polynomial-time algorithms
  - **Projective method** (Karmakar 1984)
  - **Affine Method** (Dikin 1967)
  - **Log-Barrier Methods** (Frisch 1977, Fiacco 1968, Gill et al. 1986)

Many of the interior point methods can be applied to nonlinear programs. (But not necessarily in poly time.)
State of the art

1 million variables
10 million nonzeros
No clear winner between Simplex and Interior Point
  – Depends on the problem
  – Interior point methods are subsuming more and more cases
  – All major packages supply both

**The truth**: the sparse matrix routines, make or break both methods.

The best packages are highly sophisticated.
Formulations

There are many ways to formulate linear programs:

- **objective (or cost) function**
  minimize $c^T x$, or
  maximize $c^T x$, or
  find any feasible solution

- **(in)equality**
  $Ax \leq b$, or
  $Ax \geq b$, or
  $Ax = b$, or any combination

- **nonnegative variables**
  $x \geq 0$, or not

Fortunately it is pretty easy to convert among forms
Formulations

The two **most common** formulations:

<table>
<thead>
<tr>
<th>Canonical form</th>
<th>Standard form</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>minimize</strong> $c^T x$</td>
<td><strong>minimize</strong> $c^T x$</td>
</tr>
<tr>
<td>subject to $A x \geq b$</td>
<td>subject to $A x = b$</td>
</tr>
<tr>
<td>$x \geq 0$</td>
<td>$x \geq 0$</td>
</tr>
</tbody>
</table>

**slack variables**

e.g.

$7x_1 + 5x_2 \geq 7$
$x_1, x_2 \geq 0$

$y_1$

$7x_1 + 5x_2 - y_1 = 7$
$x_1, x_2, y_1 \geq 0$

More on slack variables later.
Geometric View of Canonical Form

A polytope in n-dimensional space
Each inequality corresponds to a half-space.
The “feasible set” is the intersection of the half-spaces
This corresponds to a polytope
Polytopes are convex: if x, y is in the polytope, so is the line segment joining them.
The optimal solution is at a vertex (i.e., a corner).
Geometric View of Canonical Form

minimize:
\[ z = -2x_1 - 3x_2 \]
subject to:
\[ x_1 - 2x_2 \leq 4 \]
\[ 2x_2 \leq 18 \]
\[ 2x_1 + x_2 \leq 18 \]
\[ x_2 \leq 10 \]
\[ x_1, x_2 \geq 0 \]

An intersection of 5 halfspaces
Convex Function

f is convex if for all vectors $x, y \in S$ and $\beta \in [0,1]$

$$f(\beta x + (1- \beta)y) \leq \beta f(x) + (1- \beta)f(y)$$

(Note that linear cost functions are both convex and concave.)
"Fundamental Theorem of Linear Programming"

The optimum is at a vertex.

If the polytope $P$ is closed, every point $q$ in $P$ is a convex combination of vertices of $P$, i.e., there exist $\beta_i$ such that $q = \sum \beta_i v_i$

If $f$ is convex, then

$$f(q) = f(\sum \beta_i v_i) \leq \sum \beta_i f(v_i) \leq \max_i f(v_i)$$

(for max problem, optimum is at a vertex)

If $f$ is linear, then

$$f(q) = f(\sum \beta_i v_i) = \sum \beta_i f(v_i)$$

so

$$\min_i f(v_i) \leq f(q) \leq \max_i f(v_i)$$
A **polytope** in n-dimensional space

Each inequality corresponds to a half-space.

The “feasible set” is the intersection of the half-spaces.

This corresponds to a polytope

The optimal solution is at a corner.

**Simplex** moves around on the surface of the polytope

**Interior-Point** methods move within the polytope
Notes about higher dimensions

In canonical form $Ax \leq b$ or standard form $Ax = b$, (m equations and n variables)

Each corner (extreme point) consists of:

- At least n intersecting (n-1)-dimensional hyperplanes
  (hyperplane: $H = \{x: a_1x_1 + \ldots + a_nx_n = b\}$)
  i.e., n linearly independent constraints are tight at the corner
  (“degenerate” if more than n hyperplanes intersect)
- e.g., in 2 dimensions, hyperplanes are lines, and a corner is
  the intersection of 2 or more lines
- in 3 dimensions, hyperplanes are planes, and a corner is the
  intersection of 3 or more planes
- At least n edges intersect at a corner
- Each edge corresponds to moving off of one hyperplane (still
  constrained by at least n-1 of them)

Simplex moves from corner to corner along the edges
Polytope P

Consider Polytope P from canonical form as a graph $G = (V,E)$ with $V = \text{polytope vertices}, E = \text{polytope edges}$.

1) Find any vertex $v$ of $P$.
2) While there exists a neighbor $u$ of $v$ in $G$ with $f(u) < f(v)$, update $v$ to $u$.
3) Output $v$.

-choice of neighbor if several $u$ have $f(u) < f(v)$?
-termination? correctness? running time?
Optimality and Reduced Cost

The **reduced cost** for a hyperplane at a corner is the cost of moving one unit away from the plane along its corresponding edge.

$$r_i = c \cdot e_i$$

(e$_i$ is the distance traveled on edge for plane p$_i$, c is the cost function)

For **minimization**, if all reduced costs are non-negative, then we are at an optimal solution.

Finding the most negative reduced cost is one often-used heuristic for choosing an edge to leave on
Reduced cost example

In the example the reduced cost of leaving the hyperplane $x_2=0$ is $(-2,-3) \cdot (2,1) = -7$ since moving one unit off of $x_2=0$ will move us $(2,1)$ units along the edge. We take the dot product of this and the cost function.
Simplex Algorithm

1. Find a **corner of the feasible region**

2. **Repeat**
   A. For each of the $n$ hyperplanes intersecting at the corner, calculate its **reduced cost**
   B. If they are all non-negative, then **done** (proof later)
   C. Else, pick the most negative reduced cost
      This is called the **entering** plane
   D. Move along corresponding edge (i.e., leave that hyperplane) until we reach the next corner (i.e., reach another hyperplane)
      The new plane is called the **departing** plane
Example

Step 1

\[ z = -2x_1 - 3x_2 \]

Step 2

Entrance

Departing

Start

Page 23
Simplifying

Problem:
- The $Ax \leq b$ constraints not symmetric with the $x \geq 0$ constraints.
  We would like more symmetry.

Idea:
- Make all inequalities of the form $x \geq 0$.
Use “slack variables” to do this.

Convert into form:

$$
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad Ax = b \\
& \quad x \geq 0
\end{align*}
$$
**Standard Form**

minimize \( c^T x \)
subject to \( Ax \leq b \)
\( x \geq 0 \)

\( |A| = m \times n \)

i.e., \( m \) inequalities, \( n \) variables

\[ 2x_1 + x_2 \leq 10 \]
\[ x_1 - 2x_2 \leq 4 \]

\[ 2x_1 + x_2 \leq 18 \]

\( |A'| = m \times (m+n) \)

i.e., \( m \) equations, \( m+n \) variables

\( 2x_1 + x_2 + x_4 = 18 \)
Example, again

minimize:
\[ z = -2x_1 - 3x_2 \]
subject to:
\[ x_1 - 2x_2 + x_3 = 4 \]
\[ 2x_1 + x_2 + x_4 = 18 \]
\[ x_2 + x_5 = 10 \]
\[ x_1, x_2, x_3, x_4, x_5 \geq 0 \]

The equality constraints impose a 2d plane (a 2d subspace) embedded in 5d space, looking at the plane gives the figure above
Using Matrices

If before adding the slack variables $A$ has size $m \times n$ ($m$ inequalities, $n$ variables), then after it has size $m \times (n + m)$
m can be larger or smaller than $n$

Assuming rows are independent, the solution space of $Ax = b$ is an $n$-dimensional subspace on $n+m$ variables.
Gauss-Jordan Elimination

\[
\begin{align*}
\text{Gauss-Jordan pivoting on non-zero } A_{lj} \\
\end{align*}
\]

\[
B_{ij} = \begin{cases} 
A_{ij} - A_{ij} \frac{A_{lk}}{A_{lk}} & i \neq l \\
\frac{A_{ij}}{A_{lk}} & i = l 
\end{cases}
\]
This form is called a **Basic Solution**
- the n “**free**” variables are set to 0
- the m “**basic**” variables are set to $b'$

A valid solution to $Ax = b$ if I, F, $b'$ reached using pivoting and column swapping from the original system $A, b$

Represents n intersecting hyperplanes

If feasible (i.e., $b' \geq 0$), then the solution is called a **Basic Feasible Solution** and is a corner of the feasible set
Corner

free variables indicate corner

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>-2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td></td>
<td>18</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>-3</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

basic variables free variables

$x_3$ $x_4$ $x_5$ $x_1$ $x_2$
Corner

Exchange free/basic variables by swapping columns, using GE to restore to tableau format. (“corner” not necessarily feasible, but we won’t allow this move)
Corner

\[
\begin{array}{cccccc}
1 & 0 & 0 & 1 & -2 & 4 \\
0 & 1 & 0 & -2 & 5 & 10 \\
0 & 0 & 1 & 0 & 1 & 10 \\
0 & 0 & 0 & 2 & -7 & 8 \\
\end{array}
\]

\[
x_1 \quad x_4 \quad x_5 \quad x_3 \quad x_2
\]
Corner

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>.2</td>
<td>.4</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-.4</td>
<td>.2</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>.4</td>
<td>-.2</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-.8</td>
<td>1.4</td>
<td></td>
<td>22</td>
</tr>
</tbody>
</table>

$x_1$ $x_2$ $x_3$ $x_4$ $x_5$
Corner

Note that in general there are \( n+m \) choose \( m \) corners
Simplex Method Again

Once you have found a basic feasible solution (a corner), we can move from corner to corner by swapping columns and eliminating.

**ALGORITHM**

1. Find a basic feasible solution
2. **Repeat**
   A. If $r$ (reduced cost) $\geq 0$, DONE
   B. Else, pick column $i$ with most negative $r$ (nonbasic gradient heuristic)
   C. Pick row $j$ with least non-negative $b_j'(j$’th entry in column $i$)
   D. Swap columns
   E. Use Gaussian elimination to restore form
Tableau Method

A. If \( r \) are all non-negative then **done**

- Reduced costs if all \( \geq 0 \) then done
- Current cost

<table>
<thead>
<tr>
<th>Basic Variables</th>
<th>Free Variables</th>
<th>( b' )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 0 )</td>
<td>( r )</td>
<td>( z' )</td>
</tr>
</tbody>
</table>
Tableau Method

B. Else, pick the most negative reduced cost
This is called the **entering** plane

![Tableau Method Diagram]

\[
\begin{array}{c|c|c}
I & F & b' \\
\hline
0 & r & z' \\
\end{array}
\]

\[\min\{r_i\} \text{ entering variable}\]
C. Move along corresponding line (i.e., leave that hyperplane) until we reach the next corner (i.e. reach another hyperplane)
The new plane is called the **departing** plane

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>F</th>
<th>b'</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>r</td>
<td>z'</td>
</tr>
</tbody>
</table>

\[ \text{departing variable} \]

\[ \text{min non-negative } b_j'/u_j \]
Tableau Method

D. Swap columns

\[ \begin{array}{c|c|c}
\times & \times & \times \\
\times & \times & \times \\
\times & \times & \times \\
\hline
& r & z' \\
\hline
\end{array} \]

No longer in proper form

E. Gauss-Jordan elimination

\[ \begin{array}{c|c|c}
I & F_{i+1} & b_{i+1}' \\
\hline
0 & r_{i+1} & z'_{i+1} \\
\end{array} \]

Back to proper form
Example

\[
\begin{array}{cccccc}
\text{x}_1 & \text{x}_2 & \text{x}_3 & \text{x}_4 & \text{x}_5 & \text{Value} \\
1 & -2 & 1 & 0 & 0 & 4 \\
2 & 1 & 0 & 1 & 0 & 18 \\
0 & 1 & 0 & 0 & 1 & 10 \\
\text{-2} & \text{-3} & 0 & & & 0 \\
\end{array}
\]

Find corner

\[
\begin{array}{cccccc}
\text{x}_1 & \text{x}_2 & \text{x}_3 & \text{x}_4 & \text{x}_5 & \text{Value} \\
1 & 0 & 0 & 1 & -2 & 4 \\
0 & 1 & 0 & 2 & 1 & 18 \\
0 & 0 & 1 & 0 & 1 & 10 \\
0 & 0 & 0 & -2 & -3 & 0 \\
\end{array}
\]

\[
\begin{align*}
\text{x}_1 - 2\text{x}_2 + \text{x}_3 &= 4 \\
2\text{x}_1 + \text{x}_2 + \text{x}_4 &= 18 \\
\text{x}_2 + \text{x}_5 &= 10 \\
-2\text{x}_1 - 3\text{x}_2 &= \text{z} + 0 \\
\end{align*}
\]

\[
\begin{array}{cccccc}
\text{x}_3 & \text{x}_4 & \text{x}_5 & \text{x}_1 & \text{x}_2 & \text{Value} \\
\end{array}
\]

\[
\text{x}_1 = \text{x}_2 = 0 \text{ (start)}
\]
Example

\[
\begin{array}{cccccc}
& 0 & 0 & 1 & -2 & 4 \\
1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 2 & 1 & 18 \\
0 & 0 & 1 & 0 & 1 & 10 \\
\hline
0 & 0 & 0 & -2 & -3 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccc}
& 0 & 0 & 0 & -2 & -3 & 0 \\
x_3 & x_4 & x_5 & x_1 & x_2 & b_j/v_j \\
\end{array}
\]

\[
\begin{array}{cccccc}
& 0 & 0 & 0 & 1 & -2 & 4 \\
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 2 & 0 & 18 \\
0 & 0 & 1 & 0 & 0 & 10 \\
\hline
0 & 0 & 0 & -2 & -3 & 0 \\
x_3 & x_4 & x_5 & x_1 & x_2 & b_j/v_j \\
\end{array}
\]

min non-negative
Example

\[
\begin{align*}
\begin{array}{ccccc|c}
1 & 0 & -2 & 1 & 0 & 4 \\
0 & 1 & 1 & 2 & 0 & 18 \\
0 & 0 & 1 & 0 & 1 & 10 \\
0 & 0 & -3 & -2 & 0 & 0 \\
x_3 & x_4 & x_2 & x_1 & x_5 & \\
\end{array}
\end{align*}
\]

Gauss-Jordan Elimination

\[
\begin{align*}
\begin{array}{ccccc|c}
1 & 0 & 0 & 1 & 2 & 24 \\
0 & 1 & 0 & 2 & -1 & 8 \\
0 & 0 & 1 & 0 & 1 & 10 \\
0 & 0 & 0 & -2 & 3 & 30 \\
x_3 & x_4 & x_2 & x_1 & x_5 & \\
\end{array}
\end{align*}
\]

\[-2x_1 + 3x_5 = z + 30 = z + z'\]
**Example**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>-1</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>3</td>
<td>30</td>
</tr>
</tbody>
</table>

$x_3$  $x_4$  $x_2$  $x_1$  $x_5$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>24</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>-1</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>10</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>3</td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>

$x_3$  $x_4$  $x_2$  $x_1$  $x_5$
Example

Swap

\[
\begin{array}{ccccc|c}
1 & 1 & 0 & 0 & 2 & 24 \\
0 & 2 & 0 & 1 & -1 & 8 \\
0 & 0 & 1 & 0 & 1 & 10 \\
0 & -2 & 0 & 0 & 3 & 30 \\
\end{array}
\]

\[
\begin{array}{ccccc}
x_3 & x_1 & x_2 & x_4 & x_5 \\
\end{array}
\]

Gauss-Jordan Elimination

\[
\begin{array}{ccccc|c}
1 & 0 & 0 & -.5 & 2.5 & 20 \\
0 & 1 & 0 & .5 & -.5 & 4 \\
0 & 0 & 1 & 0 & 1 & 10 \\
0 & 0 & 0 & 1 & 2 & 38 \\
\end{array}
\]

\[
\begin{array}{ccccc}
x_3 & x_1 & x_2 & x_4 & x_5 \\
\end{array}
\]
Linear Algebra Proof that Local Min is Optimum

• Last row of tableau specifies a derived equation for cost, \( z \), that holds for all feasible solutions.
• On termination, all coefficients in the equation are non-negative.
• E.g., from previous slide:
  \[
  0x_3 + 0x_1 + 0x_2 + 1x_4 + 2x_5 = z + 38, \text{ or } \\
  z = -38 + x_4 + 2x_5
  \]
• Since \( x_4 \) and \( x_5 \) must be non-negative, minimum value of \( z \) must be -38.
• Thus, Simplex finds a global minimum by deriving a constraint on cost with all non-negative coefficients.
Geometric Proof that Local Min is Optimum

- Let \( v=v_0 \) be the corner at which Simplex stops, let \( v_1, \ldots, v_n \) be the neighbors of \( v \), and let \( f \) denote the cost function that is to be minimized. Thus \( f(v) \leq f(v_i) \) for \( 0 \leq i \leq n \).

- Within the convex hull of \( \{v_0, v_1, \ldots, v_n\} \), \( f \) takes on a minimum value at \( v \), because for any \( q \) in the hull, \( f(q) = f(\sum \beta_i v_i) = \sum \beta_i f(v_i) \) so \( f(v) = \min_i f(v_i) \leq f(q) \).

- Suppose there is a better solution at some corner \( v' \) outside of the hull. On the line segment from \( v \) to \( v' \), the cost function should decrease linearly, but this line segment must pass through the interior of the hull, and until it exits, the cost doesn't decrease.
Minimizing / Maximizing Convex Cost Functions

• If trying to maximize a convex cost function over a closed convex space, the global maximum must occur at a corner, but a local maximum may not be a global maximum, e.g., maximize $f(x)$ such that $1 \leq x \leq 2$

• If trying to minimize a convex cost function, a local minimum is always a global minimum, but it may not occur at a corner. (E.g., above.)

• Convex hull of neighbors proof shows Simplex stops at a local minimum. Since a linear cost function is convex, a local minimum is also a global minimum.
Problem?: Unbounded Solution

- What if all $b_j/u_j$ are negative (i.e., all $b_j$ are positive and all $u_j$ negative)?
- Then there is no bounded solution.
- Can move an arbitrary distance from the current corner, reducing cost by an arbitrary amount.
- Not really a problem. LP is solved.
Problem: Cycling

• If the basic variable swapped in is already zero (b’\_j=0), then the distance moved along the edge is 0.

• Although a basic variable is exchanged with a free variable, there is no reduction in cost and the algorithm stays at the same vertex.

• In this “degenerate” case more than n hyperplanes intersect at the vertex (more than n variables have value 0).

• The algorithm may cycle through classification of variables as basic and free and reach a tableau that has already been seen before! Infinite loop!

• Solution: Bland’s anticycling rule for tie breaking among columns & rows.
Simplex Concluding remarks

For dense matrices, takes $O(n(n+m))$ time per iteration.
Can take an exponential number of iterations.
In practice, sparse methods are used for the iterations.
**Spielman-Teng “Smoothed” Complexity**

**Theorem:** If Gaussian noise is added to each coefficient of $A$ and $b$, then the expected number of iterations required by the Simplex algorithm to solve $Ax \leq b$ ($n$ variables and $m$ inequalities) is polynomial in $m$ and $n$ and $1/\sigma$, where $\sigma$ is the standard deviation of the random variable “maximum, taken over all individual coefficients, amount of noise added”.

This theorem may provide a partial explanation for why the exponential worst-case running time of the Simplex algorithm is rarely seen in practice.
**Duality**

**Primal (P):**

maximize \( z = c^T x \)

subject to \( Ax \leq b \)

\( x \geq 0 \) \((m \text{ equations, } n \text{ variables})\)

**Dual (D):**

minimize \( z = y^T b \)

subject to \( A^T y \geq c \)

\( y \geq 0 \) \((n \text{ equations, } m \text{ variables})\)

**Duality Theorem:** if \( x \) is feasible for \( P \) and \( y \) is feasible for \( D \), then \( c^T x \leq y^T b \)

and at optimality \( c^T x = y^T b \).
Duality (cont.)

Optimal solution for both

feasible solutions for primal (maximization)

feasible solutions for dual (minimization)

Quite similar to duality of Maximum Flow and Minimum Cut.

Useful in many situations.
Duality Example

**Primal:**

maximize:

\[ z = 2x_1 + 3x_2 \]

subject to:

\[ x_1 - 2x_2 \leq 4 \]
\[ 2x_2 \]
\[ 2x_1 + x_2 \leq 18 \]
\[ x_2 \leq 10 \]

Solution to both is 38 \((x_1=4, x_2=10), (y_1=0, y_2=1, y_3=2)\).

\[ x_1, x_2 \geq 0 \]

**Dual:**

minimize:

\[ z = 4y_1 + 18y_2 + 10y_3 \]

subject to:

\[ y_1 + 2y_2 \geq 2 \]
\[ -2y_1 + y_2 + y_3 \geq 3 \]
\[ y_1, y_2, y_3 \geq 0 \]