# Artificial Intelligence Methods for Social Good

# Lecture 2

# Basics of (Integer) Linear Programming

17-537 (9-unit) and 17-737 (12-unit)
Fei Fang
feifang@cmu.edu

# Reminder

- Confirm course project group members
  - Due 1/23, 10pm
- Online Homework 0 (HW0)
  - Required, but worth zero points, Due 1/23, 10 pm
- Paper Reading Assignment 1 (PRA1)
  - Due 1/25, 10 pm
- Project proposal
  - Due 1/30, 10pm

# Outline

- Linear Programming
- Integer Linear Programming
- Exercise: Planning in food rescue
- Discussion

# Learning Objectives

- Understand the concept of
  - Linear Programming (LP)
  - Integer Linear Programming (ILP)
  - LP Relaxation
- Briefly describe the following algorithm
  - Simplex algorithm
- Formulate problems as LP/ILP and use solvers to solve them

# Linear Program: Definition

- Linear Program
  - A special case of convex optimization problem
  - An optimization problem whose optimization objective is a linear function and feasible region is defined by a set of linear constraints

$$\max_x c^{\mathrm{T}} x$$
$$\text{s.t. } Gx \leq h$$

Note: can also be minimization

  - $c \in \mathbb{R}^n$
  - $G \in \mathbb{R}^{m \times n}, h \in \mathbb{R}^m$

# Linear Program: Example

- Example: Maximize Income in Manufacturing

| | Price | Labor | Machine |
|---|---|---|---|
| Product 1 | \$30 | 0.2 hours | 4 hours |
| Product 2 | \$30 | 0.5 hours | 2 hours |
| Total | | <=90 | <=800 |

$$\max_{x,y} 30x + 30y$$

**s.t.**

$$0.2x + 0.5y \leq 90$$

$$4x + 2y \leq 800$$

$$x \geq 0, y \geq 0$$

$$\max_{x} c^{\mathrm{T}} x$$

**s.t.** $Gx \leq h$

$$c = \begin{bmatrix} 30 \\ 30 \end{bmatrix}$$

$$G = \begin{bmatrix} 0.2 & 0.5 \\ 4 & 2 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}, h = \begin{bmatrix} 90 \\ 800 \\ 0 \\ 0 \end{bmatrix}$$

# Linear Program: Example

- Example: Maximize Income in Manufacturing

| | Price | Labor | Machine |
|---|---|---|---|
| Product 1 | \$30 | 0.2 hours | 4 hours |
| Product 2 | \$30 | 0.5 hours | 2 hours |
| Total | | <=90 | <=800 |

$$\max_{x,y} 30x + 30y$$

s.t.

$$0.2x + 0.5y \leq 90$$

$$4x + 2y \leq 800$$

$$x \geq 0, y \geq 0$$

# Linear Program: Example

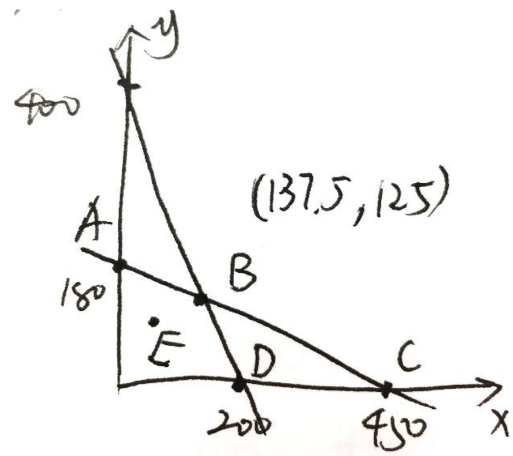
- Example: Maximize Income in Manufacturing

| | Price | Labor | Machine |
|---|---|---|---|
| Product 1 | $30 | 0.2 hours | 4 hours |
| Product 2 | $30 | 0.5 hours | 2 hours |
| Total | | <=90 | <=800 |

$$\max_{x,y} 30x + 30y$$

s.t.

$$0.2x + 0.5y \leq 90$$
$$4x + 2y \leq 800$$
$$x \geq 0, y \geq 0$$



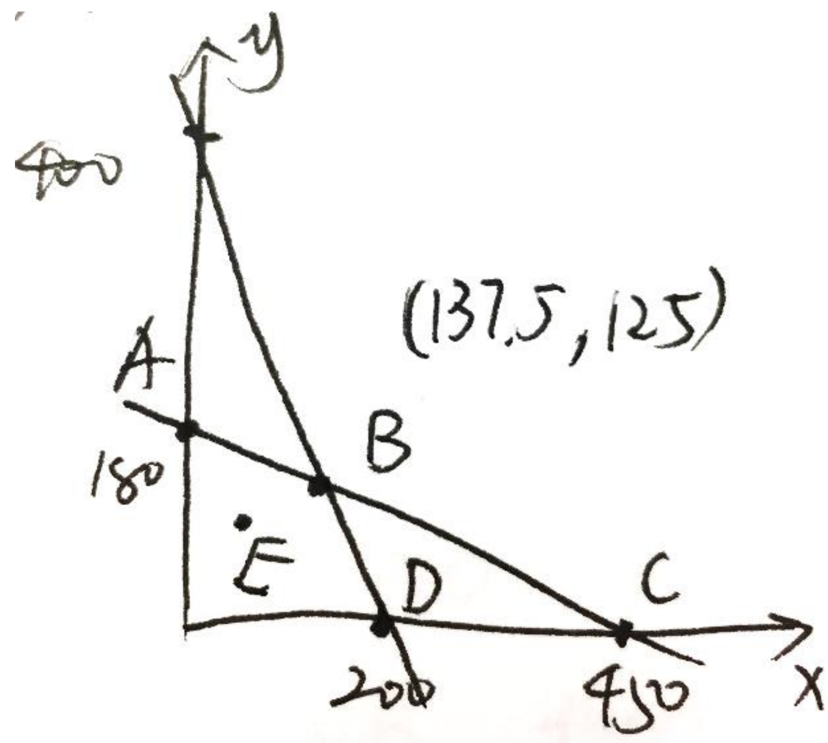
# Poll 1: Linear Program

- Which constraints determine point B?
  - A: $x = 0$
  - B: $y = 0$
  - C: $0.2x + 0.5y = 90$
  - D: $4x + 2y = 800$
  - E: I don't know

$$\max_{x,y} 30x + 30y$$

s.t.

$$0.2x + 0.5y \leq 90$$
$$4x + 2y \leq 800$$
$$x \geq 0, y \geq 0$$



# Methods for Solving Linear Programs

- Simplex Algorithm
- Ellipsoid algorithm
  - Polynomial-time in theory, poor performance in practice
- Interior Point Method

## Simplex Algorithm

- At least one vertex of the polytope is an optimal solution if feasible and bounded

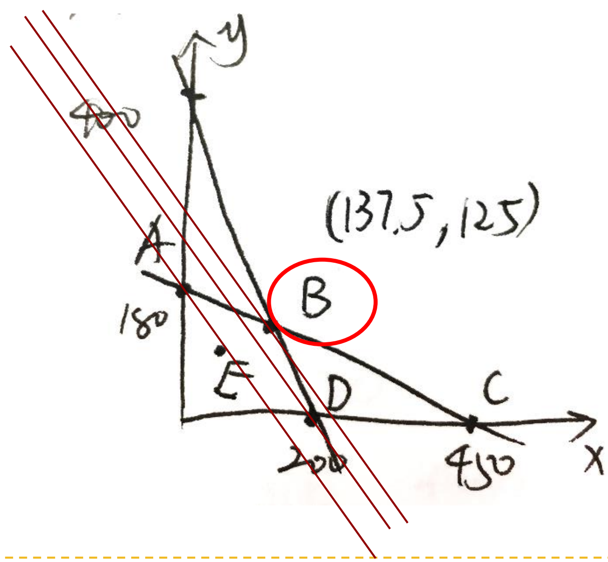
$$\max_{x,y} 30x + 30y$$

s.t.

$$0.2x + 0.5y \leq 90$$

$$4x + 2y \leq 800$$

$$x \geq 0, y \geq 0$$



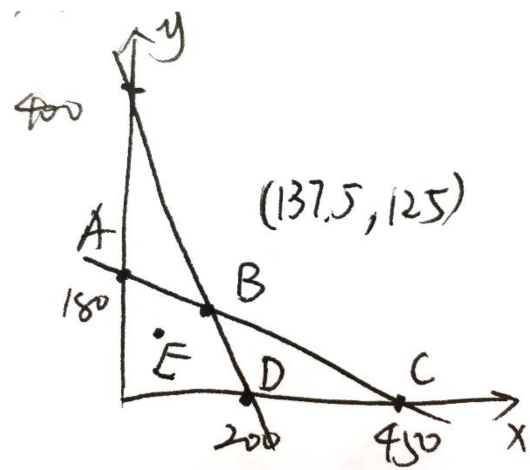
# Simplex Algorithm

- Start from one vertex, iteratively move to a neighboring vertex with better objective value
- Guaranteed to find a globally optimal solution
- May enumerate almost all the vertices in the worst case, but very efficient in most cases

$$\max_{x,y} 30x + 30y$$

s.t.

$$0.2x + 0.5y \leq 90$$
$$4x + 2y \leq 800$$
$$x \geq 0, y \geq 0$$



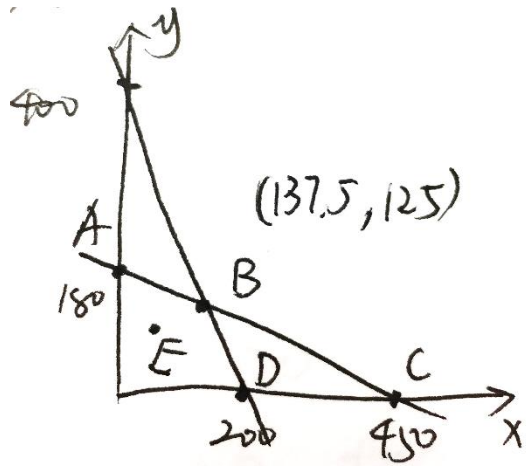
# Simplex Algorithm

- Q1: How to find a vertex?
- Q2: How to find a neighboring vertex?
- Q3: How to find a "better" neighboring vertex?

$$\max_{x,y} 30x + 30y$$

s.t.

$$0.2x + 0.5y \leq 90$$
$$4x + 2y \leq 800$$
$$x \geq 0, y \geq 0$$



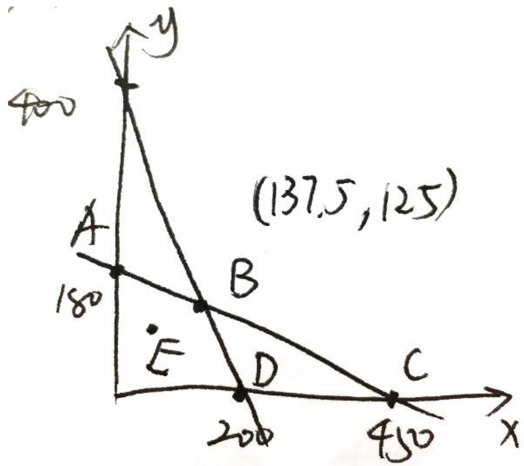
# Simplex Algorithm

- Q1: How to find a vertex?
  - If $x \in \mathbb{R}^n$, choose $n$ equalities, solve system, check feasibility
- Q2: How to find a neighboring vertex?
  - Remove one equality and add a new equality, solve system, check feasibility
- Q3: How to find a "better" neighboring vertex?
  - Check objective value

$$\max_{x,y} 30x + 30y$$

s.t.

$$0.2x + 0.5y \le 90$$
$$4x + 2y \le 800$$
$$x \ge 0, y \ge 0$$



# Linear Program: How to Solve

- Example solvers: recommend using Gurobi/Cplex, can also use linprog (MATLAB or Python), PuLP (Python)
- Some solvers require certain forms, e.g., linprog:

$$
\begin{aligned}
\min_x \ & c^T x \\
\text{such that } & A_{ub} x \leq b_{ub}, \\
& A_{eq} x = b_{eq}, \\
& l \leq x \leq u,
\end{aligned}
$$

$$\max_{x,y} 30x + 30y$$

s.t.

$$0.2x + 0.5y \leq 90$$
$$4x + 2y \leq 800$$
$$x \geq 0, y \geq 0$$

With Python 3.7 and SciPy 1.5

```
c = [-30, -30]
A = [[0.2, 0.5], [4, 2]]
b = [90, 800]
x_bounds = (0, None)
y_bounds = (0, None)
from scipy.optimize import linprog
res = linprog(c, A_ub=A, b_ub=b, bounds=[x_bounds, y_bounds])
print(res)
```



https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html

# Outline

- Linear Programming
- Integer Linear Programming
- Exercise: Planning in food rescue
- Discussion

# (Mixed) Integer Linear Program: Definition

- (Mixed) Integer Linear Program:
  - A special case of non-convex optimization problem
  - An optimization problem whose optimization objective is a linear function and feasible region is defined by a set of linear constraints + integer constraints

$$\max_x c^T x$$
$$\text{s.t. } Gx \leq h$$
$$x_i \in \mathbb{Z}, i \in J_z$$

Can be minimization
Can be $\geq h$ or $= h$

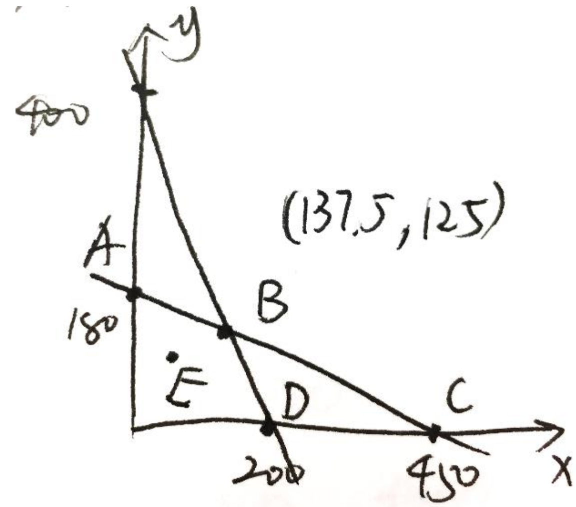
  - $c \in \mathbb{R}^n$
  - $G \in \mathbb{R}^{m \times n}, h \in \mathbb{R}^m$
  - $0 < |J_z| \leq n$
  - Integer Linear Program (ILP): $|J_z| = n$

# (Mixed) Integer Linear Program: Example

$$\max_{x,y} 30x + 30y$$

**s.t.**

$$0.2x + 0.5y \leq 90$$
$$4x + 2y \leq 800$$
$$x \geq 0, y \geq 0$$
$$x, y \in \mathbb{Z}$$




- Example: Maximize Profit in Manufacturing

| | | Price | Labor | Machine |
|---|---|---|---|---|
| | | Price | Labor | Machine |
| Earphone | Product 1 | \$30 | 0.2 hour | 4 hour |
| Charger | Product 2 | \$30 | 0.5 hour | 2 hour |
| | Total | | <=90 | <=800 |

# Binary Integer Program: Definition

- Binary Integer Program (BIP):
  - All variables are restricted to take value 0 or 1

$$\max_x c^T x$$
$$\text{s.t. } Gx \leq h$$
$$x_i \in \{0,1\}, \forall i$$

# Binary Integer Program: Example

- 0-1 Knapsack
  - Maximum weight $= 10$
  - How to select items to maximize total value?

| Items | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Weight | 5 | 4 | 2 | 6 | 7 |
| Value | 4 | 3 | 6 | 9 | 5 |

# Binary Integer Program: Example

- 0-1 Knapsack
  - Maximum weight $= 10$
  - How to select items to maximize total value?

| Items | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Weight | 5 | 4 | 2 | 6 | 7 |
| Value | 4 | 3 | 6 | 9 | 5 |

$$\max 4x_1 + 3x_2 + 6x_3 + 9x_4 + 5x_5$$
$$\text{s.t.} 5x_1 + 4x_2 + 2x_3 + 6x_4 + 7x_5 \leq 10$$
$$x_i \in \{0,1\}$$

# Binary Integer Program: Example

- 0-1 Knapsack
  - $n$ indivisible items. Item $i$ has weight $w_i$, value $v_i$.
  - Maximum weight is $W$ $(W \leq \sum_i w_i)$
  - How to pick the items to maximize total value?

# Binary Integer Program: Example

- 0-1 Knapsack
  - $n$ indivisible items. Item $i$ has weight $w_i$, value $v_i$.
  - Maximum weight is $W$ $(W \leq \sum_i w_i)$
  - How to pick the items to maximize total value?

$$\max \sum_i v_i x_i$$

$$\text{s.t. } \sum_i w_i x_i \leq W$$

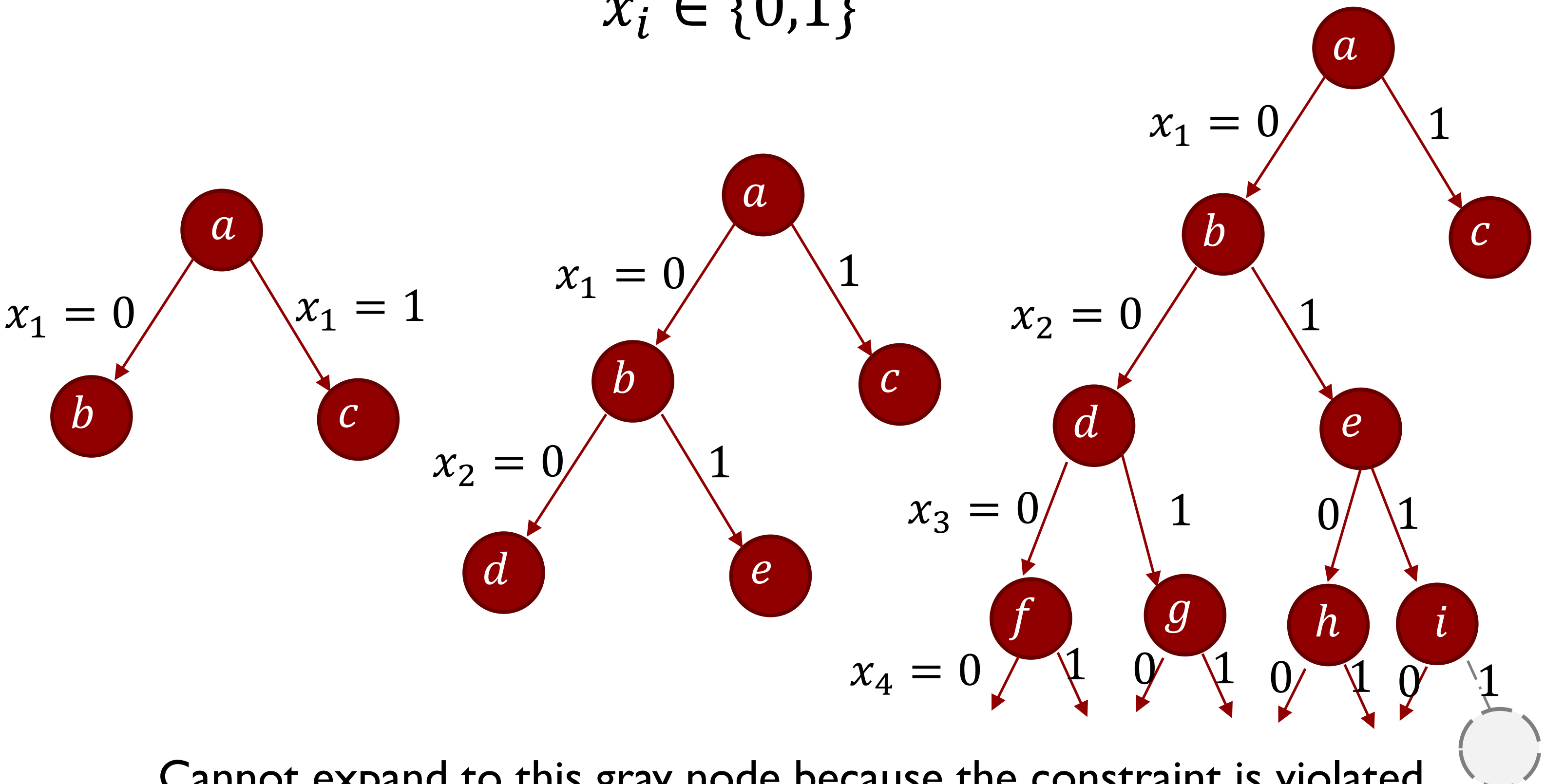
$$x_i \in \{0,1\}$$

# Depth-First Search for BIP

$$\max 4x_1 + 3x_2 + 6x_3 + 9x_4 + 5x_5$$
$$\text{s.t.}\, 5x_1 + 4x_2 + 2x_3 + 6x_4 + 7x_5 \leq 10$$
$$x_i \in \{0,1\}$$

# Depth-First Search for BIP

$$\max 4x_1 + 3x_2 + 6x_3 + 9x_4 + 5x_5$$
$$\text{s.t.} 5x_1 + 4x_2 + 2x_3 + 6x_4 + 7x_5 \le 10$$
$$x_i \in \{0,1\}$$




Cannot expand to this gray node because the constraint is violated

# (Mixed) Integer Linear Program: How to Solve

- General case: MILP is NP-Complete
  - Runtime is exponential
- Naïve approach
  - Search/enumerate values for integer variables, then solve LP

# LP Relaxation

- LP relaxation of an MILP or BIP is the LP with the same linear constraints

MILP

$$\max_x c^T x$$
$$\text{s.t. } Gx \le h$$
$$x_i \in \mathbb{Z}, i \in J_z$$

LP Relaxation

$$\max_x c^T x$$
$$\text{s.t. } Gx \le h$$

BIP

$$\max_x c^T x$$
$$\text{s.t. } Gx \le h$$
$$x_i \in \{0,1\}, \forall i$$

LP Relaxation

$$\max_x c^T x$$
$$\text{s.t. } Gx \le h$$
$$x_i \in [0,1]$$

# LP Relaxation

- What is the relaxed LP of the following BIP?

| Items | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Weight | 5 | 4 | 2 | 6 | 7 |
| Value | 4 | 3 | 6 | 9 | 5 |

$$\max 4x_1 + 3x_2 + 6x_3 + 9x_4 + 5x_5$$
$$\text{s.t.} 5x_1 + 4x_2 + 2x_3 + 6x_4 + 7x_5 \leq 10$$
$$x_i \in \{0,1\}$$

The optimal solution of the relaxed LP is $[0.4,0,1,1,0]$. Can you find a good but not necessarily optimal solution of the original BIP?

# LP Relaxation

- What is the relaxed LP of the following BIP?

| Items | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Weight | 5 | 4 | 2 | 6 | 7 |
| Value | 4 | 3 | 6 | 9 | 5 |

$$\max 4x_1 + 3x_2 + 6x_3 + 9x_4 + 5x_5$$
$$\text{s.t.} 5x_1 + 4x_2 + 2x_3 + 6x_4 + 7x_5 \leq 10$$
$$x_i \in \{0,1\} \rightarrow [0,1]$$

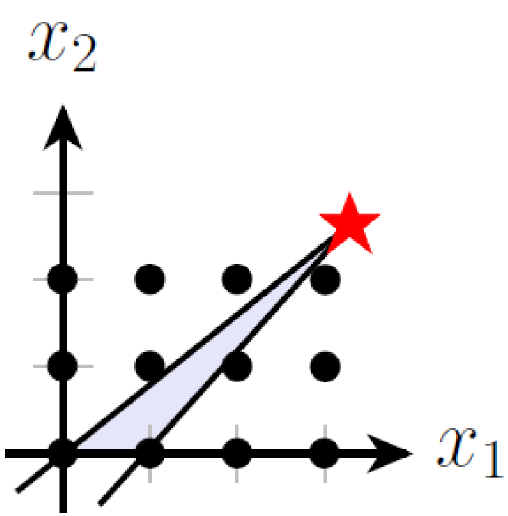
The optimal solution of the relaxed LP is $[0.4,0,1,1,0]$. Can you find a good but not necessarily optimal solution of the original BIP?

# Poll 2:

- True/False: For a ILP, it is sufficient to consider the integer points that are the closest to an optimal solution of the LP relaxation?

# Poll 2:

- True/False: For a ILP, it is sufficient to consider the integer points that are the closest to an optimal solution of the LP relaxation?




1. An LP can have infinite number of optimal solutions
2. So as IP
3. Integer points that are closest to an optimal solution of LP relaxation of IP may not be feasible
4. Integer points that are closest to an optimal solution of LP relaxation of IP may not be optimal (depends on the objective function)

# Poll 3

- Let $x^*, f^*$ be the optimal solution and the optimal value of a MILP. Let $\bar{x}^*, \bar{f}^*$ be the optimal solution and the optimal value of the LP relaxation. Which of the following statements are always true?
  - A: $x^* = \bar{x}^*$
  - B: $f^* \leq \bar{f}^*$ if it is a maximization problem
  - C: $f^* \leq \bar{f}^*$ if it is a minimization problem

MILP

$$\max_x c^T x$$
$$\text{s.t. } Gx \leq h$$
$$x_i \in \mathbb{Z}, i \in J_z$$

LP Relaxation

$$\max_x c^T x$$
$$\text{s.t. } Gx \leq h$$

When would $f^* = \bar{f}^*$?

# LP Relaxation

- If solution of relaxed LP happen to be integer solution, the solution is also optimal for the original ILP
- For some class of ILP problems, LP relaxation is guaranteed to get optimal solutions (e.g., problems satisfying total unimodularity)
- Can provide a heuristic solution through proper rounding
- Can provide a lower bound (for minimization problem) or upper bound (for maximization problem)

# (Mixed) Integer Linear Program: How to Solve

- Practically efficient solvers: Cplex, Gurobi, intlinprog (MATLAB), SCIP solver

0-1 Knapsack
Maximum weight = 10

| Items | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Weight | 5 | 4 | 2 | 6 | 7 |
| Value | 4 | 3 | 6 | 9 | 5 |

With Gurobi + free academic license >= 9.0.2
Cvxpy >= 1.0.25

```python
import cvxpy as cp

v = [4, 3, 6, 9, 5]
w = [5, 4, 2, 6, 7]
x = cp.Variable(5, boolean=True)
objective = cp.Maximize(v @ x)
constr = w @ x <= 10
milp = cp.Problem(objective, [constr])
milp.solve(solver=cp.GUROBI)
print('Solution is x = {}'.format(x.value))
```

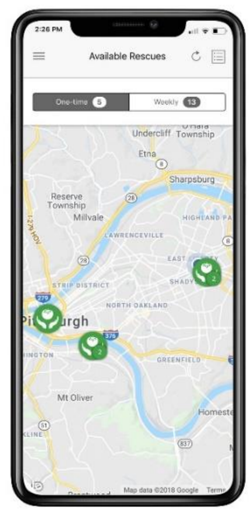
What if we assume divisible items? How should we change the code?

# Outline

- Linear Programming
- Integer Linear Programming
- Exercise: Planning in food rescue
- Discussion

# Motivation: Volunteer-Based Food Rescue Platform

- Food waste and food insecurity coexist
  - Waste up to 40% food globally
  - 1 in 8 people go hungry every day
- Rescue good food!





In collaboration with 412 Food Rescue (412FR)

# Exercise: Matching Problem in Food Rescue

- You are asked to help a food rescue organization to decide how to re-distribute the food in an efficient way.
- There are $M$ food donors (referred to as providers) and $N$ local communities in need of food (referred to as communities)
- There are $K$ type of food
- Provider $i$ has $A_{ik} \in \mathbb{N}$ unit of food of type $k \in [1..K]$
- Community $j$ needs $B_j \in \mathbb{N}$ unit of food in total, but it may have some special needs on the type
- $C_{jk} \in \mathbb{N}$ represents the minimum amount of food of type $k$ community $j$ needs
- The transportation cost per unit of food from provider $i$ to community $j$ is $T_{ij}$
- Find the optimal re-distribution plan that minimizes total transportation cost

# Exercise: Matching Problem in Food Rescue

Variables $x_{ijk}$: # of units of food of type $k$ re-distributed from provider $i$ to community $j$

What if the food sent to a community has to come from a single provider?

# Exercise: Matching Problem in Food Rescue

Variables $x_{ijk}$: # of units of food of type $k$ re-distributed from provider $i$ to community $j$

$$\max_x \sum_i \sum_j \sum_k T_{ij} x_{ijk}$$

s.t.

$$\sum_j x_{ijk} \leq A_{ik}$$

$$\sum_i \sum_k x_{ijk} \geq B_j$$

$$\sum_i x_{ijk} \geq C_{jk}$$

$$x_{ijk} \geq 0$$

What if the food sent to a community has to come from a single provider?

# Exercise: Sending Notifications in Food Rescue

- Now you are asked to help the food rescue platform decide which volunteers to send notifications to
- There are $M$ volunteers in total
- You know ahead of time that there are $N$ rescues for the upcoming day
- Assume you have access to $p_{ij} \in [0,1]$ indicating the prob. that volunteer $j$ will claim rescue $i$
- Each volunteer receives at most $L$ notifications per day
- For each rescue, you can choose $\leq K$ volunteers to notify
- Your goal is to maximize the total $p_{ij}$ of all the notified volunteer-rescue pairs

## Exercise: Sending Notifications in Food Rescue

- For each rescue, choose $\leq K$ volunteers to notify to maximize the total $p_{ij}$ of all the notified volunteer-rescue pairs

Input $p_{ij} \in [0,1]$: Prob. that volunteer $j$ will claim rescue $i$

Decision variable $x_{ij} \in \{0,1\}$: Whether to send notification of rescue $i$ to volunteer $j$

# Exercise: Sending Notifications in Food Rescue

- For each rescue, choose $\leq K$ volunteers to notify to maximize the total $p_{ij}$ of all the notified volunteer-rescue pairs

Input $p_{ij} \in [0,1]$: Prob. that volunteer $j$ will claim rescue $i$

Decision variable $x_{ij} \in \{0,1\}$: Whether to send notification of rescue $i$ to volunteer $j$

$$\max_{\mathrm{x}} \sum_i \sum_j p_{ij} x_{ij}$$

$$s.t. \sum_j x_{ij} \leq k, \forall i$$

$$\sum_i x_{ij} \leq L$$

$$x_{ij} \in \{0,1\}, \forall i, j$$

# Outline

- Linear Programming
- Integer Linear Programming
- Exercise: Planning in food rescue
- Discussion

# Discussion

- Think of another problem you face in real life for which you think LP/ILP/MILP can be used to solve

# References and Additional Resources

# Linear Program: Additional Resources

- Textbook
  - *Applied Mathematical Programming, Chapters 2-4*
  - By Bradley, Hax, and Magnanti (Addison-Wesley, 1977)
  - http://web.mit.edu/15.053/www/AMP.htm
- Online course
  - https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-251j-introduction-to-mathematical-programming-fall-2009/index.htm
- Survey of existing software: https://www.informs.org/ORMS-Today/Public-Articles/June-Volume-38-Number-3/Software-Survey-Linear-Programming
  -

# (Mixed) Integer Linear Program: Additional Resources

- A sufficient condition for ILP=LP: Total Unimodularity

- Textbook
  - *Applied Mathematical Programming, Chapter 9*
  - By Bradley, Hax, and Magnanti (Addison-Wesley, 1977)
  - http://web.mit.edu/15.053/www/AMP.htm
- Online course
  - https://ocw.mit.edu/courses/sloan-school-of-management/15-083j-integer-programming-and-combinatorial-optimization-fall-2009/index.htm

# Backup Slides

# Simplex Algorithm for LP in Standard Form

- LP in *standard form*

$$\min_x c^T x$$
$$\text{s.t. } Ax = b$$
$$x \geq 0$$

  - Any LP can be converted into standard form
    - If maximization, convert to minimization
    - For any constraint $g_i^T x \leq h_i$, add slack variables $s$ and get

$$g_i^T x + s_i = h_i, s_i \geq 0$$

$$\max_{x,y} 30x + 30y$$

s.t.

$$0.2x + 0.5y \leq 90$$
$$4x + 2y \leq 800$$
$$x \geq 0, y \geq 0$$

# Simplex Algorithm for LP in Standard Form

- LP in *standard form*

$$\min_x c^T x$$
$$\text{s.t. } Ax = b$$
$$x \geq 0$$

- Any LP can be converted into standard form
  - If maximization, convert to minimization
  - For any constraint $g_i^T x \leq h_i$, add slack variables $s$ and get

$$g_i^T x + s_i = h_i, s_i \geq 0$$

$$\max_{x,y} 30x + 30y$$

s.t.

$$0.2x + 0.5y \leq 90$$
$$4x + 2y \leq 800$$
$$x \geq 0, y \geq 0$$

$$\min_{x,y,s_1,s_2} -30x - 30y$$

s.t.

$$0.2x + 0.5y + s_1 = 90$$
$$4x + 2y + s_2 = 800$$
$$x \geq 0, y \geq 0, s_1 \geq 0, s_2 \geq 0$$

# Simplex Algorithm for LP in Standard Form

- How to find a vertex of the feasible region

$$Ax = b$$
$$x \geq 0$$

$x \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}$ (Assume $m < n, rank(A) = n$)

  - Choose a set of $m$ variables, denoted as $J$
  - Solve the following linear system, get a solution $x^*$

Only variables in $J$ can take non-zero values

$$Ax = b \qquad \qquad A_J x_J = b$$
$$x_j = 0, \forall j \notin J$$

$A_J$ denotes subselecting columns of $A$ based on $J$, $x_J$ denote subselecting element of $x$

  - $x^*$ is a vertex of the feasible region if $x^*$ satisfies remaining constraints $x \geq 0, x \in J$

# Simplex Algorithm for LP in Standard Form

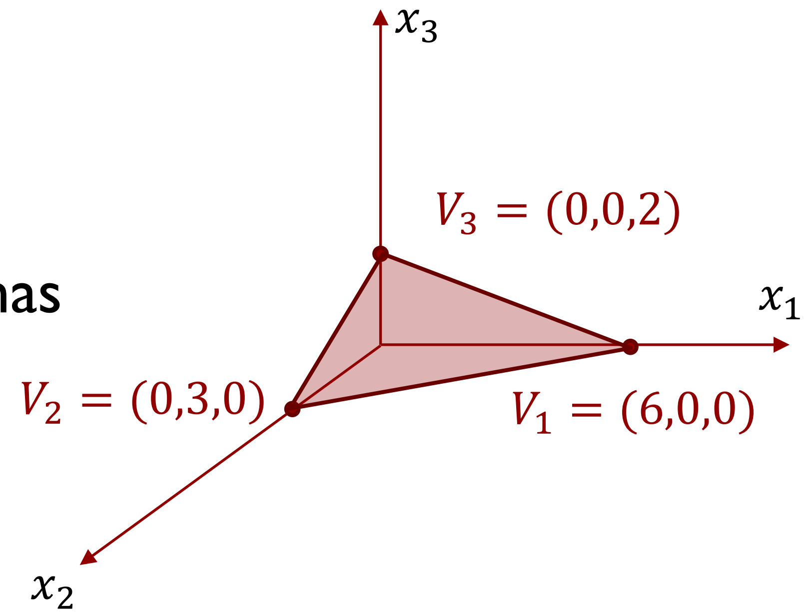
- A single step in simplex algorithm: Given a vertex $x$ and corresponding $J$, move to the neighboring vertex with a decrease in objective value

Current solution:

$$x_J = A_J^{-1} b$$

$$x_j = 0, \forall j \notin J$$

A neighboring vertex of $x$ only has one different element in $J$



## Simplex Algorithm for LP in Standard Form

- Consider adjusting $x$ to $x'$ by setting $x'_j = \alpha > 0$ for some $j \notin J$ while ensuring $x'_i = 0\ \forall i \notin J, i \neq j$ and $Ax' = b, x' \geq 0$
- All $x_i, i \in J$ has to change accordingly
- Denote $x'_J = x_J + \alpha d_J$, then (derivation omitted)
  - To ensure $Ax' = b$, we need to set $d_J = -A_J^{-1} A_j$
  - The new objective value is $f(x') = c^T x + \alpha \bar{c}_j$ where

    *reduced cost=* $\bar{c}_j = c_j - c_J^T A_J^{-1} A_j$
  - Let $i^* = \underset{\forall i: i \in J, d_i < 0}{\operatorname{argmin}} -\frac{x_i}{d_i}$. When $\alpha = -\frac{x_{i^*}}{d_{i^*}}$, we have $x'_{i^*} = 0$ and a neighboring vertex is reached

# Simplex Algorithm for LP in Standard Form

**Algorithm: Simplex Algorithm**

Input: Standard-form LP defined by $c, A, b$. Initial vertex point $x_0$ and corresponding $J_0$

Initialize $x \leftarrow x_0, J \leftarrow J_0$

Repeat

Find $j$ with $\bar{c}_j = c_j - c_J^T A_J^{-1} A_j < 0$, break if non exists

Compute $d_J \leftarrow -A_J^{-1} A_j$, $i^* \leftarrow \underset{\forall i: i \in J, d_i < 0}{\operatorname{argmin}} -\frac{x_i}{d_i}$, $\alpha^* \leftarrow -\frac{x_{i^*}}{d_{i^*}}$

Update $J \leftarrow J \backslash \{i^*\} \cup \{j\}$, $x_J \leftarrow x_J + \alpha^* d_J$, $x_j = \alpha^*$

Note: In practice, when checking if $\bar{c}_j < 0$ and $d_i < 0$ (and other tie breaking conditions), we can compare to $\pm 10^{-12}$ to avoid numerical issues

# Simplex Algorithm for LP in Standard Form

- Example

$$\min_{x,y,s_1,s_2} -30x - 30y$$

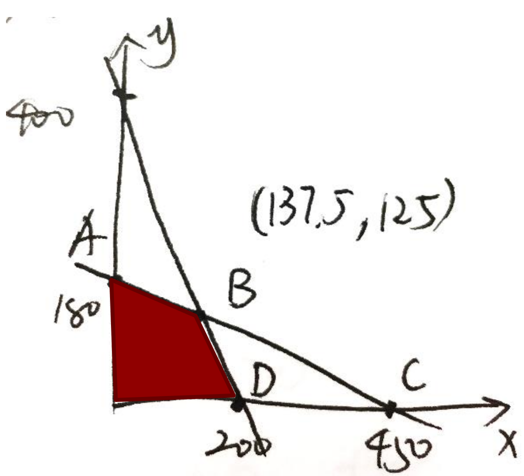
s.t.

$0.2x + 0.5y + s_1 = 90$
$4x + 2y + s_2 = 800$
$x \geq 0, y \geq 0, s_1 \geq 0, s_2 \geq 0$

$J = \{s_1, s_2\}$ $\boxed{\begin{array}{l} 0.2x + 0.5y + s_1 = 90 \\ 4x + 2y + s_2 = 800 \\ x = 0, y = 0 \end{array}}$ $\Longleftrightarrow$ $A_J = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$s_1 = 90, s_2 = 800$

- Introduce $j = \{x\}$ to $J$
- Reduced cost $\bar{c}_j = c_j - c_J^T A_J^{-1} A_j = -30 - 0 = -30.$ If $x$ increases by $\alpha$, obj value decreases by $30\alpha$
- $d_J = -A_J^{-1} A_j = \begin{bmatrix} -0.2 \\ -4 \end{bmatrix}$. So if $x$ increases by $\alpha$, $s_1$ has to decrease by $0.2\alpha$, $s_2$ has to decrease by $4\alpha$
- $x$ can increase by at most $\min\left\{\frac{90}{0.2}, \frac{800}{4}\right\} = 200$, at which point $s_2$ becomes 0
- So the new $J$ is $\{x, s_1\}$




$J = \{x, s_1\}$
$(x, y, s_1, s_2) = (200,0,50,0)$
Corresponds to vertex D in x-y space

# Simplex Algorithm for LP in Standard Form

- Example

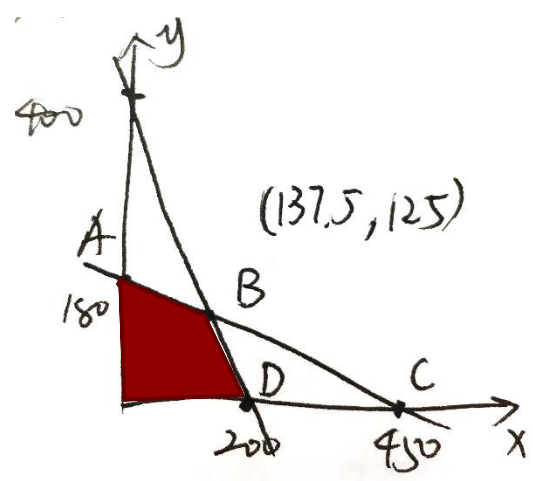
$$\min_{x,y,s_1,s_2} -30x - 30y$$

s.t.

$0.2x + 0.5y + s_1 = 90$

$4x + 2y + s_2 = 800$

$x \geq 0, y \geq 0, s_1 \geq 0, s_2 \geq 0$




$J = \{x, s_1\}$

$$x = 200, s_1 = 50$$

$$c_J = \begin{bmatrix} -30 \\ 0 \end{bmatrix} \quad A_J = \begin{bmatrix} 0.2 & 1 \\ 4 & 0 \end{bmatrix} \quad A_j = \begin{bmatrix} 0.5 \\ 2 \end{bmatrix} \quad c_j = -30$$

- Introduce $j = \{y\}$ to $J$
- Reduced cost $\bar{c}_j = c_j - c_J^T A_J^{-1} A_j = -15$
- $d_J = -A_J^{-1} A_j = \begin{bmatrix} -0.5 \\ -0.4 \end{bmatrix}$. So if $y$ increases by $\alpha$, $x$ has to decrease by $0.5\alpha$, $s_1$ has to decrease by $0.4\alpha$
- $y$ can increase by at most $\min\left\{\frac{200}{0.5}, \frac{50}{0.4}\right\} = 125$, at which point $s_1$ becomes $0$
- So the new $J$ is $\{x, y\}$

$$J = \{x, y\}$$

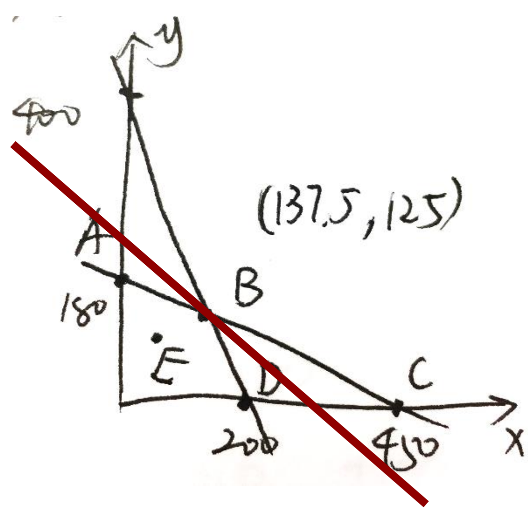
$$(x, y, s_1, s_2) = (137.5, 125, 0, 0)$$

Corresponds to vertex B in x-y space

- At this point, none of the variables $\notin J$ has a negative reduced cost $\bar{c}_j$, so optimal solution is found

# Simplex Algorithm for LP in Standard Form

- Handling degeneracy: avoid cycling over the indices
- *Bland's rule*
  - At each step, choose smallest $j$ such that $\bar{c}_j < 0$
  - For variables $x_i$ that could exit set $J$ choose the smallest $i^*$



# Two-Phase Simplex Algorithm

$$\min_{x} c^T x$$
$$\text{s.t. } Ax = b$$
$$x \geq 0$$

- Phase I (Find one vertex): Apply simplex alg to the following LP with initial vertex $(0, b)$, get $(x^*, z^*)$

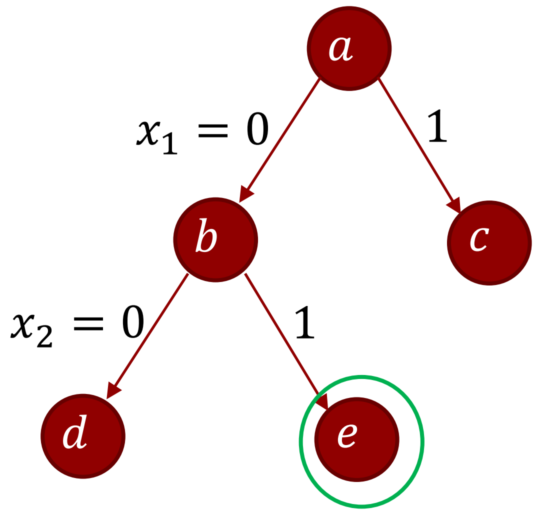
$$\min_{x,z} 1^T z$$
$$\text{s.t. } Ax + z = b$$
$$x, z \geq 0$$

For $b_i < 0$, flip the sign of the constraints
$x = 0, z = b$ provides a vertex to begin with

- If $z^* > 0$, the original LP is infeasible
- Phase II (Find best vertex): Apply simplex alg to the original LP with initial vertex $x^*$

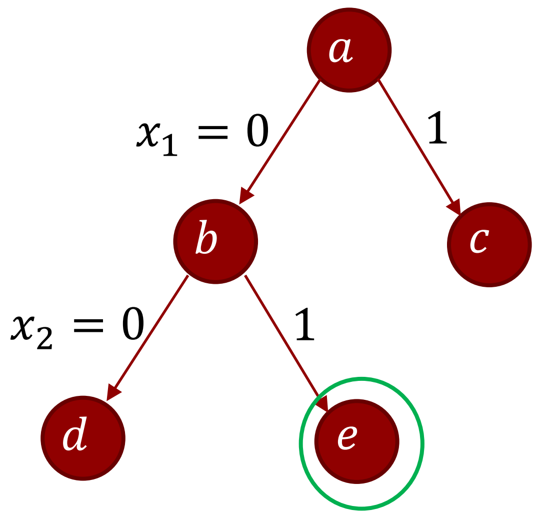
# Upper Bound during Tree Search: LP Relaxation

$$\max 4x_1 + 3x_2 + 6x_3 + 9x_4 + 5x_5$$
$$\text{s.t.}\, 5x_1 + 4x_2 + 2x_3 + 6x_4 + 7x_5 \le 10$$
$$x_i \in \{0,1\}$$



# Upper Bound during Tree Search: LP Relaxation

$$\max 4x_1 + 3x_2 + 6x_3 + 9x_4 + 5x_5$$
$$\text{s.t.} 5x_1 + 4x_2 + 2x_3 + 6x_4 + 7x_5 \leq 10$$
$$x_i \in [0,1], i = 1..5$$
$$x_1 = 0$$
$$x_2 = 1$$



# Branch and Bound for BIP

- Branch and Bound overview (assuming maximization)
  - Heuristic search
  - Use optimal objective value of LP relaxation (upper bound) as the heuristic function
  - Always expand the node with the best upper bound first (poly-time computable)
  - Terminate early when best upper bound of remaining nodes is worse than the current best solution

# Branch and Bound for BIP: Example

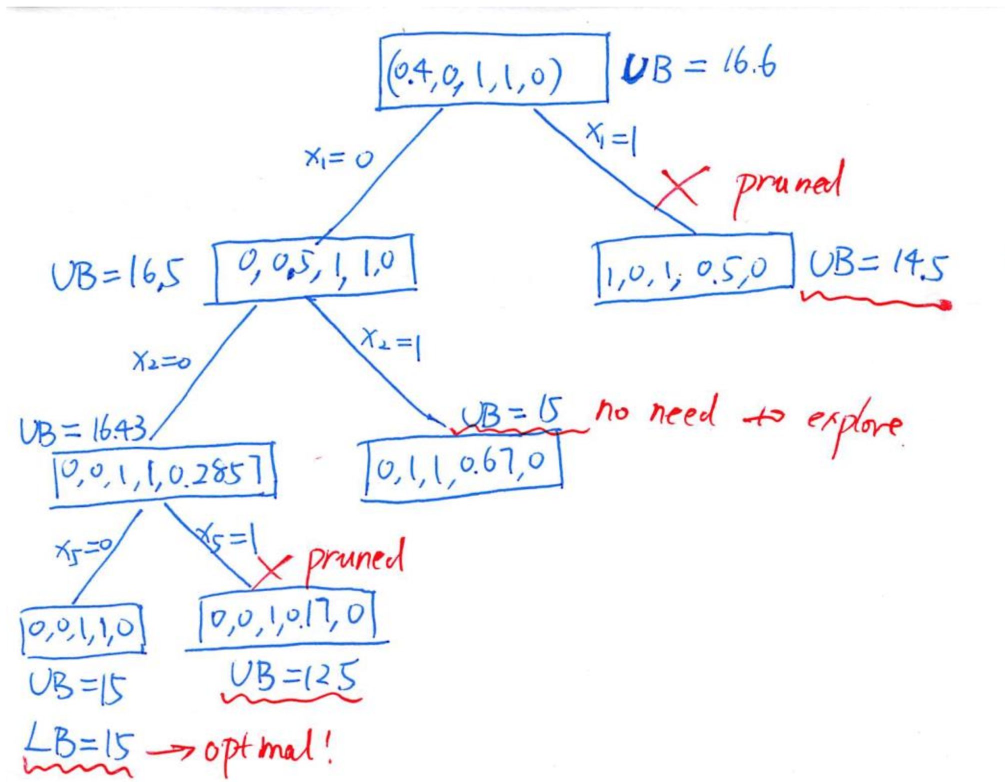
| Items | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Weight | 5 | 4 | 2 | 6 | 7 |
| Value | 4 | 3 | 6 | 9 | 5 |

$$\max 4x_1 + 3x_2 + 6x_3 + 9x_4 + 5x_5$$
$$\text{s.t.} 5x_1 + 4x_2 + 2x_3 + 6x_4 + 7x_5 \leq 10$$
$$x_i \in \{0,1\}$$

# Branch and Bound for BIP: Example

| Items | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Weight | 5 | 4 | 2 | 6 | 7 |
| Value | 4 | 3 | 6 | 9 | 5 |

$$\max 4x_1 + 3x_2 + 6x_3 + 9x_4 + 5x_5$$
$$\text{s.t.} 5x_1 + 4x_2 + 2x_3 + 6x_4 + 7x_5 \leq 10$$
$$x_i \in \{0,1\}$$



# Branch and Bound for BIP

- Solve-LP($\mathcal{C}$) returns $(f, x)$, the optimal objective value and the optimal solution for the LP relaxation of the original problem with additional constraints $\mathcal{C}$

**Algorithm: Branch and Bound for BIP**

Input: A BIP with $x_i, i = 1..n$ as variables

Initialize $nodelist$ with *Solve-LP*($\{\}$)
Repeat
    Remove a node with best $f$ from $nodelist$: $(f, x, \mathcal{C})$
    If $x$ are all integer valued, return $(f, x)$
    Get a feasible integer solution $\hat{x}$ based on $x$, update current best $(\bar{f}, \bar{x})$
    If $\bar{f} \leq f + \epsilon$, return $(\bar{f}, \bar{x})$
    Choose a variable $x_i$ that is not integer valued and add two nodes *Solve-LP*($\mathcal{C} \cup \{x_i = 0\}$) and *Solve-LP*($\mathcal{C} \cup \{x_i = 1\}$) to $nodelist$
Until $nodelist$ is empty

# Branch and Bound for MILP

- For MILP
  - BnB: For each integer variable, branching a node by considering $x_i \leq \lfloor \widetilde{x_i} \rfloor$ and $x_i \geq \lceil \widetilde{x_i} \rceil$ where $\widetilde{x_i}$ is a non-integer value

- Standard BnB has already been integrated into existing (M)ILP solvers in Cplex and Gurobi

- Extension: Branch and Cut
  - On top of branch and bound, use cutting planes (which are essentially linear constraints) to separate current non-integer solution and integer solutions