15-853: Algorithms in the Real World

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

Linear and Integer Programming
minimize \( z = c^T x \) cost or objective function
subject to \( Ax = b \) equalities
\( x \geq 0 \) inequalities
\( c \in \mathbb{R}^n, b \in \mathbb{R}^m, A \in \mathbb{R}^{n \times m} \)

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

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

Quadratic programming
\[ \text{min}(\frac{1}{2}x^TQx + c^T x : a_i^T x \leq b_i, i \in I, a_j^T x = b_j, j \in E) \]

Zero-One programming
\[ \text{min}(c^T x : Ax = b, x \in \{0,1\}^n, c \in \mathbb{R}^n, b \in \mathbb{R}^m) \]

Mixed Integer Programming
\[ \text{min}(c^T x : Ax = b, x \geq 0, x_i \in \mathbb{Z}, i \in I, x_r \in \mathbb{R}^n, r \in R) \]

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, ... 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 of 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

**Linear Programming for Max-Flow**

Create two variables per edge:

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

**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 subset 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-Barrirer Methods** (Frisch 1977, Fiacco 1968, Gill et al. 1986)

Many of the interior point methods can be applied to nonlinear programs. Not known to be 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.

Comparisons, 1994

<table>
<thead>
<tr>
<th>problem</th>
<th>Simplex (primal)</th>
<th>Simplex (dual)</th>
<th>Barrier + crossover</th>
</tr>
</thead>
<tbody>
<tr>
<td>binpacking</td>
<td>29.5</td>
<td>62.8</td>
<td>560.6</td>
</tr>
<tr>
<td>distribution</td>
<td>18,568.0</td>
<td>won't run</td>
<td>too big</td>
</tr>
<tr>
<td>forestry</td>
<td>1,354.2</td>
<td>1,911.4</td>
<td>2,348.0</td>
</tr>
<tr>
<td>maintenance</td>
<td>57,916.3</td>
<td>89,890.9</td>
<td>3,240.8</td>
</tr>
<tr>
<td>crew</td>
<td>7,182.6</td>
<td>16,172.2</td>
<td>1,264.2</td>
</tr>
<tr>
<td>airfleet</td>
<td>71,292.5</td>
<td>108,015.0</td>
<td>37,627.3</td>
</tr>
<tr>
<td>energy</td>
<td>3,091.1</td>
<td>1,943.8</td>
<td>858.0</td>
</tr>
<tr>
<td>4color</td>
<td>45,870.2</td>
<td>won't run</td>
<td>too big</td>
</tr>
</tbody>
</table>

Formulations

There are many ways to formulate linear programs:
- **objective (or cost) function**
  - Maximize $c^t x$, or
  - Minimize $c^t x$, or
  - Find any feasible solution
- **(in)equalities**
  - $Ax \leq b$, 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:

1. **minimize** $c^\top x$
   **subject to** $Ax \geq b$
   $x \geq 0$

2. **minimize** $c^\top x$
   **subject to** $Ax = b$
   $x \geq 0$

**slack variables**

*example*

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

More on slack variables later.

---

**Geometric View**

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**

For $n$ dimensions and no degeneracy

Each corner (extreme point) consists of:

- $n$ intersecting $n-1$ dimensional **hyperplanes**
  e.g. $3$, $2d$ planes in $3d$
- $n$ intersecting **edges**
  Each edge corresponds to moving off of one hyperplane (still constrained by $n-1$ of them)

**Simplex** will move from corner to corner along the edges
Optimality and Reduced Cost

The **Optimal** solution must include a corner. The **Reduced cost** for a hyperplane at a corner is the cost of moving one unit away from the plane along its corresponding edge.

\[ r_i = z \cdot e_i \]

For **minimization**, if all reduced costs are non-negative, then we are at an optimal solution. Finding the most negative reduced cost is a heuristic for choosing an edge to leave on.

Reduced cost example

In the example the reduced cost of leaving the plane \( x_1 \) is \((-2,-3) \cdot (2,1) = -7\) since moving one unit off of \( x_1 \) 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**
   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
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: $\minimize c^T x$
  subject to $Ax = b$
  $x \geq 0$

Standard and Slack Form

<table>
<thead>
<tr>
<th>Standard Form</th>
<th>Slack Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\minimize c^T x$</td>
<td>$\minimize c^T x'$</td>
</tr>
<tr>
<td>subject to $Ax \leq b$</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

$|A| = m \times n$
i.e. $m$ equations, $n$ variables
$|A'| = m \times (m+n)$
i.e. $m$ equations, $m+n$ variables

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 embedded in 5d space, looking at the plane gives the figure above

Using Matrices

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

$A = \begin{bmatrix}
1 & 0 & 0 & \ldots & 0 \\
0 & 1 & 0 & \ldots & 0 \\
0 & 0 & 1 & \ldots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \ldots & 1
\end{bmatrix}$

Assuming rows are independent, the solution space of $Ax = b$ is a $n$ dimensional subspace.
**Gauss-Jordan Elimination**

\[ \begin{array}{c|c|c|c|c|c} \hline k & & & & & \hline l & \vert & A_{lj} & & & \hline \end{array} \]

\[ B_{lj} = \begin{cases} A_{lj} - A_{lj} \frac{A_{ik}}{A_{jk}} & i \neq k \\ \frac{A_{lj}}{A_{lj}} & i = k \end{cases} \]

**Simplex Algorithm, again**

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

**Simplex Algorithm (Tableau Method)**

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 reached using Gaussian Elimination

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**
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) $\leq 0$, DONE
   B. Else, pick column with most negative $r$
   C. Pick row with least positive $b'/(selected\ column)$
   D. Swap columns
   E. Use Gaussian elimination to restore form

**Tableau Method**

A. If $r$ are all non-negative then **done**

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

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
**Tableau Method**

D. Swap columns

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

No longer in proper form

E. Gauss-Jordan elimination

<table>
<thead>
<tr>
<th>I</th>
<th>F_{i+1}</th>
<th>b_{i+1}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>r_{i+1}</td>
<td>z_{i+1}</td>
</tr>
</tbody>
</table>

Back to proper form

**Example**

\[ \begin{align*}
\begin{array}{cccccc}
1 & -2 & 1 & 0 & 0 & 4 \\
2 & 1 & 0 & 1 & 0 & 18 \\
0 & 1 & 0 & 0 & 1 & 10 \\
\end{array}
& \quad
\begin{array}{cccccc}
x_1 - 2x_2 + x_3 = 4 \\
2x_1 + x_2 + x_4 = 18 \\
x_2 + x_5 = 10 \\
\end{array}
\end{align*} \]

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

Find corner

\[ \begin{align*}
\begin{array}{cccccc}
1 & 0 & 0 & 1 & -2 & 4 \\
0 & 1 & 0 & 2 & 1 & 18 \\
0 & 0 & 1 & 0 & 1 & 10 \\
\end{array}
& \quad
\begin{array}{cccccc}
x_1 = x_2 = 0 \text{ (start)} \\
\end{array}
\end{align*} \]

**Example**

\[ \begin{align*}
\begin{array}{cccccc}
1 & 0 & 0 & 1 & -2 & 4 \\
0 & 1 & 1 & 2 & 0 & 18 \\
0 & 0 & 1 & 0 & 1 & 10 \\
\end{array}
& \quad
\begin{array}{cccccc}
x_1 = x_2 = 0 \text{ (start)} \\
\end{array}
\end{align*} \]

\[ x_5 = 3x_1 - x_2 \]

**Example**

\[ \begin{align*}
\begin{array}{cccccc}
1 & 0 & 1 & 2 & 2 & 4 \\
0 & 1 & 0 & 2 & -1 & 8 \\
0 & 0 & 1 & 0 & 1 & 10 \\
\end{array}
& \quad
\begin{array}{cccccc}
x_1 = x_2 = 0 \text{ (start)} \\
\end{array}
\end{align*} \]

\[ x_5 = 18x_1 + 10x_2 \]

Gauss-Jordan Elimination
**Example**

\[
\begin{bmatrix}
1 & 0 & 0 & 1 & 2 & 24 \\
0 & 1 & 0 & 2 & -1 & 8 \\
0 & 0 & 1 & 0 & 1 & 10 \\
0 & 0 & 0 & -2 & 3 & 30 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 0 & 0 & 1 & 2 & 24 \\
0 & 1 & 0 & 2 & -1 & 8 \\
0 & 0 & 1 & 0 & 1 & 10 \\
0 & 0 & 0 & -2 & 3 & 30 \\
\end{bmatrix}
\]

**Example**

\[
\begin{bmatrix}
1 & 1 & 0 & 0 & 2 & 24 \\
0 & 1 & 0 & 1 & -1 & 8 \\
0 & 0 & 1 & 0 & 1 & 10 \\
0 & -2 & 0 & 0 & 3 & 30 \\
\end{bmatrix}
\]

**Example**

\[
\begin{bmatrix}
1 & 0 & 0 & .5 & 2.5 & 20 \\
0 & 1 & 0 & .5 & -.5 & 4 \\
0 & 0 & 1 & 0 & 1 & 10 \\
0 & 0 & 0 & 1 & 2 & 38 \\
\end{bmatrix}
\]

**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.

**Duality**

**Primal (P):**

\[
\begin{align*}
\text{maximize} & \quad z = c^Tx \\
\text{subject to} & \quad Ax \leq b \\
& \quad x \geq 0 \quad (n \text{ equations, } m \text{ variables})
\end{align*}
\]

**Dual (D):**

\[
\begin{align*}
\text{minimize} & \quad z = y^Tb \\
\text{subject to} & \quad A^Ty \geq c \\
& \quad y \geq 0 \quad (m \text{ equations, } n \text{ variables})
\end{align*}
\]

**Duality Theorem:** if \(x\) is feasible for \(P\) and \(y\) is feasible for \(D\), then \(cx \leq yb\) and at optimality \(cx = yb\).
Duality (cont.)

Optimal solution for both

feasible solutions for Dual (maximization)
feasible solutions for Primal (minimization)

Quite similar to duality of Maximum Flow and Minimum Cut.

Useful in many situations.

Duality Example

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

Dual:
minimize:
subject to:
\[ z = 4y_1 + 18y_2 + 10y_3 \]
\[ y_1 + 2y_2 \geq 2 \]
\[ -2y_1 + Y_2 + Y_3 \geq 3 \]
\[ Y_1, Y_2, Y_3 \geq 0 \]

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