中山大學
SUN YAT-SEN UNIVERSITY

Lecture 7
Integer Programming

Algorithm

张子臻，中山大学计算机学院

zhangzizhen@gmail.com

Outline

- Integer Programming
- Big-M Transformation
- Solution to LP & IP
- Branch and Bound

Integer Programming

- Pure Integer Programming Problem (IP, 全整数规划)

  Maximize $z = 3x_1 + 2x_2$

  Subject to $x_1 + x_2 \leq 6$

  $x_1, x_2 \geq 0$, x_1, x_2 integer

- Mixed Integer Linear Programming Problem (MILP, 混合整数规划)

  Maximize $z = 3x_1 + 2x_2$

  Subject to $x_1 + x_2 \leq 6$

  $x_1, x_2 \geq 0$, x_1 integer

- Binary Integer Programming Problem (0-1规划)

  Maximize $z = x_1 - x_2$

  Subject to $x_1 + 2x_2 \leq 2$

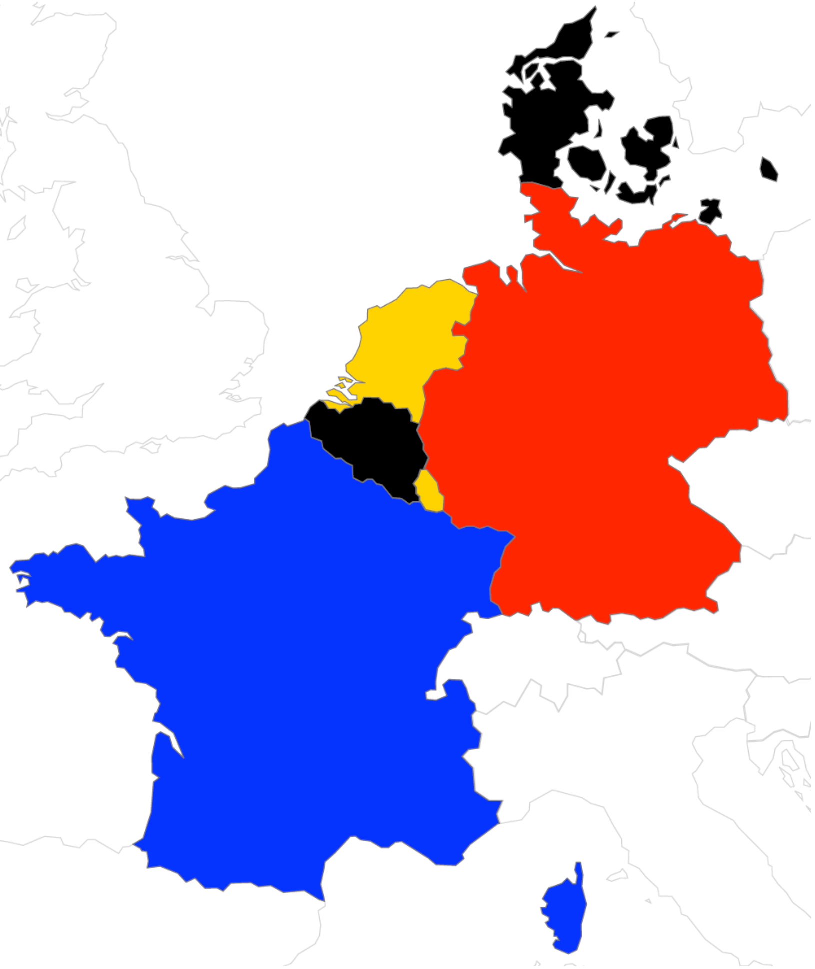
  $2x_1 - x_2 \leq 1$

  $x_1, x_2 = 0$ or 1

Coloring a Map

- Use the minimum colors

```
enum Countries = { Belgium, Denmark, France,
                   Germany, Netherlands, Luxembourg };
var{int} color[Countries] in 0..3;
minimize
 max(c in Countries) color[c]
subject to {
  color[Belgium] ≠ color[France];
  color[Belgium] ≠ color[Germany];
  color[Belgium] ≠ color[Netherlands];
  color[Belgium] ≠ color[Luxembourg];
  color[Denmark] ≠ color[Germany];
  color[France]  ≠ color[Germany];
  color[France]  ≠ color[Luxembourg];
  color[Germany] ≠ color[Netherlands];
  color[Germany] ≠ color[Luxembourg];
}
```



Big-M Transformation of Inequality Constraints

- A constraint $x \neq y$ is not a linear constraint.
- Re-express it as

$$x \neq y \leftrightarrow x \leq y - 1 \vee x \geq y + 1$$

- The disjunction ("∨") is not allowed in an MIP model.
- Introduce a 0/1 variable ***b*** and a large number ***M***.

$$\begin{cases} x \leq y - 1 + bM \\ x \geq y + 1 - (1 - b)M \\ b \in \{0,1\} \end{cases}$$

- This is the big-M rewriting of $x \neq y$.

Big-M Transformation of Inequality Constraints

$$\begin{cases} x \leq y - 1 + bM \\ x \geq y + 1 - (1 - b)M \\ b \in \{0,1\} \end{cases}$$

- The intuition is as follows.
  - When b = 1
    - Constraint $x \leq y - 1 + bM$ is trivially satisfied.
    - The second constraint then becomes $x \geq y + 1$.
  - When b = 0
    - Constraint $x \geq y + 1 - (1 - b)M$ is trivially satisfied.
    - The first constraint then becomes $x \leq y - 1$.

Big-M Transformation

- Absolute constraint:

$$|f(x)| \geq a \;\Leftrightarrow f(x) \geq a \text{ or } f(x) \leq -a$$

- Introduce a binary variable y and a big number M
  1. $-f(x) + a \leq M(1-y)$
  2. $f(x) + a \leq My$

- *N* choose *k* condition:

$$f_i(x) \leq 0, i = 1, \ldots, n$$

- Introduce *n* binary variables y_i and a big number M
  1. $f_i(x) \leq M(1-y_i)$
  2. $y_1 + \cdots + y_i = k$

Big-M Transformation: Multiplication

- Multiplication
  - $y = x_1 \cdot x_2,\ x_1, x_2 \in \{0,1\}$
  - $\begin{cases} y \le x_1 \\ y \le x_2 \\ y \ge x_1 + x_2 - 1 \\ y \in \{0,1\} \end{cases}$
  - $y = x_1 \cdot x_2,\ x_1 \in \{0,1\},\ l \le x_2 \le u$
  - $\begin{cases} y \le x_2 \\ y \ge x_2 - u(1 - x_1) \\ l x_1 \le y \le u x_1 \end{cases}$

Big-M Transformation: Min-Max

- $\min\{\max_i x_i\}$
  - $\min z$
  - $z \geq x_i, \ \forall i$

- $\min\{\min_i x_i\}$
  - $\min z$
  - $\begin{cases} z \geq x_i - M(1 - y_i), \ \forall i \\ \sum_i y_i = 1 \\ y_i \in \{0,1\} \end{cases}$

Big-M Transformation: If-Then

- If $Ax - b \leq 0$ then $Cx - d \leq 0$
- Introduce a binary variable $\delta \in \{0,1\}$, a big number M, and a tiny number $\epsilon > 0$
  1. $Ax - b \geq -\delta M + \epsilon$
  2. $Cx - d \leq (1 - \delta)M$
- Explanation:
  - When $\delta = 1$
    - $Ax - b \geq -M$, trivial
    - $Cx - d \leq 0$
  - When $\delta = 0$
    - $Ax - b \geq \epsilon > 0$
    - $Cx - d \leq M$, trivial

Big-M Transformation: If-Then-Else

- If $Ax - b \leq 0$ then $C_1x - d_1 \leq 0$ else $C_2x - d_2 \leq 0$
- Introduce a binary variable $\delta \in \{0,1\}$, a big number M, and a tiny number $\epsilon > 0$
  1. $Ax - b \leq (1-\delta)M$
  2. $Ax - b \geq -\delta M + \epsilon$
  3. $C_1x - d_1 \leq (1-\delta)M$
  4. $C_2x - d_2 \leq \delta M$
- Explanation:
  - When $\delta = 1$
    - $Ax - b \leq 0$ and $C_1x - d_1 \leq 0$
    - Constraints (2) and (4) are trivially satisfied
  - When $\delta = 0$
    - $Ax - b \geq \epsilon$ and $C_2x - d_2 \leq 0$
    - Constraints (1) and (3) are trivially satisfied

LP Relaxation

- Definition: The LP obtained by omitting all integer or 0-1 constraints on variables is called LP relaxation (松弛) of IP.
- The feasible region for any IP must be contained in the feasible region for the corresponding LP relaxation.

$$\text{Maximize } 4x_1+9x_2+6x_3$$

$$\text{Subject to} \quad 5x_1+8x_2+6x_3 \leq 12$$

$$x_1, x_2, x_3 \text{ are binary variables.}$$

- For the LP relaxation ($0 \leq x_i \leq 1$), we have
  - $x_1=0$, $x_2=1$, $x_3=2/3$, and $Z_{LP}=13$
- We can claim that the optimal solution to the IP cannot be more than 13.
  - Actually $Z_{IP}=10$

Optimal Solution to IP and LP

Max z = $21x_1+11x_2$

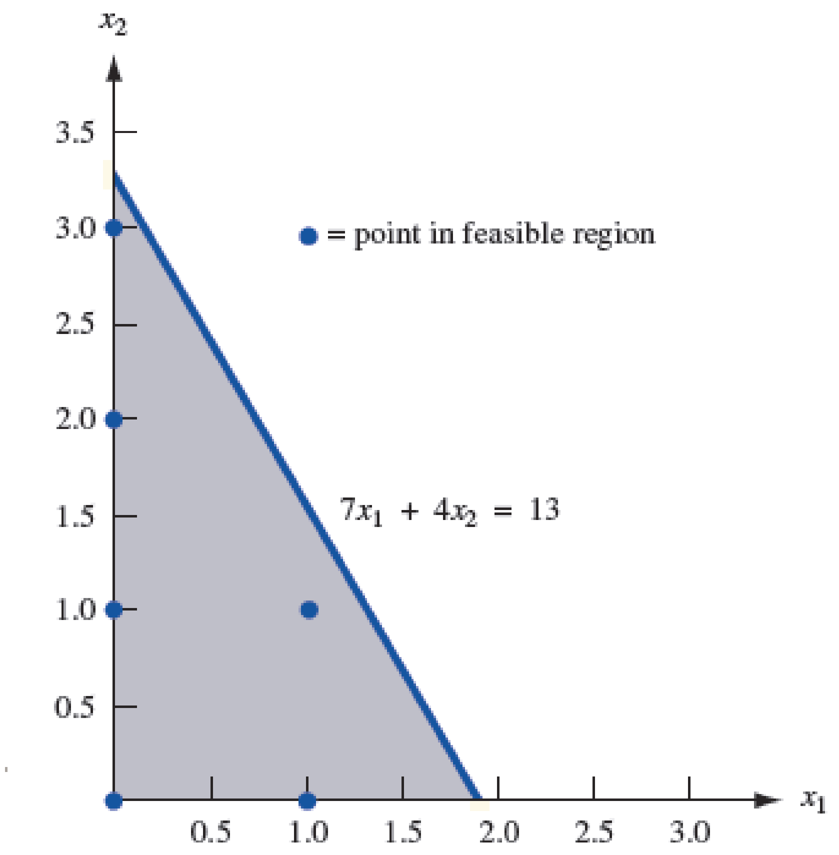
Subject to $7x_1+4x_2\leq 13$

$x_1,x_2 \geq 0$, x_1,x_2 integer

- Feasible region is:

  {(0,0), (0,1), (0,2), (0,3),(1,0),(1,1)}
- Optimal solution to the LP relaxation is:

  $(x_1, x_2) = (13/7, 0)$.
- Optimal IP solution is?



Difficulty of MILP

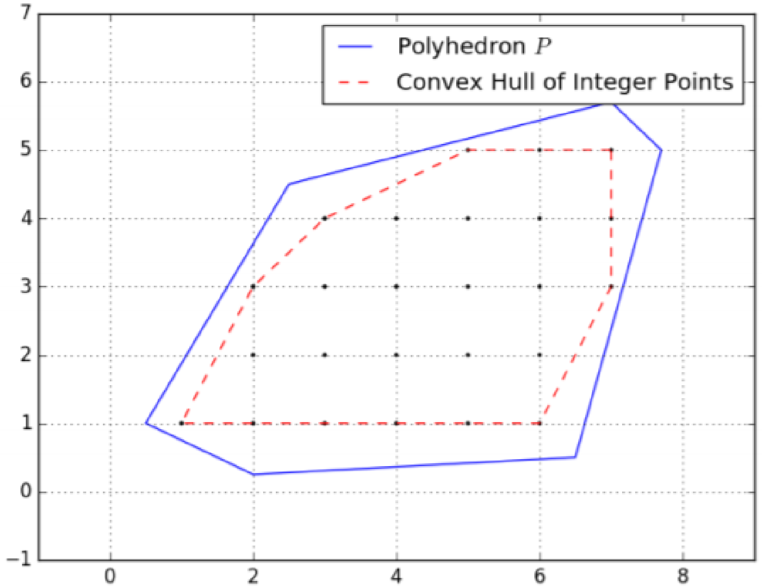
- Solving general integer MILPs can be much more difficult than solving LPs.
- There is no known polynomial-time algorithm for solving general MILPs.
- One reason why convex problems are "easy" to solve is because convexity makes it easy to find improving feasible directions.
- The feasible region of an MILP is non-convex and this makes it difficult to find feasible directions.
- Although the feasible set is non-convex, there is a convex set over which we can optimize in order to get a solution.
- The challenge is that we do not know how to describe that set.
- Even if we knew the description, it would in general be too large to write down explicitly.

The Geometry of MILP

- Let us consider an integer optimization problem:

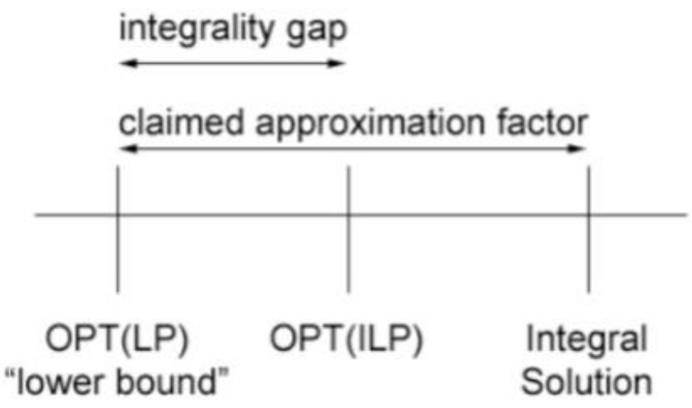
$$
\begin{aligned}
\max \quad & c^\top x \\
\text{s.t.} \quad & Ax \leq b \\
& x \in \mathbb{Z}_+^n
\end{aligned}
$$

- The feasible region is the integer points inside a polyhedron.



LP Rounding Algorithm

1. Reduce the problem to an integer program.
2. Relax the integrality constraint, that is, allow variables to take on non-integral values.
3. Solve the resulting linear program to obtain a fractional optimal solution.
4. "Round" the fractional solution to obtain an integral feasible solution.

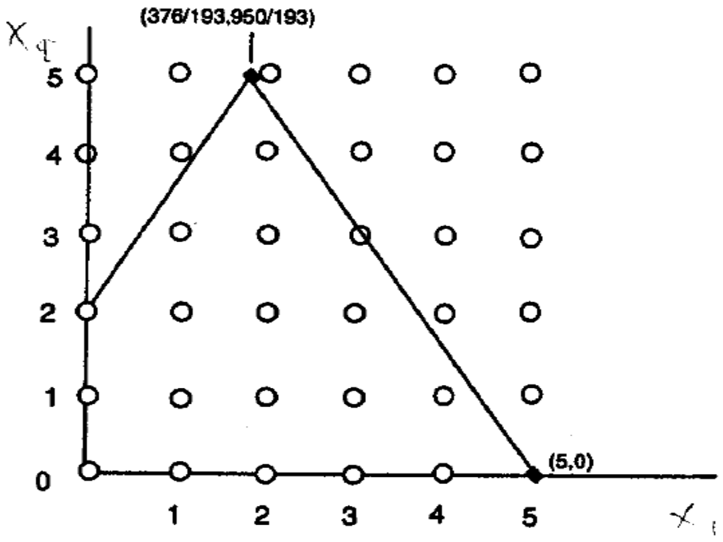


The relationship between the optimal LP and ILP values for minimization problems.

Rounding the Solution

- Rounding the solution of the corresponding LP to nearby integers usually does not work.

$$\begin{aligned} & \text{Max } 1.00x_1 + 0.64x_2 \\ \text{s.t. } \; & 50x_1 + 31x_2 \leq 250 \\ & 3x_1 - 2x_2 \geq -4 \\ & x_1, x_2 \geq 0 \text{ and integer} \end{aligned}$$



- If we solve this model as a linear programming model, the optimal solution is x=(376/193, 950/193) with objective 5.1.
- The LP rounding gives x=(2,4) with objective 4.56.
- The optimal solution is x=(5,0) with objective 5.

Worse Situation

- For Binary Integer Programming model, the situation is often even worse.
- The linear programming solution may be $(x_1, \ldots, x_n) = (0.5, \ldots, 0.5)$.
- It is typically very difficult just to answer the question whether there exists a feasible 0-1 solution.

Observation

- Elementary but important observation:

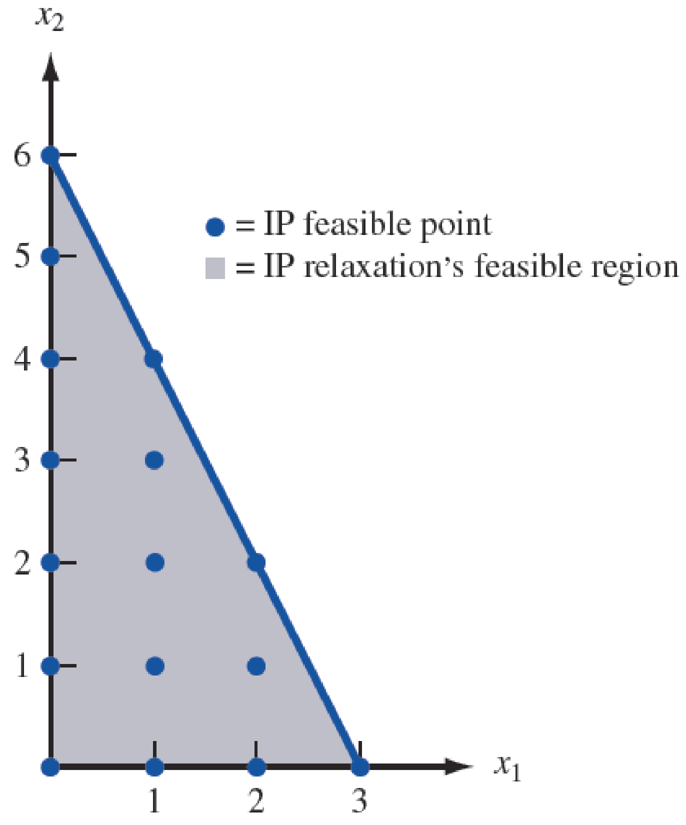
  If you solve the LP relaxation of a pure IP and obtain a solution in which all variables are integers, then the optimal solution to the LP relaxation is also the optimal solution to the IP.

- Example:

$$\max z = 3x_1 + 2x_2$$
$$\text{s.t.} \quad 2x_1 + x_2 \leq 6$$
$$x_1, x_2 \geq 0;\ x_1, x_2 \text{ integer}$$

- The optimal solution to the LP relaxation is:

  x_1=0, x_2=6, z=12.



Branch and Bound: Branch

- For an IP, we can gradually decompose it into a series of smaller IP problems.
- Example: For minimization problem, if x_1 is a binary variable, then we may have two small problems, IP_1 and IP_2.
  - In IP_1, fix x_1=0. Suppose Z_{IP1} is the optimal solution.
  - In IP_2, fix x_1=1. Suppose Z_{IP2} is the optimal solution.
  - Then, $Z_{IP}=min(Z_{IP1}, Z_{IP2})$
- If x_2 is an integer variable, then branch <=floor(x_2) and >=ceil(x_2).

Branch and Bound: Bound

- Suppose we have solved IP1 and got Z_{IP1}. (minimization problem)
- Consider IP2:
  - For the LP relaxation of IP2, Z_{LP2} can be obtained.
  - If $Z_{LP2} \geq Z_{IP1}$, we do not need to solve IP2.
  - Reason: $Z_{IP2} \geq Z_{LP2} \geq Z_{IP1}$, so Z_{IP2} cannot be the optimal solution to the original IP.
- This is called ***BOUND***
  - For each smaller IP, we can solve its LP relaxation and see if we can ***fathom*** (finish, prune) it directly, meaning that we will no longer need to solve it.

General Approach of B&B

- Suppose we have a feasible solution of the minimization IP at hand, the objective function value is z_B.
  - z_B is an upper bound of the problem.
- Considering the LP relaxation of a small IP:
  - Case 1: The LP relaxation has no feasible solution.
    - The IP has no feasible solution either.
  - Case 2: In the LP relaxation, $Z_{LP} \geq Z_B$
    - The IP cannot have a better solution than Z_B, thus can be fathomed.
  - Case 3: In the LP relaxation, an integer optimal solution is found, and $Z_{LP} < Z_B$.
    - This small IP is solved, and update $Z_B = Z_{LP}$.
  - Case 4: In the LP relaxation, an optimal solution is found with $Z_{LP} > Z_B$, but not integer value.
    - Decompose the IP into more smaller IP problems – more branches.

Combinatorial Relaxation

- We can also relax several constraints of the problem.
- Whenever the relaxed problem is a combinatorial optimization problem, we speak of a combinatorial relaxation.
- In many cases, the relaxation is an easy problem that can be solved rapidly.

TSP Model

- Decision variables: $x_{ij} = \begin{cases} 1 & \text{if edge } (i,j) \in E \text{ is in tour} \\ 0 & \text{otherwise.} \end{cases}$
- IP model:

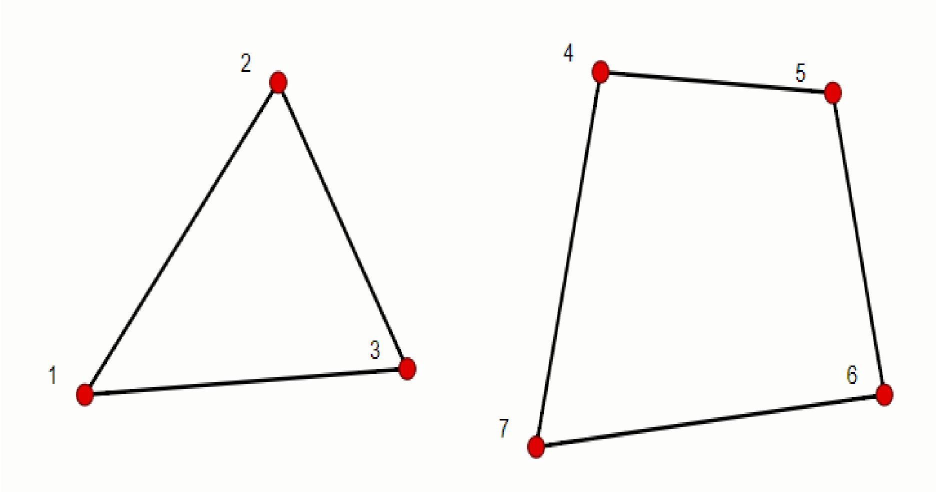
$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}.$$

$$\sum_{j:j \neq i} x_{ij} = 1 \text{ for } i = 1, \ldots, n.$$

$$\sum_{i:i \neq j} x_{ij} = 1 \text{ for } j = 1, \ldots, n.$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \text{ for } S \subset N, 2 \leq |S| \leq n-1.$$

$$x_{ij} \in \{0,1\} \text{ for } i = 1, \ldots, n, j = 1, \ldots, n, i \neq j.$$



- Sub-tour elimination constraints:

$$\sum_{i,j \in S,\, i \neq j} x_{ij} \leq |S| - 1, \quad \forall S \subset V, S \neq \emptyset$$

These constraints require that for each proper (nonempty) subset S of the set of cities V, the number of edges between the nodes of S must be at most $|S| - 1$.

Assignment Problem Bound for TSP

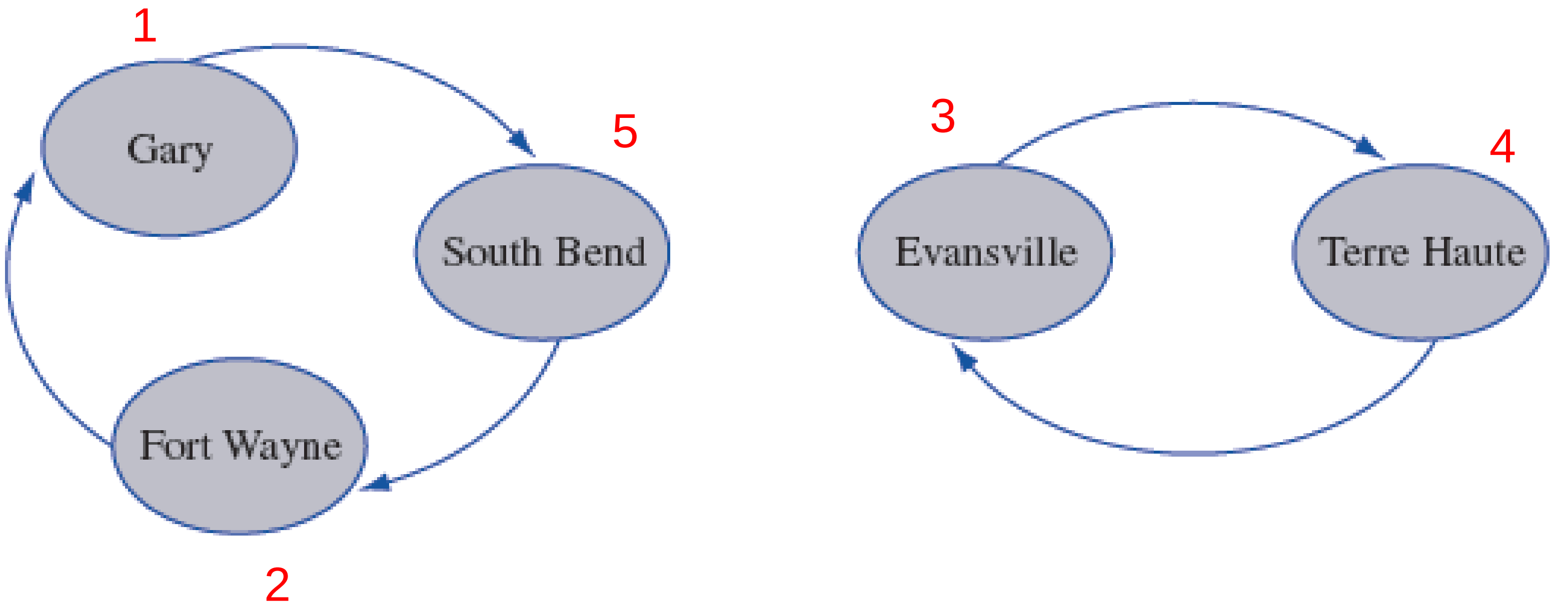
- We might be able to find the answer of TSP by solving an *assignment problem* having a cost matrix whose *ij*-th element is d_{ij} if sub-tour elimination constraints are removed.
- For instance, suppose we solved this assignment problem and obtained the solution $x_{12}=x_{24}=x_{45}=x_{53}=x_{31}=1$.
- This solution can be written as 1→2→4→5→3→1.
- If the solution to the preceding assignment problem yields a tour, then it is the optimal solution to the traveling salesman problem. (Why?)

Distance between Cities in Traveling Salesperson Problem

Day	Gary	Fort Wayne	Evansville	Terre Haute	South Bend
City 1 Gary	0	132	217	164	58
City 2 Fort Wayne	132	0	290	201	79
City 3 Evansville	217	290	0	113	303
City 4 Terre Haute	164	201	113	0	196
City 5 South Bend	58	79	303	196	0

Assignment Problem Bound for TSP

- However, the optimal solution to the assignment problem might be $x_{15}=x_{21}=x_{52}=x_{34}=x_{43}=1$.
- If we could exclude all feasible solutions that contain subtours and then solve the assignment problem, we would obtain the optimal solution to the traveling salesman problem.



B&B for TSP

- The subproblems reduce to *assignment problems*.
- We first solve the assignment problem in the following table (referred to as **subproblem 1**).
  - The optimal solution is $x_{15}=x_{21}=x_{34}=x_{43}=x_{52}=1$, $z=495$.
  - This solution contains two subtours (1-5-2-1 and 3-4-3) and cannot be the optimal solution to TSP.

Cost Matrix for Subproblem 1

	City 1	City 2	City 3	City 4	City 5
City 1	M	132	217	164	58
City 2	132	M	290	201	79
City 3	217	290	M	113	303
City 4	164	201	113	M	196
City 5	58	79	303	196	M

$t = 1$

Subproblem 1

$z = 495$

$x_{15} = x_{21} = x_{34} = x_{43} = x_{52} = 1$

B&B for TSP

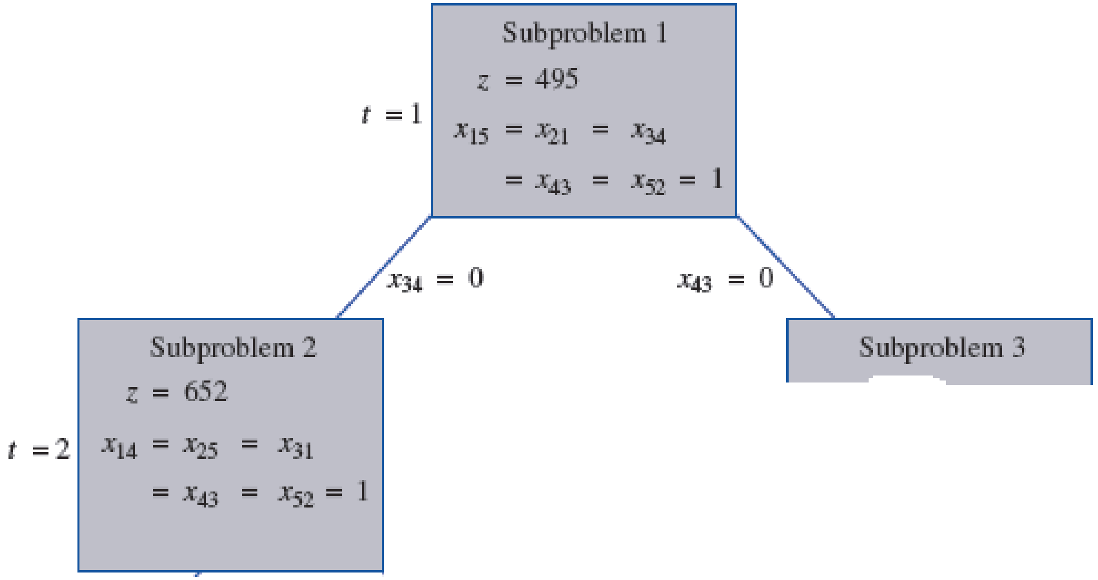
- We branch on subproblem 1 in a way that will prevent one of subproblem 1's subtours from recurring in solutions to subsequent subproblems.
- We choose to exclude the subtour 3→4→3.
- Observe that the optimal solution to TSP must have either x_{34}=0 or x_{43}=0.
- Thus, we can branch on subproblem 1 by adding the following two subproblems:
  - **Subproblem 2**: Subproblem 1 + (x_{34} = 0, or $c_{34} = M$).
  - **Subproblem 3**: Subproblem 1 + (x_{43}=0, or $c_{43} = M$).

B&B for TSP

- We arbitrarily choose subproblem 2 to solve, applying the Hungarian method to the cost matrix:

Cost Matrix for Subproblem 2

	City 1	City 2	City 3	City 4	City 5
City 1	M	132	217	164	58
City 2	132	M	290	201	79
City 3	217	290	M	M	303
City 4	164	201	113	M	196
City 5	58	79	303	196	M



- The optimal solution to subproblem 2 is $z = 652$, $x_{14}=x_{25}=x_{31}=x_{43}=x_{52}=1$.
- This solution includes the subtours 1→4→3→1 and 2→5→2, so this cannot be the optimal solution to TSP.

B&B for TSP

- We branch on subproblem 2 to exclude 2→5→2 by
  - **Subproblem 4**: Subproblem 2 + ($x_{25} = 0$, or $c_{25} = M$).
  - **Subproblem 5**: Subproblem 2 + ($x_{52} = 0$, or $c_{52} = M$).
  - We solve subproblems 4 & 5.

Cost Matrix for Subproblem 4

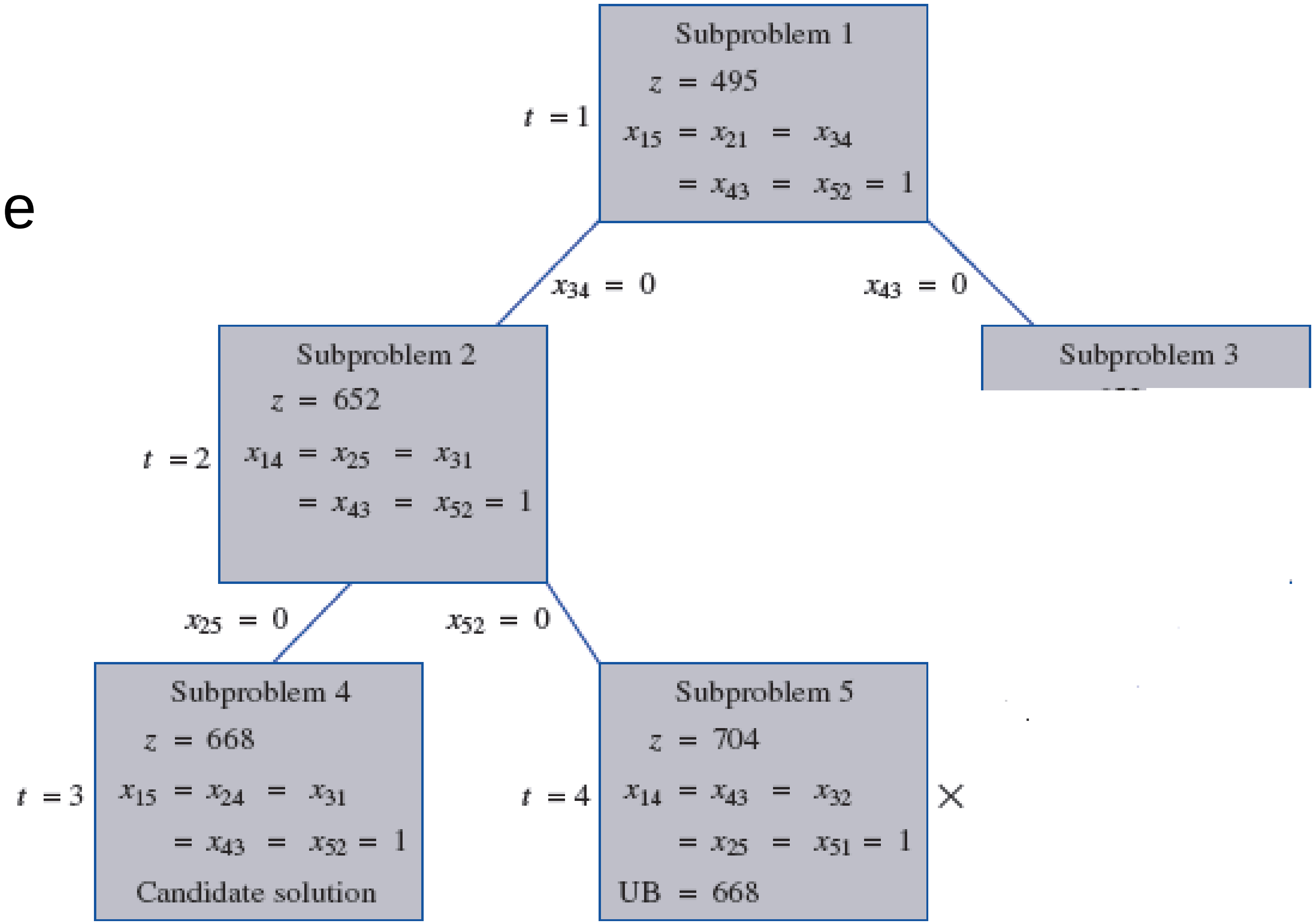
	City 1	City 2	City 3	City 4	City 5
City 1	M	132	217	164	58
City 2	132	M	290	201	M
City 3	217	290	M	M	303
City 4	164	201	113	M	196
City 5	58	79	303	196	M

Cost Matrix for Subproblem 5

	City 1	City 2	City 3	City 4	City 5
City 1	M	132	217	164	58
City 2	132	M	290	201	79
City 3	217	290	M	M	303
City 4	164	201	113	M	196
City 5	58	M	303	196	M

B&B for TSP

- Solving subproblem 4 yields a candidate solution.
- Subproblem 5 cannot update the best solution.

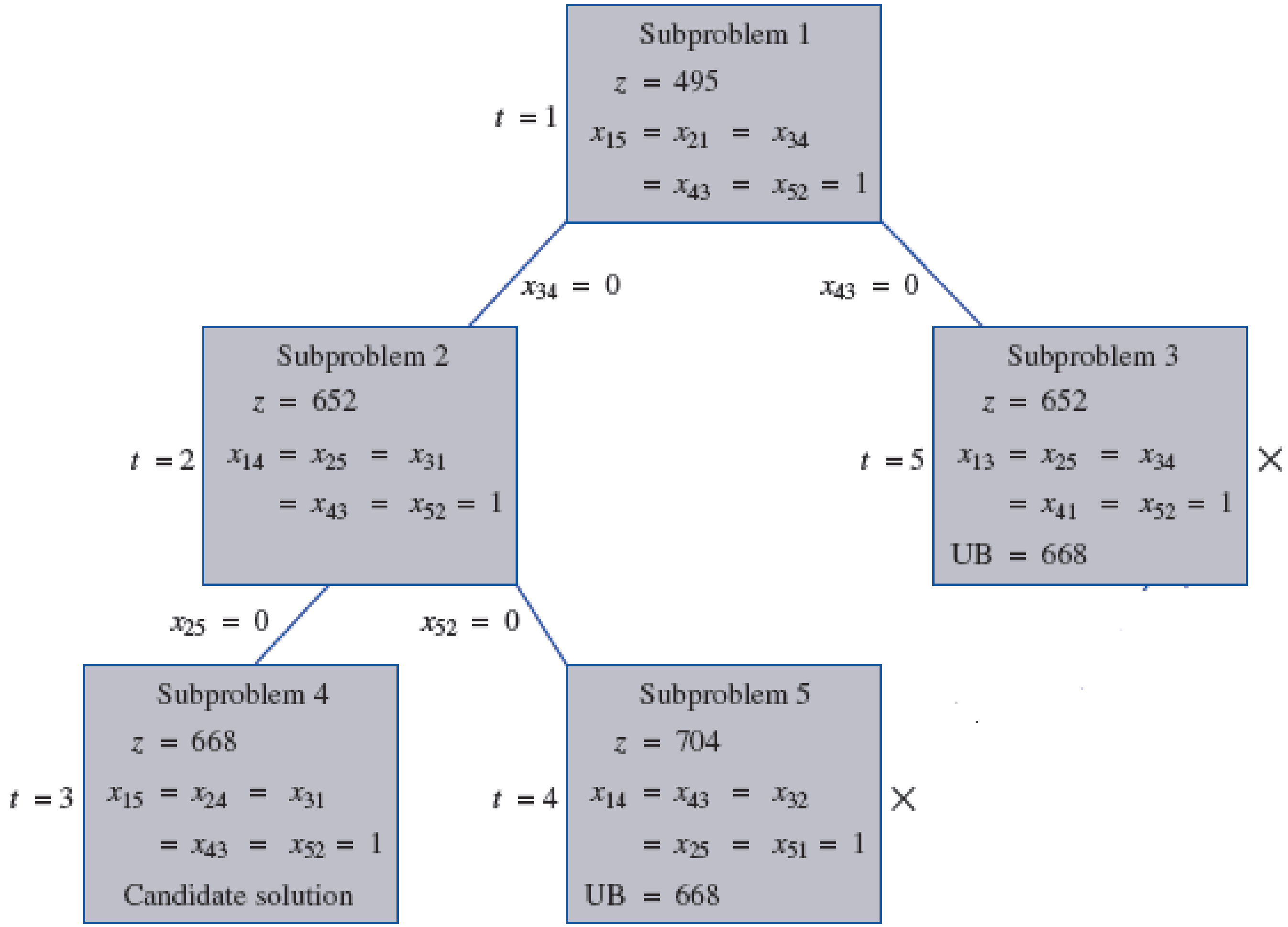


B&B for TSP

- Only subproblem 3 remains.
- We find the optimal solution to the assignment problem in the following table: $x_{13}=x_{25}=x_{34}=x_{41}=x_{52}=1$, $z=652$.

Cost Matrix for Subproblem 3

	City 1	City 2	City 3	City 4	City 5
City 1	M	132	217	164	58
City 2	132	M	290	201	79
City 3	217	290	M	113	303
City 4	164	201	M	M	196
City 5	58	79	303	196	M

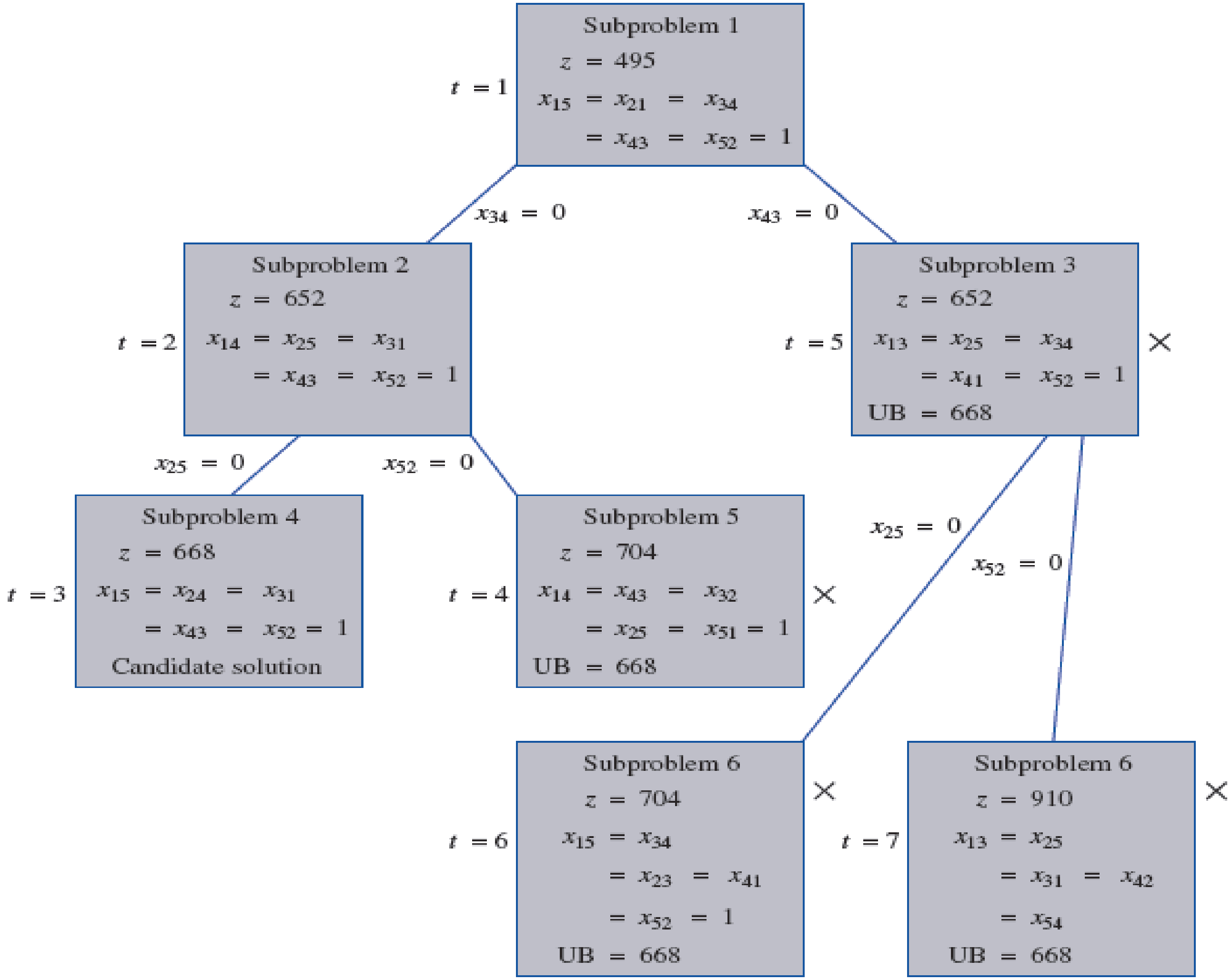


B&B for TSP

- We branch on subproblem 3 to exclude the subtour.
- Any feasible solution to the traveling salesman problem that emanates from subproblem 3 must have either x_{25}=0 or x_{52}=0.
  - **Subproblem 6**:

  Subproblem 3 + (x_{25} = 0, or c_{25} = M).
  - **Subproblem 7**:

  Subproblem 3 + (x_{52} = 0, or c_{52} = M).



1-Tree Bound

$$
\begin{aligned}
z = \min \textstyle\sum_{e \in E} c_e x_e \\
\textstyle\sum_{e \in \delta(i)} x_e = 2 \text{ for all } i \in V \\
\textstyle\sum_{e \in E(S)} x_e \le |S| - 1 \text{ for all } 2 \le |S| \le |V| - 1 \\
x \in B^{|E|}.
\end{aligned}
$$

- We dualize all the degree constraints on the nodes, but leave the degree constraint on node 1, and the constraint that the total number of edges in *n*.

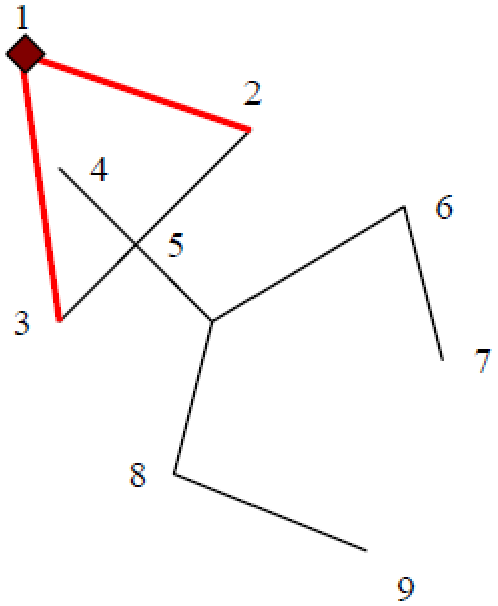
$$
(IP(u)) \qquad
\begin{aligned}
z(u) = \min \ \textstyle\sum_{e \in E} (c_e - u_i - u_j) x_e + 2 \textstyle\sum_{i \in V} u_i \\
\textstyle\sum_{e \in \delta(1)} x_e = 2 \\
\textstyle\sum_{e \in E(S)} x_e \le |S| - 1 \text{ for } 2 \le |S| \le |V| - 1, \ 1 \notin S \\
\textstyle\sum_{e \in E} x_e = n \\
x \in B^{|E|}.
\end{aligned}
$$

- IP(u) is precisely a 1-tree.

1-Tree

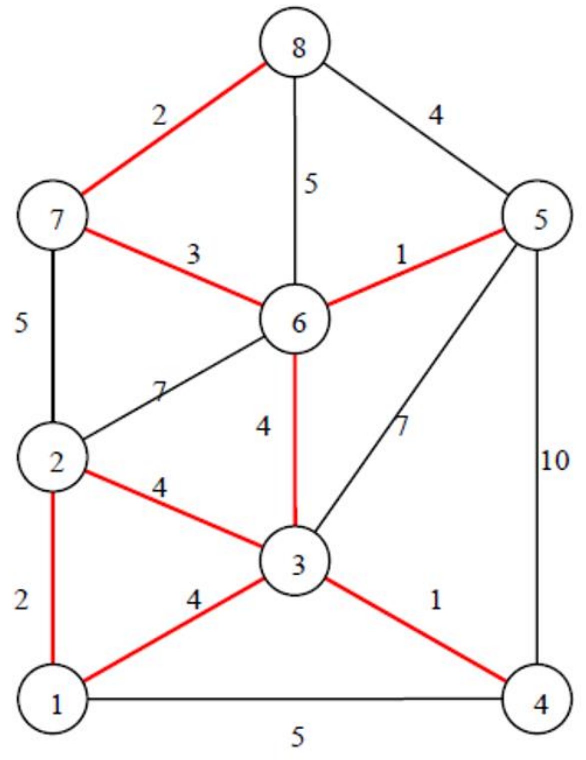
- Definition: for a given vertex, say vertex 1, a 1-Tree is a tree of {2,3,…,n} plus 2 distinct edges connected to vertex 1.

Example of 1-Tree



Minimum Weight 1-Tree

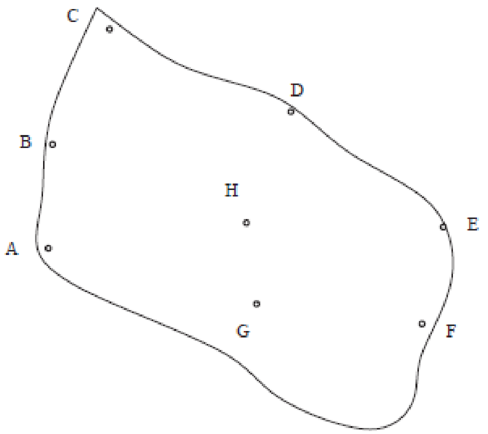
- Definition: Min cost 1-tree of all possible 1-Trees.
- To find minimum weight 1-Tree, first find minimum spanning tree of {2,3,…,n} vertices, and add two lowest cost edges incident to vertex 1.
- Any TSP tour is 1-Tree tour (with arbitrary starting node 1) in which each vertex has a degree of 2.
- If minimum weight 1-Tree is a tour, it is the optimal TSP tour.
- Thus, the minimum 1-Tree provides a lower bound on the length of the optimal TSP tour.



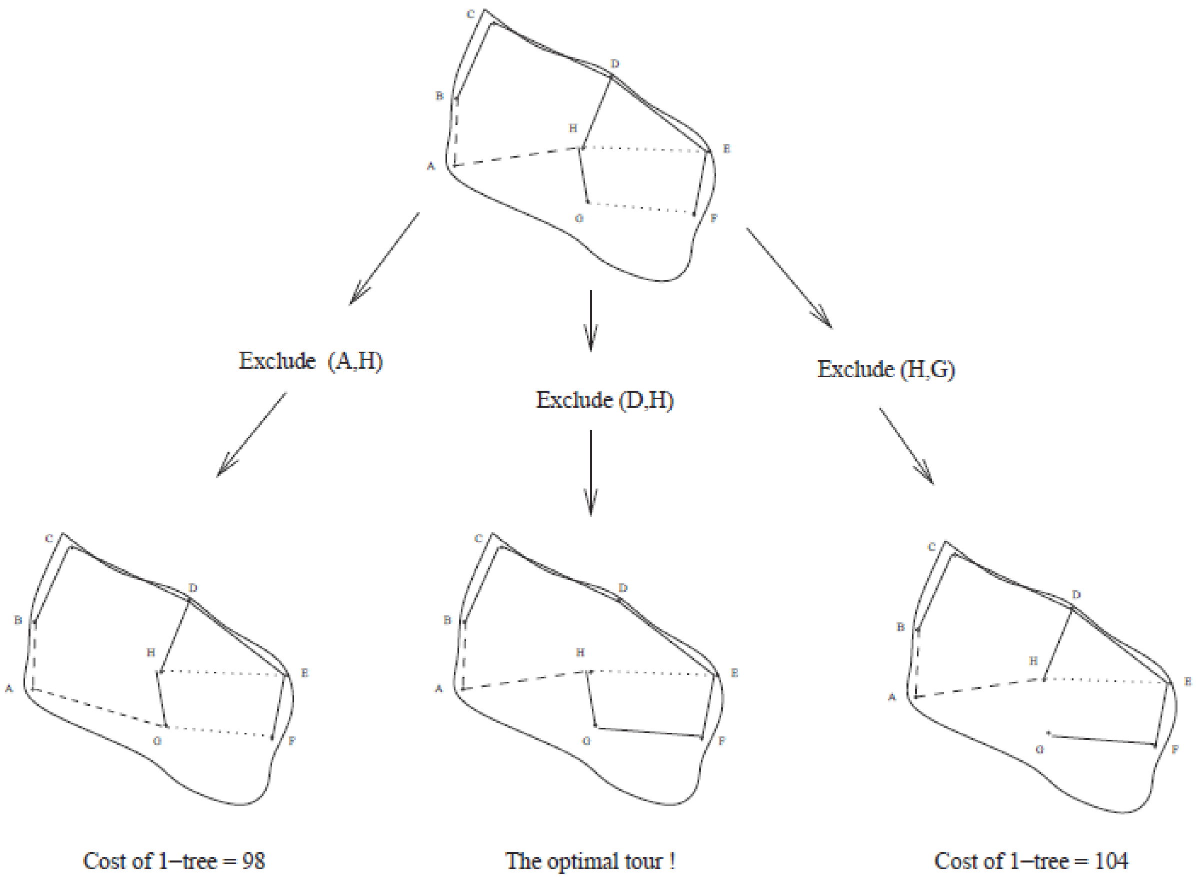
Branching on 1-Tree

- Observe that in the case 1-tree is not a tour, at least one vertex has degree 3 or more.
- So choose a vertex v with degree 3 or more.
- For each edge (u_i, v), generate a subproblem where (u_i, v) is excluded from the set of edges.

Branching on 1-Tree



	A	B	C	D	E	F	G	H
A	0	11	24	25	30	29	15	15
B	11	0	13	20	32	37	17	17
C	24	13	0	16	30	39	29	22
D	25	20	16	0	15	23	18	12
E	30	32	30	15	0	9	23	15
F	29	37	39	23	9	0	14	21
G	15	17	29	18	23	14	0	7
H	15	17	22	12	15	21	7	0



Homework

- Use Gurobi, Cplex or SCIP to solve the TSP instances.
- http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/
- Formulate the Traveling Salesman Problem.
  - Example: TSP with Miller-Tucker-Zemlin (MTZ) model
- Show the meanings of the objective and constraints.
- Use some IP solver to solve the model on the given dataset.
  - burma14, bayg29, bays29, ulysses16, ulysses22, gr17, gr21, gr24, fri26 (node<30).
  - Try larger instances (node>=30).
- Write or print out your report (model, explanations, codes, results, etc.).
- Bring your report to the class on Jun 28, 2024.

Thank you!