Linear Programming And The Simplex Algorithm

Ethan Lam Patrick Oweijane

April 27, 2022

Christian Brothers University

What Is Linear Programming?

- Linear programming is an optimization technique where we assign values to variables constrained by linear relationships.

What Is Linear Programming?

- Linear programming is an optimization technique where we assign values to variables constrained by linear relationships.
- Linear programming problems often appear in applications such as operations research and finance.
- Problems will often come in the flavor of finding the best way to allocate resource under certain constraints.

Example

- Suppose you are managing 6000 acres of land belonging to a farm co-op.

Example

- Suppose you are managing 6000 acres of land belonging to a farm co-op.
- You can either plant corn or soybeans.

Example

- Suppose you are managing 6000 acres of land belonging to a farm co-op.
- You can either plant corn or soybeans.
- You have the following resource budget:

	Corn	**Soybeans**	**Available**
Fertilizer/herbicide	9 gal/acre	3 gal/acre	40500 gal
Harvesting labor	$\frac{3}{4}$ hr/acre	1 hr/acre	5250 hr
Profit	240 \$/acre	160 \$/acre	

Example

- Suppose you are managing 6000 acres of land belonging to a farm co-op.
- You can either plant corn or soybeans.
- You have the following resource budget:

	Corn	**Soybeans**	**Available**
Fertilizer/herbicide	9 gal/acre	3 gal/acre	40500 gal
Harvesting labor	$\frac{3}{4}$ hr/acre	1 hr/acre	5250 hr
Profit	240 \$/acre	160 \$/acre	

- **How do you optimally plant corn and soybeans to maximize profit?**

Example (cont.)

We can write this optimization problem as

$$
\begin{array}{llrcrcrr}
\max & 240x_1 & + & 160x_2 & & & & \text{(Profit function)} \\
\text{s.t.} & 9x_1 & + & 3x_2 & \leq & 40500 & & \text{(Fertilizer/herbicide constraint)} \\
& \frac{3}{4}x_1 & + & x_2 & \leq & 5250 & & \text{(Labor constraint)} \\
& x_1 & + & x_2 & \leq & 6000 & & \text{(Land constraint)} \\
& x_1, x_2 & & & \geq & 0 & & \text{(Non-negativity constraint)}
\end{array}
$$

where your job is to find the best possible assignment of x_1 and x_2 which represent the amount of corn and soybeans you plant, respectively.

Formalism

- We will call instances of linear programming problems **linear programs**.

Formalism

- We will call instances of linear programming problems **linear programs**.
- Linear programs can be written in **standard form**.

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

where A is an $m \times n$ matrix, x and c are n dimensional column vectors, and b is an m dimensional column vector.

Formalism

- We will call instances of linear programming problems **linear programs**.
- Linear programs can be written in **standard form**.

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

where A is an $m \times n$ matrix, x and c are n dimensional column vectors, and b is an m dimensional column vector.

Remark
Let b and b' are m dimensional vectors. By $b \leq b'$, we mean that for all $i = 1, 2, \ldots, m$

$$b_i \leq b'_i$$

Formalism Made Concrete

Original linear program:

$$
\begin{array}{lrcrl}
\max & 240x_1 & + & 160x_2 & \\
\text{s.t.} & 9x_1 & + & 3x_2 & \leq 40500 \\
& \frac{3}{4}x_1 & + & x_2 & \leq 5250 \\
& x_1 & + & x_2 & \leq 6000 \\
& x_1, x_2 & & & \geq 0
\end{array}
$$

Formalism Made Concrete

Original linear program:

$$
\begin{array}{llrcl}
\max & 240x_1 + 160x_2 & & & \\
\text{s.t.} & 9x_1 + 3x_2 & \leq & 40500 & \rightarrow \\
& \frac{3}{4}x_1 + x_2 & \leq & 5250 & \\
& x_1 + x_2 & \leq & 6000 & \\
& x_1, x_2 & \geq & 0 &
\end{array}
$$

Formalism Made Concrete

Original linear program:

$$
\begin{array}{llcrl}
\max & 240x_1 + 160x_2 & & & \\
\text{s.t.} & 9x_1 + & 3x_2 & \leq & 40500 \\
 & \frac{3}{4}x_1 + & x_2 & \leq & 5250 \\
 & x_1 + & x_2 & \leq & 6000 \\
 & x_1, x_2 & & \geq & 0
\end{array}
\quad \rightarrow
$$

Standard form:

$$
\max \quad \overbrace{\begin{bmatrix} 240 \\ 160 \end{bmatrix}^T}^{c^T} \overbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}^{x}
$$

Formalism Made Concrete

Original linear program:

$$\begin{array}{llrl} \max & 240x_1 + 160x_2 & & \\ \text{s.t.} & 9x_1 + 3x_2 & \leq & 40500 \\ & \frac{3}{4}x_1 + x_2 & \leq & 5250 \\ & x_1 + x_2 & \leq & 6000 \\ & x_1, x_2 & \geq & 0 \end{array}$$

$\rightarrow$

Standard form:

$$\begin{array}{ll} \max & \overbrace{\begin{bmatrix} 240 \\ 160 \end{bmatrix}^T}^{c^T} \overbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}^{x} \\ \text{s.t.} & \underbrace{\begin{bmatrix} 9 & 3 \\ \frac{3}{4} & 1 \\ 1 & 1 \end{bmatrix}}_{A} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \underbrace{\begin{bmatrix} 40500 \\ 5250 \\ 6000 \end{bmatrix}}_{b} \end{array}$$

Formalism Made Concrete

Original linear program:

$$\begin{array}{lrcrcr} \max & 240x_1 & + & 160x_2 & & \\ \text{s.t.} & 9x_1 & + & 3x_2 & \leq & 40500 \\ & \frac{3}{4}x_1 & + & x_2 & \leq & 5250 \\ & x_1 & + & x_2 & \leq & 6000 \\ & x_1, x_2 & & & \geq & 0 \end{array} \quad \rightarrow$$

Standard form:

$$\begin{array}{ll} \max & \overbrace{\begin{bmatrix} 240 \\ 160 \end{bmatrix}^T}^{c^T} \overbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}^{x} \\ \text{s.t.} & \underbrace{\begin{bmatrix} 9 & 3 \\ \frac{3}{4} & 1 \\ 1 & 1 \end{bmatrix}}_{A} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \underbrace{\begin{bmatrix} 40500 \\ 5250 \\ 6000 \end{bmatrix}}_{b} \\ & \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq 0 \end{array}$$

Formalism (cont.)

- The function we are optimizing is called the **objective function**.

Formalism (cont.)

- The function we are optimizing is called the **objective function**.
- An assignment to x that satisfies all constraints is called a **feasible solution**.

Formalism (cont.)

- The function we are optimizing is called the **objective function**.
- An assignment to x that satisfies all constraints is called a **feasible solution**.
- Linear programs (in standard form) may be:
  - Feasible ... optimal solution has finite objective value

(Fundamental theorem of linear programming)

Formalism (cont.)

- The function we are optimizing is called the **objective function**.
- An assignment to x that satisfies all constraints is called a **feasible solution**.
- Linear programs (in standard form) may be:
  - Feasible ... optimal solution has finite objective value
  - Infeasible ... no solutions

(Fundamental theorem of linear programming)

Formalism (cont.)

- The function we are optimizing is called the **objective function**.
- An assignment to x that satisfies all constraints is called a **feasible solution**.
- Linear programs (in standard form) may be:
  - Feasible ... optimal solution has finite objective value
  - Infeasible ... no solutions
  - Unbounded ... the objective value can be made infinite

  (Fundamental theorem of linear programming)

Is Standard Form Too Restrictive?

- What if we want to *minimize* the objective function?

Is Standard Form Too Restrictive?

- What if we want to *minimize* the objective function?
- What if we want variables to be *unbounded*?

Is Standard Form Too Restrictive?

- What if we want to *minimize* the objective function?
- What if we want variables to be *unbounded*?
- What if we want variables to *lie in an interval*?

Is Standard Form Too Restrictive?

- What if we want to *minimize* the objective function?
- What if we want variables to be *unbounded*?
- What if we want variables to *lie in an interval*?
- What if we want constraints to use $\geq$ or $=$?

Is Standard Form Too Restrictive?

- What if we want to *minimize* the objective function?
- What if we want variables to be *unbounded*?
- What if we want variables to *lie in an interval*?
- What if we want constraints to use $\geq$ or $=$?
- **All of these situations can be reformulated into a standard form equivalent!**

Slack Form Intuition

Example
Consider

$$x_1 + x_2 \leq 6000$$

Slack Form Intuition

Example
Consider

$$x_1 + x_2 \leq 6000$$

then

$$0 \leq \underbrace{6000 - x_1 - x_2}_{s}$$

Slack Form Intuition

Example
Consider

$$x_1 + x_2 \leq 6000$$

then

$$0 \leq \underbrace{6000 - x_1 - x_2}_{s}$$

where s is the "slack" we have in increasing $x_1 + x_2$ until the constraint is violated.

Slack Form

From the standard form we can construct the slack form:

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & s = b - Ax \\ & x, s \geq 0 \end{aligned}$$

where s holds the "slack" variables which are added.

Slack Form

From the standard form we can construct the slack form:

$$\begin{aligned}\max \quad & c^T x \\ \text{s.t.} \quad & s = b - Ax \\ & x, s \geq 0\end{aligned}$$

where s holds the “slack” variables which are added.

- Slack form turns our inequality constraints into equality constraints!

Slack Form

From the standard form we can construct the slack form:

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & s = b - Ax \\ & x, s \geq 0 \end{aligned}$$

where s holds the "slack" variables which are added.

- Slack form turns our inequality constraints into equality constraints!
- Variables on the left of the $=$ are called **basic variables**

Slack Form

From the standard form we can construct the slack form:

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & s = b - Ax \\ & x, s \geq \quad 0 \end{aligned}$$

where s holds the "slack" variables which are added.

- Slack form turns our inequality constraints into equality constraints!
- Variables on the left of the $=$ are called **basic variables**
- Variables on the right of the $=$ are called **nonbasic variables**

Slack Form

From the standard form we can construct the slack form:

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & s = b - Ax \\ & x, s \geq 0 \end{aligned}$$

where s holds the “slack” variables which are added.

- Slack form turns our inequality constraints into equality constraints!
- Variables on the left of the $=$ are called **basic variables**
- Variables on the right of the $=$ are called **nonbasic variables**
- *Note: Only nonbasic variables appear in the objective function*

Slack Form

From the standard form we can construct the slack form:

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & s = b - Ax \\ & x, s \geq 0 \end{aligned}$$

where s holds the "slack" variables which are added.

- Slack form turns our inequality constraints into equality constraints!
- Variables on the left of the $=$ are called **basic variables**
- Variables on the right of the $=$ are called **nonbasic variables**
- *Note: Only nonbasic variables appear in the objective function*
- *Note: As you solve a linear program, the nonbasic and basic variables will change.*

Slack Form

From the standard form we can construct the slack form:

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & s = b - Ax \\ & x, s \geq 0 \end{aligned}$$

where s holds the "slack" variables which are added.

- Slack form turns our inequality constraints into equality constraints!
- Variables on the left of the $=$ are called **basic variables**
- Variables on the right of the $=$ are called **nonbasic variables**
- *Note: Only nonbasic variables appear in the objective function*
- *Note: As you solve a linear program, the nonbasic and basic variables will change.*
- *Note: Slack form makes solving linear programs easier.*

Slack Form Example

Standard form:

$$
\begin{array}{llll}
\max & 240x_1 + 160x_2 & & \\
\text{s.t.} & 9x_1 + 3x_2 & \leq & 40500 \\
& \frac{3}{4}x_1 + x_2 & \leq & 5250 \\
& x_1 + x_2 & \leq & 6000 \\
& x_1, x_2 & \geq & 0
\end{array}
$$

Slack Form Example

Standard form:

$$
\begin{array}{lrcrl}
\max & 240x_1 & + & 160x_2 & \\
\text{s.t.} & 9x_1 & + & 3x_2 & \leq 40500 \\
 & \frac{3}{4}x_1 & + & x_2 & \leq 5250 \\
 & x_1 & + & x_2 & \leq 6000 \\
 & x_1, x_2 & & & \geq 0
\end{array}
$$

Slack form:

$$
\begin{array}{lrl}
\max & & 240x_1 + 160x_2 \\
\text{s.t.} & s_1 = & 40500 - 9x_1 - 3x_2 \\
 & s_2 = & 5250 - \frac{3}{4}x_1 - x_2 \\
 & s_3 = & 6000 - x_1 - x_2 \\
 & x_1, x_2, s_1, s_2, s_3 \geq & 0
\end{array}
$$

Simplex Algorithm

- Developed by George Dantzig

Simplex Algorithm

- Developed by George Dantzig
- Runs in *exponential* time but there exists polynomial time algorithms that solve linear programs

Simplex Algorithm

- Developed by George Dantzig
- Runs in *exponential* time but there exists polynomial time algorithms that solve linear programs
- Runs reasonably fast in practical applications

Simplex Algorithm

- Developed by George Dantzig
- Runs in *exponential* time but there exists polynomial time algorithms that solve linear programs
- Runs reasonably fast in practical applications
- Based on a fundamental operation called **pivoting**

Simplex Algorithm

- Developed by George Dantzig
- Runs in *exponential* time but there exists polynomial time algorithms that solve linear programs
- Runs reasonably fast in practical applications
- Based on a fundamental operation called **pivoting**

Interesting Theoretical Remarks

- The ability to solve linear programs in polynomial time means we have algorithms to solve problems reduced to linear programs quickly!

Simplex Algorithm

- Developed by George Dantzig
- Runs in *exponential* time but there exists polynomial time algorithms that solve linear programs
- Runs reasonably fast in practical applications
- Based on a fundamental operation called **pivoting**

Interesting Theoretical Remarks

- The ability to solve linear programs in polynomial time means we have algorithms to solve problems reduced to linear programs quickly!
- Restricting the variables to integers makes solving linear programs NP-Hard! (*3-CNF-SAT can be reduced to determining if an integer linear program is feasible*)

Simplex Example

$$
\begin{array}{lrl}
\max & 240\,x_1 + 160x_2 & \\
\text{s.t.} & s_1 = & 40500 - 9\,x_1 - 3x_2 \\
& s_2 = & 5250 - \frac{3}{4}x_1 - x_2 \\
& s_3 = & 6000 - x_1 - x_2 \\
& x_1, x_2, s_1, s_2, s_3 \geq & 0
\end{array}
$$

Simplex Example

$$
\begin{array}{llrl}
\max & & 240\,x_1 + 160x_2 & \\
\text{s.t.} & s_1 & = 40500 - 9\,x_1 - 3x_2 & \\
 & s_2 & = 5250 - \frac{3}{4}x_1 - x_2 & \\
 & s_3 & = 6000 - x_1 - x_2 & \\
 & x_1, x_2, s_1, s_2, s_3 & \geq 0 &
\end{array}
$$

- Basic Solution: $(x_1, x_2, s_1, s_2, s_3) = (0, 0, 40500, 5250, 6000)$

Simplex Example

$$
\begin{array}{lrl}
\max & 240\,x_1 + 160x_2 & \\
\text{s.t.} & s_1 = & 40500 - 9\,x_1 - 3x_2 \\
& s_2 = & 5250 - \frac{3}{4}x_1 - x_2 \\
& s_3 = & 6000 - x_1 - x_2 \\
& x_1, x_2, s_1, s_2, s_3 \geq & 0
\end{array}
$$

- Basic Solution: $(x_1, x_2, s_1, s_2, s_3) = (0, 0, 40500, 5250, 6000)$
- Basic Solution Value: 0

Simplex Example

$$
\begin{array}{lrl}
\max & 240\boxed{x_1} + 160x_2 & \\
\text{s.t.} & s_1 = & 40500 - 9\,x_1 - 3x_2 \\
 & s_2 = & 5250 - \frac{3}{4}x_1 - x_2 \\
 & s_3 = & 6000 - x_1 - x_2 \\
 & x_1, x_2, s_1, s_2, s_3 \geq & 0
\end{array}
$$

- Basic Solution: $(x_1, x_2, s_1, s_2, s_3) = (0, 0, 40500, 5250, 6000)$
- Basic Solution Value: 0
- Entering Variable: $\boxed{x_1}$

Simplex Example

$$
\begin{array}{llrl}
\max & & 240\,x_1 + 160x_2 & \\
\text{s.t.} & \longrightarrow & s_1 = 40500 - 9\,x_1 - 3x_2 & \\
& & s_2 = 5250 - \tfrac{3}{4}x_1 - x_2 & \\
& & s_3 = 6000 - x_1 - x_2 & \\
& & x_1, x_2, s_1, s_2, s_3 \geq 0 &
\end{array}
$$

- Basic Solution: $(x_1, x_2, s_1, s_2, s_3) = (0, 0, 40500, 5250, 6000)$
- Basic Solution Value: 0
- Entering Variable: x_1
- First Constraint: x_1 can be at most $40500/9 = 4500$

Simplex Example

$$
\begin{array}{lrl}
\max & & 240\boxed{x_1} + 160x_2 \\
\text{s.t.} & s_1 & = 40500 - 9\,x_1 - 3x_2 \\
& \longrightarrow s_2 & = 5250 - \frac{3}{4}x_1 - x_2 \\
& s_3 & = 6000 - x_1 - x_2 \\
& x_1, x_2, s_1, s_2, s_3 & \geq 0
\end{array}
$$

- Basic Solution: $(x_1, x_2, s_1, s_2, s_3) = (0, 0, 40500, 5250, 6000)$
- Basic Solution Value: 0
- Entering Variable: x_1
- First Constraint: x_1 can be at most $40500/9 = 4500$
- Second Constraint: x_1 can be at most $5250/(3/4) = 7000$

Simplex Example

$$
\begin{array}{lrl}
\max & 240\,x_1 + 160x_2 & \\
\text{s.t.} & s_1 = & 40500 - 9\,x_1 - 3x_2 \\
 & s_2 = & 5250 - \tfrac{3}{4}x_1 - x_2 \\
 & \longrightarrow s_3 = & 6000 - x_1 - x_2 \\
 & x_1, x_2, s_1, s_2, s_3 \geq & 0
\end{array}
$$

- Basic Solution: $(x_1, x_2, s_1, s_2, s_3) = (0, 0, 40500, 5250, 6000)$
- Basic Solution Value: 0
- Entering Variable: x_1
- First Constraint: x_1 can be at most $40500/9 = 4500$
- Second Constraint: x_1 can be at most $5250/(3/4) = 7000$
- Third Constraint: x_1 can be at most $6000/1 = 6000$

Simplex Example

$$
\begin{array}{llrl}
\max & & 240\boxed{x_1} + 160x_2 & \\
\text{s.t.} & \boxed{s_1} & = 40500 - 9\boxed{x_1} - 3x_2 & \\
 & s_2 & = 5250 - \tfrac{3}{4}x_1 - x_2 & \\
 & s_3 & = 6000 - x_1 - x_2 & \\
 & x_1, x_2, s_1, s_2, s_3 & \geq 0 &
\end{array}
$$

- Basic Solution: $(x_1, x_2, s_1, s_2, s_3) = (0, 0, 40500, 5250, 6000)$
- Basic Solution Value: 0
- Entering Variable: $\boxed{x_1}$
- First Constraint: x_1 can be at most $40500/9 = 4500$
- Second Constraint: x_1 can be at most $5250/(3/4) = 7000$
- Third Constraint: x_1 can be at most $6000/1 = 6000$
- Leaving Variable: $\boxed{s_1}$

Simplex Example (cont.)

Increasing x_1 as much as possible means $s_1 = 0$ (there is no more "slack"), and we can turn s_1 nonbasic and x_1 basic:

$$\begin{array}{lrcl}
\max & 240\boxed{x_1} + 160x_2 & & \\
\text{s.t.} & \boxed{x_1} & = & 4500 - \frac{1}{9}\boxed{s_1} - \frac{1}{3}x_2 \\
 & s_2 & = & 5250 - \frac{3}{4}x_1 - x_2 \\
 & s_3 & = & 6000 - x_1 - x_2 \\
 & x_1, x_2, s_1, s_2, s_3 & \geq & 0
\end{array}$$

Simplex Example (cont.)

Increasing x_1 as much as possible means $s_1 = 0$ (there is no more "slack"), and we can turn s_1 nonbasic and x_1 basic:

$$\begin{array}{lrcl}
\max & 240\boxed{x_1} + 160x_2 & & \\
\text{s.t.} & \boxed{x_1} & = & 4500 - \frac{1}{9}\boxed{s_1} - \frac{1}{3}x_2 \\
& s_2 & = & 5250 - \frac{3}{4}x_1 - x_2 \\
& s_3 & = & 6000 - x_1 - x_2 \\
& x_1, x_2, s_1, s_2, s_3 & \geq & 0
\end{array}$$

We now substitute all occurrences of x_1 to get

$$\begin{array}{lrcl}
\max & -\frac{80}{3}\boxed{s_1} + 80x_2 + 1080000 & & \\
\text{s.t.} & \boxed{x_1} & = & 4500 - \frac{1}{9}\boxed{s_1} - \frac{1}{3}x_2 \\
& s_2 & = & 1875 + \frac{1}{12}s_1 - \frac{3}{4}x_2 \\
& s_3 & = & 1500 + \frac{1}{9}s_1 - \frac{2}{3}x_2 \\
& x_1, x_2, s_1, s_2, s_3 & \geq & 0
\end{array}$$

Simplex Example (cont.)

Increasing x_1 as much as possible means $s_1 = 0$ (there is no more "slack"), and we can turn s_1 nonbasic and x_1 basic:

$$
\begin{array}{lrcl}
\max & 240 x_1 + 160x_2 & & \\
\text{s.t.} & x_1 & = & 4500 - \frac{1}{9} s_1 - \frac{1}{3}x_2 \\
& s_2 & = & 5250 - \frac{3}{4}x_1 - x_2 \\
& s_3 & = & 6000 - x_1 - x_2 \\
& x_1, x_2, s_1, s_2, s_3 & \geq & 0
\end{array}
$$

We now substitute all occurrences of x_1 to get

$$
\begin{array}{lrcl}
\max & -\frac{80}{3} s_1 + 80x_2 + 1080000 & & \\
\text{s.t.} & x_1 & = & 4500 - \frac{1}{9} s_1 - \frac{1}{3}x_2 \\
& s_2 & = & 1875 + \frac{1}{12}s_1 - \frac{3}{4}x_2 \\
& s_3 & = & 1500 + \frac{1}{9}s_1 - \frac{2}{3}x_2 \\
& x_1, x_2, s_1, s_2, s_3 & \geq & 0
\end{array}
$$

We now repeat this process ...

Simplex Example (cont.)

$$
\begin{array}{llrl}
\max & & -\frac{80}{3}s_1 + 80\;x_2 & + 1080000 \\
\text{s.t.} & x_1 = & 4500 - & \frac{1}{9}s_1 - \frac{1}{3}x_2 \\
 & s_2 = & 1875 + & \frac{1}{12}s_1 - \frac{3}{4}x_2 \\
 & s_3 = & 1500 + & \frac{1}{9}s_1 - \frac{2}{3}\;x_2 \\
 & x_1, x_2, s_1, s_2, s_3 \geq & 0 &
\end{array}
$$

Simplex Example (cont.)

$$
\begin{array}{llll}
\max & -\frac{80}{3}s_1 + 80\ x_2 & + 1080000 & \\
\text{s.t.} & x_1 = & 4500 - & \frac{1}{9}s_1 - \frac{1}{3}x_2 \\
& s_2 = & 1875 + & \frac{1}{12}s_1 - \frac{3}{4}x_2 \\
& s_3 = & 1500 + & \frac{1}{9}s_1 - \frac{2}{3}\ x_2 \\
& x_1, x_2, s_1, s_2, s_3 \geq & 0 &
\end{array}
$$

- Basic Solution: $(x_1, x_2, s_1, s_2, s_3) = (4500, 0, 0, 1875, 1500)$

Simplex Example (cont.)

$$
\begin{array}{llrl}
\max & & -\frac{80}{3}s_1 + 80\ x_2 & + 1080000 \\
\text{s.t.} & x_1 = & 4500 - & \frac{1}{9}s_1 - \frac{1}{3}x_2 \\
 & s_2 = & 1875 + & \frac{1}{12}s_1 - \frac{3}{4}x_2 \\
 & s_3 = & 1500 + & \frac{1}{9}s_1 - \frac{2}{3}\ x_2 \\
 & x_1, x_2, s_1, s_2, s_3 \geq & 0 &
\end{array}
$$

- Basic Solution: $(x_1, x_2, s_1, s_2, s_3) = (4500, 0, 0, 1875, 1500)$
- Basic Solution Value: 1080000

Simplex Example (cont.)

$$
\begin{array}{llrcrcrcl}
\max & & -\frac{80}{3}s_1 + 80\, x_2 + 1080000 & & & & & & \\
\text{s.t.} & & x_1 & = & 4500 & - & \frac{1}{9}s_1 & - & \frac{1}{3}x_2 \\
& & s_2 & = & 1875 & + & \frac{1}{12}s_1 & - & \frac{3}{4}x_2 \\
& & s_3 & = & 1500 & + & \frac{1}{9}s_1 & - & \frac{2}{3}x_2 \\
& & x_1, x_2, s_1, s_2, s_3 & \geq & 0 & & & &
\end{array}
$$

- Basic Solution: $(x_1, x_2, s_1, s_2, s_3) = (4500, 0, 0, 1875, 1500)$
- Basic Solution Value: 1080000
- Entering Variable: x_2

Simplex Example (cont.)

$$
\begin{array}{lrcl}
\max & -\frac{80}{3}s_1 + 80\boxed{x_2} + 1080000 & & \\
\text{s.t.} & \longrightarrow x_1 = & 4500 - & \frac{1}{9}s_1 - \frac{1}{3}x_2 \\
& s_2 = & 1875 + & \frac{1}{12}s_1 - \frac{3}{4}x_2 \\
& s_3 = & 1500 + & \frac{1}{9}s_1 - \frac{2}{3}x_2 \\
& x_1, x_2, s_1, s_2, s_3 \geq & 0 &
\end{array}
$$

- Basic Solution: $(x_1, x_2, s_1, s_2, s_3) = (4500, 0, 0, 1875, 1500)$
- Basic Solution Value: 1080000
- Entering Variable: $\boxed{x_2}$
- First Constraint: x_2 can be at most $4500/(1/3) = 13500$

Simplex Example (cont.)

$$
\begin{array}{llrlrr}
\max & & -\frac{80}{3}s_1 + 80\, \boxed{x_2} + 1080000 & & & \\
\text{s.t.} & & x_1 = & 4500 - & \frac{1}{9}s_1 - & \frac{1}{3}x_2 \\
& & \longrightarrow s_2 = & 1875 + & \frac{1}{12}s_1 - & \frac{3}{4}x_2 \\
& & s_3 = & 1500 + & \frac{1}{9}s_1 - & \frac{2}{3}x_2 \\
& & x_1, x_2, s_1, s_2, s_3 \geq & 0 & &
\end{array}
$$

- Basic Solution: $(x_1, x_2, s_1, s_2, s_3) = (4500, 0, 0, 1875, 1500)$
- Basic Solution Value: 1080000
- Entering Variable: $\boxed{x_2}$
- First Constraint: x_2 can be at most $4500/(1/3) = 13500$
- Second Constraint: x_2 can be at most $1875/(3/4) = 2500$

Simplex Example (cont.)

$$
\begin{array}{lrcl}
\max & & & -\frac{80}{3}s_1 + 80\boxed{x_2} + 1080000 \\
\text{s.t.} & x_1 & = & 4500 - \frac{1}{9}s_1 - \frac{1}{3}x_2 \\
 & s_2 & = & 1875 + \frac{1}{12}s_1 - \frac{3}{4}x_2 \\
 & \longrightarrow s_3 & = & 1500 + \frac{1}{9}s_1 - \frac{2}{3}x_2 \\
 & x_1, x_2, s_1, s_2, s_3 & \geq & 0
\end{array}
$$

- Basic Solution: $(x_1, x_2, s_1, s_2, s_3) = (4500, 0, 0, 1875, 1500)$
- Basic Solution Value: 1080000
- Entering Variable: $\boxed{x_2}$
- First Constraint: x_2 can be at most $4500/(1/3) = 13500$
- Second Constraint: x_2 can be at most $1875/(3/4) = 2500$
- Third Constraint: x_2 can be at most $(1500)/(2/3) = 2250$

Simplex Example (cont.)

$$
\begin{array}{lrcl}
\max & & & -\frac{80}{3}s_1 + 80\,x_2 + 1080000 \\
\text{s.t.} & x_1 & = & 4500 - \frac{1}{9}s_1 - \frac{1}{3}x_2 \\
 & s_2 & = & 1875 + \frac{1}{12}s_1 - \frac{3}{4}x_2 \\
 & s_3 & = & 1500 + \frac{1}{9}s_1 - \frac{2}{3}x_2 \\
 & x_1, x_2, s_1, s_2, s_3 & \geq & 0
\end{array}
$$

- Basic Solution: $(x_1, x_2, s_1, s_2, s_3) = (4500, 0, 0, 1875, 1500)$
- Basic Solution Value: 1080000
- Entering Variable: x_2
- First Constraint: x_2 can be at most $4500/(1/3) = 13500$
- Second Constraint: x_2 can be at most $1875/(3/4) = 2500$
- Third Constraint: x_2 can be at most $(1500)/(2/3) = 2250$
- Leaving Variable: s_3

Simplex Example (cont.)

$$
\begin{array}{llrcl}
\max & & -\frac{80}{3}s_1 + 80\,x_2 + 1080000 \\
\text{s.t.} & x_1 = & 4500 - & \frac{1}{9}s_1 - & \frac{1}{3}x_2 \\
& s_2 = & 1875 + & \frac{1}{12}s_1 - & \frac{3}{4}x_2 \\
& s_3 = & 1500 + & \frac{1}{9}s_1 - & \frac{2}{3}x_2 \\
& x_1, x_2, s_1, s_2, s_3 \geq & 0 & &
\end{array}
$$

- Basic Solution: $(x_1, x_2, s_1, s_2, s_3) = (4500, 0, 0, 1875, 1500)$
- Basic Solution Value: 1080000
- Entering Variable: x_2
- First Constraint: x_2 can be at most $4500/(1/3) = 13500$
- Second Constraint: x_2 can be at most $1875/(3/4) = 2500$
- Third Constraint: x_2 can be at most $(1500)/(2/3) = 2250$
- Leaving Variable: s_3

After pivoting we get ...

Simplex Example (cont.)

$$
\begin{array}{llrl}
\max & & -\frac{40}{3}s_1 - 120s_3 + 1260000 & \\
\text{s.t.} & x_1 = & 3750 - & \frac{1}{6}s_1 + \frac{1}{2}s_3 \\
& s_2 = & \frac{375}{2} - & \frac{1}{24}s_1 + \frac{9}{8}s_3 \\
& x_2 = & 2250 + & \frac{1}{6}s_1 - \frac{3}{2}s_3 \\
& x_1, x_2, s_1, s_2, s_3 \geq & 0 &
\end{array}
$$

Simplex Example (cont.)

$$
\begin{array}{lrcl}
\max & -\frac{40}{3}s_1 - 120s_3 + 1260000 & & \\
\text{s.t.} & x_1 = & 3750 - & \frac{1}{6}s_1 + \frac{1}{2}s_3 \\
 & s_2 = & \frac{375}{2} - & \frac{1}{24}s_1 + \frac{9}{8}s_3 \\
 & x_2 = & 2250 + & \frac{1}{6}s_1 - \frac{3}{2}s_3 \\
 & x_1, x_2, s_1, s_2, s_3 \geq & 0 &
\end{array}
$$

- Basic Solution: $(x_1, x_2, s_1, s_2, s_3) = (3750, 2250, 0, 375/2, 0)$

Simplex Example (cont.)

$$
\begin{array}{lrrrl}
\max & & -\frac{40}{3}s_1 - 120s_3 + 1260000 & & \\
\text{s.t.} & x_1 = & 3750 - & \frac{1}{6}s_1 + \frac{1}{2}s_3 \\
& s_2 = & \frac{375}{2} - & \frac{1}{24}s_1 + \frac{9}{8}s_3 \\
& x_2 = & 2250 + & \frac{1}{6}s_1 - \frac{3}{2}s_3 \\
& x_1, x_2, s_1, s_2, s_3 \geq & 0 & &
\end{array}
$$

- Basic Solution: $(x_1, x_2, s_1, s_2, s_3) = (3750, 2250, 0, 375/2, 0)$
- Basic Solution Value: 1260000

Simplex Example (cont.)

$$
\begin{array}{lrcl}
\max & -\frac{40}{3}s_1 - 120s_3 + 1260000 & & \\
\text{s.t.} & x_1 = & 3750 - & \frac{1}{6}s_1 + \frac{1}{2}s_3 \\
& s_2 = & \frac{375}{2} - & \frac{1}{24}s_1 + \frac{9}{8}s_3 \\
& x_2 = & 2250 + & \frac{1}{6}s_1 - \frac{3}{2}s_3 \\
& x_1, x_2, s_1, s_2, s_3 \geq & 0 &
\end{array}
$$

- Basic Solution: $(x_1, x_2, s_1, s_2, s_3) = (3750, 2250, 0, 375/2, 0)$
- Basic Solution Value: 1260000
- We stop now because we cannot increase the objective function anymore.

Simplex Example (cont.)

$$
\begin{array}{llcrl}
\max & & & -\frac{40}{3}s_1 - 120s_3 + 1260000 & \\
\text{s.t.} & x_1 & = & 3750 - & \frac{1}{6}s_1 + \frac{1}{2}s_3 \\
& s_2 & = & \frac{375}{2} - & \frac{1}{24}s_1 + \frac{9}{8}s_3 \\
& x_2 & = & 2250 + & \frac{1}{6}s_1 - \frac{3}{2}s_3 \\
& x_1, x_2, s_1, s_2, s_3 & \geq & 0 &
\end{array}
$$

- Basic Solution: $(x_1, x_2, s_1, s_2, s_3) = (3750, 2250, 0, 375/2, 0)$
- Basic Solution Value: 1260000
- We stop now because we cannot increase the objective function anymore.
- It turns out that this is the optimal solution!

Simplex Example (cont.)

$$
\begin{array}{lrcll}
\max & & & -\frac{40}{3}s_1 - 120s_3 + 1260000 & \\
\text{s.t.} & x_1 & = & 3750 - & \frac{1}{6}s_1 + \frac{1}{2}s_3 \\
 & s_2 & = & \frac{375}{2} - & \frac{1}{24}s_1 + \frac{9}{8}s_3 \\
 & x_2 & = & 2250 + & \frac{1}{6}s_1 - \frac{3}{2}s_3 \\
 & x_1, x_2, s_1, s_2, s_3 & \geq & 0 &
\end{array}
$$

- Basic Solution: $(x_1, x_2, s_1, s_2, s_3) = (3750, 2250, 0, 375/2, 0)$
- Basic Solution Value: 1260000
- We stop now because we cannot increase the objective function anymore.
- It turns out that this is the optimal solution!
- We should plant $x_1 = 3750$ acres of corn and $x_2 = 2250$ acres of soybeans yielding a profit of $1,260,000.

1. Transform the linear program into a slack form where the basic solution is feasible.

Simplex Algorithm Overview

1. Transform the linear program into a slack form where the basic solution is feasible.
2. Find a nonbasic variable x_e, called the entering variable, in the objective function with positive coefficient.

Simplex Algorithm Overview

1. Transform the linear program into a slack form where the basic solution is feasible.
2. Find a nonbasic variable x_e, called the entering variable, in the objective function with positive coefficient.
3. Find the basic variable x_l, called the leaving variable, associated with the most restrictive constraint on x_e.

Simplex Algorithm Overview

1. Transform the linear program into a slack form where the basic solution is feasible.
2. Find a nonbasic variable x_e, called the entering variable, in the objective function with positive coefficient.
3. Find the basic variable x_l, called the leaving variable, associated with the most restrictive constraint on x_e.
4. Transform the linear program into an equivalent one by executing a pivot

Simplex Algorithm Overview

1. Transform the linear program into a slack form where the basic solution is feasible.
2. Find a nonbasic variable x_e, called the entering variable, in the objective function with positive coefficient.
3. Find the basic variable x_l, called the leaving variable, associated with the most restrictive constraint on x_e.
4. Transform the linear program into an equivalent one by executing a pivot
5. Go back to Step 2 and if an entering variable cannot be found, stop.

Simplex Algorithm Overview

1. Transform the linear program into a slack form where the basic solution is feasible.
2. Find a nonbasic variable x_e, called the entering variable, in the objective function with positive coefficient.
3. Find the basic variable x_l, called the leaving variable, associated with the most restrictive constraint on x_e.
4. Transform the linear program into an equivalent one by executing a pivot
5. Go back to Step 2 and if an entering variable cannot be found, stop.
6. The basic solution is optimal so return it.

Simplex Algorithm Technical Issues

- How do you transform a linear program into a slack form where the basic solution is feasible?

Simplex Algorithm Technical Issues

- How do you transform a linear program into a slack form where the basic solution is feasible?
- What is the "most restrictive constraint"?

Simplex Algorithm Technical Issues

- How do you transform a linear program into a slack form where the basic solution is feasible?
- What is the "most restrictive constraint"?
- Can the simplex algorithm run forever?

Our Implementation Goals

- Linear programming can be extremely technical

Our Implementation Goals

- Linear programming can be extremely technical
- Easy-to-use interface to build linear programs

Our Implementation Goals

- Linear programming can be extremely technical
- Easy-to-use interface to build linear programs
- Abstract away technical details of doing tedious transformations

Our Implementation Goals

- Linear programming can be extremely technical
- Easy-to-use interface to build linear programs
- Abstract away technical details of doing tedious transformations
- Use type-system to force users to use interface correctly

Our Implementation Goals

- Linear programming can be extremely technical
- Easy-to-use interface to build linear programs
- Abstract away technical details of doing tedious transformations
- Use type-system to force users to use interface correctly
- Create test cases to validate our implementation

Our Implementation Usage Example

Creating a linear program:

```
LinearProgram p = new LinearProgram();
Variable corn = p.registerNonnegativeVariable("corn");
Variable soybeans = p.registerNonnegativeVariable("soybeans");
```

Our Implementation Usage Example

Adding constraints:

```
p.addConstraint(new Constraint(
        new ArrayList<>(Arrays.asList(corn, soybeans)),
        new ArrayList<>(Arrays.asList(9.0, 3.0)),
        Relation.LEQ,
        40500
));
p.addConstraint(new Constraint(
        new ArrayList<>(Arrays.asList(corn, soybeans)),
        new ArrayList<>(Arrays.asList(3.0/4.0, 1.0)),
        Relation.LEQ,
        5250
));
p.addConstraint(new Constraint(
        new ArrayList<>(Arrays.asList(corn, soybeans)),
        new ArrayList<>(Arrays.asList(1.0, 1.0)),
        Relation.LEQ,
        6000
));
```

Our Implementation Usage Example

Adding objective function:

```
p.setObjective(new ObjectiveFunction(
        ObjectiveGoal.MAXIMIZE,
        new ArrayList<>(Arrays.asList(corn, soybeans)),
        new ArrayList<>(Arrays.asList(240.0, 160.0))
));
```

Our Implementation Usage Example

Getting the solution:

```
System.out.println("Optimal profit: " + p.getObjectiveValue());
System.out.println("Corn: " + p.evaluateVariable(corn));
System.out.println("Soybeans: " + p.evaluateVariable(soybeans));
```

Our Implementation Pipeline

1. Obtain linear program $\mathcal{L}$ made by the user through our simple interface

Our Implementation Pipeline

1. Obtain linear program $\mathcal{L}$ made by the user through our simple interface
2. Compile $\mathcal{L}$ into an equivalent linear program $\mathcal{L}'$ in standard form

Our Implementation Pipeline

1. Obtain linear program $\mathcal{L}$ made by the user through our simple interface
2. Compile $\mathcal{L}$ into an equivalent linear program $\mathcal{L}'$ in standard form
3. Solve $\mathcal{L}'$ using the simplex algorithm

Our Implementation Pipeline

1. Obtain linear program $\mathcal{L}$ made by the user through our simple interface
2. Compile $\mathcal{L}$ into an equivalent linear program $\mathcal{L}'$ in standard form
3. Solve $\mathcal{L}'$ using the simplex algorithm
4. Convert the answer found for $\mathcal{L}'$ back into an answer in $\mathcal{L}$

Validation Methodology

- Used exercises from a textbook[1] and checked with the answers in the back of the book

[1] Mathematical Applications for the Management, Life, and Social Sciences (12th edition) by Harshbarger and Reynolds

Validation Methodology

- Used exercises from a textbook[1] and checked with the answers in the back of the book
- Used problems from the section on solving systems of difference constraints in CLRS

[1] Mathematical Applications for the Management, Life, and Social Sciences (12th edition) by Harshbarger and Reynolds

Validation Methodology

- Used exercises from a textbook[1] and checked with the answers in the back of the book
- Used problems from the section on solving systems of difference constraints in CLRS
- Recasted example maximum flow problems in CLRS into linear programs

[1] Mathematical Applications for the Management, Life, and Social Sciences (12th edition) by Harshbarger and Reynolds

Validation Methodology

- Used exercises from a textbook[1] and checked with the answers in the back of the book
- Used problems from the section on solving systems of difference constraints in CLRS
- Recasted example maximum flow problems in CLRS into linear programs
- Created stress tester that randomly generates maximum flow problems and validated solution with Max-Flow Min-Cut Theorem

[1] Mathematical Applications for the Management, Life, and Social Sciences (12th edition) by Harshbarger and Reynolds

Maximum Flow Problem As A Linear Programming Problem

- Each edge (u, v) has positive capacity c_{uv}

Maximum Flow Problem As A Linear Programming Problem

- Each edge (u, v) has positive capacity c_{uv}
- Each edge (u, v) is assigned a flow f_{uv}: $0 \leq f_{uv} \leq c_{uv}$
- *Assume* $f_{uv} = 0$ *if* $(u, v) \notin E$

Maximum Flow Problem As A Linear Programming Problem

- Each edge (u, v) has positive capacity c_{uv}
- Each edge (u, v) is assigned a flow f_{uv}: $0 \leq f_{uv} \leq c_{uv}$
- *Assume* $f_{uv} = 0$ *if* $(u, v) \notin E$
- All non-source and non-sink vertices v must conserve flow: $\sum_{u \in V} f_{uv} = \sum_{w \in V} f_{vw}$

Maximum Flow Problem As A Linear Programming Problem

- Each edge (u, v) has positive capacity c_{uv}
- Each edge (u, v) is assigned a flow f_{uv}: $0 \leq f_{uv} \leq c_{uv}$
- *Assume* $f_{uv} = 0$ *if* $(u, v) \notin E$
- All non-source and non-sink vertices v must conserve flow: $\sum_{u \in V} f_{uv} = \sum_{w \in V} f_{vw}$
- Maximize total flow out of the source s: $\sum_{v \in V} f_{sv} - f_{vs}$

Maximum Flow Problem As A Linear Programming Problem

- Each edge (u, v) has positive capacity c_{uv}
- Each edge (u, v) is assigned a flow f_{uv}: $0 \leq f_{uv} \leq c_{uv}$
- *Assume* $f_{uv} = 0$ *if* $(u, v) \notin E$
- All non-source and non-sink vertices v must conserve flow: $\sum_{u \in V} f_{uv} = \sum_{w \in V} f_{vw}$
- Maximize total flow out of the source s: $\sum_{v \in V} f_{sv} - f_{vs}$
- All constraints and the objective function are linear relationships on the variables which means finding a maximum flow is reduced to solving a linear program!

Stress Testing Our Implementation

- Large linear programs may expose implementation issues

Stress Testing Our Implementation

- Large linear programs may expose implementation issues
- How do we generate large linear programs and also validate the answer?

Stress Testing Our Implementation

- Large linear programs may expose implementation issues
- How do we generate large linear programs and also validate the answer?
- Generate large randomized flow network parameterized by n and k (Figure 1)

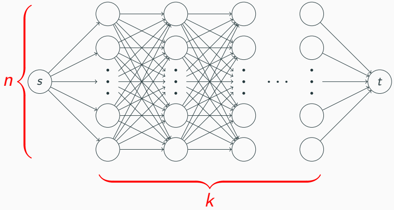

Figure 1: Stress test network architecture

Stress Testing Our Implementation

- Large linear programs may expose implementation issues
- How do we generate large linear programs and also validate the answer?
- Generate large randomized flow network parameterized by *n* and *k* (Figure 1)
- Max-Flow Min-Cut Theorem says that validating a maximum flow amounts to finding the non-existence of an augmenting path!

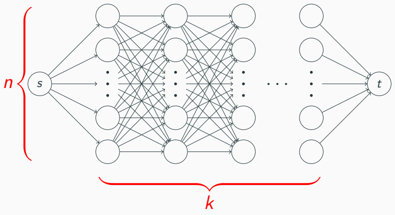

Figure 1: Stress test network architecture

Thanks!

Thanks for listening!

- Implementation Code: `https://github.com/EthanTheMaster/LinearProgrammingSimplexAlgorithm`