# 5CCS2FC2: Foundations of Computing II

# Linear Programming

## Week 8

**Dr Christopher Hampson**

*Department of Informatics*

*King's College London*

# Integer Programming

# Integer Programming

## Integer Programming IP

**Input)** — A set of linear constraints $\boldsymbol{A}\vec{\boldsymbol{x}} \leq \vec{\boldsymbol{b}}$,

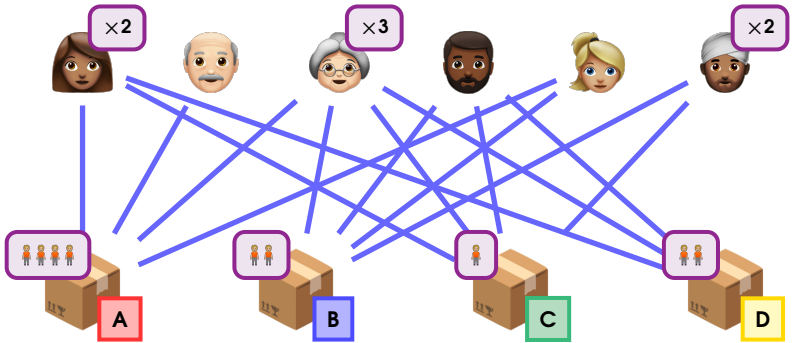
— A linear objective function $\boldsymbol{f} : \mathbb{R}^{\boldsymbol{n}} \rightarrow \mathbb{R}$

**Output)** A solution $\vec{\boldsymbol{x}} \in (\mathbb{Z}^{\geq 0})^{\boldsymbol{n}}$ such that **(i)** $\vec{\boldsymbol{x}}$ satisfies the constraints $\boldsymbol{A}\vec{\boldsymbol{x}} \leq \vec{\boldsymbol{b}}$ and **(ii)** $\boldsymbol{f}(\vec{\boldsymbol{x}})$ is maximal.

# Example

# Integer Programming Example

- Maximal Matchings



## Integer Programming Example

- **Maximal Matchings**

**Step 1)** For each edge $(i, j)$ between worker $i$ and task $j$, let $x_{i,j}$ be a **numerical variable** and add the constraint

$$x_{i,j} \leq 1 \qquad \text{for all edges } (i, j)$$

**Step 2)** Construct a constraint for each **worker** $i \in \{👩🏾, 👩🏻, 👵🏻, 👩🏿, 👩🏼, 👳🏽\}$:

$$x_{i,A} + x_{i,B} + x_{i,C} + x_{i,D} \leq \text{capacity of } i$$

# Integer Programming Example

- **Maximal Matchings**

**Step 3)** Construct a constraint for each **task** $j \in \{A, B, C, D\}$:

$$x_{\text{👩🏾},j} + x_{\text{👴🏻},j} + x_{\text{👵🏻},j} + x_{\text{👩🏾},j} + x_{\text{👱🏼‍♀️},j} + x_{\text{👴🏾},j} \geq \text{requirements for } j$$

**Step 4)** Add an objective to **maximize** the amount of connections

$$\text{Maximize} \quad \sum_{i,j} x_{i,j}$$

- **Maximal Matchings**

**Step 5)** An optimal solution to the resulting ***integer* program** correspond to a **maximal matching** of workers to tasks:

$$
\begin{array}{lll}
\text{Maximise:} & \sum_{i,j} x_{i,j} & \\
\text{Subject to:} & x_{i,j} \leq 1 & \text{for all } i,j \\
& \sum_j x_{i,j} \leq \text{capacity of } i & \text{for all } i \\
& \sum_j x_{i,j} \geq \text{requirements of } j & \text{for all } j \\
& x_{i,j} \in \mathbb{Z} & \text{for all } i,j
\end{array}
$$

## Integer Programming is NP-hard

**Theorem** Integer Programming is **NP**-hard

## Integer Programming is NP-hard

**Theorem** The Boolean Satisfiability problem is polynomially reducible to the Integer Programming problem.

***Proof:*** **Input)** Given a formula $F$ in Conjunctive Normal Form

**Step 1)** We want to **construct** an integer program $\mathbf{IP}_F$ such that

$F$ is satisfiable $\iff$ $\mathbf{IP}_F$ has an integer solution

## Integer Programming is NP-hard

**Step 2)** For each **propositional variable** $P$, let $x_P$ and $x_{\neg P}$ be **numerical variables**

**Step 3)** Convert each **clause** into a linear constraint, as follows:

$$(P \vee Q \vee \neg R)$$
$$(\neg P \vee Q \vee R)$$
$$(\neg Q \vee S)$$

$$\begin{aligned} (x_P + x_Q + x_{\neg R}) &\geq 1 \\ (x_{\neg P} + x_Q + x_R) &\geq 1 \\ (x_{\neg Q} + x_S) &\geq 1 \end{aligned}$$

## Integer Programming is NP-hard

**Step 4)** For each **propositional variable** add the following constraint

$$
\begin{aligned}
(x_P + x_{\neg P}) &\leq 1 \\
(x_Q + x_{\neg Q}) &\leq 1 \\
(x_R + x_{\neg R}) &\leq 1 \\
(x_S + x_{\neg S}) &\leq 1
\end{aligned}
$$

$$
\begin{aligned}
(x_P + x_{\neg P}) &\geq 1 \\
(x_Q + x_{\neg Q}) &\geq 1 \\
(x_R + x_{\neg R}) &\geq 1 \\
(x_S + x_{\neg S}) &\geq 1
\end{aligned}
$$

## Integer Programming is NP-hard

**Step 5)** It is easy to verify that

$F$ is satisfiable $\iff$ $\mathbf{IP}_F$ has an integer-solution

$$\mathbf{SAT} \leq_p \mathbf{IP}$$

**Q.E.D.**

# Branch-and-Bound

# Linear Relaxation and Branch-and-Bound

BRANCH-AND-BOUND $(\mathbf{LP})$:

1. Find a solution $\vec{a}$ to the *linear relaxation* of **LP**
2. **If** $\vec{a} \in \mathbb{Z}^n$ **Return** solution $\vec{a}$
3. **Else**:
    4. Choose a non-integer $x_i = a_i$ on which to branch
    5. Let $\vec{a}_L =$ BRANCH-AND-BOUND $(\mathbf{LP} \cup \{x_i \leq \lfloor a_i \rfloor\})$
    6. Let $\vec{a}_R =$ BRANCH-AND-BOUND $(\mathbf{LP} \cup \{x_i \geq \lceil a_i \rceil\})$
    7. **If** $f(\vec{a}_L) \geq f(\vec{a}_R)$ **Return** solution $\vec{a}_L$
    6. **Return** solution $\vec{a}_R$.

- **Example**

**Step 1)** Solve the **linear relaxation** using the Simplex Method

**(remove the requirement that solution must be integers)**

Maximise: $2x + 3y$

Subject to:

$$\begin{aligned} 3x + 2y &\leq 15 \\ 2y - x &\leq 5 \\ x + 2y &\leq 7 \\ \cancel{x, y \in \mathbb{Z}} & \end{aligned}$$

**Solution**

$x = 4$ and $y = 1.5$

- **Example (cont.)**

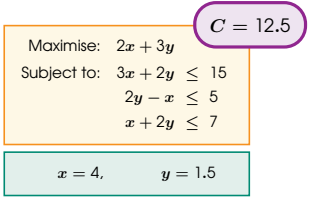
**Step 2)** **Branch** on the non-integer value $y = 1.5$ to create two **sub-problems** by adding the **bounds** $y \leq \lfloor 1.5 \rfloor$ and $y \geq \lceil 1.5 \rceil$, respectively.

$$
\begin{array}{lrcl}
\text{Maximise:} & 2x + 3y & & \\
\text{Subject to:} & 3x + 2y & \leq & 15 \\
& 2y - x & \leq & 5 \\
& x + 2y & \leq & 7 \\
& y & \leq & 1
\end{array}
$$

$$
\begin{array}{lrcl}
\text{Maximise:} & 2x + 3y & & \\
\text{Subject to:} & 3x + 2y & \leq & 15 \\
& 2y - x & \leq & 5 \\
& x + 2y & \leq & 7 \\
& y & \geq & 2
\end{array}
$$

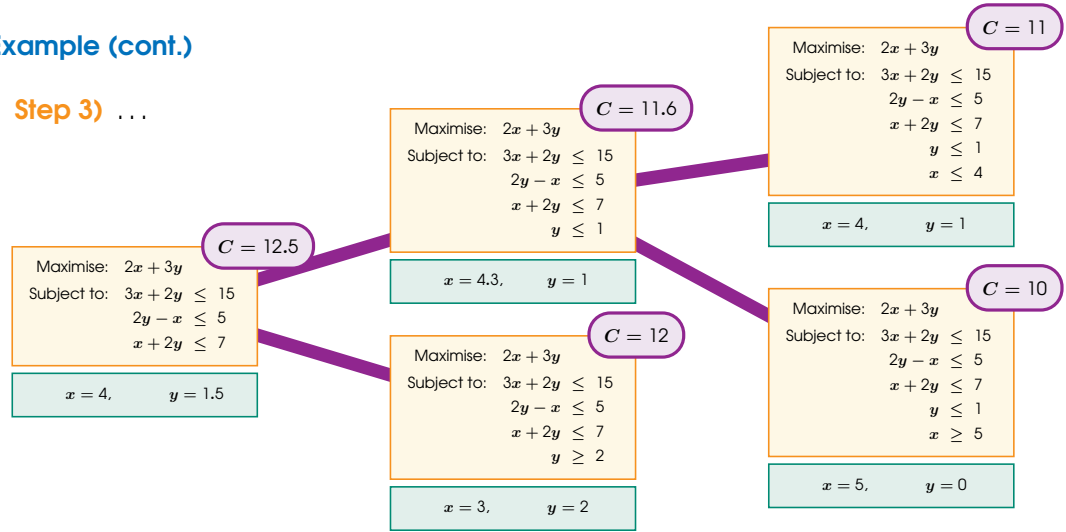
- **Example (cont.)**

**Step 3)** Recursively apply Branch-and-Bound to both sub-problems and return solution which **maximises** the objective function.



- **Example (cont.)**

**Step 3)** ...



- **Example (cont.)**

**Step 4)** Compare and return the **optimal *integral* solution** from the two sub-problems:

$C = 11$

Maximise: $2x + 3y$

Subject to:
$$3x + 2y \leq 15$$
$$2y - x \leq 5$$
$$x + 2y \leq 7$$
$$y \leq 1$$

$x = 5, \quad y = 0$

$C = 12$

Maximise: $2x + 3y$

Subject to:
$$3x + 2y \leq 15$$
$$2y - x \leq 5$$
$$x + 2y \leq 7$$
$$y \geq 2$$

$x = 3, \quad y = 2$

# Other Variants of Linear Programming

## Other Variants of Linear Programming

- **Mixed Integer Linear Programming (MILP)**
  - Hybrid of an **Integer Program** and a classical **Linear Program**,
  - Some variable are required to be **integers**,
  - Others are allowed to take **non-integral** values.
- **Zero-one Integer Programming**
  - A restriction of **Integer Programming**,
  - All variables can be either **zero** or **one**,

# End of Slides!