Randomised Algorithms
Lecture 6: Linear Programming: Introduction

Thomas Sauerwald (tms41@cam.ac.uk)
Outline

Introduction

A Simple Example of a Linear Program

Formulating Problems as Linear Programs

Standard and Slack Forms
- **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)
Introduction

A Simple Example of a Linear Program

Formulating Problems as Linear Programs

Standard and Slack Forms
What are Linear Programs?

Linear Programming (informal definition)

- maximise or minimise an objective, given limited resources (competing constraint)
- constraints are specified as (in)equalities
- objective function and constraints are linear
A Simple Example of a Linear Optimisation Problem

- Laptop
A Simple Example of a Linear Optimisation Problem

- Laptop
  - selling price to retailer: 1,000 GBP
A Simple Example of a Linear Optimisation Problem

- Laptop
  - selling price to retailer: 1,000 GBP
  - glass: 4 units
A Simple Example of a Linear Optimisation Problem

- **Laptop**
  - selling price to retailer: 1,000 GBP
  - glass: 4 units
  - copper: 2 units
A Simple Example of a Linear Optimisation Problem

- **Laptop**
  - selling price to retailer: 1,000 GBP
  - glass: 4 units
  - copper: 2 units
  - rare-earth elements: 1 unit

You have a daily supply of:
- glass: 20 units
- copper: 10 units
- rare-earth elements: 14 units
(and enough of everything else...)

How to maximise your daily earnings?
A Simple Example of a Linear Optimisation Problem

- **Laptop**
  - selling price to retailer: 1,000 GBP
  - glass: 4 units
  - copper: 2 units
  - rare-earth elements: 1 unit

- **Smartphone**
A Simple Example of a Linear Optimisation Problem

- **Laptop**
  - selling price to retailer: 1,000 GBP
  - glass: 4 units
  - copper: 2 units
  - rare-earth elements: 1 unit

- **Smartphone**
  - selling price to retailer: 1,000 GBP

You have a daily supply of:
- glass: 20 units
- copper: 10 units
- rare-earth elements: 14 units

(and enough of everything else...)
A Simple Example of a Linear Optimisation Problem

- **Laptop**
  - selling price to retailer: 1,000 GBP
  - glass: 4 units
  - copper: 2 units
  - rare-earth elements: 1 unit

- **Smartphone**
  - selling price to retailer: 1,000 GBP
  - glass: 1 unit
A Simple Example of a Linear Optimisation Problem

- **Laptop**
  - selling price to retailer: 1,000 GBP
  - glass: 4 units
  - copper: 2 units
  - rare-earth elements: 1 unit

- **Smartphone**
  - selling price to retailer: 1,000 GBP
  - glass: 1 unit
  - copper: 1 unit
A Simple Example of a Linear Optimisation Problem

- **Laptop**
  - selling price to retailer: 1,000 GBP
  - glass: 4 units
  - copper: 2 units
  - rare-earth elements: 1 unit

- **Smartphone**
  - selling price to retailer: 1,000 GBP
  - glass: 1 unit
  - copper: 1 unit
  - rare-earth elements: 2 units
A Simple Example of a Linear Optimisation Problem

- **Laptop**
  - selling price to retailer: 1,000 GBP
  - glass: 4 units
  - copper: 2 units
  - rare-earth elements: 1 unit

- **Smartphone**
  - selling price to retailer: 1,000 GBP
  - glass: 1 unit
  - copper: 1 unit
  - rare-earth elements: 2 units

- You have a **daily supply** of:
A Simple Example of a Linear Optimisation Problem

- **Laptop**
  - selling price to retailer: 1,000 GBP
  - glass: 4 units
  - copper: 2 units
  - rare-earth elements: 1 unit

- **Smartphone**
  - selling price to retailer: 1,000 GBP
  - glass: 1 unit
  - copper: 1 unit
  - rare-earth elements: 2 units

- You have a **daily supply** of:
  - glass: 20 units
A Simple Example of a Linear Optimisation Problem

- **Laptop**
  - selling price to retailer: 1,000 GBP
  - glass: 4 units
  - copper: 2 units
  - rare-earth elements: 1 unit

- **Smartphone**
  - selling price to retailer: 1,000 GBP
  - glass: 1 unit
  - copper: 1 unit
  - rare-earth elements: 2 units

- You have a **daily supply** of:
  - glass: 20 units
  - copper: 10 units
A Simple Example of a Linear Optimisation Problem

- **Laptop**
  - selling price to retailer: 1,000 GBP
  - glass: 4 units
  - copper: 2 units
  - rare-earth elements: 1 unit

- **Smartphone**
  - selling price to retailer: 1,000 GBP
  - glass: 1 unit
  - copper: 1 unit
  - rare-earth elements: 2 units

- **You have a daily supply of:**
  - glass: 20 units
  - copper: 10 units
  - rare-earth elements: 14 units
### A Simple Example of a Linear Optimisation Problem

- **Laptop**
  - Selling price to retailer: 1,000 GBP
  - Glass: 4 units
  - Copper: 2 units
  - Rare-earth elements: 1 unit

- **Smartphone**
  - Selling price to retailer: 1,000 GBP
  - Glass: 1 unit
  - Copper: 1 unit
  - Rare-earth elements: 2 units

- **You have a daily supply of:**
  - Glass: 20 units
  - Copper: 10 units
  - Rare-earth elements: 14 units
  - (And enough of everything else...)
A Simple Example of a Linear Optimisation Problem

- **Laptop**
  - selling price to retailer: 1,000 GBP
  - glass: 4 units
  - copper: 2 units
  - rare-earth elements: 1 unit

- **Smartphone**
  - selling price to retailer: 1,000 GBP
  - glass: 1 unit
  - copper: 1 unit
  - rare-earth elements: 2 units

- **You have a daily supply of:**
  - glass: 20 units
  - copper: 10 units
  - rare-earth elements: 14 units
  - (and enough of everything else...)

How to maximise your daily earnings?
The Linear Program

Linear Program for the Production Problem

\[
\begin{align*}
\text{maximise} & \quad x_1 + x_2 \\
\text{subject to} & \\
4x_1 + x_2 & \leq 20 \\
2x_1 + x_2 & \leq 10 \\
x_1 + 2x_2 & \leq 14 \\
x_1, x_2 & \geq 0
\end{align*}
\]
The solution of this linear program yields the optimal production schedule.
The solution of this linear program yields the optimal production schedule.

Formal Definition of Linear Program
The Linear Program

Linear Program for the Production Problem

maximise \( x_1 + x_2 \)

subject to

\[
\begin{align*}
4x_1 + x_2 & \leq 20 \\
2x_1 + x_2 & \leq 10 \\
x_1 + 2x_2 & \leq 14 \\
x_1, x_2 & \geq 0
\end{align*}
\]

The solution of this linear program yields the optimal production schedule.

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

\[
f(x_1, x_2, \ldots, x_n) = a_1x_1 + a_2x_2 + \cdots + a_nx_n.
\]
The Linear Program

Linear Program for the Production Problem

maximise $x_1 + x_2$
subject to
$4x_1 + x_2 \leq 20$
$2x_1 + x_2 \leq 10$
$x_1 + 2x_2 \leq 14$
$x_1, x_2 \geq 0$

The solution of this linear program yields the optimal production schedule.

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: $f(x_1, x_2, \ldots, x_n) \geq b$
The Linear Program

Linear Program for the Production Problem

maximise \( x_1 + x_2 \)
subject to
\[
\begin{align*}
4x_1 + x_2 & \leq 20 \\
2x_1 + x_2 & \leq 10 \\
x_1 + 2x_2 & \leq 14 \\
x_1, x_2 & \geq 0 
\end{align*}
\]

The solution of this linear program yields the optimal production schedule.

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
  \[
  f(x_1, x_2, \ldots, x_n) = a_1x_1 + a_2x_2 + \cdots + a_nx_n.
  \]
- Linear Equality: \( f(x_1, x_2, \ldots, x_n) = b \)
- Linear Inequality: \( f(x_1, x_2, \ldots, x_n) \geq b \)
The Linear Program

Linear Program for the Production Problem

maximise \( x_1 + x_2 \)
subject to
\[
\begin{align*}
4x_1 + x_2 & \leq 20 \\
2x_1 + x_2 & \leq 10 \\
x_1 + 2x_2 & \leq 14 \\
x_1, x_2 & \geq 0
\end{align*}
\]
The solution of this linear program yields the optimal production schedule.

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
\[
f(x_1, x_2, \ldots, x_n) = a_1x_1 + a_2x_2 + \cdots + a_nx_n.
\]
- Linear Equality: \( f(x_1, x_2, \ldots, x_n) = b \)
- Linear Inequality: \( f(x_1, x_2, \ldots, x_n) \geq \leq b \)
- Linear Programming Problem: either minimise or maximise a linear function subject to a set of linear constraints
maximise \[ x_1 + x_2 \]
subject to
\[ 4x_1 + x_2 \leq 20 \]
\[ 2x_1 + x_2 \leq 10 \]
\[ x_1 + 2x_2 \leq 14 \]
\[ x_1, x_2 \geq 0 \]
Finding the Optimal Production Schedule

maximise \( x_1 + x_2 \)
subject to
\[
\begin{align*}
4x_1 + x_2 & \leq 20 \\
2x_1 + x_2 & \leq 10 \\
x_1 + 2x_2 & \leq 14 \\
x_1, x_2 & \geq 0
\end{align*}
\]

Any setting of \( x_1 \) and \( x_2 \) satisfying all constraints is a feasible solution.
Finding the Optimal Production Schedule

maximise \( x_1 + x_2 \)
subject to
\[
\begin{align*}
4x_1 + x_2 &\leq 20 \\
2x_1 + x_2 &\leq 10 \\
x_1 + 2x_2 &\leq 14 \\
x_1, x_2 &\geq 0
\end{align*}
\]

Any setting of \( x_1 \) and \( x_2 \) satisfying all constraints is a feasible solution
Finding the Optimal Production Schedule

maximise \( x_1 + x_2 \)

subject to

\[ 4x_1 + x_2 \leq 20 \]
\[ 2x_1 + x_2 \leq 10 \]
\[ x_1 + 2x_2 \leq 14 \]
\[ x_1, x_2 \geq 0 \]

Any setting of \( x_1 \) and \( x_2 \) satisfying all constraints is a feasible solution
Finding the Optimal Production Schedule

maximise $x_1 + x_2$

subject to

$4x_1 + x_2 \leq 20$
$2x_1 + x_2 \leq 10$
$x_1 + 2x_2 \leq 14$
$x_1, x_2 \geq 0$

Any setting of $x_1$ and $x_2$ satisfying all constraints is a feasible solution
Finding the Optimal Production Schedule

maximise \( x_1 + x_2 \)
subject to
\[
\begin{align*}
4x_1 + x_2 &\leq 20 \\
2x_1 + x_2 &\leq 10 \\
x_1 + 2x_2 &\leq 14 \\
x_1, x_2 &\geq 0
\end{align*}
\]

Any setting of \( x_1 \) and \( x_2 \) satisfying all constraints is a feasible solution
Finding the Optimal Production Schedule

maximise \( x_1 + x_2 \)

subject to

\[ 4x_1 + x_2 \leq 20 \]
\[ 2x_1 + x_2 \leq 10 \]
\[ x_1 + 2x_2 \leq 14 \]
\[ x_1, x_2 \geq 0 \]

Any setting of \( x_1 \) and \( x_2 \) satisfying all constraints is a feasible solution.
Finding the Optimal Production Schedule

maximise $x_1 + x_2$

subject to

$4x_1 + x_2 \leq 20$
$2x_1 + x_2 \leq 10$
$x_1 + 2x_2 \leq 14$
$x_1, x_2 \geq 0$

Any setting of $x_1$ and $x_2$ satisfying all constraints is a feasible solution.
Finding the Optimal Production Schedule

maximise \( x_1 + x_2 \)

subject to

\[ 4x_1 + x_2 \leq 20 \]
\[ 2x_1 + x_2 \leq 10 \]
\[ x_1 + 2x_2 \leq 14 \]
\[ x_1, x_2 \geq 0 \]

Any setting of \( x_1 \) and \( x_2 \) satisfying all constraints is a feasible solution.
Finding the Optimal Production Schedule

maximise $x_1 + x_2$

subject to

$4x_1 + x_2 \leq 20$
$2x_1 + x_2 \leq 10$
$x_1 + 2x_2 \leq 14$

$x_1$, $x_2 \geq 0$

Graphical Procedure: Move the line $x_1 + x_2 = z$ as far up as possible.
Finding the Optimal Production Schedule

maximise $x_1 + x_2$

subject to

$4x_1 + x_2 \leq 20$

$2x_1 + x_2 \leq 10$

$x_1 + 2x_2 \leq 14$

$x_1, x_2 \geq 0$

Graphical Procedure: Move the line $x_1 + x_2 = z$ as far up as possible.
Finding the Optimal Production Schedule

maximise \( x_1 + x_2 \)

subject to

\[
\begin{align*}
4x_1 + x_2 & \leq 20 \\
2x_1 + x_2 & \leq 10 \\
x_1 + 2x_2 & \leq 14 \\
x_1, x_2 & \geq 0
\end{align*}
\]

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

maximise \( x_1 + x_2 \)
subject to
\[
4x_1 + x_2 \leq 20 \\
2x_1 + x_2 \leq 10 \\
x_1 + 2x_2 \leq 14 \\
x_1, x_2 \geq 0
\]

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

maximise \( x_1 + x_2 \)

subject to

\[
\begin{align*}
4x_1 + x_2 & \leq 20 \\
2x_1 + x_2 & \leq 10 \\
x_1 + 2x_2 & \leq 14 \\
x_1, x_2 & \geq 0
\end{align*}
\]

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

maximise \( x_1 + x_2 \)

subject to

\[
\begin{align*}
4x_1 + x_2 &\leq 20 \\
2x_1 + x_2 &\leq 10 \\
x_1 + 2x_2 &\leq 14 \\
x_1, x_2 &\geq 0
\end{align*}
\]

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

maximise $x_1 + x_2$

subject to

$4x_1 + x_2 \leq 20$

$2x_1 + x_2 \leq 10$

$x_1 + 2x_2 \leq 14$

$x_1, x_2 \geq 0$

Graphical Procedure: Move the line $x_1 + x_2 = z$ as far up as possible.
Finding the Optimal Production Schedule

maximise \( x_1 + x_2 \)
subject to
\[
\begin{align*}
4x_1 + x_2 &\leq 20 \\
2x_1 + x_2 &\leq 10 \\
x_1 + 2x_2 &\leq 14 \\
x_1, x_2 &\geq 0
\end{align*}
\]

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

maximise $x_1 + x_2$

subject to

$4x_1 + x_2 \leq 20$
$2x_1 + x_2 \leq 10$
$x_1 + 2x_2 \leq 14$
$x_1, x_2 \geq 0$

Graphical Procedure: Move the line $x_1 + x_2 = z$ as far up as possible.
Finding the Optimal Production Schedule

maximise \( x_1 + x_2 \)

subject to

\[
\begin{align*}
4x_1 &+ x_2 \leq 20 \\
2x_1 &+ x_2 \leq 10 \\
x_1 &+ 2x_2 \leq 14 \\
x_1, x_2 &\geq 0
\end{align*}
\]

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

maximise \( x_1 + x_2 \)
subject to

\[ 4x_1 + x_2 \leq 20 \]
\[ 2x_1 + x_2 \leq 10 \]
\[ x_1 + 2x_2 \leq 14 \]
\[ x_1, x_2 \geq 0 \]

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

subject to

$4x_1 + x_2 \leq 20$

$2x_1 + x_2 \leq 10$

$x_1 + 2x_2 \leq 14$

$x_1, x_2 \geq 0$

Graphical Procedure: Move the line $x_1 + x_2 = z$ as far up as possible.
Finding the Optimal Production Schedule

maximise \[ x_1 + x_2 \]
subject to
\[ 4x_1 + x_2 \leq 20 \]
\[ 2x_1 + x_2 \leq 10 \]
\[ x_1 + 2x_2 \leq 14 \]
\[ x_1, x_2 \geq 0 \]

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

Question: Which aspect did we ignore in the formulation of the linear program?
Finding the Optimal Production Schedule

maximise \( x_1 + x_2 \)
subject to
\[
\begin{align*}
4x_1 + x_2 &\leq 20 \\
2x_1 + x_2 &\leq 10 \\
x_1 + 2x_2 &\leq 14 \\
x_1, x_2 &\geq 0
\end{align*}
\]

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

A Simple Example of a Linear Program

Formulating Problems as Linear Programs

Standard and Slack Forms
Shortest Paths

Single-Pair Shortest Path Problem

- **Given**: directed graph $G = (V, E)$ with edge weights $w : E \rightarrow \mathbb{R}$, pair of vertices $s, t \in V$
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 $G$
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 \( G \).

Let \( p = (v_0 = s, v_1, \ldots, v_k = t) \) such that

\[
w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)
\]

is minimised.
Shortest Paths

Single-Pair Shortest Path Problem

- **Given:** directed graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}$, 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 minimised.
Shortest Paths

**Single-Pair Shortest Path Problem**

- **Given:** directed graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}$, 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 minimised.

**Exercise:** How can we translate the SPSP problem into a linear program?
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 minimised.

Shortest Paths as LP

subject to
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 minimised.

Shortest Paths as LP

\[
\begin{align*}
\text{s.t. } & d_v \leq d_u + w(u, v) & \text{for each edge } (u, v) \in E, \\
& d_s = 0.
\end{align*}
\]
**Shortest Paths**

**Single-Pair Shortest Path Problem**

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

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

**Shortest Paths as LP**

\[
\begin{align*}
\text{maximise} & \quad d_t \\
\text{subject to} & \quad d_v \leq d_u + w(u, v) \quad \text{for each edge } (u, v) \in E, \\
& \quad d_s = 0.
\end{align*}
\]
Shortest Paths

Single-Pair Shortest Path Problem

- **Given**: directed graph \( G = (V, E) \) with edge weights \( w : E \to \mathbb{R} \), 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 minimised.

Shortest Paths as LP

\[
\begin{align*}
\text{maximise} & \quad d_t \\
\text{subject to} & \quad d_v \leq d_u + w(u, v) \quad \text{for each edge } (u, v) \in E, \\
& \quad d_s = 0.
\end{align*}
\]

this is a maximisation problem!
Shortest Paths

Single-Pair Shortest Path Problem

- **Given:** directed graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}$, 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 minimised.

Recall: When **Bellman-Ford** terminates, all these inequalities are satisfied.

Shortest Paths as LP

maximise $d_t$
subject to $d_v \leq d_u + w(u, v)$ for each edge $(u, v) \in E$,

$s = 0$.

this is a maximisation problem!
Shortest Paths

Single-Pair Shortest Path Problem

- **Given:** directed graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}$, 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 minimised.

Shortest Paths as LP

maximise $d_t$
subject to

$\begin{align*}
    d_v &\leq d_u + w(u, v) \quad \text{for each edge } (u, v) \in E, \\
    d_s &\leq 0.
\end{align*}$

this is a maximisation problem!

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

Solution $\overline{d}$ satisfies $\overline{d}_v = \min_{(u,v) \in E} \{ d_u + w(u, v) \}$

6. Linear Programming © T. Sauerwald
Formulating Problems as Linear Programs
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 $s, t \in V$
**Maximum Flow**

- **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 $s, t \in V$
Maximum Flow

- **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

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 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

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

**Maximum Flow Problem**

- \( s \to 2 \to 4 \to t \)
- \( f_{24} = 4 \)
- \( f_{4t} = 9/10 \)
- \( f_{32} = 0/2 \)
- \( f_{3s} = 10/10 \)
- \( |f| = 19 \)

**Maximum Flow as LP**

\[
\begin{align*}
\text{maximise} & \quad \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs} \\
\text{subject to} & \quad f_{uv} \leq c(u, v) \quad \text{for each } u, v \in V, \\
& \quad \sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv} \quad \text{for each } u \in V \setminus \{s, t\}, \\
& \quad f_{uv} \geq 0 \quad \text{for each } u, v \in V.
\end{align*}
\]
Minimum-Cost Flow

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

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

Extension of the Maximum Flow Problem

Minimum-Cost-Flow Problem

Optimal Solution with total cost:

$$\sum_{(u, v) \in E} a(u, v) f_{u,v} = (2 \cdot 2) + (5 \cdot 2) + (3 \cdot 1) + (7 \cdot 1) + (1 \cdot 3) = 27$$
Minimum-Cost Flow

**Extension of the Maximum Flow Problem**

**Minimum-Cost-Flow Problem**

- **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

---

**Optimal Solution with total cost:**

\[
\sum_{(u, v) \in E} a(u, v)f_{uv} = (2 \times 2) + (5 \times 2) + (3 \times 1) + (7 \times 1) + (1 \times 3) = 27
\]
Minimum-Cost Flow

Minimum-Cost-Flow Problem

- **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.
Minimum-Cost Flow

Minimum-Cost-Flow Problem

- **Given:** directed graph \( G = (V, E) \) with capacities \( c : E \rightarrow \mathbb{R}^+ \), pair of vertices \( s, t \in V \), cost function \( a : E \rightarrow \mathbb{R}^+ \), flow demand of \( d \) units
- **Goal:** Find a flow \( f : V \times V \rightarrow \mathbb{R} \) 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.

**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

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

\[
\text{minimise} \quad \sum_{(u, v) \in E} a(u, v) f_{uv}
\]

subject to

\[
\begin{align*}
 f_{uv} & \leq c(u, v) \quad \text{for } u, v \in V, \\
\sum_{v \in V} f_{vu} - \sum_{v \in V} f_{uv} & = 0 \quad \text{for } u \in V \setminus \{s, t\}, \\
\sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs} & = d, \\
 f_{uv} & \geq 0 \quad \text{for } u, v \in V.
\end{align*}
\]
Minimum Cost Flow as a LP

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

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

Introduction

A Simple Example of a Linear Program

Formulating Problems as Linear Programs

Standard and Slack Forms
Standard and Slack Forms

Standard Form

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

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

\[ x_j \geq 0 \quad \text{for } j = 1, 2, \ldots, n \]
Standard and Slack Forms

**Standard Form**

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

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 and Slack Forms

Standard Form

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

subject to

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

\[ x_j \geq 0 \quad \text{for } j = 1, 2, \ldots, n \]

6. Linear Programming © T. Sauerwald

Standard and Slack Forms

15
Standard and Slack Forms

**Standard Form**

- **Objective Function**
  \[ \text{maximise} \quad \sum_{j=1}^{n} c_j x_j \]
  
- **Subject to**
  \[ \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad \text{for } i = 1, 2, \ldots, m \]
  \[ x_j \geq 0 \quad \text{for } j = 1, 2, \ldots, n \]

**n + m constraints**
Standard and Slack Forms

Standard Form

maximise \( \sum_{j=1}^{n} c_j x_j \) 

subject to 

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

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

\( n + m \) constraints

Objective Function

Non-Negativity Constraints

Standard Form (Matrix-Vector-Notation)

maximise \( c^T x \) 

subject to 

\( Ax \leq b \) 

\( x \geq 0 \)
Reasons for a LP not being in standard form:
1. The objective might be a minimisation rather than maximisation.
2. There might be variables without nonnegativity constraints.
3. There might be equality constraints.
4. There might be inequality constraints (with $\geq$ instead of $\leq$).
Converting Linear Programs into Standard Form

Reasons for a LP not being in standard form:
1. The objective might be a minimisation rather than maximisation.
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
Reasons for a LP not being in standard form:
1. The objective might be a **minimisation** rather than **maximisation**.
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 minimisation rather than maximisation.
Reasons for a LP not being in standard form:

1. The objective might be a minimisation rather than maximisation.

\[
\begin{align*}
\text{minimise} & \quad -2x_1 + 3x_2 \\
\text{subject to} & \\
& x_1 + x_2 = 7 \\
& x_1 - 2x_2 \leq 4 \\
& x_1 \geq 0
\end{align*}
\]
Converting into Standard Form (1/5)

Reasons for a LP not being in standard form:
1. The objective might be a minimisation rather than maximisation.

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

Negate objective function
Reasons for a LP not being in standard form:
1. The objective might be a minimisation rather than maximisation.

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

Negate objective function

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

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

maximise \( 2x_1 - 3x_2 \)
subject to
\[
\begin{align*}
x_1 + x_2 &= 7 \\
x_1 - 2x_2 &\leq 4 \\
x_1 &\geq 0
\end{align*}
\]
Reasons for a LP not being in standard form:
2. There might be variables without nonnegativity constraints.

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

Replace \( x_2 \) by two non-negative variables \( x'_2 \) and \( x''_2 \).
Reasons for a LP not being in standard form:
2. There might be variables without nonnegativity constraints.

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

Replace \( x_2 \) by two non-negative variables \( x'_2 \) and \( x''_2 \)

\[
\begin{align*}
\text{maximise} & \quad 2x_1 - 3x'_2 + 3x''_2 \\
\text{subject to} & \quad x_1 + x'_2 - x''_2 = 7 \\
& \quad x_1 - 2x'_2 + 2x''_2 \leq 4 \\
& \quad x_1, x'_2, x''_2 \geq 0 \\
\end{align*}
\]
Converting into Standard Form (3/5)

Reasons for a LP not being in standard form:

3. There might be equality constraints.

Replace each equality by two inequalities.
Reasons for a LP not being in standard form:
3. There might be equality constraints.

maximise \(2x_1 - 3x'_2 + 3x''_2\)
subject to
\[
\begin{align*}
x_1 + x'_2 - x''_2 &= 7 \\
x_1 - 2x'_2 + 2x''_2 &\leq 4 \\
x_1, x'_2, x''_2 &\geq 0
\end{align*}
\]
Reasons for a LP not being in standard form:

3. There might be equality constraints.

maximise $2x_1 - 3x'_2 + 3x''_2$

subject to

$x_1 + x'_2 - x''_2 = 7$

$x_1 - 2x'_2 + 2x''_2 \leq 4$

$x_1, x'_2, x''_2 \geq 0$

Replace each equality by two inequalities.
Converting into Standard Form (3/5)

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

maximise \( 2x_1 - 3x_2' + 3x_2'' \)
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.

maximise \( 2x_1 - 3x_2' + 3x_2'' \)
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 $\leq$).
Reasons for a LP not being in standard form:

4. There might be inequality constraints (with ≥ instead of ≤).

maximise $2x_1 - 3x'_2 + 3x''_2$

subject to

$x_1 + x'_2 - x''_2 \leq 7$
$x_1 + x'_2 - x''_2 \geq 7$
$x_1 - 2x'_2 + 2x''_2 \leq 4$
$x_1, x'_2, x''_2 \geq 0$
Reasons for a LP not being in standard form:

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

maximise $2x_1 - 3x'_2 + 3x''_2$

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.
Reasons for a LP not being in standard form:

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

\[
\begin{align*}
\text{maximise} & \quad 2x_1 - 3x'_2 + 3x''_2 \\
\text{subject to} & \quad x_1 + x'_2 - x''_2 \leq 7 \\
& \quad x_1 + x'_2 - x''_2 \geq 7 \\
& \quad x_1 - 2x'_2 + 2x''_2 \leq 4 \\
& \quad x_1, x'_2, x''_2 \geq 0
\end{align*}
\]

Negate respective inequalities.

\[
\begin{align*}
\text{maximise} & \quad 2x_1 - 3x'_2 + 3x''_2 \\
\text{subject to} & \quad x_1 + x'_2 - x''_2 \leq 7 \\
& \quad -x_1 - x'_2 + x''_2 \leq -7 \\
& \quad x_1 - 2x'_2 + 2x''_2 \leq 4 \\
& \quad x_1, x'_2, x''_2 \geq 0
\end{align*}
\]
Converting into Standard Form (5/5)

maximise \( 2x_1 - 3x_2 + 3x_3 \)

subject to

\[
\begin{align*}
x_1 + x_2 - x_3 & \leq 7 \\
-x_1 - x_2 + x_3 & \leq -7 \\
x_1 - 2x_2 + 2x_3 & \leq 4 \\
x_1, x_2, x_3 & \geq 0
\end{align*}
\]
Converting into Standard Form (5/5)

Rename variable names (for consistency).

maximise \[ 2x_1 - 3x_2 + 3x_3 \]
subject to
\[ x_1 + x_2 - x_3 \leq 7 \]
\[ -x_1 - x_2 + x_3 \leq -7 \]
\[ x_1 - 2x_2 + 2x_3 \leq 4 \]
\[ x_1, x_2, x_3 \geq 0 \]
Convert into Standard Form (5/5)

Rename variable names (for consistency).

maximise \(2x_1 - 3x_2 + 3x_3\)

subject to

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

It is always possible to convert a linear program into standard form.
**Goal:** Convert standard form into slack form, where all constraints except for the non-negativity constraints are equalities.
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.
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
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.
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 $s_i$ by
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

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

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

$s \geq 0$.  

---

6. Linear Programming © T. Sauerwald  
Standard and Slack Forms  
22
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

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

$s$ measures the slack between the two sides of the inequality.

$s \geq 0$. 

---

6. Linear Programming © T. Sauerwald

**Standard and Slack Forms**
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 $x_{n+i}$
Converting Standard Form into Slack Form (2/3)

maximise $2x_1 - 3x_2 + 3x_3$

subject to

$x_1 + x_2 - x_3 \leq 7$
$-x_1 - x_2 + x_3 \leq -7$
$x_1 - 2x_2 + 2x_3 \leq 4$

$x_1, x_2, x_3 \geq 0$
Converting Standard Form into Slack Form (2/3)

maximise \( 2x_1 - 3x_2 + 3x_3 \)

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
Converting Standard Form into Slack Form (2/3)

maximise \( 2x_1 - 3x_2 + 3x_3 \)

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

subject to

\[
x_4 = 7 - x_1 - x_2 + x_3
\]
Converting Standard Form into Slack Form (2/3)

maximise $2x_1 - 3x_2 + 3x_3$

subject to

$x_1 + x_2 - x_3 \leq 7$
$-x_1 - x_2 + x_3 \leq -7$
$x_1 - 2x_2 + 2x_3 \leq 4$

$x_1, x_2, x_3 \geq 0$

Introduce slack variables

subject to

$x_4 = 7 - x_1 - x_2 + x_3$
$x_5 = -7 + x_1 + x_2 - x_3$
Converting Standard Form into Slack Form (2/3)

maximise \[ 2x_1 - 3x_2 + 3x_3 \]
subject to
\[ x_1 + x_2 - x_3 \leq 7 \]
\[ -x_1 - x_2 + x_3 \leq -7 \]
\[ x_1 - 2x_2 + 2x_3 \leq 4 \]
\[ x_1, x_2, x_3 \geq 0 \]

Introduce slack variables

subject to
\[ x_4 = 7 - x_1 - x_2 + x_3 \]
\[ x_5 = -7 + x_1 + x_2 - x_3 \]
\[ x_6 = 4 - x_1 + 2x_2 - 2x_3 \]
Converting Standard Form into Slack Form (2/3)

maximise \[ 2x_1 - 3x_2 + 3x_3 \]

subject to
\[ x_1 + x_2 - x_3 \leq 7 \]
\[ -x_1 - x_2 + x_3 \leq -7 \]
\[ x_1 - 2x_2 + 2x_3 \leq 4 \]
\[ x_1, x_2, x_3 \geq 0 \]

Introduce slack variables

subject to
\[ x_4 = 7 - x_1 - x_2 + x_3 \]
\[ x_5 = -7 + x_1 + x_2 - x_3 \]
\[ x_6 = 4 - x_1 + 2x_2 - 2x_3 \]
\[ x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \]
maximise \[ 2x_1 - 3x_2 + 3x_3 \]
subject to
\[ x_1 + x_2 - x_3 \leq 7 \]
\[ -x_1 - x_2 + x_3 \leq -7 \]
\[ x_1 - 2x_2 + 2x_3 \leq 4 \]
\[ x_1, x_2, x_3 \geq 0 \]

Introduce slack variables

maximise \[ 2x_1 - 3x_2 + 3x_3 \]
subject to
\[ x_4 = 7 - x_1 - x_2 + x_3 \]
\[ x_5 = -7 + x_1 + x_2 - x_3 \]
\[ x_6 = 4 - x_1 + 2x_2 - 2x_3 \]
\[ x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \]
Converting Standard Form into Slack Form (3/3)

maximise \[ 2x_1 - 3x_2 + 3x_3 \]

subject to

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

Use variable \( z \) to denote objective function and omit the nonnegativity constraints. This is called slack form.
Converting Standard Form into Slack Form (3/3)

maximise \[ 2x_1 - 3x_2 + 3x_3 \]
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 slack form.
Converting Standard Form into Slack Form (3/3)

maximise \[ 2x_1 - 3x_2 + 3x_3 \]

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.

\[
\begin{align*}
z &= 2x_1 - 3x_2 + 3x_3 \\
x_4 &= 7 - x_1 - x_2 + x_3 \\
x_5 &= -7 + x_1 + x_2 - x_3 \\
x_6 &= 4 - x_1 + 2x_2 - 2x_3 \\
\end{align*}
\]
Converting Standard Form into Slack Form (3/3)

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

\[ x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \]

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

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

This is called slack form.
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 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: \( B = \{4, 5, 6\} \)
Basic and Non-Basic Variables

\[
\begin{align*}
    z &= 2x_1 - 3x_2 + 3x_3 \\
x_4 &= 7 - x_1 - x_2 + x_3 \\
x_5 &= -7 + x_1 + x_2 - x_3 \\
x_6 &= 4 - x_1 + 2x_2 - 2x_3
\end{align*}
\]

Basic Variables: \( B = \{4, 5, 6\} \)  
Non-Basic Variables: \( N = \{1, 2, 3\} \)
Basic and Non-Basic Variables

\[
\begin{align*}
  z &= 2x_1 - 3x_2 + 3x_3 \\
  x_4 &= 7 - x_1 - x_2 + x_3 \\
  x_5 &= -7 + x_1 + x_2 - x_3 \\
  x_6 &= 4 - x_1 + 2x_2 - 2x_3
\end{align*}
\]

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

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

Slack Form (Formal Definition)

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

\[
\begin{align*}
  z &= \nu + \sum_{j \in N} c_j x_j \\
  x_i &= b_i - \sum_{j \in N} a_{ij} x_j \quad \text{for } i \in B,
\end{align*}
\]

and all variables are non-negative.
Basic and Non-Basic Variables

\[
\begin{align*}
    z &= 2x_1 - 3x_2 + 3x_3 \\
    x_4 &= 7 - x_1 - x_2 + x_3 \\
    x_5 &= -7 + x_1 + x_2 - x_3 \\
    x_6 &= 4 - x_1 + 2x_2 - 2x_3 \\
\end{align*}
\]

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

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

and all variables are non-negative.

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

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

- \( B = \{1, 2, 4\} \), \( N = \{3, 5, 6\} \)
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

- \( B = \{1, 2, 4\}, \ N = \{3, 5, 6\} \)
- \[
A = \begin{pmatrix}
a_{13} & a_{15} & a_{16} \\
a_{23} & a_{25} & a_{26} \\
a_{43} & a_{45} & a_{46}
\end{pmatrix}
= \begin{pmatrix}
-1/6 & -1/6 & 1/3 \\
8/3 & 2/3 & -1/3 \\
1/2 & -1/2 & 0
\end{pmatrix}
\]
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**

- \( B = \{1, 2, 4\}, \; N = \{3, 5, 6\} \)
- \( A = \begin{pmatrix}
    a_{13} & a_{15} & a_{16} \\
    a_{23} & a_{25} & a_{26} \\
    a_{43} & a_{45} & a_{46}
\end{pmatrix} = \begin{pmatrix}
    -1/6 & -1/6 & 1/3 \\
    8/3 & 2/3 & -1/3 \\
    1/2 & -1/2 & 0
\end{pmatrix} \)
- \( b = \begin{pmatrix}
    b_1 \\
    b_2 \\
    b_4
\end{pmatrix} = \begin{pmatrix}
    8 \\
    4 \\
    18
\end{pmatrix} \)
Slack Form (Example)

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

Slack Form Notation

- \( B = \{1, 2, 4\}, \ N = \{3, 5, 6\} \)
- \( A = \begin{pmatrix} a_{13} & a_{15} & a_{16} \\ a_{23} & a_{25} & a_{26} \\ a_{43} & a_{45} & a_{46} \end{pmatrix} = \begin{pmatrix} -1/6 & -1/6 & 1/3 \\ 8/3 & 2/3 & -1/3 \\ 1/2 & -1/2 & 0 \end{pmatrix} \)
- \( b = \begin{pmatrix} b_1 \\ b_2 \\ b_4 \end{pmatrix} = \begin{pmatrix} 8 \\ 4 \\ 18 \end{pmatrix}, \ c = \begin{pmatrix} c_3 \\ c_5 \\ c_6 \end{pmatrix} = \begin{pmatrix} -1/6 \\ -1/6 \\ -2/3 \end{pmatrix} \)
Slack Form (Example)

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

**Slack Form Notation**

- \( B = \{1, 2, 4\}, \ N = \{3, 5, 6\} \)

- \[
    A = \begin{pmatrix}
        a_{13} & a_{15} & a_{16} \\
        a_{23} & a_{25} & a_{26} \\
        a_{43} & a_{45} & a_{46}
    \end{pmatrix}
    = \begin{pmatrix}
        -1/6 & -1/6 & 1/3 \\
        8/3 & 2/3 & -1/3 \\
        1/2 & -1/2 & 0
    \end{pmatrix}
\]

- \[
    b = \begin{pmatrix}
        b_1 \\
        b_2 \\
        b_4
    \end{pmatrix}
    = \begin{pmatrix}
        8 \\
        4 \\
        18
    \end{pmatrix},
    \quad
    c = \begin{pmatrix}
        c_3 \\
        c_5 \\
        c_6
    \end{pmatrix}
    = \begin{pmatrix}
        -1/6 \\
        -1/6 \\
        -2/3
    \end{pmatrix}
\]

- \( v = 28 \)