

# Artificial Intelligence Methods (AE2AIM)

## Lec. 06: Linear Programming

**Huan Jin**

# About me

- Huan (Joyce) Jin
- Email: [huan.jin@nottingham.edu.cn](mailto:huan.jin@nottingham.edu.cn)
- Office: PMB438
- Research interests
  - Artificial Intelligence, OPTimization, Algorithms, Transportation Logistics, Vehicle Routing Problem, Scheduling
  - Web: <https://www.nottingham.edu.cn/en/science-engineering/staffprofile/huan-jin.aspx>

# Agenda

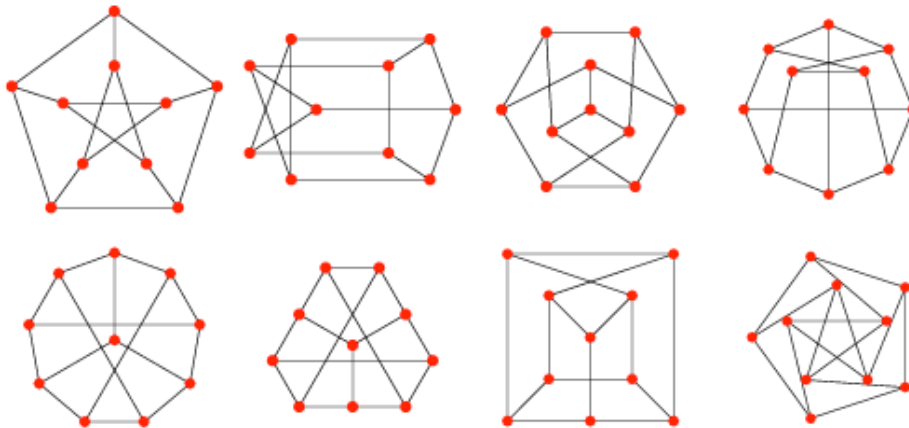
- **Matrix**
- Formulate a Linear Program (LP)
- Linear Programming-General Form
- LP Algorithm: The Simplex Method
- Integer Program (IP)
- IP Algorithm: Branch and bound
- Classic LP and IP Solvers

# Matrices

- Matrix notation
- Matrix addition and subtraction
- Matrix multiplication
- Matrix Product
- Matrix transposition
- Symmetric matrix

# Applications of matrices

- Linear Programming
- Markov Chains
- Graph Theory
- Cryptography
- Image processing
- Machine Learning



# Agenda

- Matrix
- **Formulate a Linear Program (LP)**
- Linear Programming-General Form
- LP Algorithm: The Simplex Method
- Integer Program (IP)
- IP Algorithm: Branch and bound
- Classic LP and IP Solvers

# Linear Programming

Linear programming (LP) is a mathematical optimization technique used to determine the best possible outcome (such as maximum profit, minimum cost, or efficient resource allocation) within a model defined by linear relationships.

- **Objective Function:** A linear equation representing the goal to maximize (e.g., profit) or minimize (e.g., cost).  
*Example:* Maximize  $Z=3x+5y$ , where  $x$  and  $y$  are decision variables.
- **Decision Variables:** The quantities to be determined (e.g., units of products to produce).
- **Constraints:** Linear inequalities or equations that limit the feasible solutions (e.g., resource availability, time).  
*Example:*  $2x+4y\leq 100$  (resource constraint).

# A basic example to wake up 😊

**A Hot Tub factory produces two types of hot tubs: Aqua-Spas & Hydro-Luxes. Each tub consists of common materials and uses the same labor skills. However, pumps and tubing (common to both) are scarce and expensive. The following data is relevant to the problem of determining the **optimal product mix**:**

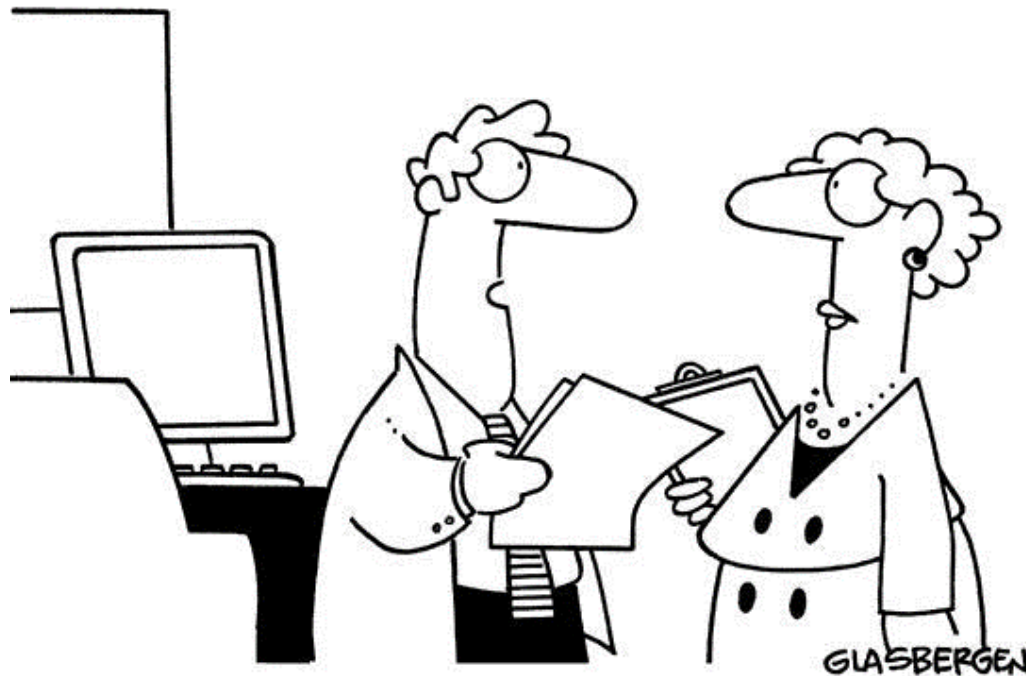
- The Aqua-Spas use one pump and 12 feet of tubing and require 9 hours of labor. Hydro-Luxes also use one pump but require 16 feet of tubing and 6 hours of labor to build.
- Each Aqua-Spa can be sold for a profit of \$350 while each Hydro-Lux generates \$300. There are 200 pumps, 1566 hours of labor, and 2880 feet of tubing available.

**How can we maximize the profit?**



# Solution approach (Three basic questions)

- 1. What needs to be solved? What is the problem? What are my levers? i.e. What are the Decision Variables?**



**“My team has created a very innovative solution,  
but we’re still looking for a problem to go with it.”**

# Solution approach (Three basic questions)

## 2. What is the criterion? How do we evaluate the solutions? i.e. What is the Objective Function?



# Solution approach (Three basic questions)

## 3. What are the limitations? What are my constraints?



# Formulate a LP

- **Step 1: Determine Decision Variables**

$x_1$  = The number of Aqua-Spas produced

$x_2$  = The number of Hydro-Lux produced

- **Step 2: Formulate Objective Function**

$$\text{Max } 350x_1 + 300x_2$$

- **Step 3: Formulate Constraints**

$$x_1 + x_2 \leq 200$$

$$9x_1 + 6x_2 \leq 1,566$$

$$12x_1 + 16x_2 \leq 2,880$$

# Formulate a LP

$$\begin{array}{ll}
 \text{MAX: } 350x_1 + 300x_2 & \text{\} profit \\
 \text{s.t.:} & \\
 & x_1 + x_2 \leq 200 \quad \text{\} pumps} \\
 & 9x_1 + 6x_2 \leq 1566 \quad \text{\} labor} \\
 & 12x_1 + 16x_2 \leq 2880 \quad \text{\} tubing} \\
 & x_1 \geq 0 \quad \text{\} non-negativity} \\
 & x_2 \geq 0 \quad \text{\} non-negativity}
 \end{array}$$

Column 1
Column 2

# Agenda

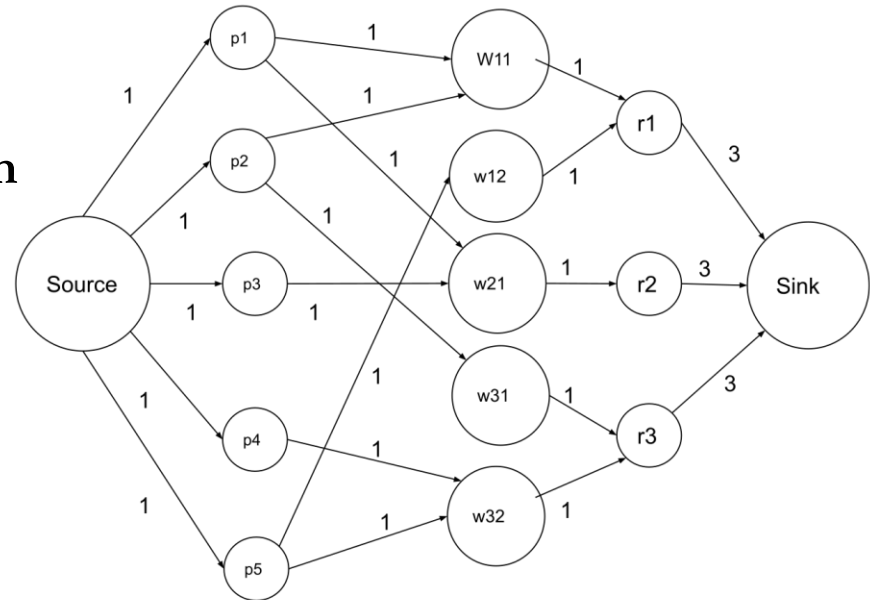
- Matrix
- Formulate a Linear Program (LP)
- **Linear Programming-General Form**
- LP Algorithm: The Simplex Method
- Integer Program (IP)
- IP Algorithm: Branch and bound
- Classic LP and IP Solvers

# Linear Programming (LP) – example 1

## Maximum flow problem

- Directed graph  $G=(V,E)$
- $c_{ij}$ : capacity on edge  $(i,j)$
- $x_{ij}$ : flow on edge  $(i,j)$
- **Objective:** maximize the flow from source to sink in the network

$$\begin{aligned} \max \quad & \sum_{j:(s,j) \in E} x_{sj} \\ \text{s.t.} \quad & x_{ij} \leq c_{ij}, \quad (i,j) \in E \\ & \sum_{i:(i,j) \in E} x_{ij} = \sum_{i:(j,i) \in E} x_{ji}, \quad i \in V \\ & x_{ij} \geq 0, \quad (i,j) \in E \end{aligned}$$



source: [wiki](#)

# Classic Optimisation Problems

- **Maximum flow problem:**

Given a capacitated direct graph  $G(N, A)$  with a source  $s \in N$  and sink  $d \in N$ .

What is the maximum flow from  $s$  to  $d$ ?

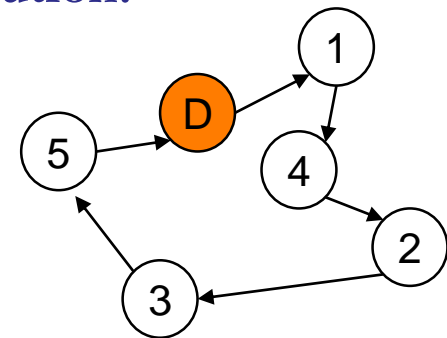
- How many possible solutions?  
How to solve it?

- **1D Knapsack problem:** Given a knapsack of capacity  $C$  and a set of  $n$  items, each item  $i$  has a volume  $s_i$  and a value  $v_i$ . Determine the list of items to be packed so that the total value is maximized.

- How many possible solutions?  
How do we solve it?

- **TSP (Traveling Salesman Problem)**

- How many possible solution in total? How to find the best solution?



**Among the 3 problems, which one is easier to solve, which is harder to solve?**



**What are these problems  
in common?**

# A standard LP formulation

- Standard format

$$\mathbf{max} \quad c_1x_1 + c_2x_2 + c_3x_3 + \dots + c_nx_n$$

*s.t.*

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots a_{1n}x_n \leq b_1$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \dots a_{2n}x_n \leq b_2$$

...

...

...

$$a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \dots a_{mn}x_n \leq b_m$$

- Matrix format

$$\mathbf{max} \quad \mathbf{c}'\mathbf{x}$$

subject to

$$\mathbf{Ax} \leq \mathbf{b}$$

$$\mathbf{x} \geq \mathbf{0}$$

---

$$\mathbf{min} \quad \mathbf{c}'\mathbf{x}$$

subject to


$$\mathbf{Ax} \geq \mathbf{b}$$

$$\mathbf{x} \geq \mathbf{0}$$

This type of problem can be solved fairly efficiently thanks to a famous algorithm, Simplex method, by George B. Dantzig.

# LP– General Formulation

minimize  $\mathbf{c}'\mathbf{x}$   
subject to  $\mathbf{Ax} = \mathbf{b}$   
 $\mathbf{x} \geq \mathbf{0}$



$\mathbf{c} = \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}$        $\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$   
 $\mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$

minimize  $\sum_{i=1}^n c_i x_i$       Objective Function

subject to  $\sum_{j=1}^n a_{ij} x_j = b_i$        $\forall i$       Regular Constraints

$x_i \geq 0$        $\forall i$       Non-negativity Constraints

# Agenda

- Matrix
- Formulate a Linear Program (LP)
- Linear Programing-General Form
- **LP Algorithm: The Simplex Method**
- Integer Program (IP)
- IP Algorithm: Branch and bound
- Classic LP and IP Solvers

# Formulate a LP

$$\begin{array}{ll}\text{Max} & 350 x_1 + 300 x_2 \\ \text{s.t.:} & x_1 + x_2 \leq 200 \\ & 9 x_1 + 6 x_2 \leq 1566 \\ & 12 x_1 + 16 x_2 \leq 2880 \\ & x_1 \geq 0 \\ & x_2 \geq 0\end{array}$$

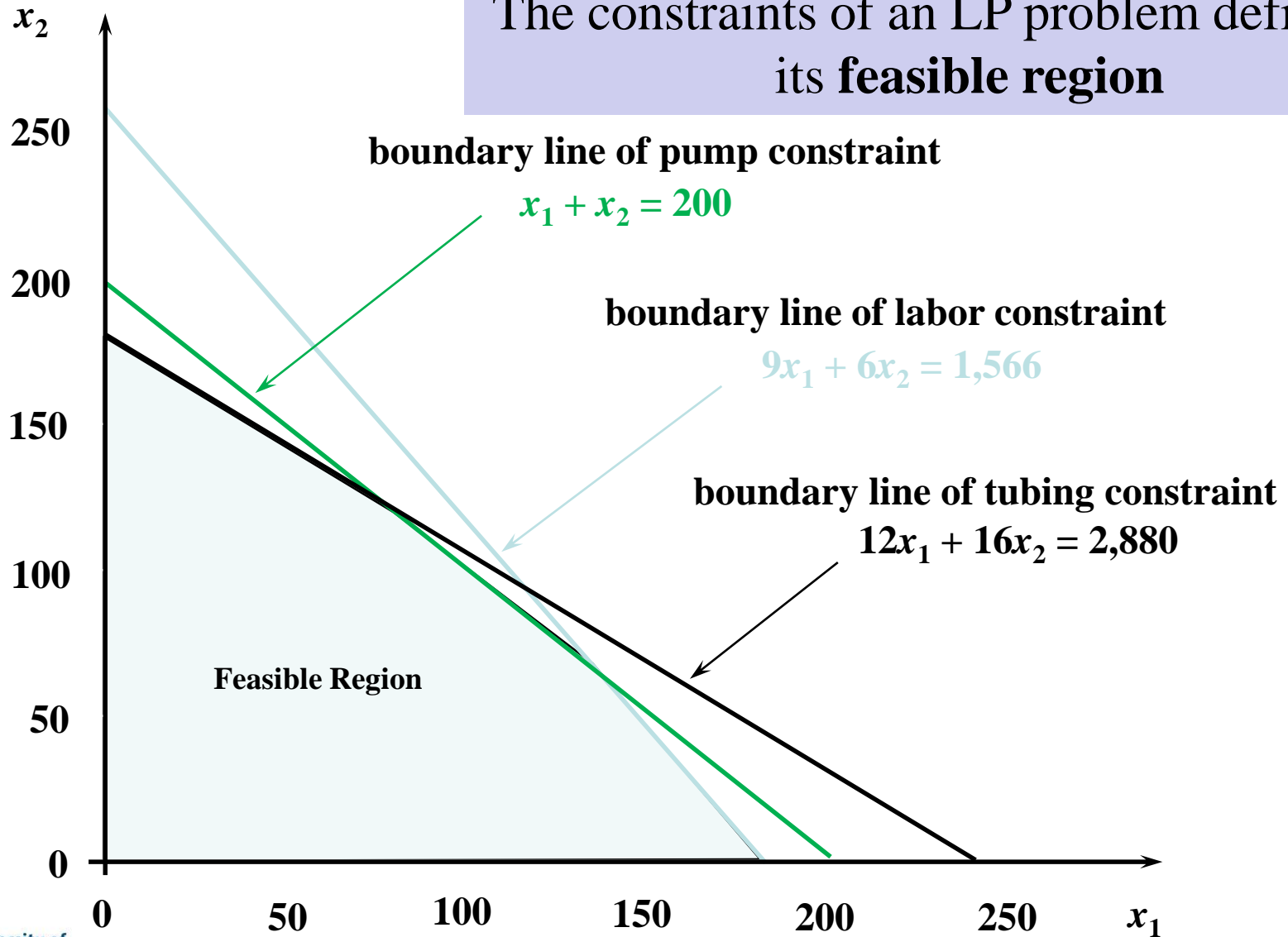
Objective  
Function

“Regular”  
Constraints

Non-negativity  
Constraints

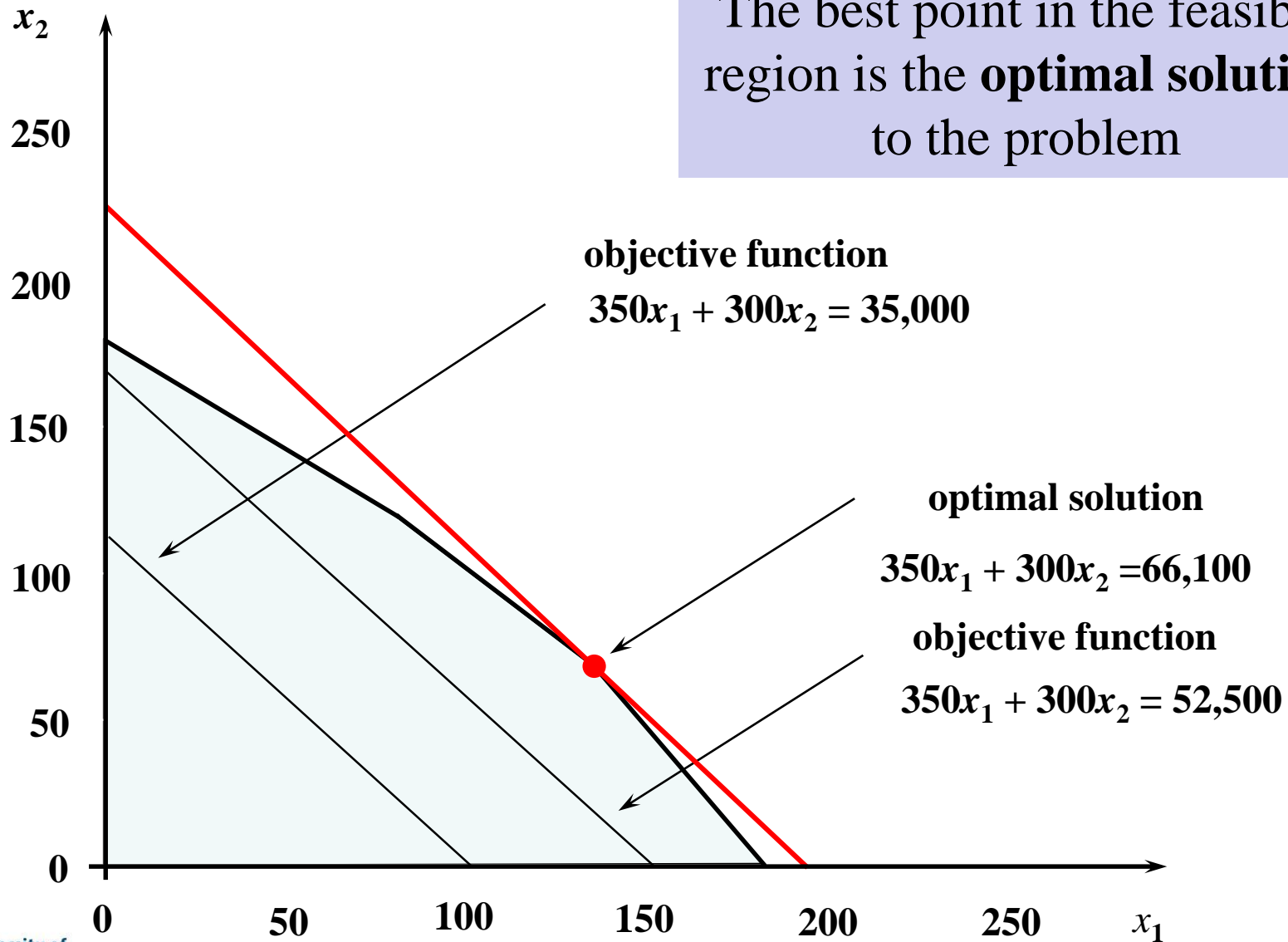
# Solving LP Problems: A Graphical Approach

The constraints of an LP problem defines its **feasible region**

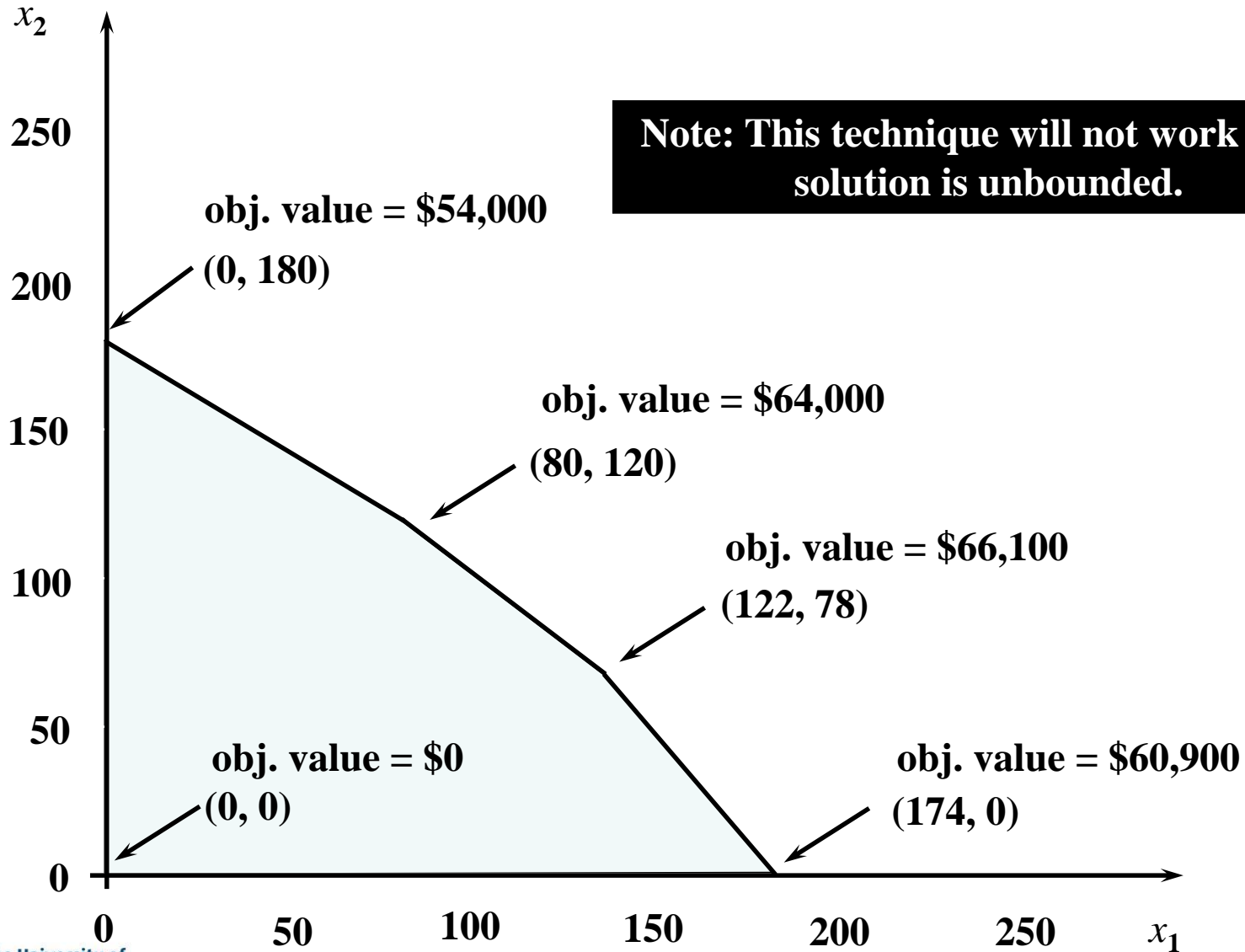


# Solving LP Problems: A Graphical Approach

The best point in the feasible region is the **optimal solution** to the problem



# Enumerating the Corner Points

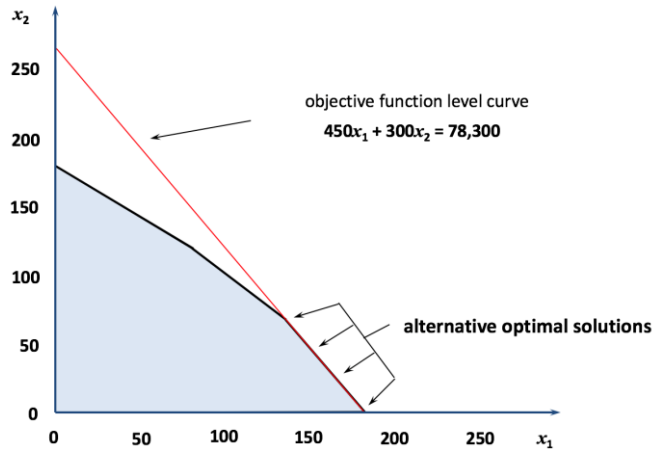




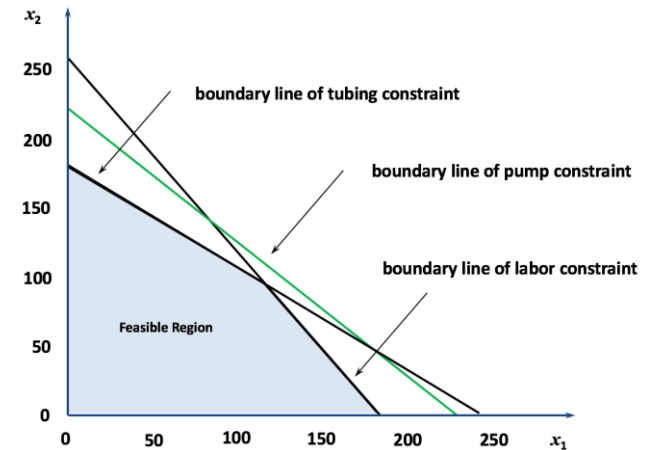
# Special Conditions in LP Models

## A number of anomalies can occur in LP problems:

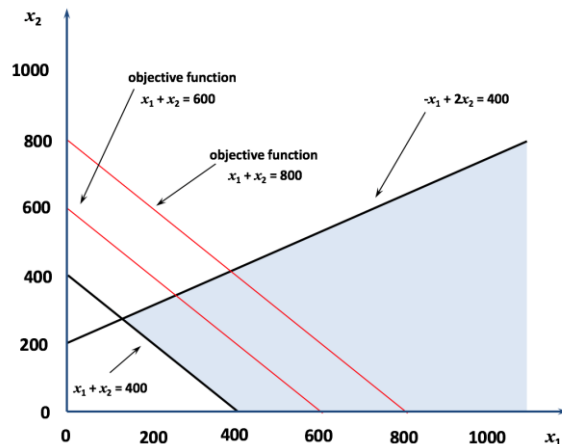
### Alternative Optimal Solutions



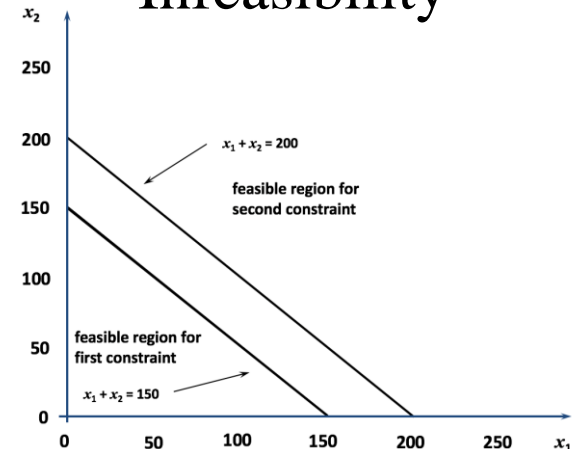
### Redundant Constraints



### Unbounded Solutions



### Infeasibility

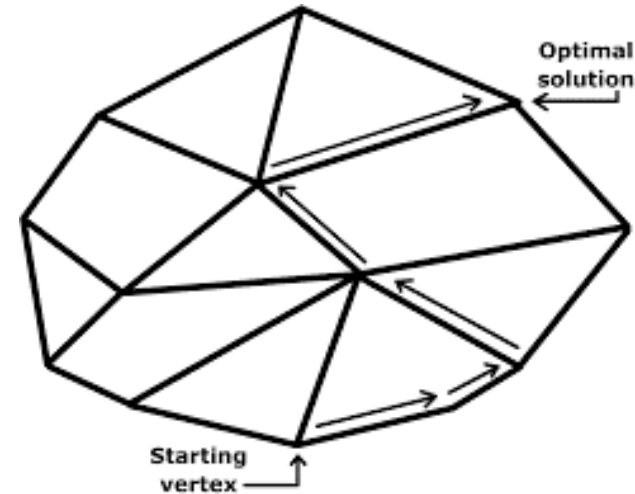


# Algorithmic Approaches

## Simplex method

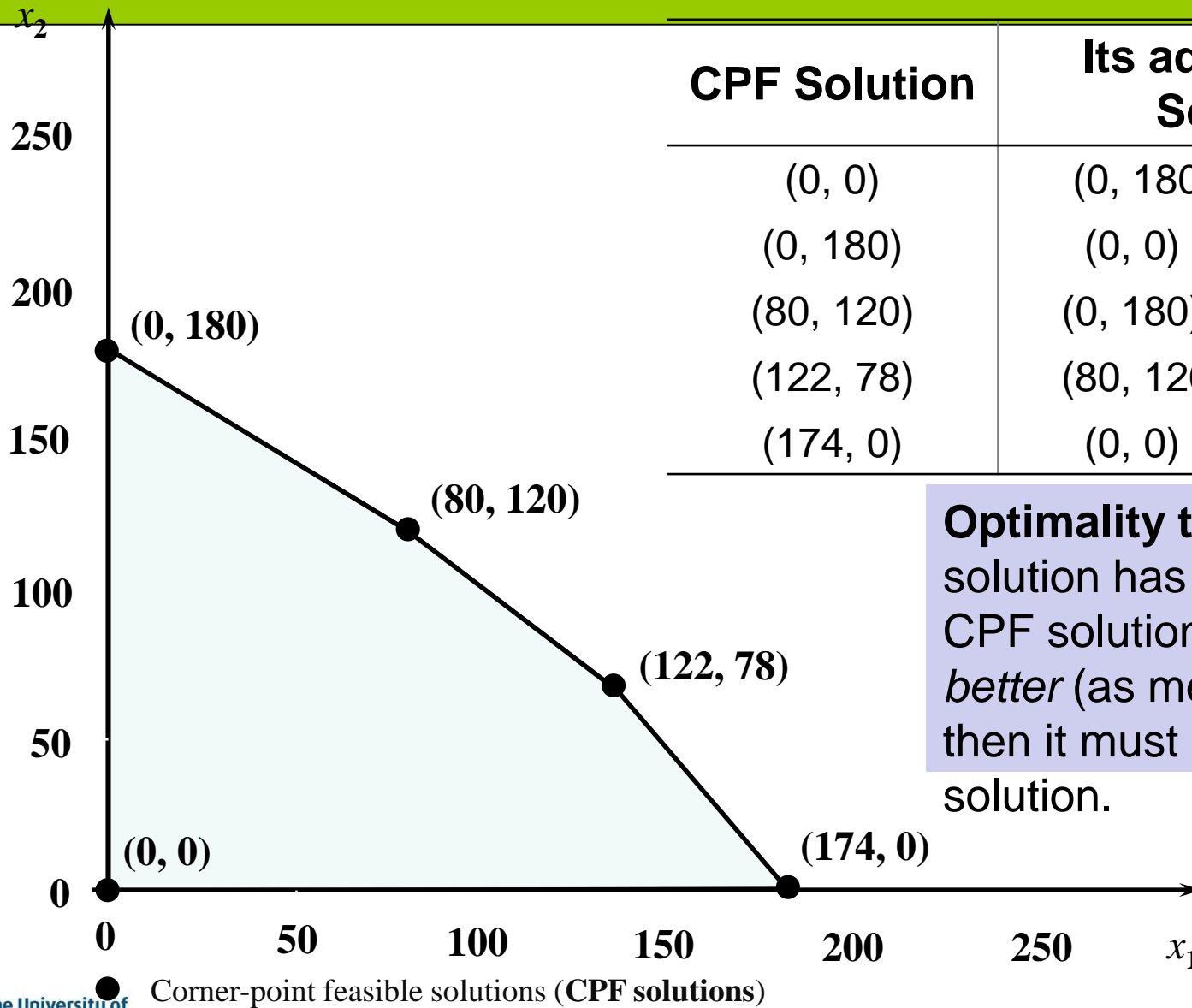
It focuses on CPF solutions. In the worst case, it requires a number of steps that is exponential in the size of the problem.

However, it performs well in practice, visiting only a small fraction of the total number of vertices.



	$y_1$	$s_2$	$\dots$	$s_n$	
$s_1$	$\hat{a}_{11}$	$\hat{a}_{12}$	$\dots$	$\hat{a}_{1n}$	$\hat{b}_1$
$y_2$	$\hat{a}_{21}$	$\hat{a}_{22}$	$\dots$	$\hat{a}_{2n}$	$\hat{b}_2$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
$y_m$	$\hat{a}_{m1}$	$\hat{a}_{m2}$	$\dots$	$\hat{a}_{mn}$	$\hat{b}_m$
1	$-\hat{c}_1$	$-\hat{c}_2$	$\dots$	$-\hat{c}_n$	$\hat{v}$

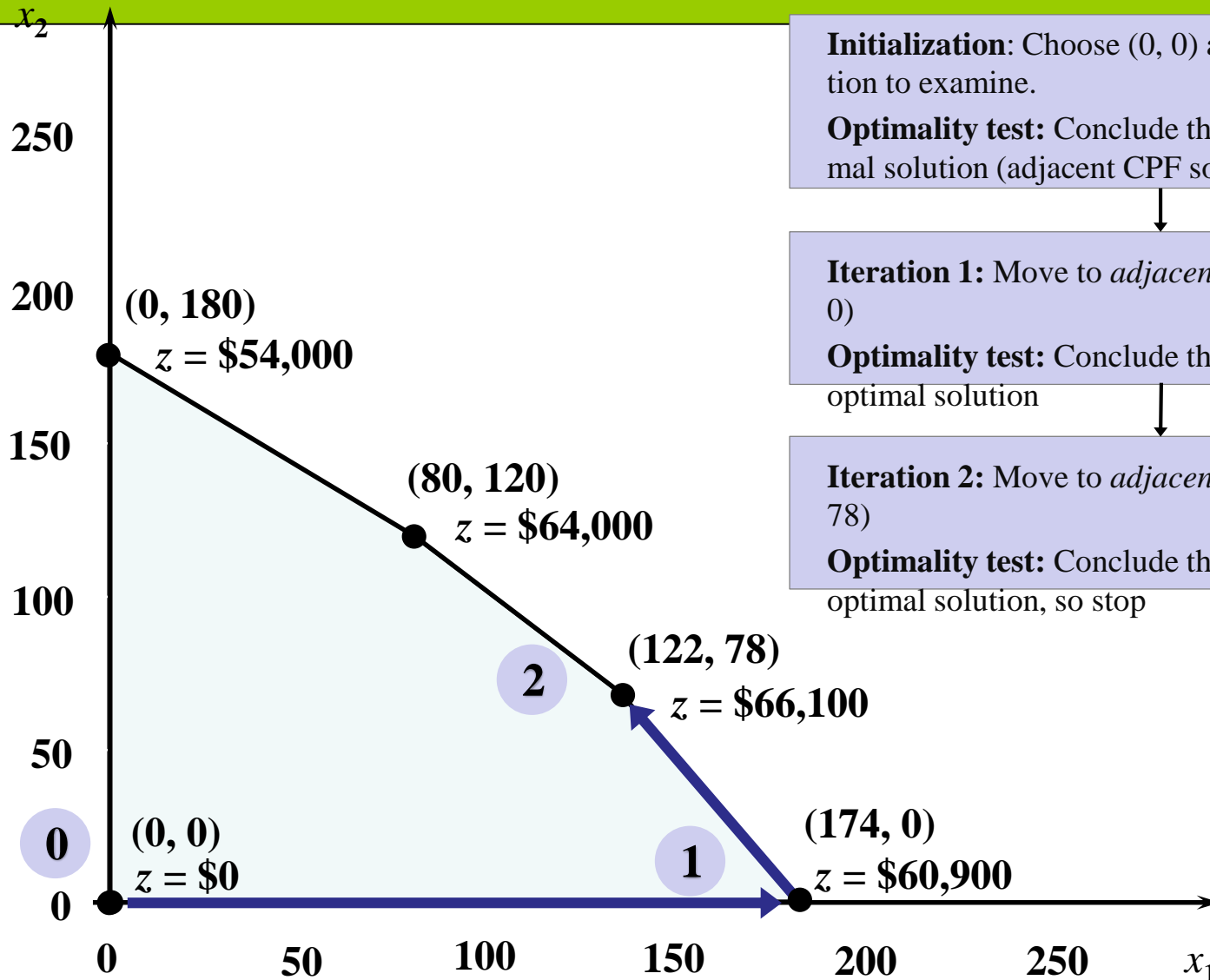
# The essence of the Simplex method



CPF Solution	Its adjacent CPF Solutions
(0, 0)	(0, 180) and (174, 0)
(0, 180)	(0, 0) and (80, 120)
(80, 120)	(0, 180) and (122, 78)
(122, 78)	(80, 120) and (174, 0)
(174, 0)	(0, 0) and (122, 78)

**Optimality test:** If a CPF solution has no *adjacent* CPF solutions that are *better* (as measured by  $Z$ ), then it must be an *optimal* solution.

# The essence of the Simplex method



**Initialization:** Choose  $(0, 0)$  as the *initial* CPF solution to examine.

**Optimality test:** Conclude that  $(0, 0)$  is *not* an optimal solution (adjacent CPF solutions are better)

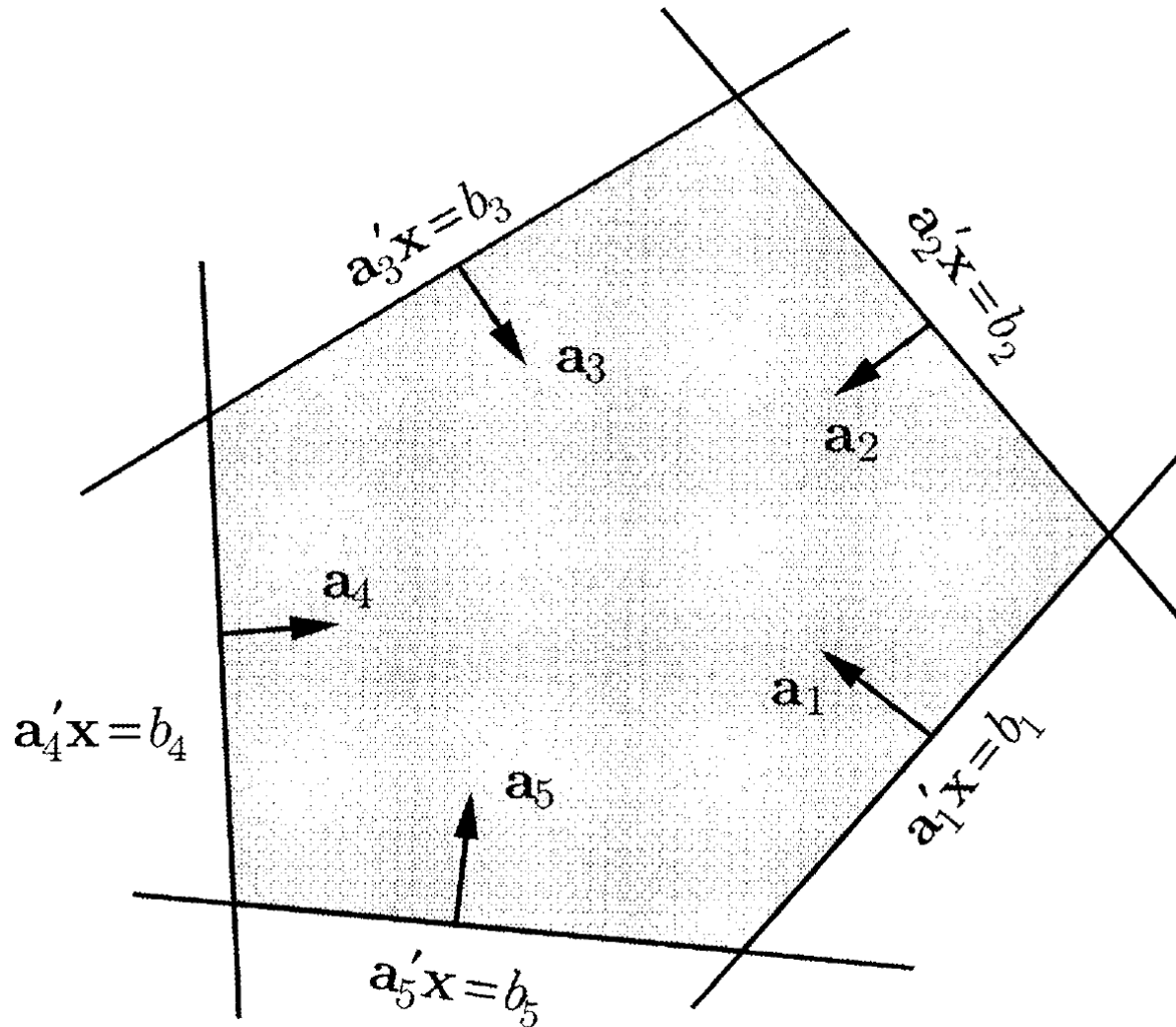
**Iteration 1:** Move to *adjacent* CPF solution,  $(174, 0)$

**Optimality test:** Conclude that  $(174, 0)$  is *not* an optimal solution

**Iteration 2:** Move to *adjacent* CPF solution,  $(122, 78)$

**Optimality test:** Conclude that  $(122, 78)$  is an optimal solution, so stop

# The Convex Hull



The convex hull is the smallest convex set that contains  $X$

Source: Bertsimas, D., & Tsitsiklis, J. N. (1997). Introduction to linear optimization. Belmont, MA: Athena Scientific.

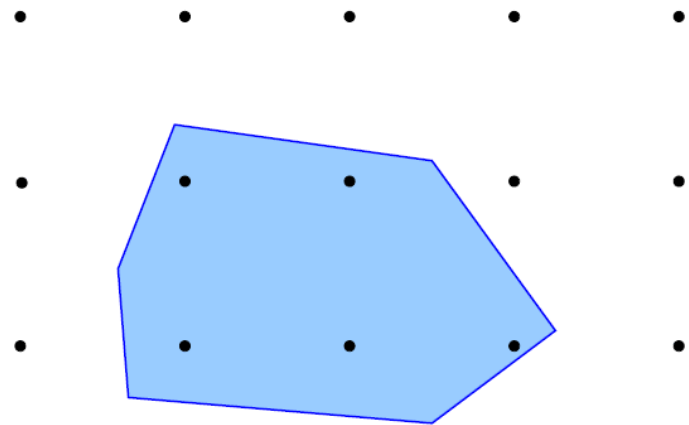
# Agenda

- Matrix
- Formulate a Linear Program (LP)
- Linear Programming-General Form
- LP Algorithm: The Simplex Method
- **Integer Program (IP)**
- IP Algorithm: Branch and bound
- Classic LP and IP Solvers

# LP and IP

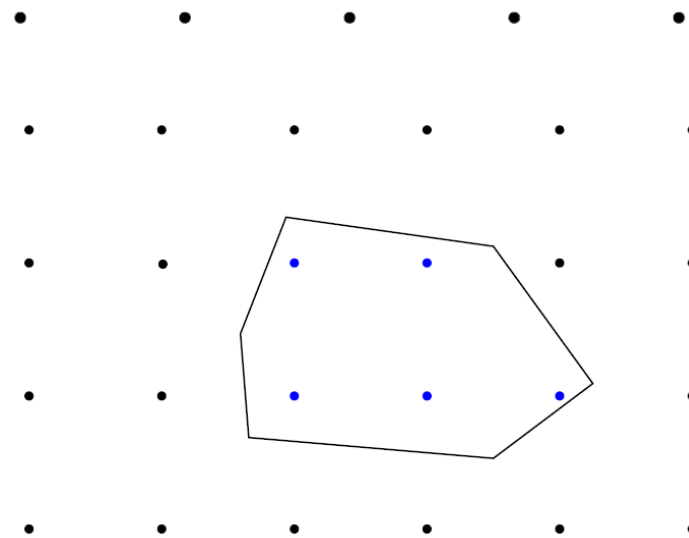
minimize  
subject to

$$\begin{aligned} & \mathbf{c}'\mathbf{x} \\ & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned}$$



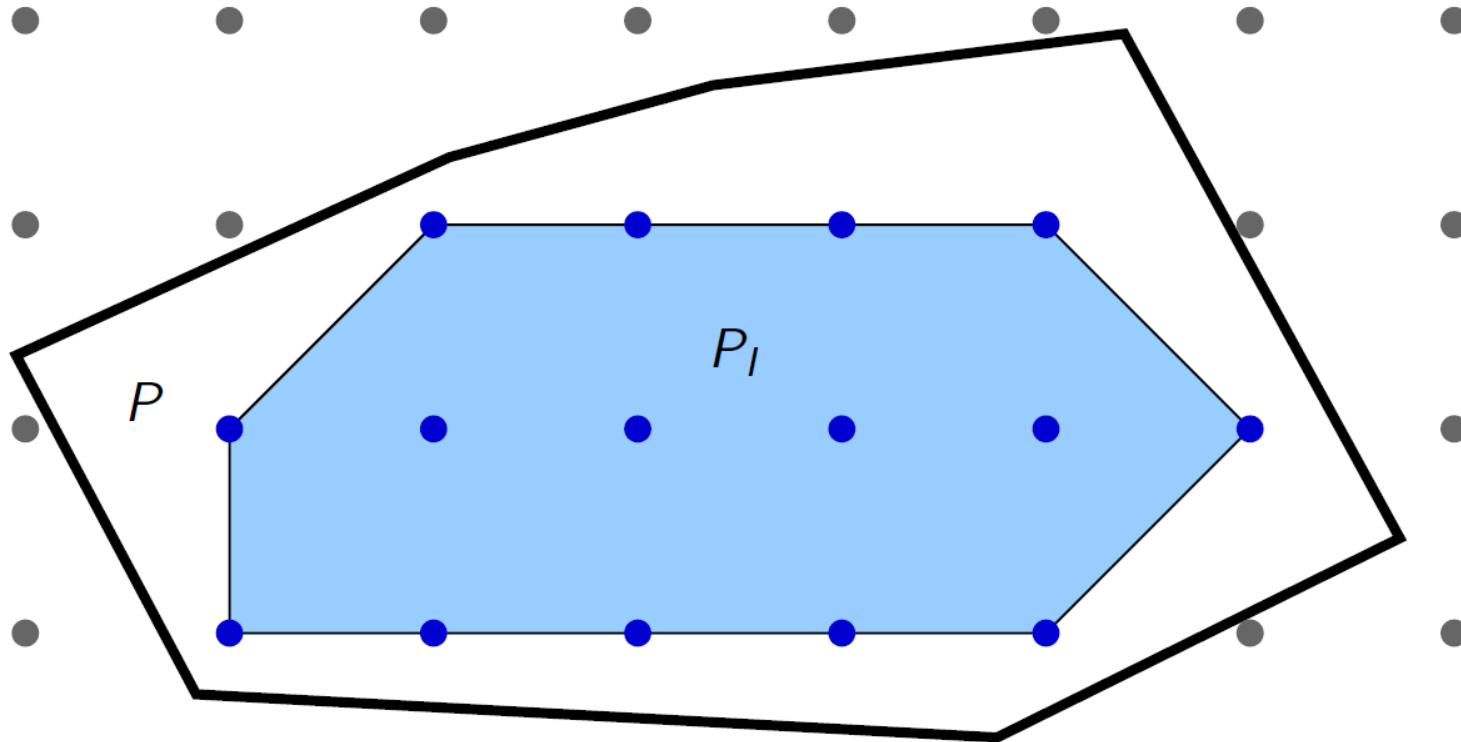
minimize  
subject to

$$\begin{aligned} & \mathbf{c}'\mathbf{x} \\ & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \in \mathbb{Z}_+^n \end{aligned}$$



Important special case: Binary  
Programming:  $\mathbf{x} \in \{0,1\}^n$

# Integer Hull Vs Convex Hull



**Integer optimization over  $P \Leftrightarrow$  linear optimization over  $P_I$**



# Modeling with binary variables

minimize	$\sum_{j=1}^n c_j x_j$	Objective Function
subject to	$\sum_{j=1}^n a_j x_j \leq b$	Constraints
	$x_j \in \{0,1\} \quad \forall j$	Binary variables

## Potential relations:

$$\sum_j x_j \leq 1$$

At most one event occurs

$$x_2 - x_1 = 0$$

Neither or both events occur

$$x_2 \leq x_1$$

If one event occurs then, another occurs

$$y \leq Ux_1$$

If  $x_1 = 0$ , then  $y = 0$ ; if  $x_1 = 1$ , then  $y$  is bounded by  $U$

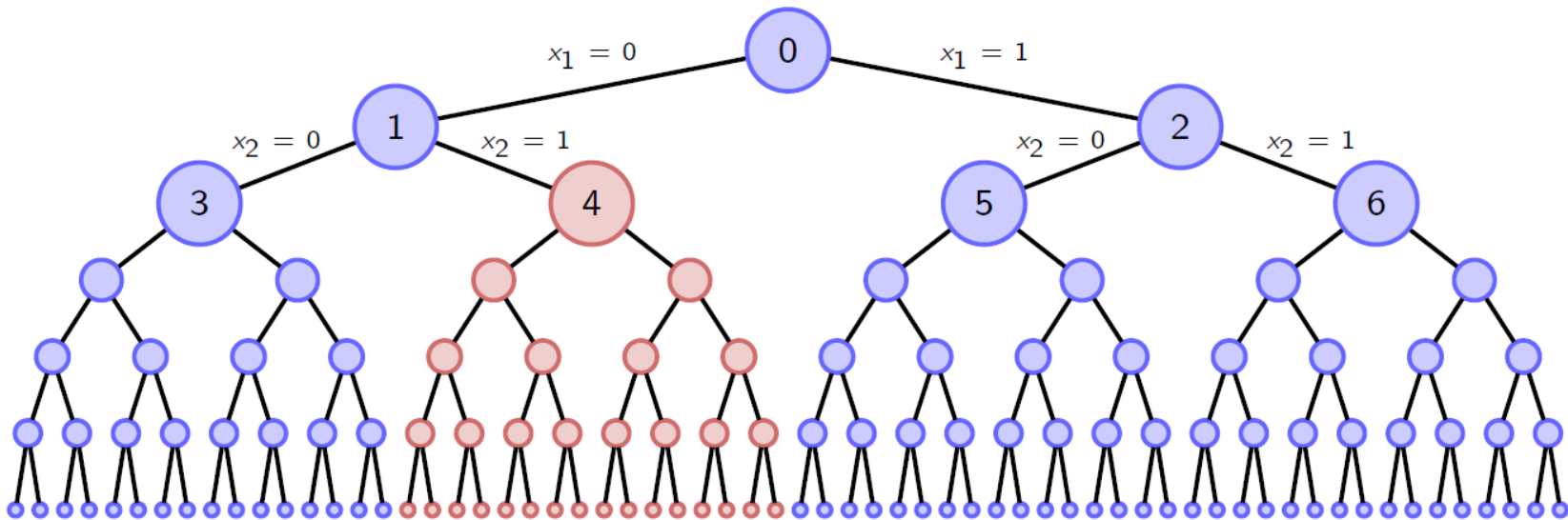
# Agenda

- Matrix
- Formulate a Linear Program (LP)
- Linear Programming-General Form
- LP Algorithm: The Simplex Method
- Integer Program (IP)
- **IP Algorithm: Branch and bound**
- Classic LP and IP Solvers

# Solution Methods

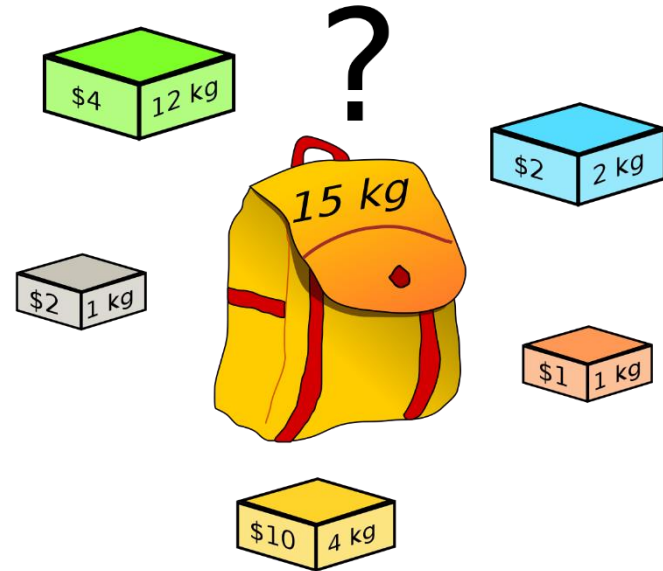
## Branch-and-Bound

Idea: enumerate all possible solutions systematically using a tree (branch). Stop branching from a node as soon as possible (e.g. suppose we look at node 4 and conclude none of its descendants can be optimal), thus we can eliminate 1/4 the solutions at once!



# Knapsack Problem

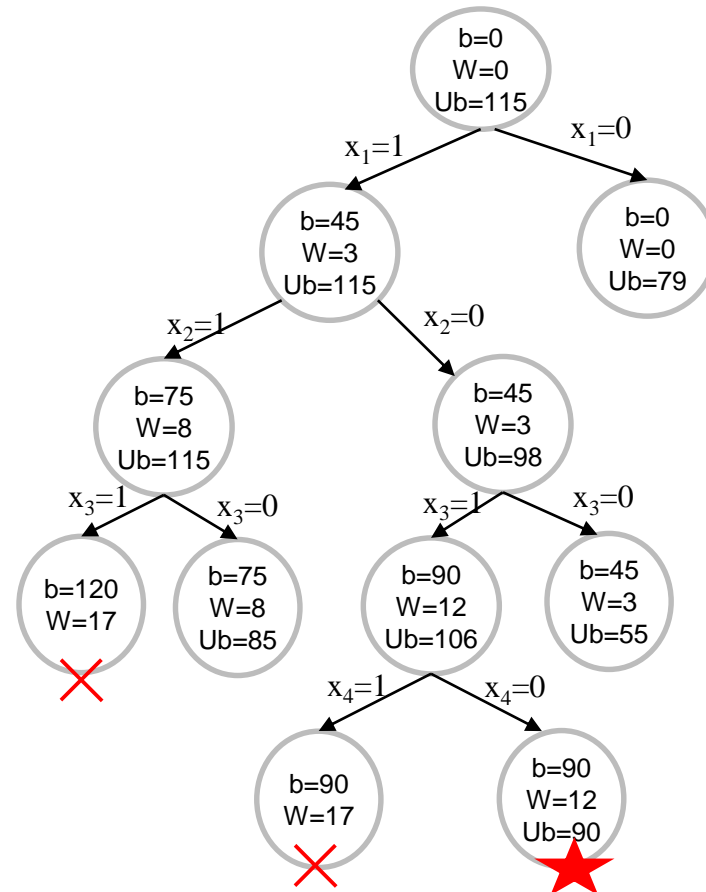
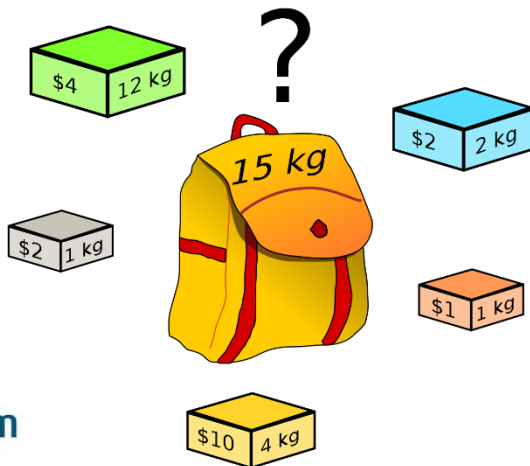
- Problem setting
  - Bag size: 16
  - Benefit of items: {45, 30, 45, 10}
  - Weight of items: {3, 5, 9, 5}
  - Benefit weight ratio: {15, 6, 5, 2}



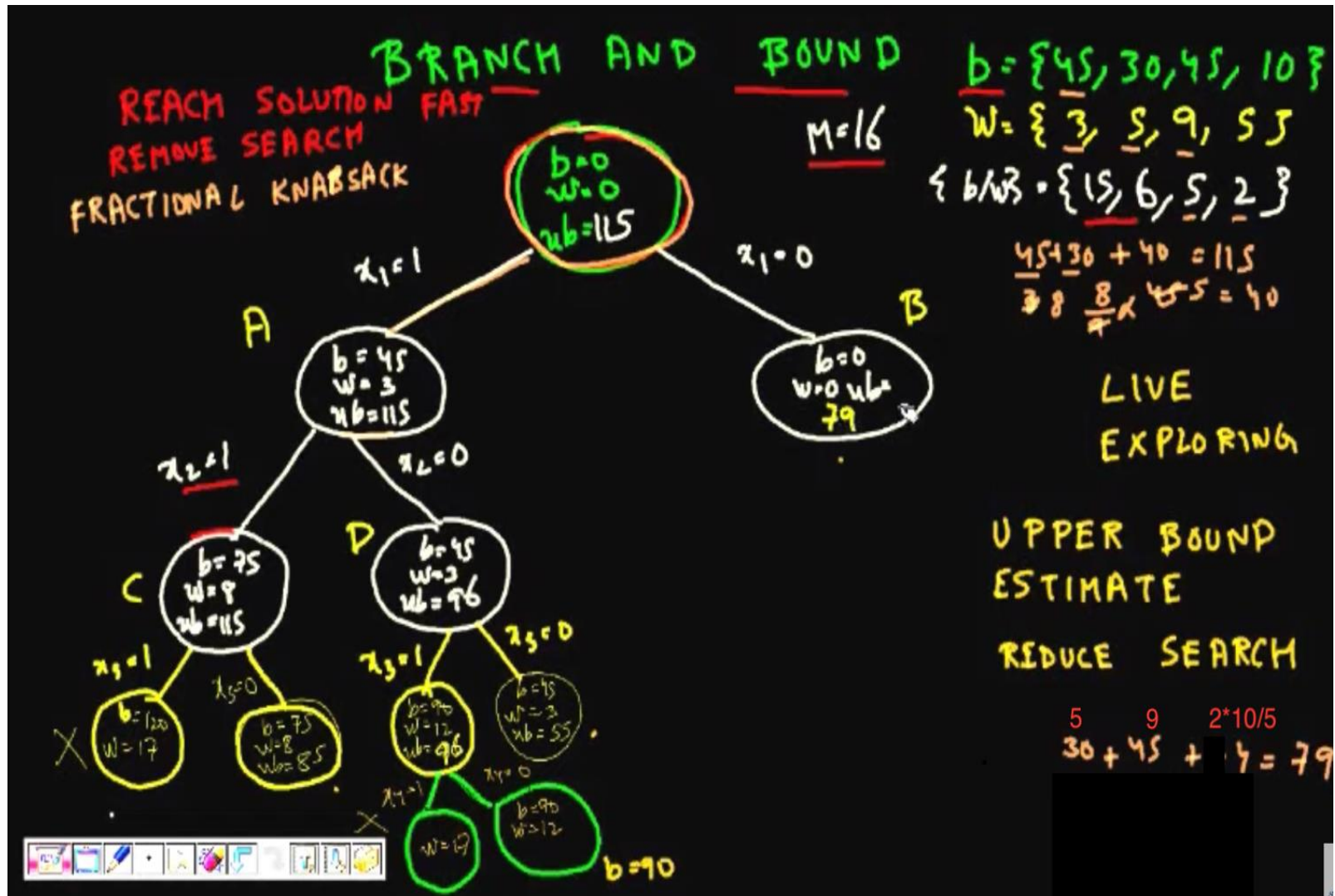
# Knapsack Problem

- Problem setting

- Bag size: 16
- Benefit of items: {45, 30, 45, 10}
- Weight of items: {3, 5, 9, 5}
- Benefit weight ratio: {15, 6, 5, 2}



# Branch and Bound



Source: <https://www.youtube.com/watch?v=R6BQ3gBrfjQ>

# Agenda

- Matrix
- Formulate a Linear Program (LP)
- Linear Programming-General Form
- LP Algorithm: The Simplex Method
- Integer Program (IP)
- IP Algorithm: Branch and bound
- **Classic LP and IP Solvers**

# LP Solvers

- LP problems involving 1000s of variables and 1000s of constraints are now routinely solved with computer packages.
- Linear programming solvers are now part of many spreadsheet packages, such as Microsoft Excel.
- For the previous example, we can easily solve it using Excel solver.
- Leading commercial packages include CPLEX, GUROBI, LINGO, MOSEK, Xpress-MP, and Premium Solver for Excel.



# Further Reference

- **Reference**

- Introduction to operations research / Frederick S. Hillier, Gerald J. Lieberman. McGraw-Hill 2015 Available at Main Library Room

- **Getting familiar with an LP solver**

- IBM Cplex, **free** for academics and students
- Supports C/C++, .Net, Java, R, Python etc...
- <https://ibm.onthehub.com/WebStore/ProductSearchOfferingList.aspx?srch=ilog+cplex>
- <https://www.ibm.com/products/ilog-cplex-optimization-studio>

- **Using IBM Cplex to solve the relaxed knapsack problem**

- You can either use the OPL or
- Call LP solver from a C/C++ program