Introduction to Optimisation:

Integer Programming

Lectures 10-11

Lecture notes by Dr. Julia Memar and Dr. Hanyu Gu and with an acknowledgement to Dr.FJ Hwang and Dr.Van Ha Do

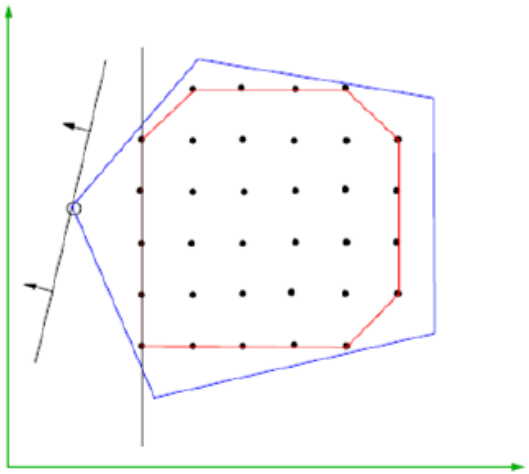
Introduction

- An integer program (IP) is a mathematical programming problem in which some or all of the variables are required to be integers.
- An IP is called
  - a pure IP if all variables are required to be integers,
  - a 0-1 IP or binary IP if all variables must be 0 or 1, and
  - a mixed IP (MIP) if some of the variables are required to be integers.
- There is no "nice" optimality conditions **as for LP and NLP**

$$\min \quad \sum_{i}^{n} c_i x_i$$

$$s.t. \quad \sum_{j}^{n} a_{ij} x_j \leq b_i, \quad i = 1, \cdots, m$$

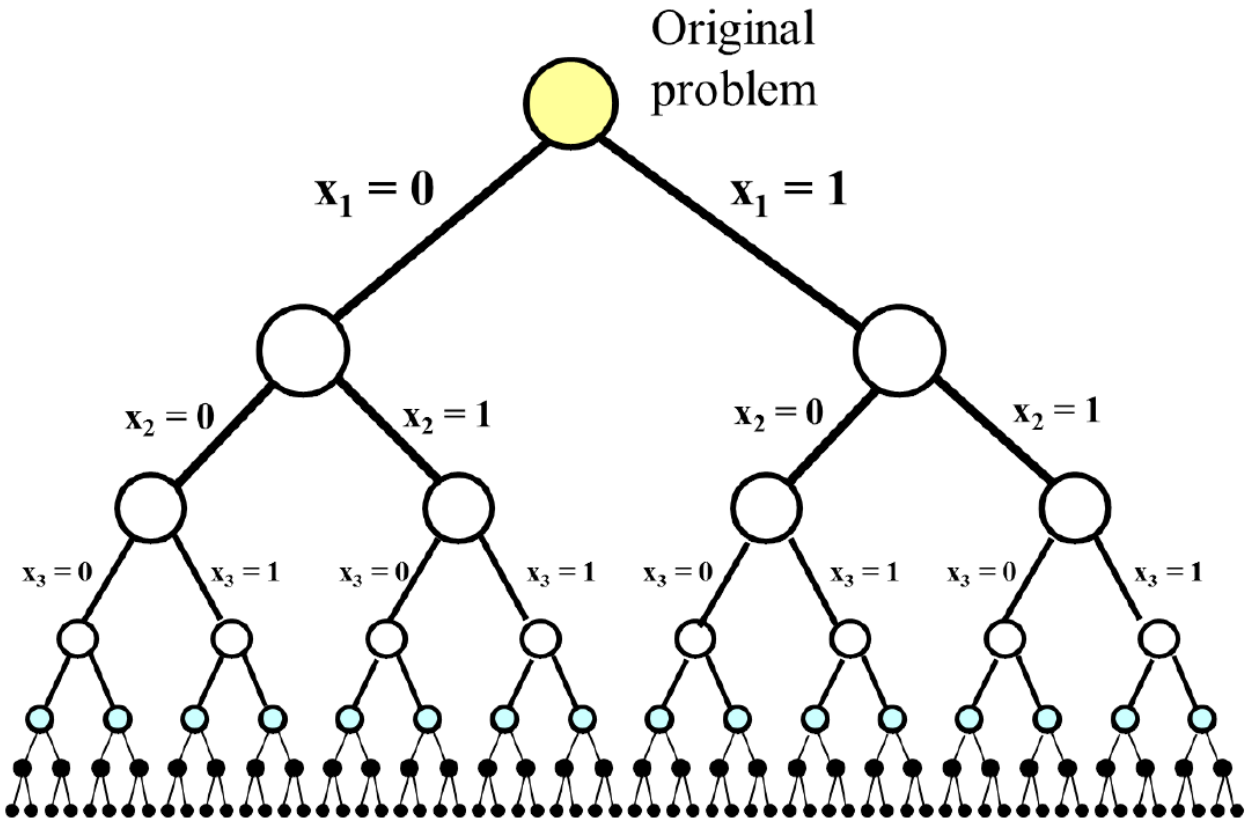
$$x \quad \text{integer}$$



Branch-and-bound algorithm

Branch-and-bound method:

- The feasible region S is replaced by smaller problems S_i- branching.
- Each S_i is a basis of another branching



Branch-and-bound algorithm

Pruning:

To avoid enumeration of all solutions, the search tree is ”pruned” and the following assumptions are made:

- there is an algorithm to calculate lower bound L_i on objective values of feasible solutions of a subproblem S_i
- the upper bound *U* of objective value on *S* can be obtained as an objective value on some solution

Branch-and-bound algorithm

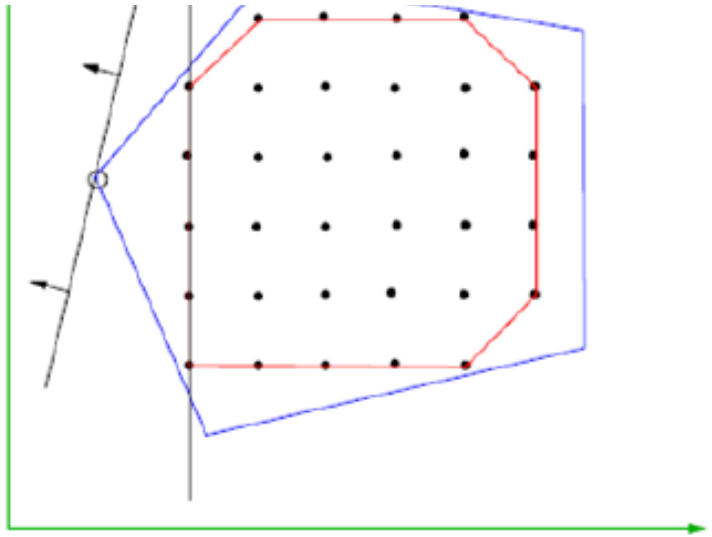
Pruning:

- If for a subproblem S_i

$$L_i \geq U$$

then S_i can/can not improve the objective value on S, hence

- LP relaxation – the integer requirement is ignored/relaxed



Branch-and-bound algorithm

Branching:

- each LP relaxation will represent a node on the solution tree;
- solve the LP relaxation corresponding to the current node.
  - If in resultant solution $x' = (x_1', x_2', \ldots, x_n')$ component x_j' is not integer, then branch on this node by adding constraint

    either $x_j \geq \lfloor x_j' \rfloor + 1$ or $x_j \leq \lfloor x_j' \rfloor$;

Branch-and-bound algorithm

Branching:

- each LP relaxation will represent a node on the solution tree;
- solve the LP relaxation corresponding to the current node.
  - If in resultant solution $x' = (x_1', x_2', \ldots, x_n')$ component x_j' is not integer, then branch on this node by adding constraint

Branch-and-bound algorithm

Bounding:

- If for the current node the LP relaxation provides solution with OF value not better than the incumbent solution, no further branching required, and the node is
  - Optimality gap is useful in practice for large problems
- If the node's LP is infeasible, the node is

Rules to travel the solution tree:

- LIFO rule : Last-In-First- Out (LIFO) rule (depth-first)
- The Jumptracking rule: solves all the problems created by a branching and then branches on the node with the best OF-value (best-first).
- Many other rules for variable order and value order
  - Active research area to make use of machine learning

Example

- $max\ z = 8x_1 + 5x_2$ *(1)*

  s.t. $x_1 + x_2 \leq 6$

  $9x_1 + 5x_2 \leq 45$

  $x_1, x_2 \geq 0$, integer

- Solve the problem graphically:

Example

- $max\ z = 8x_1 + 5x_2$ (*)

  $s.t.$ $x_1 + x_2 \leq 6$

  $9x_1 + 5x_2 \leq 45$

  $x_1, x_2 \geq 0$, integer

Solve the problem with branch-n-bound algorithm:

Node 1 (Subproblem 1): Solve (*) ignoring integer constraint:

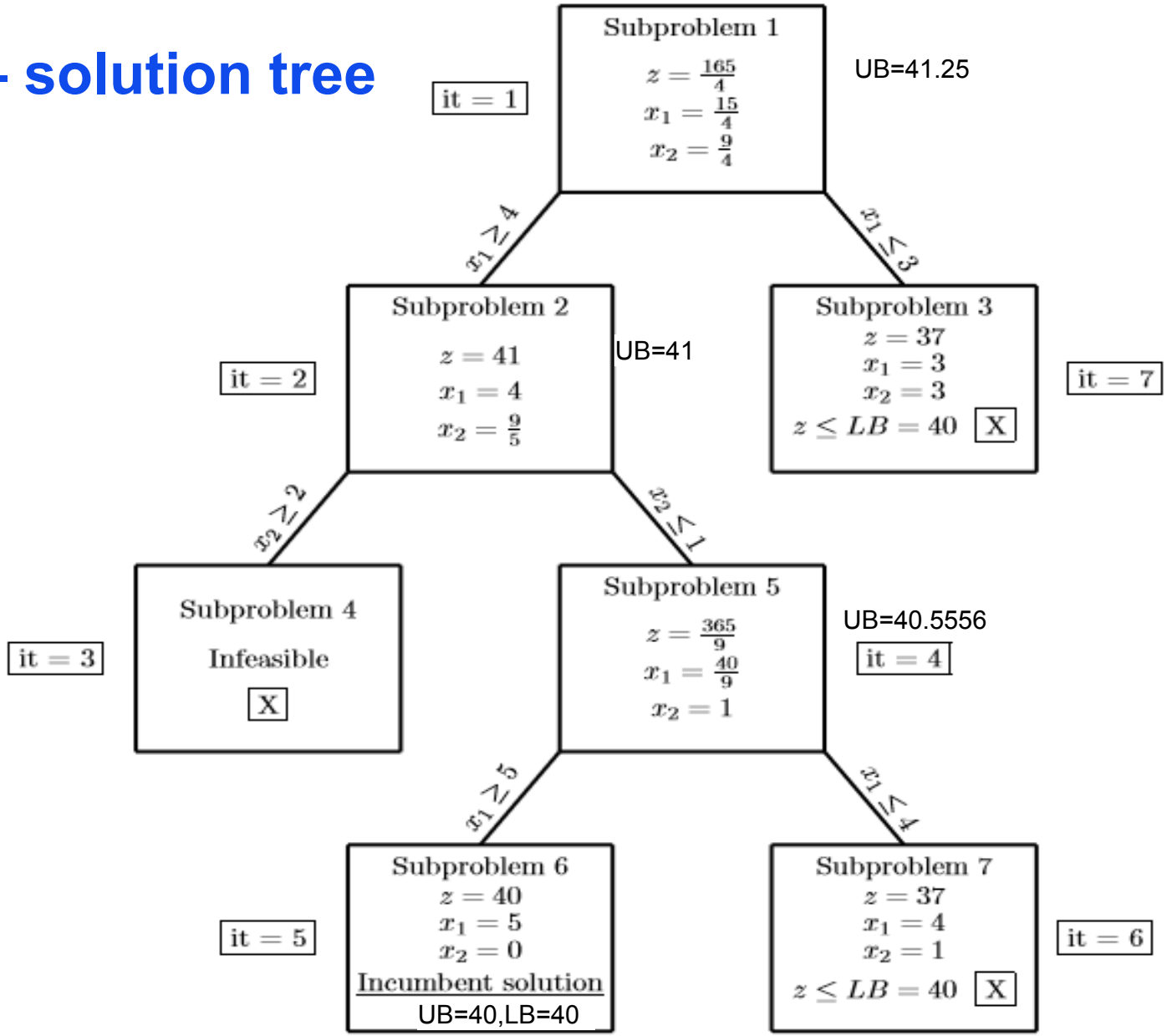
- $max\ z = 8x_1 + 5x_2$ (*)

  $s.t.$ $x_1 + x_2 \leq 6$

  $9x_1 + 5x_2 \leq 45$

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

Example – solution tree



Combinatorial Optimisation problems

- Solution has a combinatorial structure, e.g., an object on a graph; the number of solutions increase exponentially fast
- Can be solved as Mixed Integer Program(MIP), but not efficient for large problems; active research area:
  - Knapsack problem
  - Assignment problem
  - Travelling salesman problem

Knapsack problem

There is a knapsack with a capacity of 14 units. There are 4 items, each of which retains a size and a value, e.g. item 1 has a size of 5 units and a value of 16 dollars, item 2 has a size of 7 units and a value of 22 dollars, item 3 has a size of 4 units and a value of 12 dollars, and item 4 has a size of 3 units and a value of 8 dollars. The objective is to decide which items shall be packed in the knapsack so as to maximise the total value of items in the knapsack. Then the knapsack problem can be formulated as follows:

- $max\ z = 16x_1 + 22x_2 + 12x_3 + 8x_4$

  $s.t. \quad 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14$

  $x_1, x_2, x_3, x_4 \geq 0$ and _____

Knapsack problem

There is no need to use Simplex algorithm ☺

- Finding ratio and placing first the item with

then till only part fits

Solution:

- Branching:

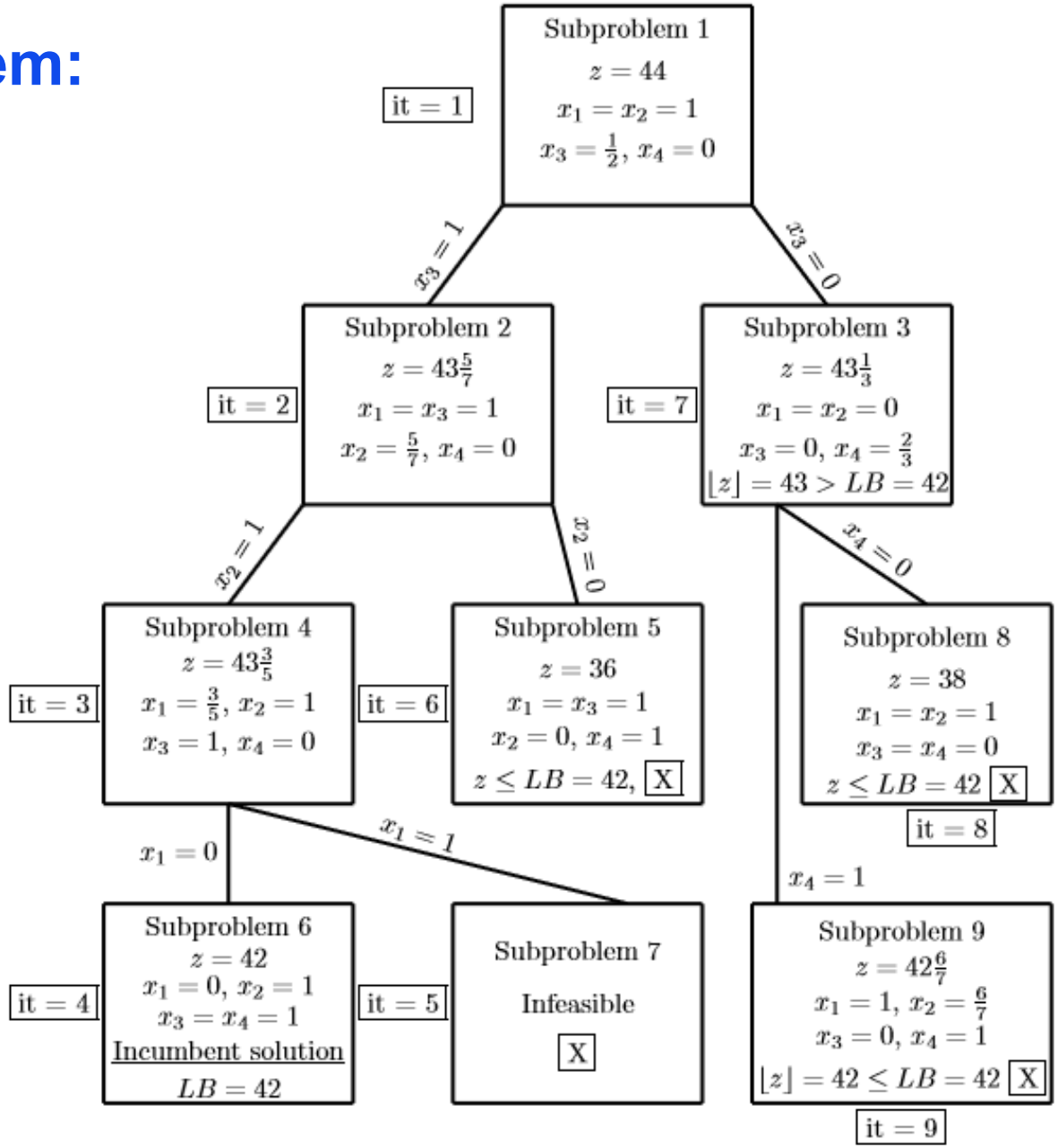
Subproblem 2 or Subproblem 3

$max\ z = 16x_1 + 22x_2 + 8x_4 +$

$s.t.\ \ 5x_1 + 7x_2 + 3x_4 \leq 14 -$

$x_1, x_2, x_4\ are\ binary\ variables$

Knapsack problem: solution tree



Assignment problem - example

A manufacturing factory has four machines which can process four sorts of tasks to be completed. The time required to set up each machine for completing each job is shown in the table below. Any machine which has been assigned to process a task cannot be reassigned to another task. The factory manager aims to minimise the total setup time needed to complete four jobs with these four machines.

	Time (Hours)			
Machine	Task 1	Task 2	Task 3	Task 4
1	14	5	8	7
2	2	12	6	5
3	7	8	3	9
4	2	4	6	10

How many possible assignments are there?

Assignment problem - example

Formulation: Let $x_{ij} =$

$min\, z =$

$s.t.$

Assignment problem - example

- *Totally unimodular matrix:* matrix A is totally unimodular (TU) if every square submatrix of A has determinant −1, 0 or +1

A =

	x_11	x_12	x_13	x_14	x_21	x_22	x_23	x_24	x_31	x_32	x_33	x_34	x_41	x_42	x_43	x_44
i=1	1	1	1	1												
i=2					1	1	1	1								
i=3									1	1	1	1				
i=4													1	1	1	1
j=1	1				1				1				1			
j=2		1				1				1				1		
j=3			1				1				1				1	
j=4				1				1				1				1

Assignment problem - example

- Extreme points of LP are integral if A is *unimodular* and b is integral
  - Network flow problem, shortest path
  - Simplex algorithm can find optimal solution for the assignment problem, i.e., no need for B&B!
  - Simplex algorithm is/is not polynomial…
  - Can be solved by “Hungarian method” in polynomial time

Hungarian method

Theorem: If a number is added to or subtracted from all of the entries of any one row or column of a cost matrix, then on optimal assignment for the resulting cost matrix is also an optimal assignment for the original cost matrix.

Hungarian method

The following algorithm applies the above theorem to a given $n \times n$ cost matrix to find an optimal assignment.

Step 1. Subtract the smallest entry in each row from all the entries of its row.

Step 2. Subtract the smallest entry in each column from all the entries of its column.

Step 3. Draw lines through appropriate rows and columns so that all the zero entries of the cost matrix are covered and the minimum number of such lines is used.

Step 4. Test for Optimality:

(i) If the minimum number of covering lines is n, an optimal assignment of zeros is possible and we are finished.

(ii) If the minimum number of covering lines is less than n, an optimal

assignment of zeros is not yet possible. In that case, proceed to Step 5.

Step 5. Determine the smallest entry not covered by any line. Subtract

this entry from each uncovered row, and then add it to each covered

column. Return to Step 3.

Hungarian method

Row min

Column min

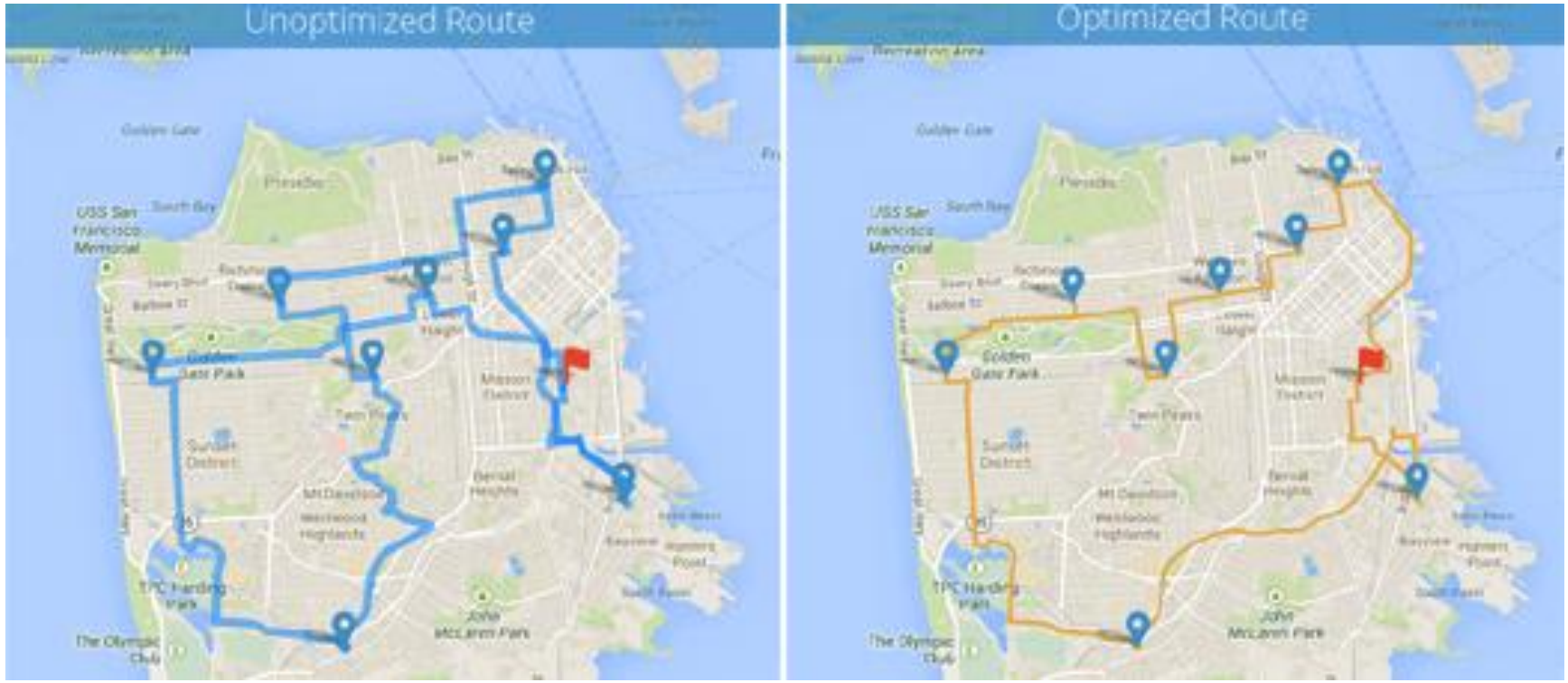
Hungarian method

Row min

Column min

Travelling salesman problem

- Starting from home city, a travelling salesman must travel to each of m cities exactly once before returning home. There is a cost c_{ij}^{*} associated with a travel from city i to city j. Find the route minimizing the total trip cost.



Travelling salesman problem

Formulation:

Let $x_{ij} =$

$c_{ij} =$

$min\, z =$

s.t.

Issues:

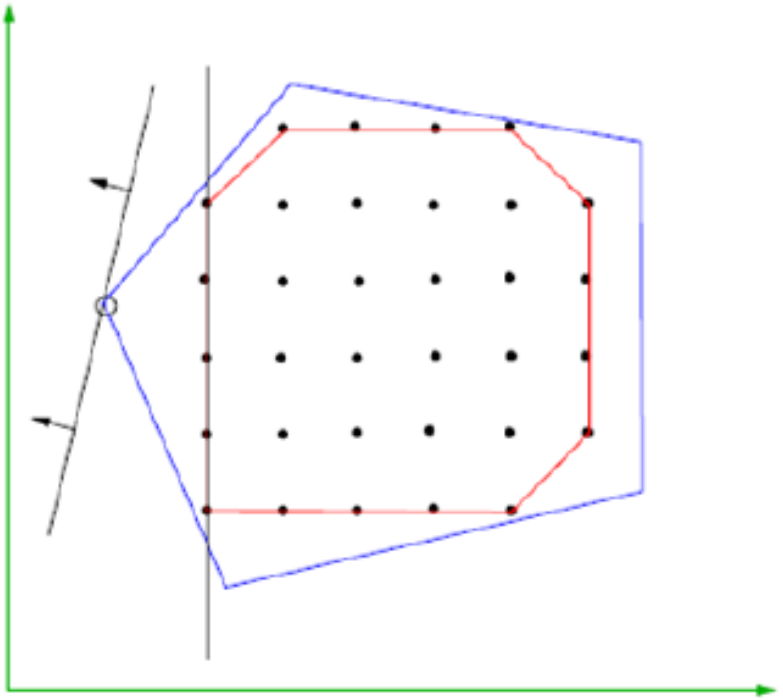
In many cases, combinatorial optimisation problem is easy to state, but hard to model.

Travelling salesman problem –

- Can be solved by combination of Hungarian and branch-and –bound algorithm
- Heuristics are important to get good UB: classic Christofides Algorithm
- Local search: k-opt of Lin and Kernighan, LKH
- Metaheuristic, Matheuristic, nature inspired algorithms, AI/machine learning,
- Strong relaxation for LB: Held-Karp LP
- Cutting Planes

TSP inspires many research directions for Combinatorial Optimasation

Cutting Plane Algorithm



Separation algorithm

Separation as hard as optimisation:

- Given a (an arbitrary) $x \in P \backslash P_I$, find an inequality that separates x from P_I

General cuts

Special cuts

Cutting Plane Algorithm

Consider $min(or\ \max)\ z = f(x)$
s.t. $Ax = b$,
$x \geq 0$ and integer

Step 1. Solve LP relaxation with Simplex. In the optimal tableau select row with non-integer RHS:

Step 2. Rearrange as follows – terms with integer coefficients on LHS and fractional coefficient – on RHS:

And introduce the "cut":

Observe that the "cut" will make the optimal non-integer bfs infeasible:

Note: we can select a row from the original formulation or the linear combination and apply the same rounding technique

Step 3 Add the cut to the constraints***,*** introduce new slack variable x_k and use Dual Simplex method to solve the subproblem. Choose non –integer RHS, and make another cut.

Cutting Plane Algorithm - example

$max\ z = 8x_1 + 5x_2$

s.t.

$$x_1 + x_2 \leq 6$$
$$9x_1 + 5x_2 \leq 45$$
$$x_1, x_2 \geq 0 \text{ and integer}$$

- Standard form:

basis	x_1	x_2	s_1	s_2	rhs
z	0	0	1.25	0.75	41.25
x_2	0	1	2.25	-0.25	2.25
x_1	1	0	-1.25	0.25	3.75

Cutting Plane Algorithm - example

basis	x_1	x_2	s_1	s_2	rhs
z	0	0	1.25	0.75	41.25
x_2	0	1	2.25	-0.25	2.25
x_1	1	0	-1.25	0.25	3.75

Cutting Plane Algorithm - example

$max\ z = 7x_1 + 9x_2$

$$
\begin{aligned}
s.t. \quad & -x_1 + 3x_2 \leq 6 \\
& 7x_1 + x_2 \leq 35 \\
& x_1, x_2 \geq 0 \text{ and integer}
\end{aligned}
$$

- Optimal tableau:

basis	x_1	x_2	s_1	s_2	rhs
z	0	0	$\frac{28}{11}$	$\frac{15}{11}$	63
x_2	0	1	$\frac{7}{22}$	$\frac{1}{22}$	$\frac{7}{2}$
x_1	1	0	$-\frac{1}{22}$	$\frac{3}{22}$	$\frac{9}{2}$

Cutting Plane Algorithm - example

basis	x_1	x_2	s_1	s_2	rhs
z	0	0	$\frac{28}{11}$	$\frac{15}{11}$	63
x_2	0	1	$\frac{7}{22}$	$\frac{1}{22}$	$\frac{7}{2}$
x_1	1	0	$-\frac{1}{22}$	$\frac{3}{22}$	$\frac{9}{2}$

Cutting Plane Algorithm - example

$$
\begin{aligned}
min\ z = & -18x_1 - 12x_2 \\
s.t. \quad & 2x_1 - x_2 \leq 5 \\
& 2x_1 + 3x_2 \leq 13 \\
& x_1, x_2 \geq 0 \text{ and integer}
\end{aligned}
$$