BDC5101

Deterministic Operations Research Models

Zhenyu Hu

Semester II, 2018/2019

National University of Singapore

Integer Programming

- **Integer Program Modeling**
- **Further Applications**

- **Linear Programming Relaxation**
- **Branch and Bound Method**

Integer Program Modeling

Mix IP

$$
\begin{aligned}
\max \quad & c'x + h'y \\
s.t. \quad & Ax + By \le b \\
& x \ge 0, \text{integer} \\
& y \ge 0.
\end{aligned}
$$

Pure IP

IP

$$\boxed{\begin{aligned} \max \quad & c'x \\ s.t. \quad & Ax \le b \\ & x \ge 0, \text{integer} \end{aligned}}$$

BIP

$$\boxed{\begin{aligned} \max \quad & c'x \\ s.t. \quad & Ax \le b \\ & x \in \{0,1\}^n \end{aligned}}$$

Modeling with Binary Variables

$$x = \begin{cases} 1 & \text{if event occurs} \\ 0 & \text{otherwise} \end{cases}$$

Choice Among Several Possibilities

- **Due to resource constraint, at most one of project 1,2, and 3 can be implemented. How to model this constraint?**

- **If exactly two out of the five projects 2,3,4,5, and 6 must be implemented. How to model this constraint?**

Simple Implications

- If project 1 is chosen, then project 4 is also chosen.
- If project 2 is chosen, then we cannot choose project 5.
- If we don’t choose project 1, then project 6 must be chosen.
- Project 4 and 5 belong to the same company. Either both of them are chosen or neither of them is chosen.

Implication with three variables

- **If we do project 2, then we must do project 3 and 4.**
- **If we do project 3, then we must do project 1 or 5.**
- **If we do both 3 and 4, then we must do 6.**

Product of Binary Variables

- **How to convert the following constraint into linear constraints?**

$$x_1 = x_2 \times x_3, \;\; x_1, x_2, x_3 \text{ binary.}$$

  - **If project 2 is not chosen, then project 1 can not be chosen: $x_1 \le x_2$**
  - **If project 3 is not chosen, then project 1 can not be chosen: $x_1 \le x_3$**
  - **If we do 2 and 3, then we must do 1:**

$$x_1 \ge x_2 + x_3 - 1$$

Product of Binary and Nonnegative Variables

- **How to convert the following constraint into linear constraints?**

$$\boldsymbol{x_1 = x_2 \times x_3,\ x_2} \text{ binary, } \boldsymbol{x_3 \ge 0} \text{ .}$$

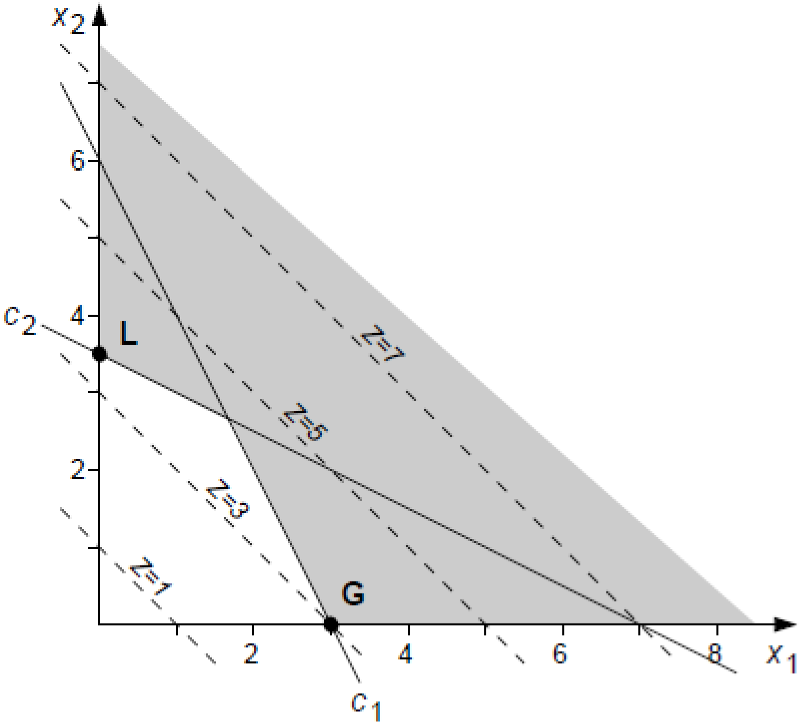
- **If $x_2 = 1$, then $x_1 = x_3$: $x_1 \le x_3$, $x_1 \ge x_3 - (1 - x_2)$**
- **If $x_2 = 0$, then $x_1 = 0$: $0 \le x_1 \le x_2$.**
- **Big M:**

$$x_1 \le x_3,\ 0 \le x_1 \le M x_2\ x_1 \ge x_3 - M(1 - x_2)$$

Either / Or Constraints

$$\begin{aligned} &\text{Minimize} && z = x_1 + x_2 \\ &s.t. && \text{Either } 2x_1 + x_2 \geq 6 \\ & && \text{Or} \quad x_1 + 2x_2 \geq 7 \\ & && x_1, x_2 \geq 0 \end{aligned}$$

Graphical View



With the help of binary variable

$$
\begin{aligned}
\text{Minimize} \quad & z = x_1 + x_2 \\
s.t. \quad & 2x_1 + x_2 \geq 6b \\
& x_1 + 2x_2 \geq 7(1-b) \\
& x_1, x_2 \geq 0, \quad b \text{ binary}
\end{aligned}
$$

Production Problem with Fixed Cost

- **Let *K* be the cost of setting up a piece of machinery.**
- **Let *x* be the production level and *c* be the variable cost of the production.**
- **Let *T* be the production capacity.**
- **How to model the total cost?**

$$\text{Cost} = Kb + cx$$

$$x \leq bT$$

$$x \geq 0 \text{ and } b \text{ binary.}$$

Production Problem with Economy of Scale

- **Let *c* be the original variable cost of the production.**
- **If the production quantity is above 100, there is a 10% discount to the total production cost.**
- **How to model the total cost?**

$$\text{Cost} = cx_1 + 0.9cx_2$$
$$0 \leq x_1 \leq 100b_1$$
$$100b_2 \leq x_2 \leq Mb_2$$
$$b_1 + b_2 = 1, b_1, b_2 \text{ binary, M is a large number}$$

Problem Involving Counting

- **Suppose you have a sum of money $*M* to invest in a basket of 20 investment products.**
- **But you want to limit the final number of investments that you choose to be no more than 5.**
- **How to model this constraint?**

x_i : amount of money invested in product i.

$$\sum_{i=1}^{20} x_i \leq M$$

$$x_i \leq M b_i$$

$$\sum_{i=1}^{20} b_i \leq 5$$

Assignment Problem

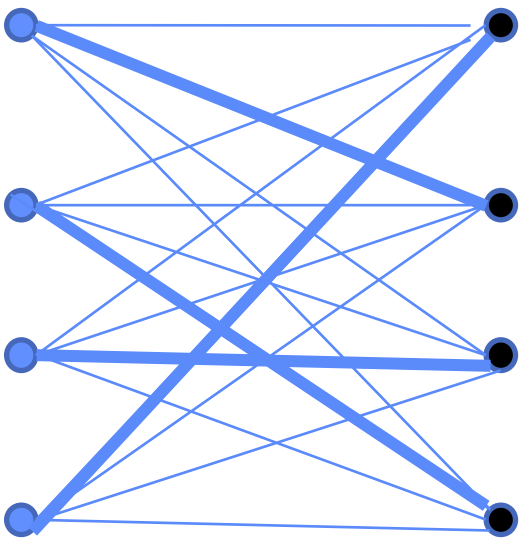
$$\text{Maximize } z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij},$$

subject to:

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad (i = 1, 2, \ldots, n),$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (j = 1, 2, \ldots, n),$$

$$x_{ij} = 0 \quad \text{or} \quad 1 \qquad (i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, n).$$



Shortest Path Problem

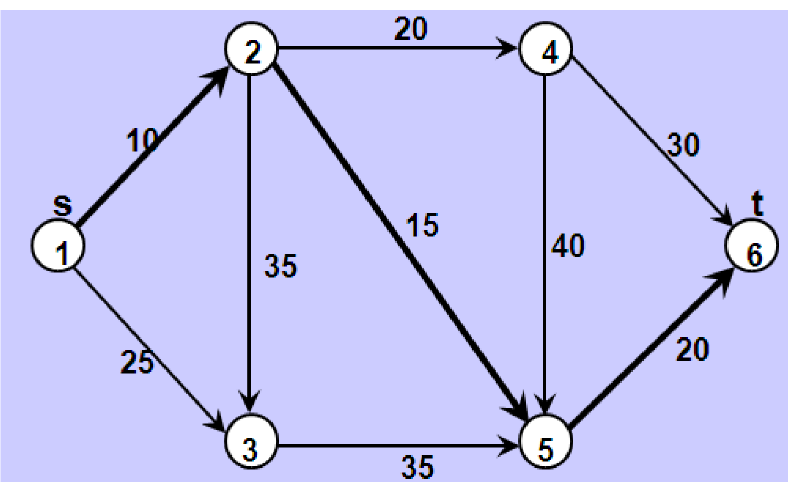
$$\text{Minimize } z = \sum_i \sum_j c_{ij} x_{ij},$$

subject to:

$$\sum_j x_{ij} - \sum_k x_{ki} = \begin{cases} 1 & \text{if } i = s \text{ (source)}, \\ 0 & \text{otherwise}, \\ -1 & \text{if } i = t \text{ (sink)} \end{cases}$$

$$x_{ij} \geq 0 \quad \text{for all arcs } i\text{–}j \text{ in the network.}$$

$$x_{ij} = 0 \quad \text{or} \quad 1$$



Knapsack Problem

- **There are n projects.**
- **Project i costs c_i to implement.**
- **Project i returns p_i.**
- **Total budget is b.**

Which projects to invest to maximize return?

Knapsack Problem

- $x_i = 1$ **if project** i **is invested,** $x_i = 0$ **oterwise.**

$$\text{Maximize: } \sum_{i=1}^{n} p_i x_i$$
$$\text{s.t. } \sum_{i=1}^{n} c_i x_i \leq b$$
$$x_i \text{ binary}$$

Facility Location Problem

- $\{1,2,\ldots,n\}$ be potential locations to place a facility, e.g., a plant. Once plant is built, there is no capacity in its production quantities.
- $\{1,2,\ldots,m\}$ be the locations of the demands, e.g., distribution centers. Location j requires d_j units of product.
- c_{ij} is the cost (e.g., transportation distance) of satisfying one unit demand at location j using supply at location i.
- K_i is the cost of building a plant at location i.

Which locations to open plants such that the total cost of building plants and transportation cost from satisfying demands is minimized?

Facility Location Problem

- $x_i = 1$ **if a plant is built at location** i**,** $x_i = 0$ **otherwise.**
- y_{ij}**: the amount of products shipped from plant** i **to demand** j**.**

$$\text{Minimize:} \quad \sum_{i=1}^{n} K_i x_i + \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} y_{ij}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} y_{ij} = d_j, \; j = 1, \dots, m$$

$$\sum_{j=1}^{m} y_{ij} \leq M x_i, \; i = 1, \dots, n$$

$$y_{ij} \geq 0, x_i \text{ binary}$$

- M **is a large number (one can take** $M = \sum_{j=1}^{m} d_j$**)**

Further Applications

Travelling Salesman Problem

- **Given a list of cities and their pairwise distances, the task is to find a shortest possible tour that visits each city exactly once.**

SCIENCE

Newsweek—Bensi

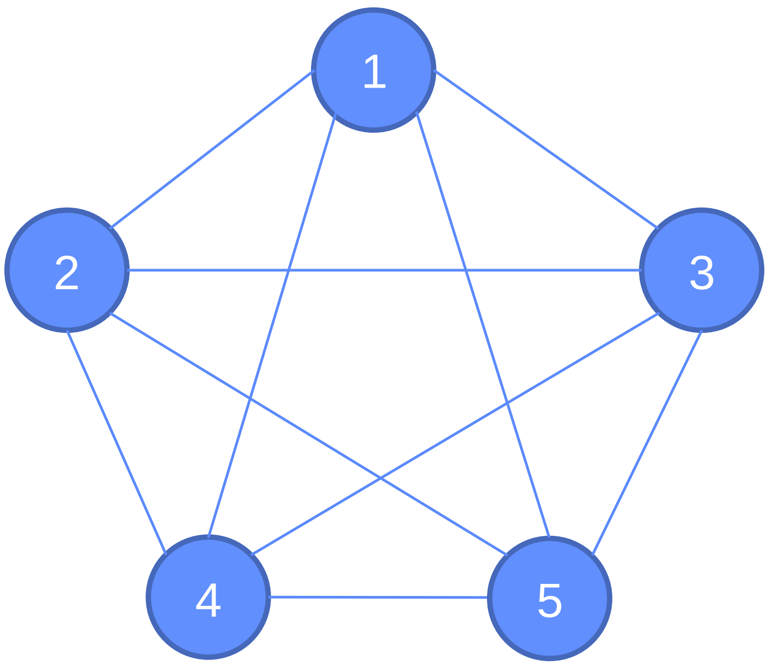
FINDING the shortest route for a traveling salesman—starting from a given city, visiting each of a series of other cities, and then returning to his original point of departure—is more than an after-dinner teaser. For years it has baffled not only goods- and salesmen-routing businessmen but mathematicians as well. If a drummer visits 50 cities, for example, he has 10^{62} (62 zeros) possible itineraries. No electronic computer in existence could sort out such a large number of routes and find the shortest.

Three Rand Corp. mathematicians, using Rand McNally road-map distances between the District of Columbia and major cities in each of the 48 states, have finally produced a solution (see above). By an ingenious application of linear programming—a mathematical tool recently used to solve production-scheduling problems—it took only a few weeks for the California experts to calculate "by hand" the shortest route to cover the 49 cities: 12,345 miles.

http://www.math.uwaterloo.ca/tsp/index.html

Complete Graph

- **Every pair of node is connected by an arc.**



Mathematical Definition (Symmetric TSP)

- **Given a complete undirected graph and a cost on each arc, find a path starting and finishing at a specified node after having visited each other node exactly once (called *Hamiltonian cycle*) that minimizes the total cost.**

- **If in application, no arc exists between two nodes, we can assign an arbitrarily large cost on the arc connecting these two nodes.**

Naïve Method

- **Brute force enumeration.**

How many feasible solutions are there?

How does $n!$ compare with 2^n?

Model

- **Suppose a graph with n nodes. The cost from node i to j is c_{ij}. For convenience, let $c_{ii} = \infty$.**
  - $x_{i,j} = 1$ **if we travel from i to j**
  - $= 0$ **otherwise.**

Model

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$s.t. \quad \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, 2, \ldots, n$$

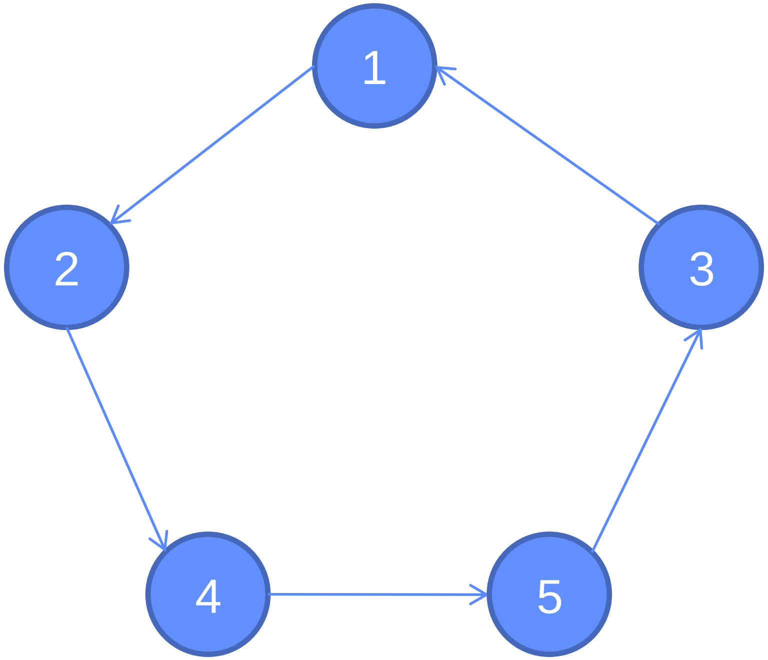
$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, 2, \ldots, n$$

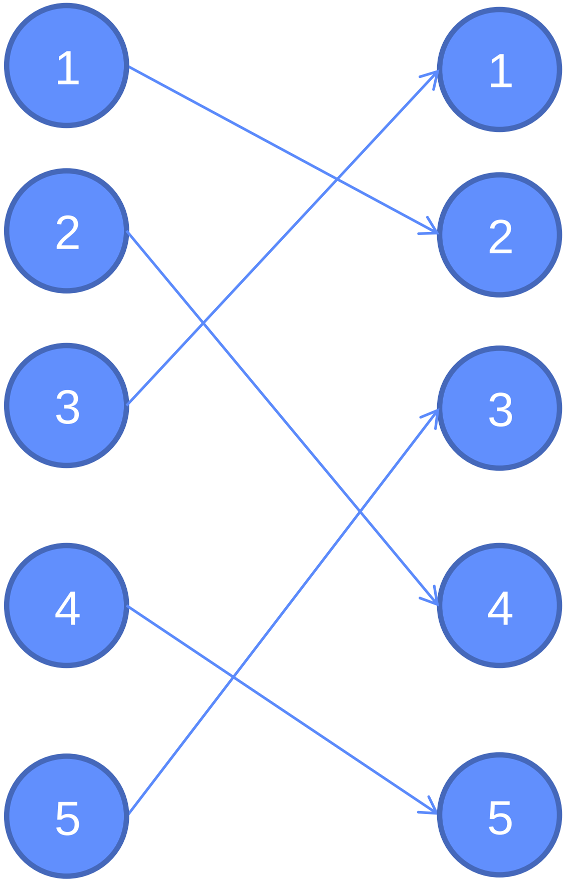
$$x_{ij} \in \{0,1\}$$

Assignment problem gives a lower bound to the TSP.

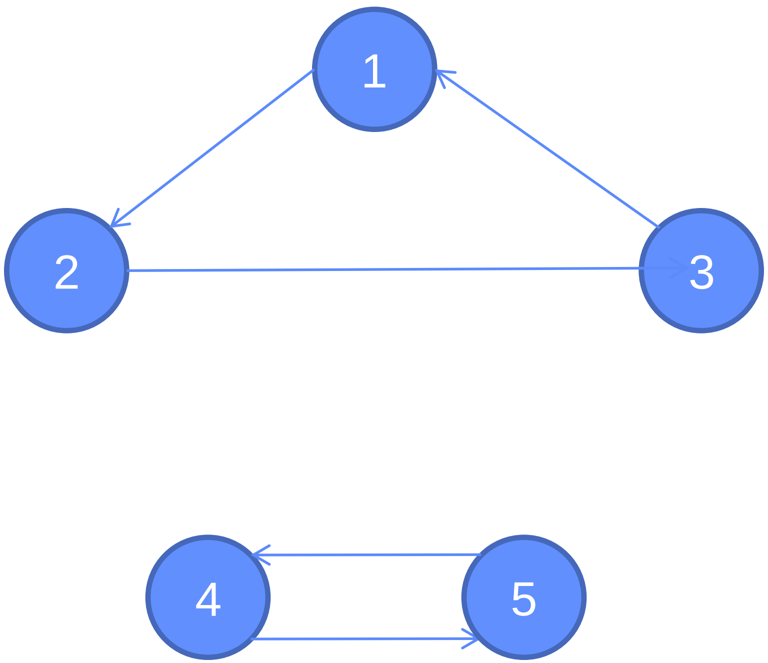
(why?)

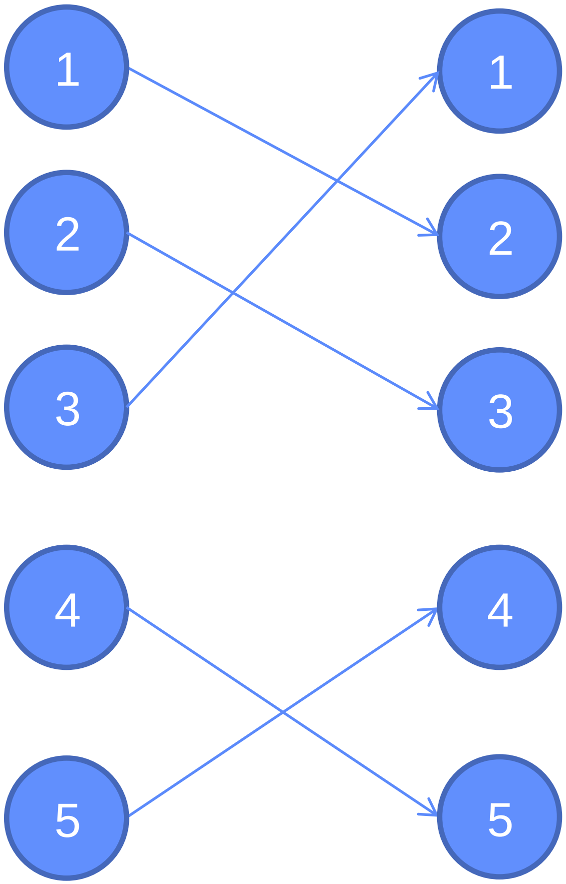
Assignment Problem





Assignment Problem





Model (Dantzig, Fulkerson & Johnson)

$$\min \quad \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$s.t. \quad \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, 2, \ldots, n$$

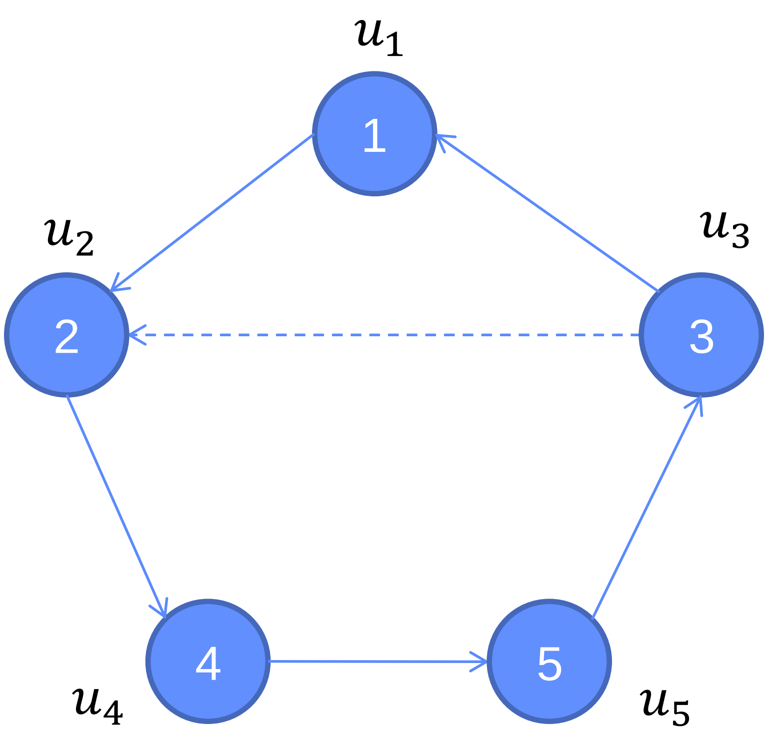
$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, 2, \ldots, n$$

$$x_{ij} \in \{0, 1\}$$

Sub tour breaking constraints:

$\sum_{i,j \in S} x_{ij} \leq |S| - 1$, for every subset S

Counting Cities Traveled



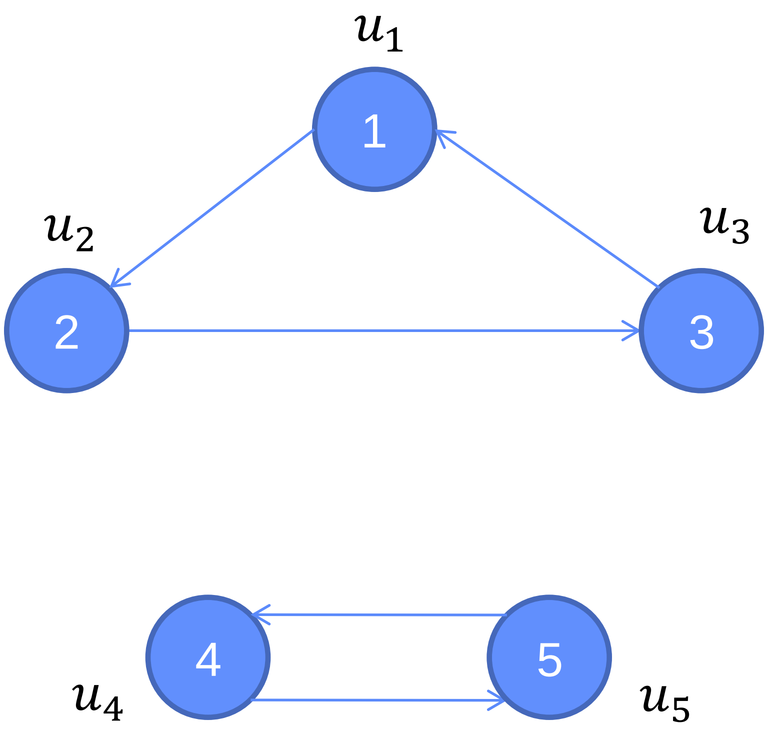
$$u_2 \geq (u_1+1)x_{12} \quad u_2 \geq (u_3+1)x_{32}$$

$$u_4 \geq (u_2+1)x_{24}$$

$$u_5 \geq (u_4+1)x_{45}$$

$$u_3 \geq (u_5+1)x_{53}$$

Counting Cities Traveled



$$u_4 \geq (u_5+1)x_{54}$$

$$u_5 \geq (u_4+1)x_{45}$$

Model

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$s.t. \quad \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1,2,\dots,n$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1,2,\dots,n$$

$$x_{ij} \in \{0,1\}$$

Introduce additional variables:

$$u_i \geq 0, i = 1,2,\dots,n$$

and counting constraint:

$$u_j \geq (u_i+1)x_{ij}, i \neq j, 2 \leq j \leq n, 1 \leq i \leq n$$

or the equivalent counting constraint:

$$(u_i+1-u_j)x_{ij} \leq 0, i \neq j, 2 \leq j \leq n, 1 \leq i \leq n$$

Transformation

- **The constraint below is not linear**

$$(u_i+1-u_j)x_{ij} \le 0, i \neq j, 2 \le j \le n, 1 \le i \le n$$

- **For a very large constant** M ($M = n - 1$ is sufficient)

$$u_i + 1 - u_j \le M(1 - x_{ij}), i \neq j, 2 \le j \le n, 1 \le i \le n$$

Model (Miller, Tucker and Zemlin)

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$s.t. \quad \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, 2, \ldots, n$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, 2, \ldots, n$$

$$x_{ij} \in \{0,1\}$$

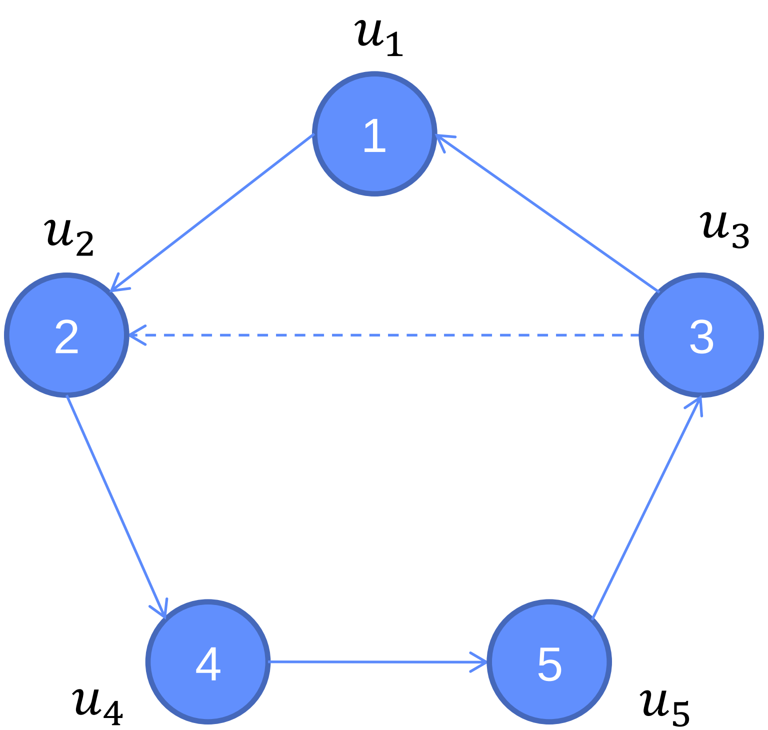
Introduce additional variables:

$$u_i \geq 0, i = 1, 2, \ldots, n$$

and constraints:

$$u_i + 1 - u_j \leq M(1 - x_{ij}), i \neq j, 2 \leq j \leq n, 1 \leq i \leq n$$

Counting Accumulated Cost/Distance/Time



$$u_2 \geq (u_1 + c_{12})x_{12} \quad u_2 \geq (u_3 + c_{32})x_{32}$$

$$u_4 \geq (u_2 + c_{24})x_{24}$$

$$u_5 \geq (u_4 + c_{45})x_{45}$$

$$u_3 \geq (u_5 + c_{53})x_{53}$$

Model

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$s.t. \quad \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, 2, \ldots, n$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, 2, \ldots, n$$

$$x_{ij} \in \{0,1\}$$

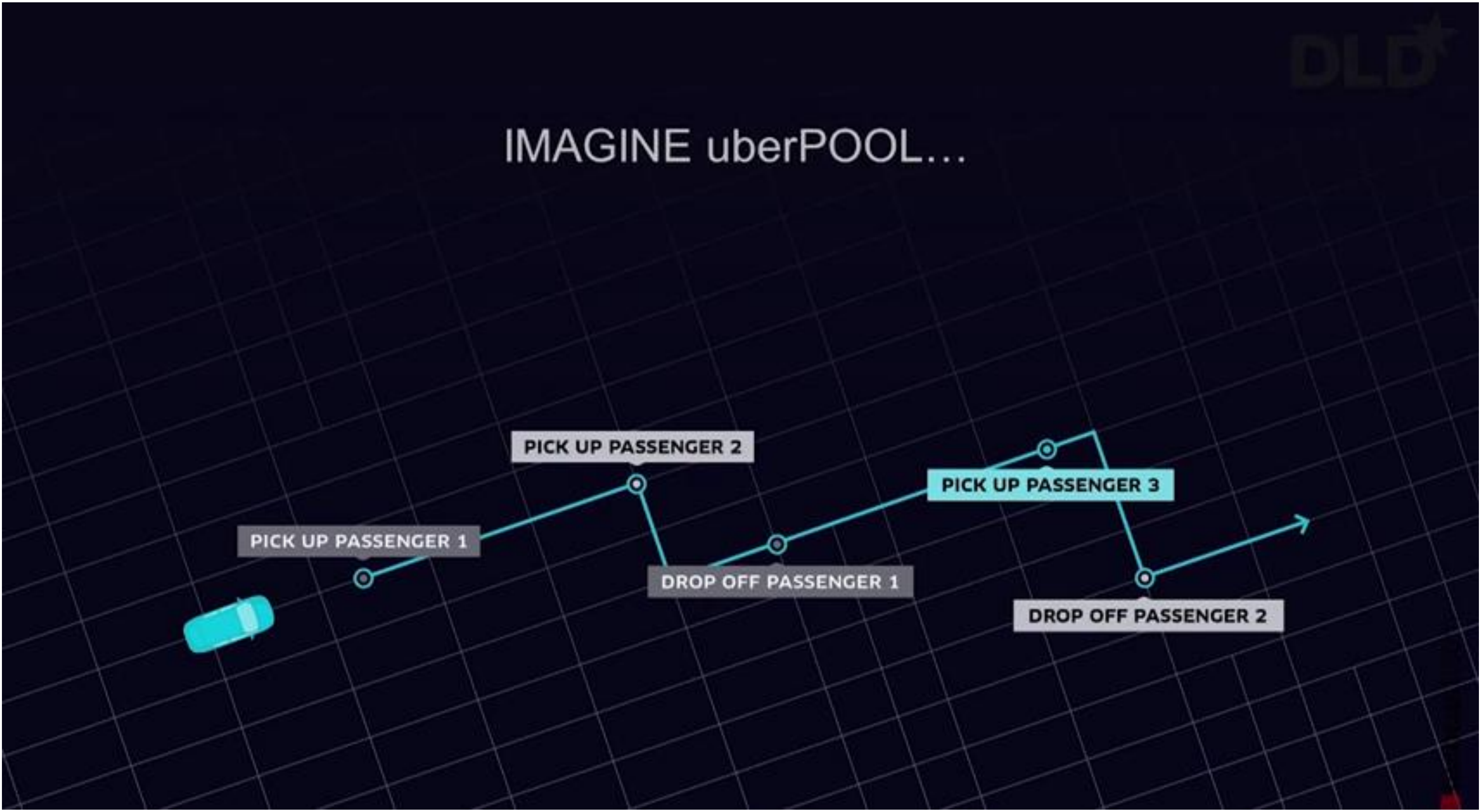
Introduce additional variables:

$$u_i \geq 0, i = 1, 2, \ldots, n$$

and constraints:

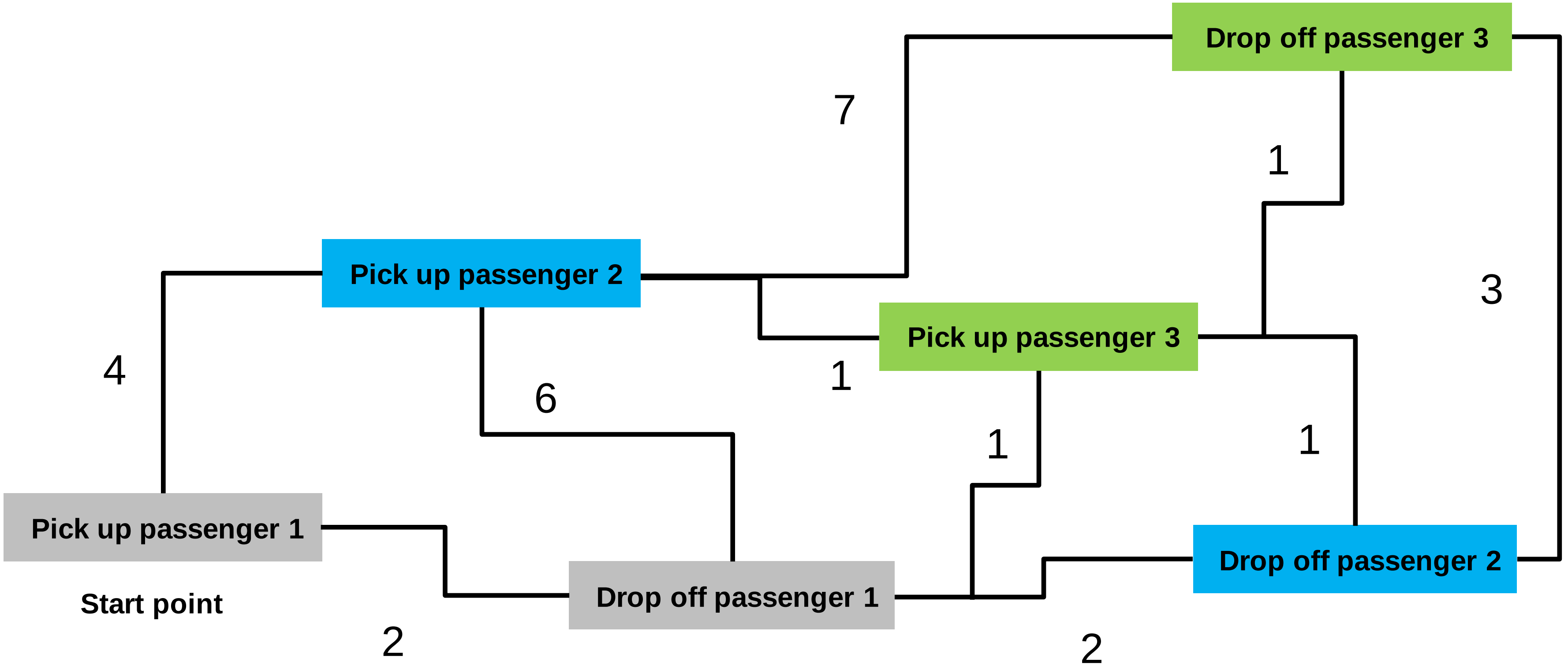
$$u_i + c_{ij} - u_j \leq M(1 - x_{ij}), i \neq j, 2 \leq j \leq n, 1 \leq i \leq n$$

UberPOOL



Source: http://blog.under.com/uberPOOL-2015

An Example



Drop off passenger 3
7
1
Pick up passenger 2
3
Pick up passenger 3
4
6
1
1
1
Pick up passenger 1
Drop off passenger 2
Drop off passenger 1
Start point
2
2

The Problem

- The objective is to find a route that achieves:
  - Send every customer to his/her destination
  - Minimizes the total time/distance traveled

- Assumptions:
  - We know every positions in advance
  - The vehicle has unlimited capacity

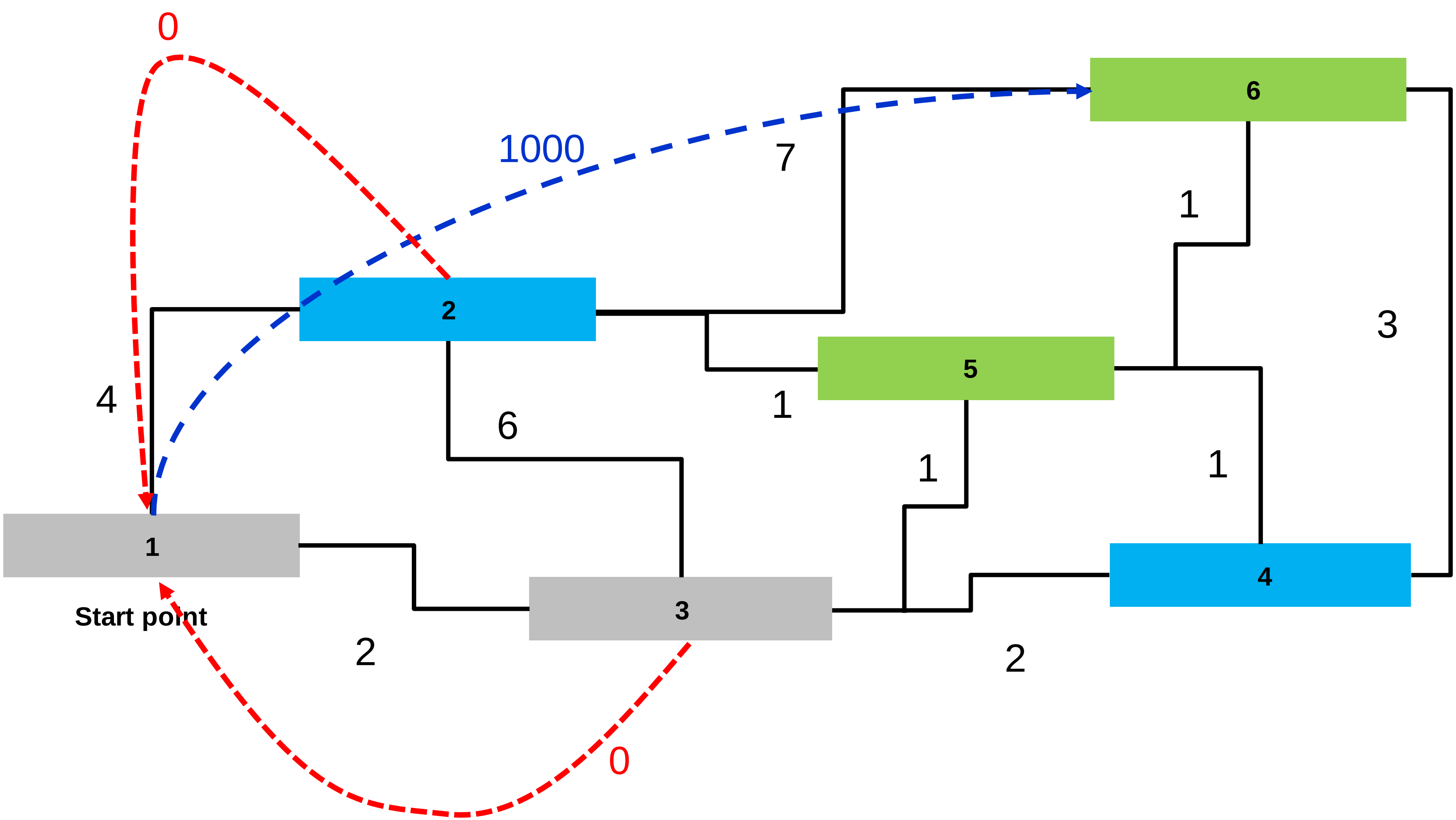
- How should we model the problem?

Model

- **Graph is not complete.**
  - **Set** $c_{ij} = \infty$ **if** (i, j) **is not in the graph.**

- **It is not required to return to the starting node.**
  - **We can assign 0 cost to all arcs flowing into the starting node.**

Modification



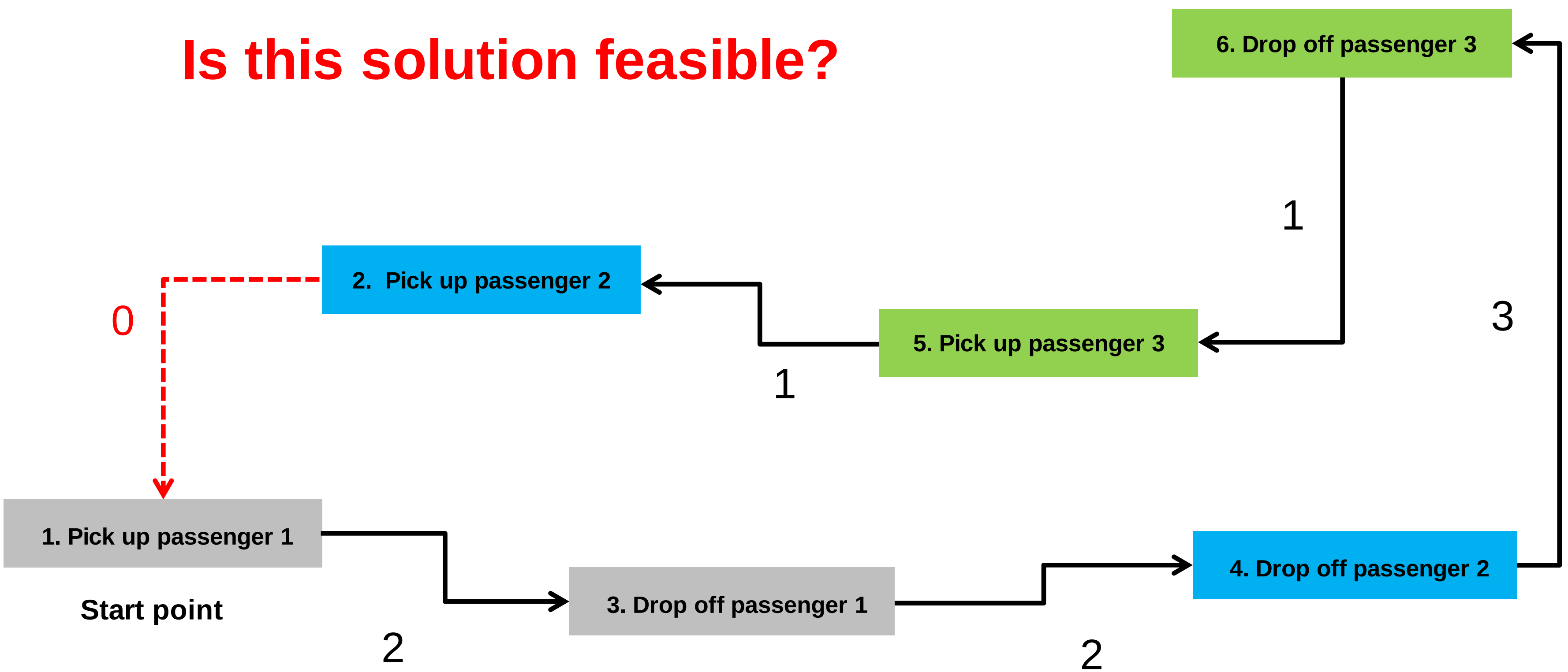
TSP: Data

```
from gurobipy import *
import numpy as np

#########Parameters Set-up############

#traveling cost from node i to node j
cost = np.array([[1000, 4, 2, 1000, 1000, 1000],
                 [0, 1000, 6, 1000, 1, 7],
                 [0, 6, 1000, 2, 1, 1000],
                 [0, 1000, 2, 1000, 1, 3],
                 [0, 1, 1, 1, 1000, 1],
                 [0, 7, 1000, 3, 1, 1000]])

N = cost.shape[0]

#the big M
M = 10000
```

TSP: Model

```python
########Model Set-up###########



tsp = Model("traveling_salesman")

# Creat variables
x = tsp.addVars(N, N, vtype=GRB.BINARY, name = "x")

u = tsp.addVars(N, name = "u")

# Set objective
tsp.setObjective( quicksum(cost[i,j]*x[i,j] for i in range(N) for j in range(N)), GRB.MINIMIZE)

# Assignment constraints:
tsp.addConstrs(( quicksum(x[i,j] for j in range(N)) == 1 for i in range(N) ))

tsp.addConstrs(( quicksum(x[i,j] for i in range(N)) == 1 for j in range(N) ))

# Subtour-breaking constraints:
tsp.addConstrs(( u[i] + 1 - u[j] <= M*(1 - x[i,j])  for i in range(N) for j in range(1,N) ))


# Solving the model
tsp.optimize()
```

Solution: Optimal Value 9



Precedence Constraints

- In order to get a feasible solution, we need to impose precedence constraints:
  - Node 1 (pick customer 1) need to be traveled before node 3 (drop customer 1) .
  - Node 2 (pick customer 2) need to be traveled before node 4 (drop customer 2) .
  - Node 5 (pick customer 3) need to be traveled before node 6 (drop customer 3) .

Precedence Constraints

- Recall the meaning of u_i at each node i: u_i is the number of nodes visited so far.

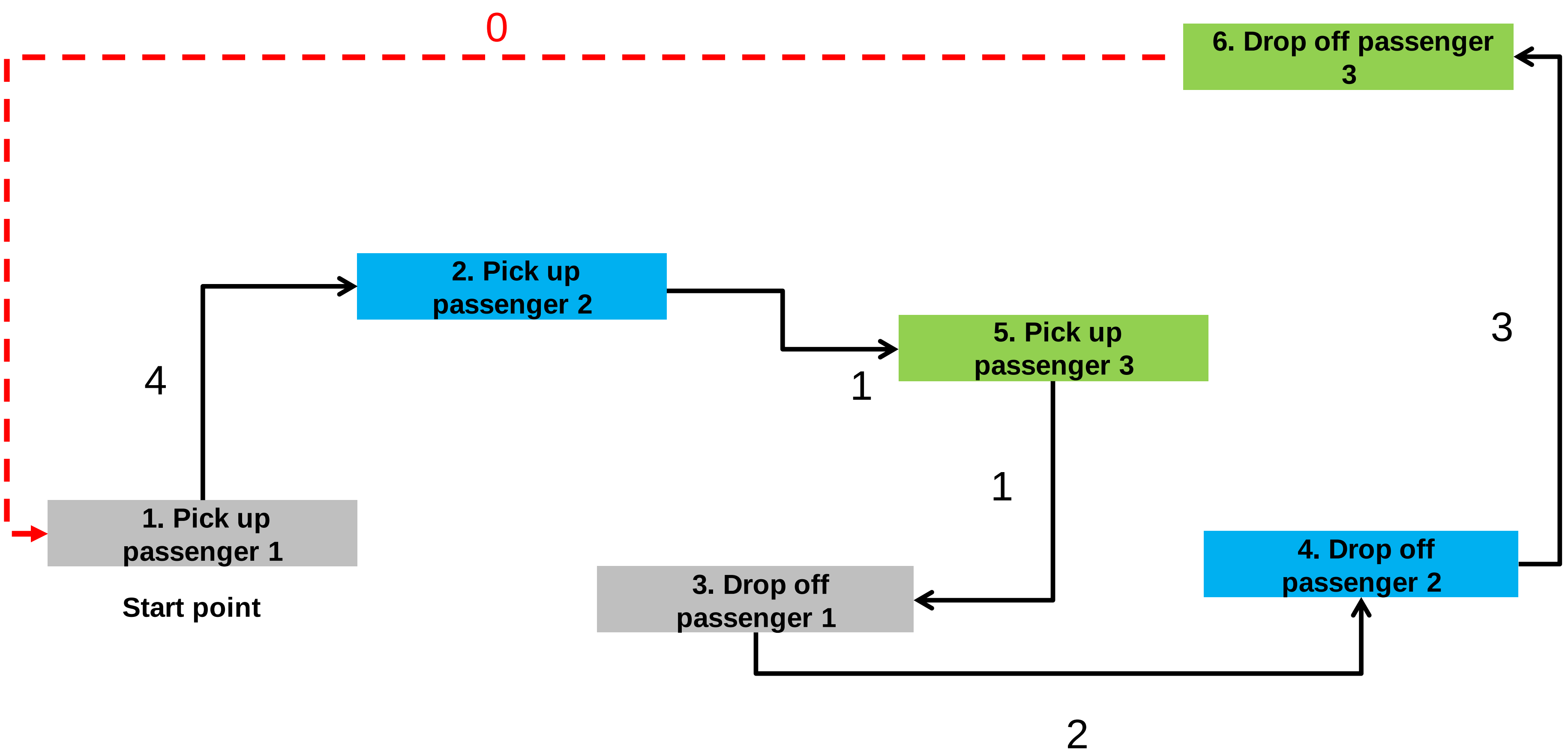
- If we define a precedent pairs P containing (1, 3), (2, 4) and (5, 6), then the precedence constraints can be expressed as

$$u_i \leq u_j,\ for\ (i, j) \in P$$

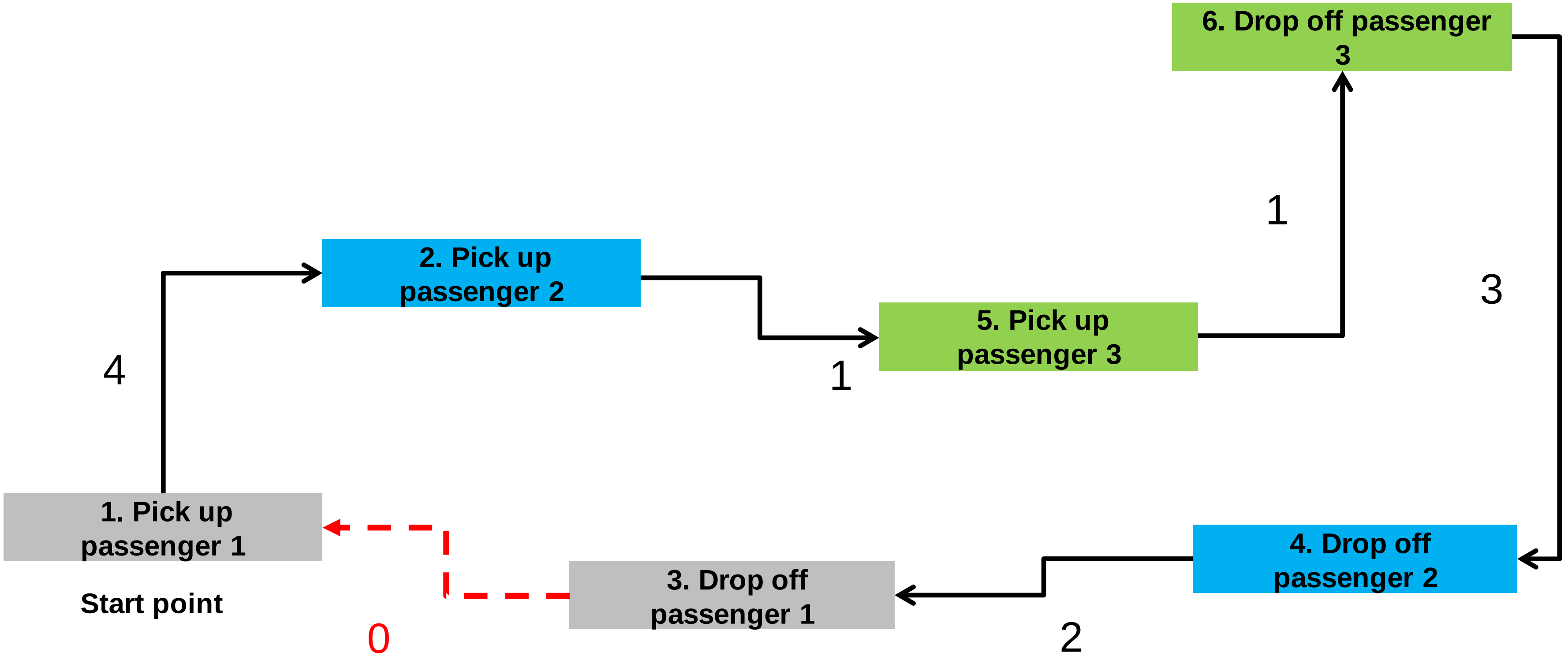
Precedence Constraints

```
# data for the precedent pair

Precedent_Pair = tuplelist([(0,2), (1,3), (4,5)])


tsp.addConstrs( (u[i] <= u[j] for (i,j) in Precedent_Pair)  )


# Solving the new model
tsp.optimize()
```

Solution: Optimal Value 11



OR Solution: Optimal Value 11



Sudoku

8			6			9		5
				2		3	1	
		7	3	1	8		6	
2	4						7	3
		2	7	9		1		
5				8			3	6
		3						

Sudoku

Lee Hsien Loong

Public Figure · 869,723 Likes · May 4 · Edited ·

Liked

I told the Founders Forum two weeks ago that the last computer program I wrote was a Sudoku solver, written in C++ several years ago (http://bit.ly/1DMK5Zk). Someone asked me for it. Here is the source code, the exe file, and a sample printout - http://bit.ly/1zAXbua

The program is pretty basic: it runs at the command prompt, in a DOS window. Type in the data line by line (e.g. 1-3-8---6), then the solver will print out the solution (or all the solutions if there are several), the number of steps the program took searching for the solution, plus some search statistics.

For techies: the program does a backtrack search, choosing the next cell to guess which minimises the fanout.

Here's a question for those reading the source code: if x is an (binary) integer, what does (x & -x) compute?

Hope you have fun playing with this. Please tell me if you find any bugs! – LHL

#SmartNation

===========

Answer: As several of you noted, (x & –x) returns the least significant '1' bit of x, i.e. the highest power of two that divides x. This assumes two's complement notation for negative numbers, as some of you also pointed out. e.g. if x=12 (binary 1100), then (x & -x) = 4 (binary 100). I didn't invent this; it is an old programming trick. 🙂

===========

Update: A few people suggested that I add a licence to the code. Have added it in the Google Drive folder.

Sudoku2.cpp

```cpp

void Place(int S)
{
    LevelCount[S]++;
    Count++;

    if (S >= 81) {
        Succeed();
        return;
    }

    int S2 = NextSeq(S);
    SwapSeqEntries(S, S2);

    int Square = Sequence[S];

    int     BlockIndex = InBlock[Square],
            RowIndex = InRow[Square],
            ColIndex = InCol[Square];

    int     Possibles = Block[BlockIndex] & Row[RowIndex] & Col[ColIndex];
    while (Possibles) {
        int valbit = Possibles & (-Possibles); // Lowest 1 bit in Possibles
        Possibles &= ~valbit;
        Entry[Square] = valbit;
        Block[BlockIndex] &= ~valbit;
        Row[RowIndex] &= ~valbit;
        Col[ColIndex] &= ~valbit;

        Place(S + 1);

        Entry[Square] = BLANK; // Could be moved out of the Loop
        Block[BlockIndex] |= valbit;
        Row[RowIndex] |= valbit;
        Col[ColIndex] |= valbit;
    }

    SwapSeqEntries(S, S2);
}


int main(int argc, char* argv[])
{
    int i, j, Square;

    for (i = 0; i < 9; i++)
        for (j = 0; j < 9; j++) {
            Square = 9 * i + j;
            InRow[Square] = i;
            InCol[Square] = j;
            InBlock[Square] = (i / 3) * 3 + ( j / 3);
        }


    for (Square = 0; Square < 81; Square++) {
        Sequence[Square] = Square;
        Entry[Square] = BLANK;
        LevelCount[Square] = 0;
    }

    for (i = 0; i < 9; i++)
        Block[i] = Row[i] = Col[i] = ONES;

    ConsoleInput();
    Place(SeqPtr);
    printf("\n\nTotal Count = %d\n", Count);

    return 0;
}
```

Sudoku: IP Formulation

- **Decision variable is $x_{ijk} \in \{0, 1\}, k = 1, \dots, 9$**
- **Input data: $y_{ij} = k, k = 1, \dots, 9$**
  - $y_{ij} = 0$, **if it is not filled.**

Sudoku: IP Formulation

$$\max 0$$

$$x_{i,j,y_{i,j}} = 1,\ for\ y_{i,j} \neq 0$$

$$\sum_{i=1}^{9} x_{i,j,k} = 1,\ \forall j,k$$

$$\sum_{j=1}^{9} x_{i,j,k} = 1,\ \forall i,k$$

$$\sum_{k=1}^{9} x_{i,j,k} = 1,\ \forall i,j$$

$$\sum_{i=3p-2}^{3p} \sum_{j=3q-2}^{3q} x_{i,j,k} = 1,\ \forall k; p,q \in \{1,2,3\}$$

$$x_{i,j,k} \in \{0,1\}$$

Rue La La

Source: https://www.ruelala.com/boutique/

The Kooples T-Shirt
$49.99 ~~$95.00~~
3 LEFT

The Kooples T-Shirt
$49.99 ~~$95.00~~
2 LEFT

The Kooples Polo Shirt
$59.99 ~~$115.00~~
1 LEFT

Price Grids

$39.99	$39.99	$39.99
$44.99	$44.99	$44.99
$49.99	$49.99	$49.99
$54.99	$54.99	$54.99
$59.99	$59.99	$59.99

What do we need?

The Kooples T-Shirt
$49.99 ~~$95.00~~
3 LEFT

The Kooples T-Shirt
$39.99 ~~$95.00~~
2 LEFT

The Kooples Polo Shirt
$59.99 ~~$115.00~~
1 LEFT

What are the demands for products 1, 2 and 3 under prices ($49.99, $49.99, $59.99)?

Will demand for product 1, 2 and 3 change under prices ($49.99, $39.99, $59.99)?

How many possible price points are there? 5^3 (M^N)

Simplification Assumption

- **Let p_m denote the m-th price point, e.g. $p_1 = 39.99$**
- **Demand for a product n depends on its own price p_m and the sum of prices (or average price) of all products k.**
  - **Demand for product 1 is the same under**
    - ***($49.99, $39.99, $59.99) or ($49.99, $49.99, $49.99)***
  - **$D(n, m, k)$ is the demand for product n at price point m and when total price is k.**
  - **The possible values for k are 3x39.99, 3x39.99 + 5, 3x39.99 + 10, …, 3x59.99 in total 3x4 + 1 points.**
    - *Need to estimate demands at $N \times M \times [N \times (M-1) + 1]$ points instead of M^N points (consider M=2, N=50).*

Optimization Model

- **Here, $D(n,m,k)$ is not a parameter, since k depends on the prices of each of the product.**
- **If k is fixed, say, $k = 3 \times 49.99$, can we model the problem?**
  - $x_{n,m} = 1$ **if product n is priced at price point p_m, $x_{n,m} = 0$ otherwise.**
  - **Objective:** $z_k = \max \sum_{n=1}^{N} \sum_{m=1}^{M} p_m D(n,m,k)\, x_{n,m}$
  - **Choose only one price:** $\sum_{m=1}^{M} x_{n,m} = 1$
  - **Total price is k:** $\sum_{n=1}^{N} \sum_{m=1}^{M} p_m x_{n,m} = k$

Optimization Model

- **But k is not fixed…**
  - **Try every possible k !**
  - **For k = 3x39.99, 3x39.99 + 5, …, 3x59.99**
  - **Solve the IP to obtain z_k**
  - **Find the maximum z_k.**

Linear Programming Relaxation

Method of LP Relaxation

Minimize: $\boldsymbol{c}'\boldsymbol{x}$
s.t. $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$
$\boldsymbol{x} \geq \boldsymbol{0}$, integer

Minimize: $\boldsymbol{c}'\boldsymbol{x}$
s.t. $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$
$\boldsymbol{x} \geq \boldsymbol{0}$, binary

LP Relaxation

Minimize: $\boldsymbol{c}'\boldsymbol{x}$
s.t. $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$
$\boldsymbol{x} \geq \boldsymbol{0}$

Minimize: $\boldsymbol{c}'\boldsymbol{x}$
s.t. $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$
$\boldsymbol{0} \leq \boldsymbol{x} \leq \boldsymbol{1}$

Round the solution to the nearest integers

Investment Example

- **Consider two investments opportunities:**
  - **Invest $100K, return 1 million 40 years later.**
    - *Annual rate of return: 5.9%*
  - **Invest $80K, return 0.95 million 40 years later.**
    - *Annual rate of return: 6.4%*
- **Budget constraint: $100K.**
- **Objective: Maximize the return after 40 years.**

Model

- **IP formulation:**

$$\begin{aligned} &\text{Maximize: } 1000x_1 + 950x_2 \\ &\text{s.t.} \quad 100x_1 + 80x_2 \le 100 \\ &x_1, x_2 \text{ binary} \end{aligned}$$

- **LP relaxation**

$$\begin{aligned} &\text{Maximize:} 1000x_1 + 950x_2 \\ &\text{s.t.} \quad 100x_1 + 80x_2 \le 100 \\ &0 \le x_1, x_2 \le 1 \end{aligned}$$

$\Rightarrow$ $\boxed{(x_1, x_2) = (0.2, 1)}$ $\Rightarrow$ $\boxed{(x_1, x_2) = (0, 1)}$

Method of LP Relaxation

$$\begin{aligned} \text{Minimize:} \quad & \boldsymbol{c}'\boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0}, \text{ integer} \end{aligned}$$

$$\begin{aligned} \text{Minimize:} \quad & \boldsymbol{c}'\boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0}, \text{ binary} \end{aligned}$$

LP Relaxation $\Downarrow$

$$\begin{aligned} \text{Minimize:} \quad & \boldsymbol{c}'\boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

$$\begin{aligned} \text{Minimize:} \quad & \boldsymbol{c}'\boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{0} \leq \boldsymbol{x} \leq \boldsymbol{1} \end{aligned}$$

$\Downarrow$

When are they the same?

Cramer's Rule

The solution for LP: $\boldsymbol{B x_B = b}$

Cramer's Rule: consider a system of n linear equations and n unknowns

$$\boldsymbol{Bx = b}$$

$$\boldsymbol{x_i = \frac{det(M_i)}{det(B)}}$$

$$\boldsymbol{M_i = [B_1, \dots, B_{i-1}, b, B_{i+1}, \dots, B_n]}$$

Totally Unimodular Matrix

Unimodular Matrix: A square integer matrix having determinant +1 or -1.

If the optimal basis is unimodular, then we are done.

Totally Unimodular Matrix: A matrix for which every square non-singular submatrix is unimodular.

If A is totally unimodular, then we are done.

Necessary Condition and Example

Every entry of ***A*** is 0, +1, or -1

	x_{12}	x_{13}	x_{23}	x_{24}	x_{25}	x_{34}	x_{35}	x_{45}	x_{53}	*Righthand side*
Node 1	1	1								20
Node 2	-1		1	1	1					0
Node 3		-1	-1			1	1		-1	0
Node 4				-1		-1		1		-5
Node 5					-1		-1	-1	1	-15
Capacities	15	8	∞	4	10	15	5	∞	4	
Objective function	4	4	2	2	6	1	3	2	1	(Min)

Branch and Bound Method

Knapsack Problem

$$\text{Maximize: } \sum_{i=1}^{n} p_i x_i$$
$$\text{s.t. } \sum_{i=1}^{n} c_i x_i \leq b$$
$$x_i \text{ binary}$$

- **How many feasible solutions do we have?**
  - **Consider** $n = 50$

- **NP-hard**

Investment Example

- **Consider three investments opportunities:**
  - **Invest \$100K, return 1 million 40 years later.**
    - *Annual rate of return: 5.9%*
  - **Invest \$80K, return 0.95 million 40 years later.**
    - *Annual rate of return: 6.4%*
  - **Invest \$50K, return 0.35 million 40 years later.**
    - *Annual rate of return: 5.0%*

- **Budget constraint: \$200K.**

- **Objective: Maximize z: the return after 40 years.**

How to Branch

- **LP relaxation**

$$\text{Maximize:} 1000x_1 + 950x_2 + 350x_3$$
$$\text{s.t.} \quad 100x_1 + 80x_2 + 50x_3 \leq 200$$
$$0 \leq x_1, x_2, x_3 \leq 1$$

$\Rightarrow$ $(x_1, x_2, x_3) = (1, 1, 0.4)$; Objective: 2090

- **Branch: consider two sub-problems:**
  - $x_3 = 0$
  - $x_3 = 1$

How to Bound

- **First branch: $x_3 = 0$**

$$\text{Maximize:} 1000x_1 + 950x_2 + 350x_3$$
$$\text{s.t.} \quad 100x_1 + 80x_2 + 50x_3 \leq 200$$
$$0 \leq x_1, x_2 \leq 1, x_3 = 0$$

$\Rightarrow$ $\boxed{(x_1, x_2, x_3) = (1, 1, 0)\text{; Objective: 1950}}$

- **What information do we get from this?**
  - $\boldsymbol{z} \geq 1950$
  - $\boldsymbol{z} \leq 2090$ (from the full problem)

How to Bound

- **Second branch: $x_3 = 1$**

$$\text{Maximize:} 1000x_1 + 950x_2 + 350x_3$$
$$\text{s.t.} \quad 100x_1 + 80x_2 + 50x_3 \leq 200$$
$$0 \leq x_1, x_2 \leq 1, x_3 = 1$$

$\Rightarrow$ $(x_1, x_2, x_3) = (0.7, 1, 1)$; Objective: 2000

- **What information do we get from this?**
  - $1950 \leq \boldsymbol{z} \leq 2000$
  - Need to further branch: maybe a better solution than (1, 1, 0) lies in this branch.

Branches in Branch $x_3 = 1$

- **First branch: $x_1 = 0$**

$$\text{Maximize:} 1000x_1 + 950x_2 + 350x_3$$
$$\text{s.t.} \quad 100x_1 + 80x_2 + 50x_3 \leq 200$$
$$0 \leq x_2 \leq 1, x_1 = 0, x_3 = 1$$

$\boxed{(x_1, x_2, x_3) = (0, 1, 1)\text{; Objective: 1300}}$

- **What information do we get from this?**
  - Since $1950 \leq \boldsymbol{z} \leq 2000$, this branch can not contain optimal solution, no need to explore further.

Branches in Branch $x_3 = 1$

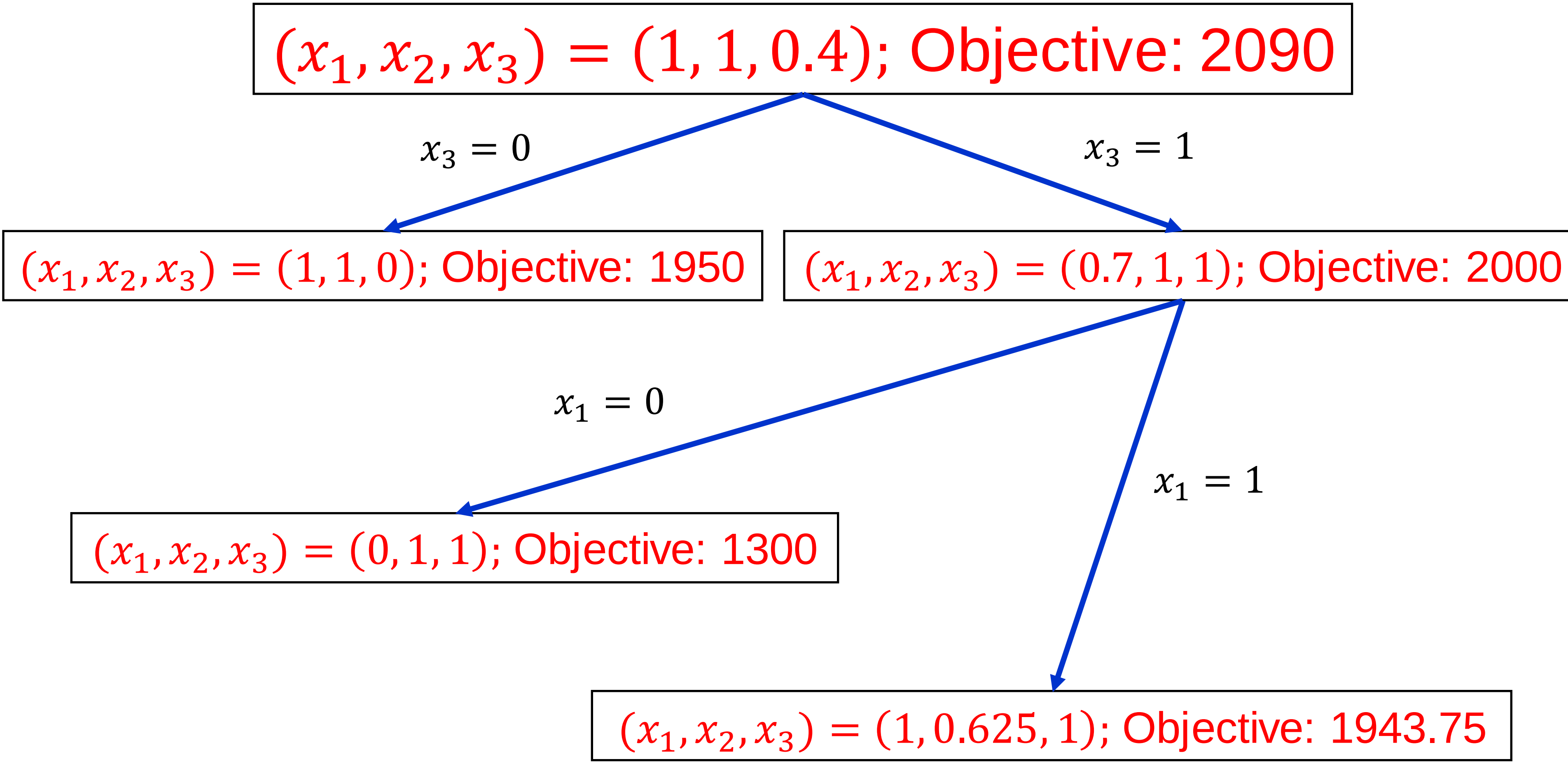
- **Second branch:** $x_1 = 1$

$$\text{Maximize:} 1000x_1 + 950x_2 + 350x_3$$
$$\text{s.t.} \quad 100x_1 + 80x_2 + 50x_3 \leq 200$$
$$0 \leq x_2 \leq 1, x_1 = 1, x_3 = 1$$

$\boxed{(x_1, x_2, x_3) = (1, 0.625, 1)\text{; Objective: 1943.75}}$

- **What information do we get from this?**
  - Since $1950 \leq \boldsymbol{z} \leq 2000$, this branch can not contain optimal solution, no need to explore further.

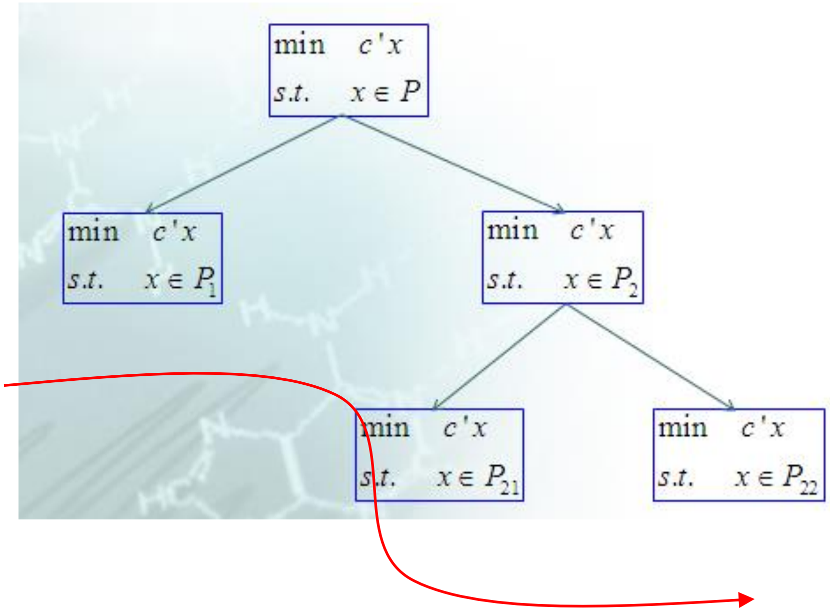
Tree View



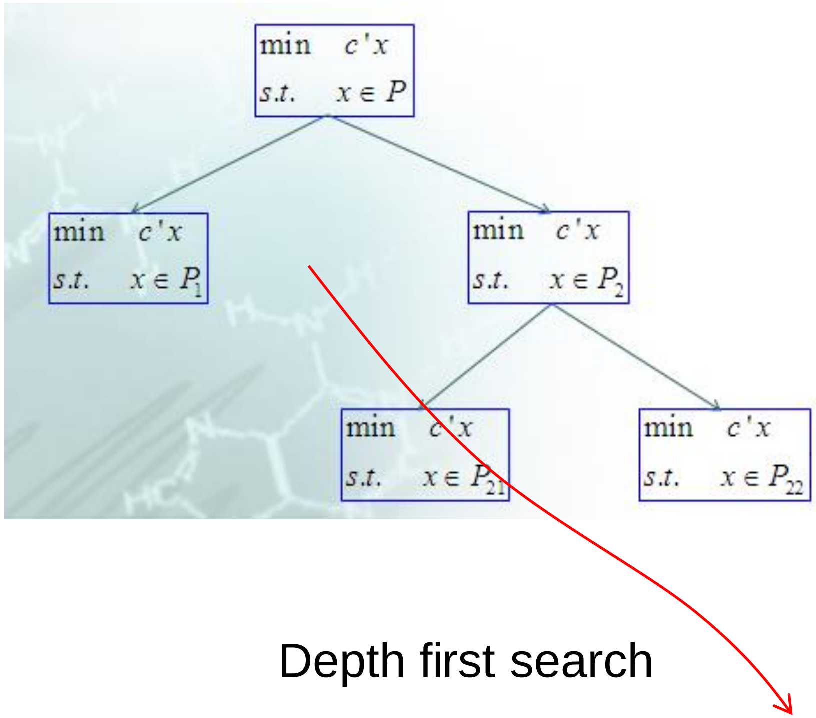
$(x_1, x_2, x_3) = (1, 1, 0.4)$; Objective: 2090
$x_3 = 0$
$x_3 = 1$
$(x_1, x_2, x_3) = (1, 1, 0)$; Objective: 1950
$(x_1, x_2, x_3) = (0.7, 1, 1)$; Objective: 2000
$x_1 = 0$
$x_1 = 1$
$(x_1, x_2, x_3) = (0, 1, 1)$; Objective: 1300
$(x_1, x_2, x_3) = (1, 0.625, 1)$; Objective: 1943.75

Variations

- **Choices of active sub-problems.**



Breath first search



Depth first search