296.3: 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 \( z = c^T x \) cost or objective function
subject to \( Ax \leq b \) inequalities
\( x \geq 0 \)
\( c \in R^n, b \in R^m, A \in R^{n \times m} \)

Linear programming:
\( x \in R^n \) (polynomial time)

Integer programming:
\( x \in Z^n \) (NP-complete)
Extremely general framework, especially IP

Related Optimization Problems

Unconstrained optimization
\( \min \{ f(x) : x \in R^n \} \)

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

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

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

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

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 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

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

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 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)

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

- **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:

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

\[ 7x_1 + 5x_2 \geq 7 \]
\[ x_1, x_2 \geq 0 \]

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

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).

Optimum (max problem) is at a Vertex

Holds even if the objective function is merely convex: $f$ is convex if for all vectors $x, y \in S$ and $\beta \in [0, 1]$

\[ f(\beta x + (1- \beta)y) \leq \beta f(x) + (1- \beta)f(y) \]
**Optimum (max problem) is at a Vertex**

Every point \( q \) in \( P \) is a convex combination of vertices of \( P \).

There exist \( \beta_i \),

\[
q = \sum \beta_i v_i
\]

Fix convex \( f \), then

\[
f(q) = f(\sum \beta_i v_i) \leq \sum \beta_i f(v_i) \leq \max f(v_i)
\]

If \( f \) is linear, then

\[
f(q) = f(\sum \beta_i v_i) = \sum \beta_i f(v_i)
\]

\[
\min f(v_i) \leq f(q) \leq \max f(v_i)
\]

---

**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

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

(i.e., \( A \) has full row rank)

Each corner (extreme point) consists of:

- \( n \) intersecting \( (n-1) \)-dimensional **hyperplanes**
  e.g. \( n = 3 \), 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

---

**The Simple Essense of Simplex**

Consider Polytope \( P \) from canonical form as a graph \( G = (V,E) \) with

\( V = \) polytope vertices,
\( E = \) 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.

\[ r_i = z \cdot e_i \]

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

**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
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:

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

Example, again

minimize:
\[ z = -2x_1 - 3x_2 \]
subject to:
\[
\begin{align*}
x_1 - 2x_2 + x_3 &= 4 \\
2x_1 + x_2 + x_4 &= 18 \\
x_2 + x_5 &= 10 \\
x_1, x_2, x_3, x_4, x_5 &\geq 0
\end{align*}
\]

The equality constraints impose a 2d plane embedded in 5d space, looking at the plane gives the figure above.

Standard Form

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad Ax \leq b \\
& \quad x \geq 0
\end{align*}
\]

\[
\begin{align*}
\text{minimize} & \quad c^T x' \\
\text{subject to} & \quad A'x' = b \\
& \quad x' \geq 0
\end{align*}
\]

Using Matrices

If before adding the slack variables \( A \) has size \( m \times n \) then after it has size \( m \times (n + m) \).

\[ A = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 & 1 & 0 & \ldots & 0 & 1 & \ldots \end{bmatrix} \]

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

\[ A_{ij} \]

\[ B_{ij} = \begin{cases} 
    A_{ij} - \frac{A_{ik}}{A_{ii}} & i \neq l \\
    \frac{A_{ij}}{A_{ii}} & i = l 
\end{cases} \]

**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)**

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

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**

<table>
<thead>
<tr>
<th>x_1</th>
<th>x_2</th>
<th>x_3</th>
<th>x_4</th>
<th>x_5</th>
<th>x_6</th>
<th>x_7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>

Corner, free variables indicate corner
Exchange free/basic variables by swapping columns, using GE to restore to tableau format. ("corner" not necessarily feasible)

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**

- \[ \begin{array}{ccc} I & F & b' \\ 0 & r & z \\ \end{array} \]
  - current cost
  - reduced costs if all \( \geq 0 \) then done

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

- \[ \begin{array}{ccc} I & F & b' \\ 0 & r & z \\ \end{array} \]
  - \( \min(r_i) \) 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

- \[ \begin{array}{ccc} I & F & b' \\ 0 & r & z \\ \end{array} \]
  - min positive \( b_j / u_j \)
  - departing variable
Tableau Method

D. Swap columns

<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>

No longer in proper form

E. Gauss-Jordan elimination

Back to proper form

Example

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

\[x_1 = 2x_2 + x_3 = 4\]
\[2x_1 + x_2 + x_4 = 18\]
\[x_5 + x_5 = 10\]
\[z = -2x_1 - 3x_2\]

Find corner

\[x_1 = x_2 = 0\text{ (start)}\]

Example

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

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

Gauss-Jordan Elimination
**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 basic variable swapped in is already zero ($b'_j=0$), don’t reduce total cost.
- Can cycle back to already seen vertex!
- 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.

**Duality**

**Primal (P):**

maximize $z = c^T x$

subject to $Ax \leq b$

$x \geq 0$ (n equations, m variables)

**Dual (D):**

minimize $z = y^T b$

subject to $A^Ty \geq c$

$y \geq 0$ (m equations, n 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:

$x_1 - 2x_2 \leq 4$

$2x_1 + x_2 \leq 18$

$x_2 \leq 10$

$x_1, x_2 \geq 0$

**Dual:**

minimize: $z = 4y_1 + 18y_2 + 10y_3$

subject to:

$y_1 + 2y_2 \geq 2$

$-2y_1 + y_2 + y_3 \geq 3$

$y_1, y_2, y_3 \geq 0$

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