II. Linear Programming

Thomas Sauerwald

Easter 2020
Outline

Introduction

Formulating Problems as Linear Programs

Standard and Slack Forms

Simplex Algorithm

Finding an Initial Solution
Introduction

- linear programming is a powerful tool in optimisation
- inspired more sophisticated techniques such as quadratic optimisation, convex optimisation, integer programming and semi-definite programming
- we will later use the connection between linear and integer programming to tackle several problems (Vertex-Cover, Set-Cover, TSP, satisfiability)
What are Linear Programs?

Linear Programming (informal definition)

- maximize or minimize an objective, given limited resources and competing constraint
- constraints are specified as (in)equalities

Example: Political Advertising (from CLRS3)

- Imagine you are a politician trying to win an election
- Your district has three different types of areas: Urban, suburban and rural, each with, respectively, 100,000, 200,000 and 50,000 registered voters
- **Aim**: at least half of the registered voters in each of the three regions should vote for you
- **Possible Actions**: Advertise on one of the primary issues which are (i) building more roads, (ii) gun control, (iii) farm subsidies and (iv) a gasoline tax dedicated to improve public transit.
The effects of policies on voters. Each entry describes the number of thousands of voters who could be won (lost) over by spending $1,000 on advertising support of a policy on a particular issue.

### Possible Solution:
- $20,000 on advertising to building roads
- $0 on advertising to gun control
- $4,000 on advertising to farm subsidies
- $9,000 on advertising to a gasoline tax

**Total cost: $33,000**

What is the best possible strategy?
Towards a Linear Program

<table>
<thead>
<tr>
<th>policy</th>
<th>urban</th>
<th>suburban</th>
<th>rural</th>
</tr>
</thead>
<tbody>
<tr>
<td>build roads</td>
<td>−2</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>gun control</td>
<td>8</td>
<td>2</td>
<td>−5</td>
</tr>
<tr>
<td>farm subsidies</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>gasoline tax</td>
<td>10</td>
<td>0</td>
<td>−2</td>
</tr>
</tbody>
</table>

The effects of policies on voters. Each entry describes the number of thousands of voters who could be won (lost) over by spending $1,000 on advertising support of a policy on a particular issue.

- $x_1 =$ number of thousands of dollars spent on advertising on building roads
- $x_2 =$ number of thousands of dollars spent on advertising on gun control
- $x_3 =$ number of thousands of dollars spent on advertising on farm subsidies
- $x_4 =$ number of thousands of dollars spent on advertising on gasoline tax

Constraints:

- $−2x_1 + 8x_2 + 0x_3 + 10x_4 \geq 50$
- $5x_1 + 2x_2 + 0x_3 + 0x_4 \geq 100$
- $3x_1 − 5x_2 + 10x_3 − 2x_4 \geq 25$

Objective: Minimize $x_1 + x_2 + x_3 + x_4$
The Linear Program

Linear Program for the Advertising Problem

minimize \( x_1 + x_2 + x_3 + x_4 \)
subject to
\[
\begin{align*}
-2x_1 + 8x_2 + 0x_3 + 10x_4 & \geq 50 \\
5x_1 + 2x_2 + 0x_3 + 0x_4 & \geq 100 \\
3x_1 - 5x_2 + 10x_3 - 2x_4 & \geq 25 \\
x_1, x_2, x_3, x_4 & \geq 0
\end{align*}
\]

The solution of this linear program yields the optimal advertising strategy.

Formal Definition of Linear Program

- Given \( a_1, a_2, \ldots, a_n \) and a set of variables \( x_1, x_2, \ldots, x_n \), a linear function \( f \) is defined by
  \[
  f(x_1, x_2, \ldots, x_n) = a_1 x_1 + a_2 x_2 + \cdots + a_n x_n.
  \]
- Linear Equality: \( f(x_1, x_2, \ldots, x_n) = b \)
- Linear Inequality: \( f(x_1, x_2, \ldots, x_n) \geq b \)
- Linear-Programming Problem: either minimize or maximize a linear function subject to a set of linear constraints
A Small(er) Example

maximize \[ x_1 + x_2 \]
subject to
\[
\begin{align*}
4x_1 - x_2 & \leq 8 \\
2x_1 + x_2 & \leq 10 \\
5x_1 - 2x_2 & \geq -2 \\
x_1, x_2 & \geq 0
\end{align*}
\]

Any setting of \( x_1 \) and \( x_2 \) satisfying all constraints is a feasible solution.

Graphical Procedure: Move the line \( x_1 + x_2 = z \) as far up as possible.
A Small(er) Example

maximize \( x_1 + x_2 \)
subject to
\[
\begin{align*}
4x_1 - x_2 & \leq 8 \\
2x_1 + x_2 & \leq 10 \\
5x_1 - 2x_2 & \geq -2 \\
x_1, x_2 & \geq 0
\end{align*}
\]

Graphical Procedure: Move the line \( x_1 + x_2 = z \) as far up as possible.

While the same approach also works for higher-dimensions, we need to take a more systematic and algebraic procedure.
Outline

Introduction

Formulating Problems as Linear Programs

Standard and Slack Forms

Simplex Algorithm

Finding an Initial Solution
Shortest Paths

Single-Pair Shortest Path Problem

- **Given:** directed graph $G = (V, E)$ with edge weights $w: E \rightarrow \mathbb{R}$, pair of vertices $s, t \in V$
- **Goal:** Find a path of minimum weight from $s$ to $t$ in $G$

$p = (v_0 = s, v_1, \ldots, v_k = t)$ such that $w(p) = \sum_{i=1}^{k} w(v_{k-1}, v_k)$ is minimized.

Shortest Paths as LP

maximize $d_t$

subject to

- $d_v \leq d_u + w(u, v)$ for each edge $(u, v) \in E$,
- $d_s = 0$.

this is a maximization problem!

Recall: When BELLMAN-FORD terminates, all these inequalities are satisfied.

Solution $\bar{d}$ satisfies $\bar{d}_v = \min_{u: (u, v) \in E} \{\bar{d}_u + w(u, v)\}$
Maximum Flow

**Maximum Flow Problem**

- **Given:** directed graph $G = (V, E)$ with edge capacities $c : E \rightarrow \mathbb{R}^+$ (recall $c(u, v) = 0$ if $(u, v) \notin E$), pair of vertices $s, t \in V$
- **Goal:** Find a maximum flow $f : V \times V \rightarrow \mathbb{R}$ from $s$ to $t$ which satisfies the capacity constraints and flow conservation

### Maximum Flow as LP

Maximize

$$\sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$$

Subject to

- $f_{uv} \leq c(u, v)$ for each $u, v \in V$,
- $\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}$ for each $u \in V \setminus \{s, t\}$,
- $f_{uv} \geq 0$ for each $u, v \in V$. 

- Maximum Flow as LP

- $|f| = 19$
Minimum-Cost Flow

Given: directed graph \( G = (V, E) \) with capacities \( c : E \to \mathbb{R}^+ \), pair of vertices \( s, t \in V \), cost function \( a : E \to \mathbb{R}^+ \), flow demand of \( d \) units

Goal: Find a flow \( f : V \times V \to \mathbb{R} \) from \( s \) to \( t \) with \( |f| = d \) while minimising the total cost \( \sum_{(u,v) \in E} a(u, v)f_{uv} \) incurred by the flow.

Optimal Solution with total cost:
\[
\sum_{(u,v) \in E} a(u, v)f_{uv} = (2 \cdot 2) + (5 \cdot 2) + (3 \cdot 1) + (7 \cdot 1) + (1 \cdot 3) = 27
\]

Figure 29.3  (a) An example of a minimum-cost-flow problem. We denote the capacities by \( c \) and the costs by \( a \). Vertex \( s \) is the source and vertex \( t \) is the sink, and we wish to send 4 units of flow from \( s \) to \( t \).  (b) A solution to the minimum-cost flow problem in which 4 units of flow are sent from \( s \) to \( t \). For each edge, the flow and capacity are written as flow/capacity.
Minimum-Cost Flow as a LP

minimize \[ \sum_{(u,v) \in E} a(u, v) f_{uv} \]
subject to
- \[ f_{uv} \leq c(u, v) \quad \text{for each } u, v \in V, \]
- \[ \sum_{v \in V} f_{vu} - \sum_{v \in V} f_{uv} = 0 \quad \text{for each } u \in V \setminus \{s, t\}, \]
- \[ \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs} = d, \]
- \[ f_{uv} \geq 0 \quad \text{for each } u, v \in V. \]

Real power of Linear Programming comes from the ability to solve new problems!
Outline

Introduction

Formulating Problems as Linear Programs

Standard and Slack Forms

Simplex Algorithm

Finding an Initial Solution
Standard and Slack Forms

**Standard Form**

maximize \[ \sum_{j=1}^{n} c_j x_j \]  

subject to \[ \sum_{j=1}^{n} a_{ij} x_j \leq b_i \]  
for \( i = 1, 2, \ldots, m \) 

\[ x_j \geq 0 \]  
for \( j = 1, 2, \ldots, n \)

**Standard Form (Matrix-Vector-Notation)**

maximize \[ c^T x \]  
Inner product of two vectors

subject to \[ Ax \leq b \]  
Matrix-vector product

\[ x \geq 0 \]
Converting Linear Programs into Standard Form

Reasons for a LP not being in standard form:
1. The objective might be a minimization rather than maximization.
2. There might be variables without nonnegativity constraints.
3. There might be equality constraints.
4. There might be inequality constraints (with $\geq$ instead of $\leq$).

Goal: Convert linear program into an equivalent program which is in standard form

Equivalence: a correspondence (not necessarily a bijection) between solutions.
Reasons for a LP not being in standard form:

1. The objective might be a minimization rather than maximization.

\[
\begin{align*}
\text{minimize} & \quad -2x_1 + 3x_2 \\
\text{subject to} & \quad x_1 + x_2 = 7 \\
& \quad x_1 - 2x_2 \leq 4 \\
& \quad x_1 \geq 0
\end{align*}
\]

Negate objective function

\[
\begin{align*}
\text{maximize} & \quad 2x_1 - 3x_2 \\
\text{subject to} & \quad x_1 + x_2 = 7 \\
& \quad x_1 - 2x_2 \leq 4 \\
& \quad x_1 \geq 0
\end{align*}
\]
Converting into Standard Form (2/5)

Reasons for a LP not being in standard form:
2. There might be variables without nonnegativity constraints.

maximize 

subject to 

\[ \begin{align*}
2x_1 &- 3x_2 \\
\end{align*} \]

Replace \( x_2 \) by two non-negative variables \( x_2' \) and \( x_2'' \)

maximize 

subject to 

\[ \begin{align*}
2x_1 &- 3x_2' + 3x_2'' \\
\end{align*} \]

\[ \begin{align*}
x_1 + x_2 & = 7 \\
x_1 - 2x_2 & < 4 \\
x_1 & \geq 0 \\
\end{align*} \]

\[ \begin{align*}
x_1, x_2', x_2'' & \geq 0 \\
\end{align*} \]
Converting into Standard Form (3/5)

Reasons for a LP not being in standard form:
3. There might be equality constraints.

maximize \[ 2x_1 - 3x_2' + 3x_2'' \]
subject to
\[
\begin{align*}
x_1 + x_2' - x_2'' & = 7 \\
x_1 - 2x_2' + 2x_2'' & \leq 4 \\
x_1, x_2', x_2'' & \geq 0
\end{align*}
\]
Replace each equality by two inequalities.

maximize \[ 2x_1 - 3x_2' + 3x_2'' \]
subject to
\[
\begin{align*}
x_1 + x_2' - x_2'' & \leq 7 \\
x_1 + x_2' - x_2'' & \geq 7 \\
x_1 - 2x_2' + 2x_2'' & \leq 4 \\
x_1, x_2', x_2'' & \geq 0
\end{align*}
\]
Reasons for a LP not being in standard form:

4. There might be inequality constraints (with $\geq$ instead of $\leq$).

maximize $2x_1 - 3x'_2 + 3x''_2$
subject to

\[
\begin{align*}
x_1 + x'_2 - x''_2 & \leq 7 \\
x_1 + x'_2 - x''_2 & \geq 7 \\
x_1 - 2x'_2 + 2x''_2 & \leq 4 \\
x_1, x'_2, x''_2 & \geq 0
\end{align*}
\]

Negate respective inequalities.

maximize $2x_1 - 3x'_2 + 3x''_2$
subject to

\[
\begin{align*}
-x_1 + x'_2 - x''_2 & \leq 7 \\
-x_1 - x'_2 + x''_2 & \leq -7 \\
x_1 - 2x'_2 + 2x''_2 & \leq 4 \\
x_1, x'_2, x''_2 & \geq 0
\end{align*}
\]
Converting into Standard Form (5/5)

Rename variable names (for consistency).

\[
\begin{align*}
\text{maximize } & \quad 2x_1 - 3x_2 + 3x_3 \\
\text{subject to } & \quad x_1 + x_2 - x_3 \leq 7 \\
& \quad -x_1 - x_2 + x_3 \leq -7 \\
& \quad x_1 - 2x_2 + 2x_3 \leq 4 \\
& \quad x_1, x_2, x_3 \geq 0
\end{align*}
\]

It is always possible to convert a linear program into standard form.
Converting Standard Form into Slack Form (1/3)

**Goal:** Convert standard form into slack form, where all constraints except for the non-negativity constraints are equalities.

For the simplex algorithm, it is more convenient to work with equality constraints.

Introducing Slack Variables

- Let $\sum_{j=1}^{n} a_{ij}x_j \leq b_i$ be an inequality constraint
- Introduce a slack variable $s$ by

$$s = b_i - \sum_{j=1}^{n} a_{ij}x_j$$

$s \geq 0$.

- Denote slack variable of the $i$th inequality by $x_{n+i}$

$s$ measures the slack between the two sides of the inequality.
Converting Standard Form into Slack Form (2/3)

maximize 2x₁ − 3x₂ + 3x₃
subject to

\[
\begin{align*}
x_1 + x_2 - x_3 & \leq 7 \\
-x_1 - x_2 + x_3 & \leq -7 \\
x_1 - 2x_2 + 2x_3 & \leq 4 \\
x_1, x_2, x_3 & \geq 0
\end{align*}
\]

Introduce slack variables

maximize 2x₁ − 3x₂ + 3x₃
subject to

\[
\begin{align*}
x_4 & = 7 - x_1 - x_2 + x_3 \\
x_5 & = -7 + x_1 + x_2 - x_3 \\
x_6 & = 4 - x_1 + 2x_2 - 2x_3 \\
x_1, x_2, x_3, x_4, x_5, x_6 & \geq 0
\end{align*}
\]
maximize \[ 2x_1 - 3x_2 + 3x_3 \]
subject to
\[
\begin{align*}
x_4 &= 7 - x_1 - x_2 + x_3 \\
x_5 &= -7 + x_1 + x_2 - x_3 \\
x_6 &= 4 - x_1 + 2x_2 - 2x_3
\end{align*}
\]
\[ x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \]

Use variable \( z \) to denote objective function and omit the nonnegativity constraints.

This is called \textit{slack form}. 

II. Linear Programming

Standard and Slack Forms 24
Basic and Non-Basic Variables

\[ z = 2x_1 - 3x_2 + 3x_3 \]
\[ x_4 = 7 - x_1 - x_2 + x_3 \]
\[ x_5 = -7 + x_1 + x_2 - x_3 \]
\[ x_6 = 4 - x_1 + 2x_2 - 2x_3 \]

**Basic Variables:** \( B = \{4, 5, 6\} \)

**Non-Basic Variables:** \( N = \{1, 2, 3\} \)

Slack Form (Formal Definition)

Slack form is given by a tuple \((N, B, A, b, c, v)\) so that

\[ z = v + \sum_{j \in N} c_j x_j \]
\[ x_i = b_i - \sum_{j \in N} a_{ij} x_j \quad \text{for } i \in B, \]

and all variables are non-negative.

Variables/Coefficients on the right hand side are indexed by \( B \) and \( N \).
Slack Form (Example)

\[ z = 28 - \frac{x_3}{6} - \frac{x_5}{6} - \frac{2x_6}{3} \]
\[ x_1 = 8 + \frac{x_3}{6} + \frac{x_5}{6} - \frac{x_6}{3} \]
\[ x_2 = 4 - \frac{8x_3}{3} - \frac{2x_5}{3} + \frac{x_6}{3} \]
\[ x_4 = 18 - \frac{x_3}{2} + \frac{x_5}{2} \]

Slack Form Notation

- \( B = \{1, 2, 4\} \), \( N = \{3, 5, 6\} \)
- \( A = \begin{pmatrix} a_{13} & a_{15} & a_{16} \\ a_{23} & a_{25} & a_{26} \\ a_{43} & a_{45} & a_{46} \end{pmatrix} = \begin{pmatrix} -1/6 & -1/6 & 1/3 \\ 8/3 & 2/3 & -1/3 \\ 1/2 & -1/2 & 0 \end{pmatrix} \)
- \( b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 8 \\ 4 \\ 18 \end{pmatrix} \), \( c = \begin{pmatrix} c_3 \\ c_5 \\ c_6 \end{pmatrix} = \begin{pmatrix} -1/6 \\ -1/6 \\ -2/3 \end{pmatrix} \)
- \( v = 28 \)
The Structure of Optimal Solutions

**Definition**

A point $x$ is a **vertex** if it cannot be represented as a strict convex combination of two other points in the feasible set.

**Theorem**

If the slack form has an optimal solution, **one of them** occurs at a vertex.

Proof Sketch (informal and non-examinable):

- Rewrite LP s.t. $Ax = b$. Let $x$ be optimal but not a vertex
  $\Rightarrow \exists$ vector $d$ s.t. $x - d$ and $x + d$ are feasible
- Since $A(x + d) = b$ and $Ax = b \Rightarrow Ad = 0$
- W.l.o.g. assume $c^T d \geq 0$ (otherwise replace $d$ by $-d$)
- Consider $x + \lambda d$ as a function of $\lambda \geq 0$

**Case 1:** There exists $j$ with $d_j < 0$

- Increase $\lambda$ from 0 to $\lambda'$ until a new entry of $x + \lambda d$ becomes zero
- $x + \lambda' d$ feasible, since $A(x + \lambda' d) = Ax = b$ and $x + \lambda' d \geq 0$
- $c^T (x + \lambda' d) = c^T x + c^T \lambda' d \geq c^T x$
The Structure of Optimal Solutions

**Definition**

A point \( x \) is a **vertex** if it cannot be represented as a strict convex combination of two other points in the feasible set.

The set of feasible solutions is a convex set.

**Theorem**

If the slack form has an optimal solution, one of them occurs at a vertex.

Proof Sketch (informal and non-examinable):

- Rewrite LP s.t. \( Ax = b \). Let \( x \) be optimal but not a vertex
  \( \Rightarrow \exists \) vector \( d \) s.t. \( x - d \) and \( x + d \) are feasible
- Since \( A(x + d) = b \) and \( Ax = b \Rightarrow Ad = 0 \)
- W.l.o.g. assume \( c^T d \geq 0 \) (otherwise replace \( d \) by \(-d\))
- Consider \( x + \lambda d \) as a function of \( \lambda \geq 0 \)

**Case 2:** For all \( j \), \( d_j \geq 0 \)

- \( x + \lambda d \) is feasible for all \( \lambda \geq 0 \): \( A(x + \lambda d) = b \) and
  \( x + \lambda d \geq x \geq 0 \)
- If \( \lambda \to \infty \), then \( c^T (x + \lambda d) \to \infty \)
  \( \Rightarrow \) This contradicts the assumption that there exists an optimal solution.
Outline

Introduction

Formulating Problems as Linear Programs

Standard and Slack Forms

Simplex Algorithm

Finding an Initial Solution
Simplex Algorithm: Introduction

- classical method for solving linear programs (Dantzig, 1947)
- usually fast in practice although worst-case runtime not polynomial
- iterative procedure somewhat similar to Gaussian elimination

Basic Idea:
- Each iteration corresponds to a “basic solution” of the slack form
- All non-basic variables are 0, and the basic variables are determined from the equality constraints
- Each iteration converts one slack form into an equivalent one while the objective value will not decrease
- Conversion (“pivoting”) is achieved by switching the roles of one basic and one non-basic variable

In that sense, it is a greedy algorithm.
Extended Example: Conversion into Slack Form

maximize \[ 3x_1 \quad + \quad x_2 \quad + \quad 2x_3 \]
subject to
\[ x_1 \quad + \quad x_2 \quad + \quad 3x_3 \quad \leq \quad 30 \]
\[ 2x_1 \quad + \quad 2x_2 \quad + \quad 5x_3 \quad \leq \quad 24 \]
\[ 4x_1 \quad + \quad x_2 \quad + \quad 2x_3 \quad \leq \quad 36 \]
\[ x_1, x_2, x_3 \quad \geq \quad 0 \]

Conversion into slack form

\[ z \quad = \quad 3x_1 \quad + \quad x_2 \quad + \quad 2x_3 \]
\[ x_4 \quad = \quad 30 \quad - \quad x_1 \quad - \quad x_2 \quad - \quad 3x_3 \]
\[ x_5 \quad = \quad 24 \quad - \quad 2x_1 \quad - \quad 2x_2 \quad - \quad 5x_3 \]
\[ x_6 \quad = \quad 36 \quad - \quad 4x_1 \quad - \quad x_2 \quad - \quad 2x_3 \]
Extended Example: Iteration 1

\[ z = 3x_1 + x_2 + 2x_3 \]
\[ x_4 = 30 - x_1 - x_2 - 3x_3 \]
\[ x_5 = 24 - 2x_1 - 2x_2 - 5x_3 \]
\[ x_6 = 36 - 4x_1 - x_2 - 2x_3 \]

Basic solution: \((x_1, x_2, \ldots, x_6) = (0, 0, 0, 30, 24, 36)\)

This basic solution is **feasible**

Objective value is 0.
Extended Example: Iteration 1

Increasing the value of $x_1$ would increase the objective value.

\[
\begin{align*}
  z &= 3x_1 + x_2 + 2x_3 \\
  x_4 &= 30 - x_1 - x_2 - 3x_3 \\
  x_5 &= 24 - 2x_1 - 2x_2 - 5x_3 \\
  x_6 &= 36 - 4x_1 - x_2 - 2x_3
\end{align*}
\]

The third constraint is the tightest and limits how much we can increase $x_1$.

Switch roles of $x_1$ and $x_6$:

- Solving for $x_1$ yields:
  \[
  x_1 = 9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4}.
  \]
- Substitute this into $x_1$ in the other three equations.
Extended Example: Iteration 2

Increasing the value of $x_3$ would increase the objective value.

$$z = 27 + \frac{x_2}{4} + \frac{x_3}{2} - \frac{3x_6}{4}$$

$$x_1 = 9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4}$$

$$x_4 = 21 - \frac{3x_2}{4} - \frac{5x_3}{2} + \frac{x_6}{4}$$

$$x_5 = 6 - \frac{3x_2}{2} - 4x_3 + \frac{x_6}{2}$$

Basic solution: $(x_1, x_2, \ldots, x_6) = (9, 0, 0, 21, 6, 0)$ with objective value 27
Extended Example: Iteration 2

\[ z = 27 + \frac{x_2}{4} + \frac{x_3}{2} - \frac{3x_6}{4} \]
\[ x_1 = 9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4} \]
\[ x_4 = 21 - \frac{3x_2}{4} - \frac{5x_3}{2} + \frac{x_6}{4} \]
\[ x_5 = 6 - \frac{3x_2}{2} - 4x_3 + \frac{x_6}{2} \]

The third constraint is the tightest and limits how much we can increase \( x_3 \).

Switch roles of \( x_3 \) and \( x_5 \):

- Solving for \( x_3 \) yields:

\[ x_3 = \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} - \frac{x_6}{8} \]

- Substitute this into \( x_3 \) in the other three equations
Extended Example: Iteration 3

Increasing the value of $x_2$ would increase the objective value.

$$z = \frac{111}{4} + \frac{x_2}{16} - \frac{x_5}{8} - \frac{11x_6}{16}$$

$$x_1 = \frac{33}{4} - \frac{x_2}{16} + \frac{x_5}{8} - \frac{5x_6}{16}$$

$$x_3 = \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} + \frac{x_6}{8}$$

$$x_4 = \frac{69}{4} + \frac{3x_2}{16} + \frac{5x_5}{8} - \frac{x_6}{16}$$

Basic solution: $(x_1, x_2, \ldots, x_6) = (\frac{33}{4}, 0, \frac{3}{2}, \frac{69}{4}, 0, 0)$ with objective value $\frac{111}{4} = 27.75$
Extended Example: Iteration 3

\[ z = \frac{111}{4} + \frac{x_2}{16} - \frac{x_5}{8} - \frac{11x_6}{16} \]

\[ x_1 = \frac{33}{4} - \frac{x_2}{16} + \frac{x_5}{8} - \frac{5x_6}{16} \]

\[ x_3 = \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} + \frac{x_6}{8} \]

\[ x_4 = \frac{69}{4} + \frac{3x_2}{16} + \frac{5x_5}{8} - \frac{x_6}{16} \]

The second constraint is the tightest and limits how much we can increase \( x_{2,6} \).

**Switch roles of \( x_2 \) and \( x_3 \):**

- Solving for \( x_2 \) yields:
  \[ x_2 = 4 - \frac{8x_3}{3} - \frac{2x_5}{3} + \frac{x_6}{3} \]

- Substitute this into \( x_2 \) in the other three equations
Extended Example: Iteration 4

All coefficients are negative, and hence this basic solution is optimal!

\[
\begin{align*}
    z &= 28 - \frac{x_3}{6} - \frac{x_5}{6} - \frac{2x_6}{3} \\
    x_1 &= 8 + \frac{x_3}{6} + \frac{x_5}{6} - \frac{x_6}{3} \\
    x_2 &= 4 - \frac{8x_3}{3} - \frac{2x_5}{3} + \frac{x_6}{3} \\
    x_4 &= 18 - \frac{x_3}{2} + \frac{x_5}{2}
\end{align*}
\]

Basic solution: \((\overline{x_1}, \overline{x_2}, \ldots, \overline{x_6}) = (8, 4, 0, 18, 0, 0)\) with objective value 28
Extended Example: Visualization of SIMPLEX

Exercise: How many basic solutions (including non-feasible ones) are there?

II. Linear Programming
Simplex Algorithm
32
Extended Example: Alternative Runs (1/2)

\[ z = 3x_1 + x_2 + 2x_3 \]
\[ x_4 = 30 - x_1 - x_2 - 3x_3 \]
\[ x_5 = 24 - 2x_1 - 2x_2 - 5x_3 \]
\[ x_6 = 36 - 4x_1 - x_2 - 2x_3 \]

Switch roles of \( x_2 \) and \( x_5 \)

\[ z = 12 + 2x_1 - \frac{x_3}{2} - \frac{x_5}{2} \]
\[ x_2 = 12 - x_1 - \frac{5x_3}{2} - \frac{x_5}{2} \]
\[ x_4 = 18 - x_2 - \frac{x_3}{2} + \frac{x_5}{2} \]
\[ x_6 = 24 - 3x_1 + \frac{x_3}{2} + \frac{x_5}{2} \]

Switch roles of \( x_1 \) and \( x_6 \)

\[ z = 28 - \frac{x_3}{6} - \frac{x_5}{6} - \frac{2x_6}{3} \]
\[ x_1 = 8 + \frac{x_3}{6} + \frac{x_5}{6} - \frac{x_6}{3} \]
\[ x_2 = 4 - \frac{8x_3}{3} - \frac{2x_5}{3} + \frac{x_6}{3} \]
\[ x_4 = 18 - \frac{x_3}{2} + \frac{x_5}{2} \]
Extended Example: Alternative Runs (2/2)

\[ z = 3x_1 + x_2 + 2x_3 \]
\[ x_4 = 30 - x_1 - x_2 - 3x_3 \]
\[ x_5 = 24 - 2x_1 - 2x_2 - 5x_3 \]
\[ x_6 = 36 - 4x_1 - x_2 - 2x_3 \]

\[ z = 48 + \frac{11x_1}{5} + \frac{x_2}{5} - \frac{2x_5}{5} \]
\[ x_4 = 78 + \frac{x_1}{5} + \frac{x_2}{5} + \frac{3x_5}{5} \]
\[ x_3 = \frac{24}{5} - \frac{2x_1}{5} - \frac{2x_2}{5} - \frac{x_5}{5} \]
\[ x_6 = \frac{132}{5} - \frac{16x_1}{5} - \frac{x_2}{5} + \frac{2x_3}{5} \]

Switch roles of \( x_3 \) and \( x_5 \)

Switch roles of \( x_1 \) and \( x_6 \)

Switch roles of \( x_2 \) and \( x_3 \)

\[ z = \frac{111}{4} + \frac{x_2}{16} - \frac{x_5}{8} - \frac{11x_6}{16} \]
\[ x_1 = \frac{33}{4} - \frac{x_2}{16} + \frac{x_5}{8} - \frac{5x_6}{16} \]
\[ x_3 = \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} + \frac{x_6}{8} \]
\[ x_4 = \frac{69}{4} + \frac{3x_2}{16} + \frac{5x_5}{8} - \frac{x_6}{16} \]

\[ z = 28 - \frac{x_3}{6} - \frac{x_5}{6} - \frac{2x_6}{3} \]
\[ x_1 = 8 + \frac{x_3}{6} + \frac{x_5}{6} - \frac{x_6}{3} \]
\[ x_2 = 4 - \frac{8x_3}{3} - \frac{2x_5}{3} + \frac{x_6}{3} \]
\[ x_4 = 18 - \frac{x_3}{2} + \frac{x_5}{2} \]
The Pivot Step Formally

\[ \text{PIVOT}(N, B, A, b, c, v, l, e) \]

1. // Compute the coefficients of the equation for new basic variable \( x_e \).
2. let \( \hat{A} \) be a new \( m \times n \) matrix
3. \( \hat{b}_e = b_l / a_{le} \)
4. for each \( j \in N - \{e\} \)
5. \( \hat{a}_{ej} = a_{lj} / a_{le} \)
6. \( \hat{a}_{el} = 1 / a_{le} \)
7. // Compute the coefficients of the remaining constraints.
8. for each \( i \in B - \{l\} \)
9. \( \hat{b}_i = b_i - a_{ie} \hat{b}_e \)
10. for each \( j \in N - \{e\} \)
11. \( \hat{a}_{ij} = a_{ij} - a_{ie} \hat{a}_{ej} \)
12. \( \hat{a}_{il} = -a_{ie} \hat{a}_{el} \)
13. // Compute the objective function.
14. \( \hat{v} = v + c_e \hat{b}_e \)
15. for each \( j \in N - \{e\} \)
16. \( \hat{c}_j = c_j - c_e \hat{a}_{ej} \)
17. \( \hat{c}_l = -c_e \hat{a}_{el} \)
18. // Compute new sets of basic and nonbasic variables.
19. \( \hat{N} = N - \{e\} \cup \{l\} \)
20. \( \hat{B} = B - \{l\} \cup \{e\} \)
21. return \( (\hat{N}, \hat{B}, \hat{A}, \hat{b}, \hat{c}, \hat{v}) \)

Rewrite “tight” equation for entering variable \( x_e \).
Substituting \( x_e \) into other equations.
Substituting \( x_e \) into objective function.
Update non-basic and basic variables.
Consider a call to Pivot\((N, B, A, b, c, v, l, e)\) in which \(a_{le} \neq 0\). Let the values returned from the call be \((\hat{N}, \hat{B}, \hat{A}, \hat{b}, \hat{c}, \hat{v})\), and let \(\bar{x}\) denote the basic solution after the call. Then

1. \(\bar{x}_j = 0\) for each \(j \in \hat{N}\).
2. \(\bar{x}_e = b_l/a_{le}\).
3. \(\bar{x}_i = b_i - a_{ie}\hat{b}_e\) for each \(i \in \hat{B} \setminus \{e\}\).

Proof:
1. Holds since the basic solution always sets all non-basic variables to zero.
2. When we set each non-basic variable to 0 in a constraint
   \[ x_i = \hat{b}_i - \sum_{j \in \hat{N}} \hat{a}_{ij} x_j, \]
   we have \(\bar{x}_i = \hat{b}_i\) for each \(i \in \hat{B}\). Hence \(\bar{x}_e = \hat{b}_e = b_l/a_{le}\).
3. After substituting into the other constraints, we have
   \[ \bar{x}_i = \hat{b}_i = b_i - a_{ie}\hat{b}_e. \]
Questions:

- How do we determine whether a linear program is feasible?
- What do we do if the linear program is feasible, but the initial basic solution is not feasible?
- How do we determine whether a linear program is unbounded?
- How do we choose the entering and leaving variables?

Example before was a particularly nice one!
The formal procedure \textsc{simplex} \( (A,b,c) \)

1. \((N, B, A, b, c, v) = \text{INITIALIZE-SIMPLEX}(A, b, c)\)
2. let \(\Delta\) be a new vector of length \(m\)
3. while some index \(j \in N\) has \(c_j > 0\)
4. choose an index \(e \in N\) for which \(c_e > 0\)
5. for each index \(i \in B\)
   6. if \(a_{ie} > 0\)
      7. \(\Delta_i = b_i / a_{ie}\)
   8. else \(\Delta_i = \infty\)
9. choose an index \(l \in B\) that minimizes \(\Delta_i\)
10. if \(\Delta_l == \infty\)
    11. return “unbounded”
12. else \((N, B, A, b, c, v) = \text{PIVOT}(N, B, A, b, c, v, l, e)\)
13. for \(i = 1\) to \(n\)
    14. if \(i \in B\)
      15. \(\bar{x}_i = b_i\)
    16. else \(\bar{x}_i = 0\)
17. return \((\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)\)

Returns a slack form with a feasible basic solution (if it exists)

Main Loop:
- terminates if all coefficients in objective function are negative
- Line 4 picks entering variable \(x_e\) with negative coefficient
- Lines 6 – 9 pick the tightest constraint, associated with \(x_l\)
- Line 11 returns “unbounded” if there are no constraints
- Line 12 calls \textsc{pivot}, switching roles of \(x_l\) and \(x_e\)

Return corresponding solution.
The formal procedure SIMPLEX

SIMPLEX \((A, b, c)\)
1. \((N, B, A, b, c, v) = \text{INITIALIZE-SIMPLEX}(A, b, c)\)
2. let \(\Delta\) be a new vector of length \(m\)
3. while some index \(j \in N\) has \(c_j > 0\)
4. choose an index \(e \in N\) for which \(c_e > 0\)
5. for each index \(i \in B\)
6. if \(a_{ie} > 0\)
7. \(\Delta_i = b_i / a_{ie}\)
8. else \(\Delta_i = \infty\)
9. choose an index \(l \in B\) that minimizes \(\Delta_i\)
10. if \(\Delta_l = \infty\)
11. return “unbounded”

Proof is based on the following three-part loop invariant:
1. the slack form is always equivalent to the one returned by \text{INITIALIZE-SIMPLEX},
2. for each \(i \in B\), we have \(b_i \geq 0\),
3. the basic solution associated with the (current) slack form is feasible.

Lemma 29.2
Suppose the call to \text{INITIALIZE-SIMPLEX} in line 1 returns a slack form for which the basic solution is feasible. Then if \text{SIMPLEX} returns a solution, it is a feasible solution. If \text{SIMPLEX} returns “unbounded”, the linear program is unbounded.
Termination

**Degeneracy:** One iteration of SIMPLEX leaves the objective value unchanged.

\[
\begin{align*}
  z &= x_1 + x_2 + x_3 \\
  x_4 &= 8 - x_1 - x_2 \\
  x_5 &= x_2 - x_3
\end{align*}
\]

Pivot with \( x_1 \) entering and \( x_4 \) leaving

\[
\begin{align*}
  z &= 8 + x_3 - x_4 \\
  x_1 &= 8 - x_2 - x_4 \\
  x_5 &= x_2 - x_3
\end{align*}
\]

Cycling: If additionally slack form at two iterations are identical, SIMPLEX fails to terminate!

\[
\begin{align*}
  z &= 8 + x_2 - x_4 - x_5 \\
  x_1 &= 8 - x_2 - x_4 \\
  x_3 &= x_2 - x_5
\end{align*}
\]
Exercise: Execute one more step of the Simplex Algorithm on the tableau from the previous slide.
Termination and Running Time

Cycling: SIMPLEX may fail to terminate.

Anti-Cycling Strategies

1. Bland’s rule: Choose entering variable with smallest index
2. Random rule: Choose entering variable uniformly at random
3. Perturbation: Perturb the input slightly so that it is impossible to have two solutions with the same objective value

Replace each $b_i$ by $\hat{b}_i = b_i + \epsilon_i$, where $\epsilon_i \gg \epsilon_{i+1}$ are all small.

Lemma 29.7

Assuming INITIALIZE-SIMPLEX returns a slack form for which the basic solution is feasible, SIMPLEX either reports that the program is unbounded or returns a feasible solution in at most \( \binom{n+m}{m} \) iterations.

Every set $B$ of basic variables uniquely determines a slack form, and there are at most \( \binom{n+m}{m} \) unique slack forms.
Introduction

Formulating Problems as Linear Programs

Standard and Slack Forms

Simplex Algorithm

Finding an Initial Solution
Finding an Initial Solution

maximize \[ 2x_1 - x_2 \]
subject to
\[ \begin{align*}
2x_1 - x_2 & \leq 2 \\
x_1 - 5x_2 & \leq -4 \\
x_1, x_2 & \geq 0
\end{align*} \]

Conversion into slack form

\[ \begin{align*}
z & = 2x_1 - x_2 \\
x_3 & = 2 - 2x_1 + x_2 \\
x_4 & = -4 - x_1 + 5x_2
\end{align*} \]

Basic solution \((x_1, x_2, x_3, x_4) = (0, 0, 2, -4)\) is not feasible!
maximize \( 2x_1 - x_2 \)

subject to

\[
\begin{align*}
2x_1 - x_2 & \leq 2 \\
x_1 - 5x_2 & \leq -4 \\
x_1, x_2 & \geq 0
\end{align*}
\]

Questions:

- How to determine whether there is any feasible solution?
- If there is one, how to determine an initial basic solution?
Formulating an Auxiliary Linear Program

Let $L_{aux}$ be the auxiliary LP of a linear program $L$ in standard form. Then $L$ is feasible if and only if the optimal objective value of $L_{aux}$ is 0.

Proof.

- “⇒”: Suppose $L$ has a feasible solution $\bar{x} = (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)$
  - $\bar{x}_0 = 0$ combined with $\bar{x}$ is a feasible solution to $L_{aux}$ with objective value 0.
  - Since $\bar{x}_0 \geq 0$ and the objective is to maximize $-x_0$, this is optimal for $L_{aux}$
- “⇐”: Suppose that the optimal objective value of $L_{aux}$ is 0
  - Then $\bar{x}_0 = 0$, and the remaining solution values $(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)$ satisfy $L$. \qed
The initial basic feasible solution

\[ \text{maximize } \sum_{j=1}^{n} a_{ij} x_j \quad \text{subject to} \quad \sum_{j=1}^{n} b_i \leq 0 \quad \text{for } i = 1, 2, \ldots, m; \]
\[ x_j \leq 0 \quad \text{for } j = 0, 1, \ldots, n. \]

Then \( L \) is feasible if and only if the optimal objective value of \( L_{aux} \) is 0.

**Proof**

Suppose that \( L \) has a feasible solution \( \{x_0, x_1, \ldots, x_n\} \).

Then the solution \( \{0, x_1, \ldots, x_n\} \) combined with \( x_0 \) is a feasible solution to \( L_{aux} \) with objective value 0.

Since \( x_0 \leq 0 \) is a constraint of \( L_{aux} \) and the objective function is to maximize \( x_0 \), this solution must be optimal for \( L_{aux} \).

Conversely, suppose that the optimal objective value of \( L_{aux} \) is 0.

Then \( \{0, x_1, \ldots, x_n\} \), and the remaining solution values of \( \{x_0, x_1, \ldots, x_n\} \) satisfy the constraints of \( L_{aux} \).

We now describe our strategy to find an initial basic feasible solution for a linear program \( L \) in standard form:

\[ \text{INITIALIZE-SIMPLEX} \]

\[ \begin{align*}
&\text{let } k \text{ be the index of the minimum } b_i \\
&\text{if } b_k \geq 0  \\
&\quad \text{return } \{1, 2, \ldots, n\}, \{n+1, n+2, \ldots, n+m\}, A, b, c, 0 \\
&\text{form } L_{aux} \text{ by adding } -x_0 \text{ to the left-hand side of each constraint} \\
&\text{and setting the objective function to } -x_0 \\
&\text{let } (N, B, A, b, c, \nu) \text{ be the resulting slack form for } L_{aux} \\
&l = n + k \\
&\text{// } L_{aux} \text{ has } n+1 \text{ nonbasic variables and } m \text{ basic variables.} \\
&(N, B, A, b, c, \nu) = \text{PIVOT}(N, B, A, b, c, \nu, l, 0) \\
&\text{// The basic solution is now feasible for } L_{aux}. \\
&\text{iterate the while loop of lines 3–12 of SIMPLEX until an optimal solution} \\
&\text{to } L_{aux} \text{ is found} \\
&\text{if the optimal solution to } L_{aux} \text{ sets } x_0 \text{ to 0} \\
&\quad \text{if } x_0 \text{ is basic} \\
&\quad \quad \text{perform one (degenerate) pivot to make it nonbasic} \\
&\quad \quad \text{from the final slack form of } L_{aux}, \text{ remove } x_0 \text{ from the constraints and} \\
&\quad \quad \text{restore the original objective function of } L, \text{ but replace each basic} \\
&\quad \quad \text{variable in this objective function by the right-hand side of its} \\
&\quad \quad \text{associated constraint} \\
&\quad \text{return the modified final slack form} \\
&\text{else return } \text{“infeasible”}
\end{align*} \]
Example of INITIALIZE-SIMPLEX (1/3)

maximize \(2x_1 - x_2\)
subject to
\[
\begin{align*}
2x_1 & - x_2 \leq 2 \\
x_1 & - 5x_2 \leq -4 \\
x_1, x_2 & \geq 0
\end{align*}
\]

Formulating the auxiliary linear program

maximize \(-x_0\)
subject to
\[
\begin{align*}
2x_1 & - x_2 - x_0 \leq 2 \\
x_1 & - 5x_2 - x_0 \leq -4 \\
x_1, x_2, x_0 & \geq 0
\end{align*}
\]

Basic solution \((0, 0, 0, 2, -4)\) not feasible!

Converting into slack form

\[
\begin{align*}
z & = -x_0 \\
x_3 & = 2 - 2x_1 + x_2 + x_0 \\
x_4 & = -4 - x_1 + 5x_2 + x_0
\end{align*}
\]
Example of INITIALIZE-SIMPLEX (2/3)

\[ z = -x_0, \]
\[ x_3 = 2 - 2x_1 + x_2 + x_0, \]
\[ x_4 = -4 - x_1 + 5x_2 + x_0. \]

Pivot with \( x_0 \) entering and \( x_4 \) leaving

\[ z = -4 - x_0 + 5x_2 - x_4, \]
\[ x_0 = 4 + x_1 - 5x_2 + x_4, \]
\[ x_3 = 6 - x_1 - 4x_2 + x_4. \]

Basic solution \((4, 0, 0, 6, 0)\) is feasible!

Pivot with \( x_2 \) entering and \( x_0 \) leaving

\[ z = \frac{4}{5} - x_0 - \frac{x_0}{5} + \frac{x_1}{5} + \frac{x_4}{5}, \]
\[ x_2 = \frac{14}{5} + \frac{4x_0}{5} - \frac{9x_1}{5} + \frac{x_4}{5}, \]
\[ x_3 = \frac{9}{5} - \frac{9x_1}{5} + \frac{x_4}{5}. \]

Optimal solution has \( x_0 = 0 \), hence the initial problem was feasible!
Example of INITIALIZE-SIMPLEX (3/3)

\[
\begin{align*}
\mathbf{x}_2 &= \frac{4}{5} - \frac{x_0}{5} + \frac{x_1}{5} + \frac{x_4}{5} \\
\mathbf{x}_3 &= \frac{14}{5} + \frac{4x_0}{5} - \frac{9x_1}{5} + \frac{x_4}{5}
\end{align*}
\]

Set \( x_0 = 0 \) and express objective function by non-basic variables

\[
\begin{align*}
\mathbf{z} &= -\frac{4}{5} + \frac{9x_1}{5} - \frac{x_4}{5} \\
\mathbf{x}_2 &= \frac{4}{5} + \frac{5}{x_1} + \frac{5}{x_4} \\
\mathbf{x}_3 &= \frac{14}{5} - \frac{9x_1}{5} + \frac{x_4}{5}
\end{align*}
\]

Basic solution \((0, \frac{4}{5}, \frac{14}{5}, 0)\), which is feasible!

Lemma 29.12

If a linear program \( L \) has no feasible solution, then INITIALIZE-SIMPLEX returns “infeasible”. Otherwise, it returns a valid slack form for which the basic solution is feasible.
Fundamental Theorem of Linear Programming

Any linear program $L$, given in standard form, either
1. has an optimal solution with a finite objective value,
2. is infeasible, or
3. is unbounded.

Proof requires the concept of duality, which is not covered in this course (for details see CLRS3, Chapter 29.4)
Workflow for Solving Linear Programs

1. Linear Program (in any form)
   - Standard Form
     - Slack Form
       - No Feasible Solution
         - INITIALIZE-SIMPLEX terminates
       - Feasible Basic Solution
         - INITIALIZE-SIMPLEX followed by SIMPLEX
           - LP unbounded
             - SIMPLEX terminates
           - LP bounded
             - SIMPLEX returns optimum
Linear Programming

- extremely versatile tool for modelling problems of all kinds
- basis of Integer Programming, to be discussed in later lectures

Simplex Algorithm

- In practice: usually terminates in polynomial time, i.e., $O(m + n)$
- In theory: even with anti-cycling may need exponential time

Research Problem: Is there a pivoting rule which makes SIMPLEX a polynomial-time algorithm?

Polynomial-Time Algorithms

- Interior-Point Methods: traverses the interior of the feasible set of solutions (not just vertices!)
Which of the following statements are true?

1. In each iteration of the Simplex algorithm, the objective function increases.
2. There exist linear programs that have exactly two optimal solutions.
3. There exist linear programs that have infinitely many optimal solutions.
4. The Simplex algorithm always runs in worst-case polynomial time.