Decision Making and Optimization

Master in Data Analytics for Business

Lisbon School of Economics & Management
Universidade de Lisboa

2024-2025

Integer Linear Programming

Integer Linear Programming Example

TBA Airlines is a small air company, specialized in regional flights. The management is considering an expansion and it has the possibility to buy small or medium size airplanes. Find the best strategy, knowing that at the moment no more than two small airplanes can be bought and that $100 millions are available to invest. Consider also the values in the following table:

	Small airplane	Medium size airplane
Annual profit per airplane	$1 million	$5 millions
Cost per airplane	$5 millions	$50 millions

Model of the example

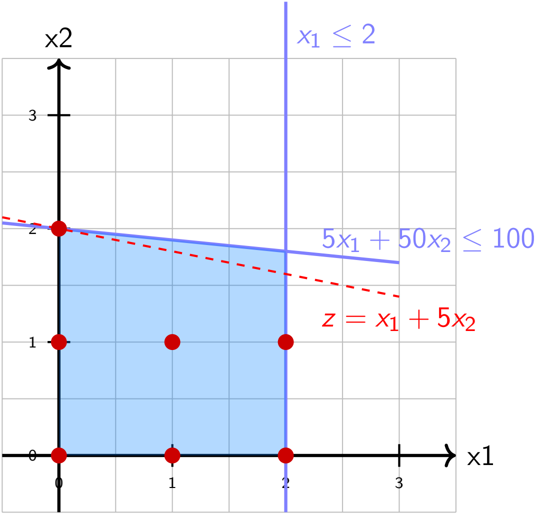
Consider the variables
x_1 - number of small airplanes to buy
x_2 - number of medium size airplanes to buy

The ILP model is

$$\begin{aligned} \max z &= x_1 + 5x_2 \\ \text{s.t. } & 5x_1 + 50x_2 \leq 100 \\ & x_1 \leq 2 \\ & x_1, x_2 \geq 0 \text{ and integer} \end{aligned}$$

Graphical solution

Optimal solution (0,2) with value 10



Feasible set $= \{(0,0),(0,1),(0,2),(1,0),(2,0),(1,1),(2,1)\}$

Integer Linear Programming (ILP)

(ILP) arises when in the context of Linear Programming the decision variables only make sense if they have integer values, that is if the assumption of divisibility of LP doesn't fit to the problem in hands.

Integer Linear Programming problems where variables only take

– integer values $x_j \in \mathbb{N},\ j = 1, \ldots, n$
or
– binary values $x_j \in \{0, 1\},\ j = 1, \ldots, n$

Integer Linear Programming (ILP)

Integer Linear Programming (ILP) model

$$
\begin{aligned}
\max \quad & z = \sum_{j=1}^{n} c_j x_j \\
\text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \; i = 1, \ldots, m \\
& x_j \geq 0 \text{ and integer}, \; j = 1, \ldots, n \\
& \text{or} \\
& x_j \in \{0, 1\}, \; j = 1, \ldots, n
\end{aligned}
$$

Integer Linear Programming

(ILP) arises when to decide on:

- indivisible number decisions (for, example decisions on the number of machines to purchase, the number of people to select for a job)
- “yes-or-no” decisions (for example, the type of projects in which to invest or not to invest)

A ILP is a pure ILP if all the decision variables are required to have integer values;

A mixed ILP is a ILP where only some variables are required to have integer values.

Lower and Upper Bounds

Lower and Upper Bounds

Given an integer linear program (IP)

$$\begin{aligned} \max \quad & z = \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \;\; i = 1, \ldots, m \\ & x_j \geq 0 \text{ and integer}, \;\; j = 1, \ldots, n \end{aligned}$$

we can obtain
lower bounds $\underline{z}_\ell$ and upper bounds $\overline{z}_u$ for its optimal value z^*:

$$\underline{z}_\ell \leq z^* \leq \overline{z}_u.$$

How to obtain lower and upper bounds?

Feasible solutions

the **value of any feasible solution z_a** is a

lower bound for a maximization integer linear problem

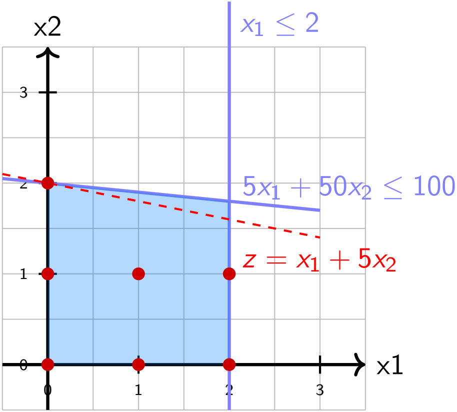
$$z_a \leq z^*$$

upper bound for a minimization integer linear problem

$$z^* \leq z_a$$

Example

Optimal solution (0,2)
with value $z^* = 10$



Feasible set $= \{(0,0),(0,1),(0,2),(1,0),(2,0),(1,1),(2,1)\}$

Lower bounds: $z_{(0,0)} = 0,\quad z_{(0,1)} = 5,\quad z_{(0,2)} = 10,\quad z_{(1,0)} = 1,\quad z_{(2,0)} = 2,\quad z_{(1,1)} = 6,\quad z_{(2,1)} = 7$

Relaxation

We obtain a relaxation when we replace the ILP problem with another one for which it is easier to obtain a solution so that we can obtain an approximation to the optimal value of the ILP problem.

Given an ILP we obtain a relaxation when:

(i) the set of feasible solutions is enlarged (obtaining a larger set) and/or

(ii) the objective function is replaced by another one

There are several relaxations that can be obtained, we will use a linear relaxation.

Properties of the Relaxation

If the relaxed problem has no feasible solutions (is impossible), then the ILP has no feasible solutions (is impossible).

If the optimal solution of the relaxed problem is feasible for the ILP problem, then the solution is optimal to the ILP problem.

Relaxation

the **optimal value z_r of a relaxation** is

upper bound for a maximization integer linear problem

$$z^* \leq z_r$$

lower bound for a minimization integer linear problem

$$z_r \leq z^*$$

Linear Relaxation

The linear relaxation of an ILP is the LP obtained by dropping the constraints on the integer values for the variables (the integrality constraints).

we obtain the linear relaxation as follows

$$
\begin{aligned}
\max \quad & z = \sum_{j=1}^{n} c_j x_j \\
\text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \;\; i = 1, \ldots, m \\
& x_j \geq 0 \; \cancel{\text{and integer}}, \;\; j = 1, \ldots, n
\end{aligned}
$$

Example

The linear relaxation of the ILP model is

$$\begin{aligned} \max z &= x_1 + 5x_2 \\ \text{s.t. } & 5x_1 + 50x_2 \leq 100 \\ & x_1 \leq 2 \\ & x_1, x_2 \geq 0 \ \sout{\text{and integer}} \end{aligned}$$

The optimal solution is $x^*_{LR} = (2, \frac{9}{5})$ with value $z^*_{LR} = 11$

Thus $z^*_{LR} = 11$ is an upper bound

Duality

the **value of a dual feasible solution** z_d of the linear relaxation is

upper bound for a maximization integer linear problem

$$z^* \leq z_d$$

lower bound for a minimization integer linear problem

$$z_d \leq z^*$$

Example

The dual of the linear relaxation of the ILP model is

$$\begin{aligned} \min w = 100y_1 + 2y_2 \\ \text{s.t. } 5y_1 + y_2 \geq 1 \\ 50y_1 \geq 5 \\ y_1, y_2 \geq 0 \end{aligned}$$

A feasible solution is $y = (1, 1)$ with value $w = 102$

Thus $w = 102$ is an upper bound

another dual feasible solution is $y = (0.1, 1)$ with value $w = 12$

Solving ILP

Solve ILP problems

Methods to (exactly) solve ILP problems

- Branch & Bound
- Cutting Planes
- Branch & Cut
- Branch & Price
- etc.

Approximation methods (give lower and/or upper bounds)

- any exact method that is stopped before completing its search
- Relaxations
- Heuristics
- etc.

Branch & Bound

- divides the feasible set and builds an enumeration tree (**branch**)
- removes branchs of the tree that corresponds to infeasible situations (**bound**)
- starts by solving the linear relaxation of the initial program
- only solves linear relaxations of the linear problems that are generated by adding constraints to the initial problem

Branch & Bound algorithm - max problem

Step 1: Initialisation

- solve the linear relaxation (P_L) of P.L.I.
  If (P_L) is impossible, STOP! (P) is also impossible.
  If its solution is integer, STOP! The optimal solution of (P) has been found.
  Otherwise
- let $\bar{z}$ be the corresponding optimal value
- let $\underline{z} = -\infty$ or else equal to the value of the o.f. associated with a feasible solution

Step 2: Branching

- partition the problem from a variable that violates the integrality constraint
- let x_k be the chosen variable with fractional value $\bar{x}_k$
- in one of the problems include the constraints $x_k \leq \lfloor \bar{x}_k \rfloor$
- in the other problem include the constraint $x_k \geq \lfloor \bar{x}_k \rfloor + 1$
- place these problems on the list of problems to be analysed

Step 3: Subproblem selection

- if there are no more subproblems to analyse, go to Step 5
- otherwise, select a new problem from the list and go to Step 4

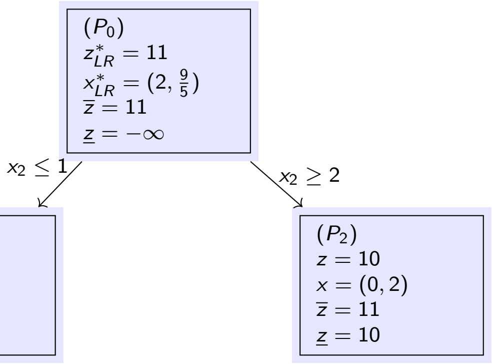
Step 4: Solve the selected subproblem/Bounding

- solve the linear relaxation of the selected problem
- if the linear relation is impossible, abandon the subproblem, cancel the node in the search tree and go to Step 3
- otherwise, let z be the value of the o.f. corresponding to the optimal solution of the linear relaxation
  - if $z < \underline{z}$, abandon this problem, cancel this node, and go to Step 3
  - if $z \geq \underline{z}$ and if in the optimal solution there is at least one integer variable with a fractional value, then go to Step 2
  - if $z \geq \underline{z}$ and if the optimal solution satisfies all the integrality constraints, then cancel the tree node, replace $\underline{z}$ for z and move on to Step 3

Step 5: Optimality test

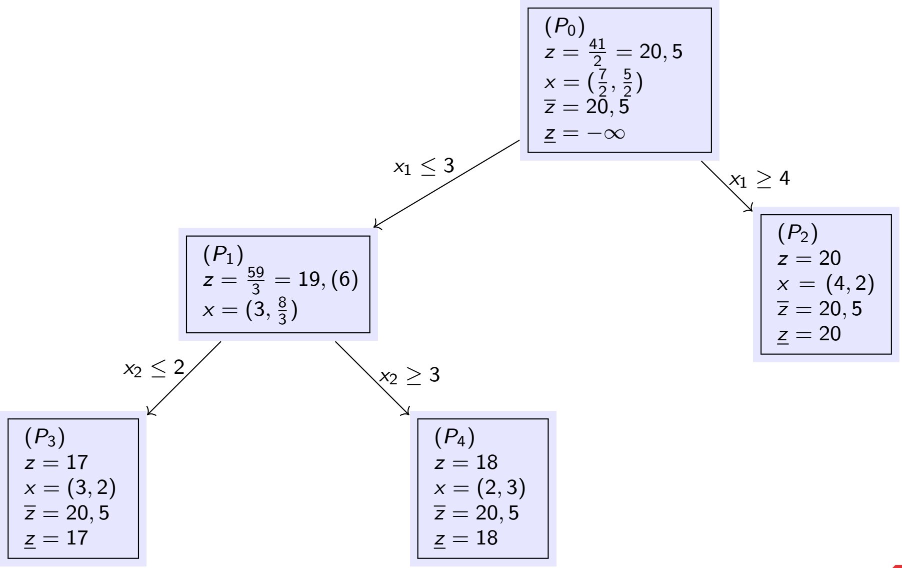
- if $\underline{z} = -\infty$ then the problem is impossible and the process ends
- otherwise the optimal solution has been obtained and the process ends with $z^* = \underline{z}$

Branch & Bound for the Example

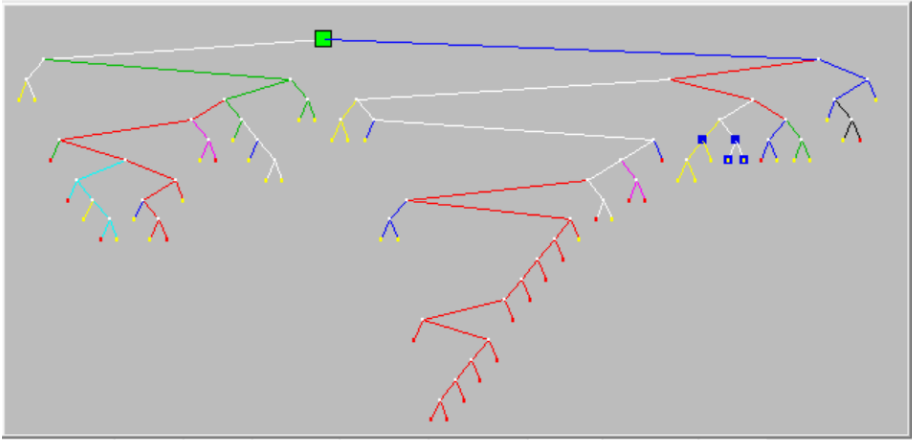


Another Example

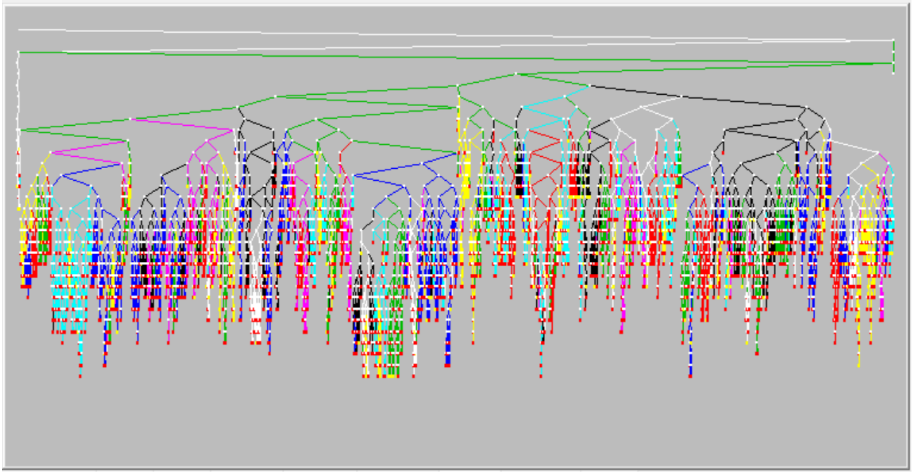
$$
\begin{aligned}
\max \quad & z = 3x_1 + 4x_2 \\
\text{s. a:} \quad & -3x_1 + 2x_2 \leq 2 \\
& x_1 + 3x_2 \leq 11 \\
& x_1 + x_2 \leq 6 \\
& x_1, x_2 \geq 0 \text{ and integer}
\end{aligned}
$$


(P_0)
$z = \frac{41}{2} = 20,5$
$x = (\frac{7}{2}, \frac{5}{2})$
$\overline{z} = 20,5$
$\underline{z} = -\infty$
$x_1 \leq 3$
$x_1 \geq 4$
(P_1)
$z = \frac{59}{3} = 19,(6)$
$x = (3, \frac{8}{3})$
(P_2)
$z = 20$
$x = (4,2)$
$\overline{z} = 20,5$
$\underline{z} = 20$
$x_2 \leq 2$
$x_2 \geq 3$
(P_3)
$z = 17$
$x = (3,2)$
$\overline{z} = 20,5$
$\underline{z} = 17$
(P_4)
$z = 18$
$x = (2,3)$
$\overline{z} = 20,5$
$\underline{z} = 18$

Example of a Branch & Bound Tree (Xpress)



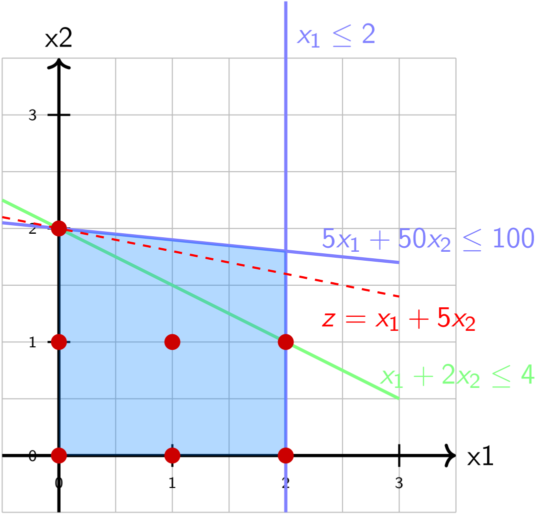
Example of a Branch & Bound Tree (Xpress)



Cutting Planes

Adds valid inequalities, such as $x_1 + 2x_2 \leq 4$ for our example

Optimal solution (0,2)
with value 10



Software

Small size

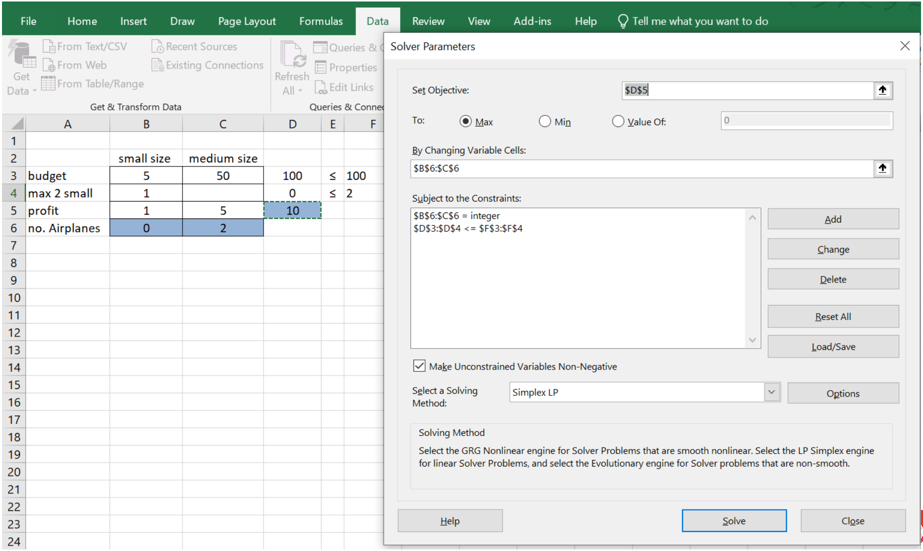
Solver of the Excel
PuLP with Python

Large size

XPress https://www.fico.com/en/products/fico-xpress-optimization
Gurobi https://www.gurobi.com/
CPLEX https://www.ibm.com/analytics/cplex-optimizer

among others
and you may use them with Python, C, C++

Solving using the Solver of the Excel



Heuristics

A heuristic, or a heuristic technique, is any approach to problem-solving that uses a practical method or various shortcuts in order to produce solutions that may not be optimal but are good enough, sufficient, given a limited timeframe or deadline.

Heuristics usually produce feasible solutions.

Later we'll learn some heuristics and metaheuristics.

Modelling with binary variables

Integer/Binary Linear Program

– decision variables $x_j \geq 0,\ j = 1, \ldots, n$
– binary variables $y_j \in \{0, 1\},\ j = 1, \ldots, nn$
Mixed Integer Linear Programming (ILP) model

$$\begin{aligned} \max \quad & z = \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i,\ i = 1, \ldots, m \\ & \text{additional constraints using } x_j \text{ and } y_j \\ & x_j \geq 0,\ j = 1, \ldots, n \\ & y_j \in \{0, 1\},\ j = 1, \ldots, nn \end{aligned}$$

Binary decision variables:

$$y_j = \begin{cases} 1, & \text{if } j \text{ is selected}, \\ 0, & \text{otherwise}, \end{cases} \qquad j = 1, \ldots, n,$$

Modelling Fixed Costs

f_j is the fixed cost for considering (selecting/using/producing) activity x_j

$$\begin{aligned} \max \quad & z = \sum_{j=1}^{n} c_j x_j - \sum_{j=1}^{n} f_j y_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \; i = 1, \ldots, m \\ & x_j \leq M \; y_j, \; j = 1, \ldots, n \text{ (linking constraints)} \\ & x_j \geq 0, \; j = 1, \ldots, n \\ & y_j \in \{0, 1\}, \; j = 1, \ldots, n \end{aligned}$$

M should be defined so that the value of the variable x_j is bounded only by the functional constraints and not by the linking constraints

Limitations on activities

$$y_j = \begin{cases} 1, & \text{if } j \text{ is selected,} \\ 0, & \text{otherwise,} \end{cases} \qquad j = 1, \ldots, n,$$

- on the maximum number of activities: at most K

$$\begin{array}{ll} \sum_{j=1}^{n} y_j \leq K, & \\ x_j \leq M \; y_j, & j = 1, \ldots, n, \end{array}$$

- incompatible activities: r and s are incompatible

$$\begin{array}{l} y_r + y_s \leq 1, \\ x_r \leq M \; y_r, \\ x_s \leq M \; y_s, \end{array}$$

- complementary activities: s is selected only if r is selected

$$y_s \leq y_r$$

Disjunction of constraints

$$\sum_{j=1}^{n} a_{1j} x_j \leq b_1 + M\,(1 - y_1),$$

$$\sum_{j=1}^{n} a_{2j} x_j \leq b_2 + M\,(1 - y_2),$$

$$y_1 + y_2 = 1$$

$$y_1, y_2 \in \{0, 1\}$$

Example 1

A company wishes to expand its business by acquiring 3 new properties and is considering 5 possible locations for this purchase, L1, L2, L3, L4 and L5. Each site should be no less than 50 ha and no more than 300 ha. Due to the proximity of locations L1, L2, L3, no more than 2 of these 3 locations should be selected, and locations L4 and L5 are incompatible.
The company has the following information about the locations.

	L1	L2	L3	L4	L5
annual expected profit / ha	25	30	20	20	25
annual fixed costs	100	120	110	110	120
initial investment / ha	30	45	20	25	30

The company has a budget of 18 500 u.m. for an initial investment, and aims to maximise its expected (annual) profit. Model this problem.

Example 1 (more constraints)

Now consider that if the company chooses locations L4 or L5, it will have an additional 500 u.m. or 300 u.m. respectively to the initial amount.

Example 2

An oil company intends to select 5 out of 10 wells: $P1, P2, \ldots, P10$, to which are associated the cost $c_1, c_2, \ldots, c_{10}$, respectively. According to commitments with the local government, the company must comply with the following restrictions for regional development:

- **r1)** the selection of both P1 and P7 block selection of P8;
- **r2)** the selection of P3 or P4 block selection of P5;
- **r3)** from P5, P6, P7 and P8 at most two can be selected;
- **r4)** the selection of P1 forces selection of P10.

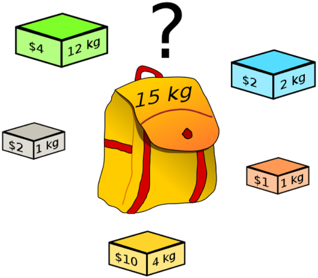
Formulate the problem and solve it assigning costs at your choice.

Application Examples

- Knapsack Problem
- Set Covering Problem
- Sequencing Problem
- Facility Location Problem
- Travelling Salesperson Problem

Knapasack Problem

Example of a Knapasack Problem



utility =(4,2,2,1,10)
weight=(12,2,1,1,4)
W=15

$$\max \quad 4x_1 + 2x_2 + 2x_3 + x_4 + 10x_5$$

$$\text{s. to:} \quad 12x_1 + 2x_2 + x_3 + x_4 + 4x_5 \leq 15$$
$$x_j \in \{0,1\}, j = 1, \ldots, n$$

$x = (1,1,1,0,0), z = 8, v = 15$
$x = (0,1,1,1,1), z = 15, v = 8$

Knapasack Problem

in Knapsack type problems, given n objects, each with an associated cost or utility u_j, $j = 1, \ldots, n$, and weight or volume v_j, $j = 1, \ldots, n$, the decision to be made is whether or not to select the object j in order to optimize the total utility and not to violate the the imposed volume constraint of V.

one must decide if $x_j = 1$, which means that the object j is selected, or if $x_j = 0$, which means that the object j is not selected

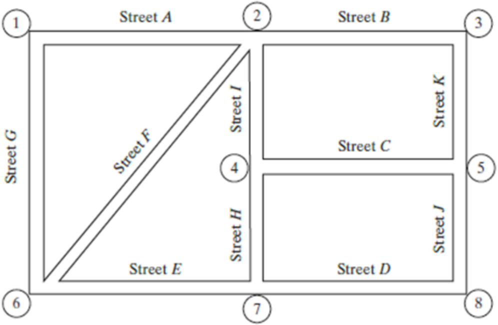
the ILP model of the **Binary Knapsack** is

$$
\begin{aligned}
\max \quad & \sum_{j=1}^{n} u_j x_j \\
\text{s. to:} \quad & \sum_{j=1}^{n} v_j x_j \leq V \\
& x_j \in \{0, 1\}, j = 1, \ldots, n
\end{aligned}
$$

Set Covering Problem

Example of a Set Covering Problem

To promote safety on campus, the Security Department is in the process of installing emergency equipment in selected locations. The department would like to install a minimum number of these devices to serve each of the main campus streets. The figure below shows the main campus roads.



Set Covering Problem

one must decide if $x_j = \begin{cases} 1, & \text{if } j \text{ is selected,} \\ 0, & j \text{ is not selected,} \end{cases} \quad j = 1, \ldots, n,$

the ILP model of the **Set Covering Problem** is

$$
\begin{aligned}
\min \quad & \sum_{j=1}^{n} c_j x_j \\
\text{s. to:} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq 1, i = 1, \ldots, m \\
& x_j \in \{0, 1\}, j = 1, \ldots, n
\end{aligned}
$$

with $a_{ij} = \begin{cases} 1, & \text{if } j \text{ serves } i, \\ 0, & \text{otherwise,} \end{cases} \quad i = 1, \ldots, m, \ j = 1, \ldots, n,$

Sequencing Problem

Example of a Job Sequencing Problem

Jobco uses a single machine to process three jobs. For each job, both the processing time and the due date (in days) are given in the following table. The due dates are measured from zero, the assumed start time of the first job.

Job	Processing time (day)	Due date (day)	Late penalty ($/day)
1	5	25	19
2	20	22	12
3	15	35	34

The objective of the problem is to determine the job sequence that minimizes the late penalty for processing all three jobs.

Sequencing Problem

if $x_{ij} = \begin{cases} 1, & \text{if } i \text{ precedes } j, \\ 0, & \text{otherwise,} \end{cases} \quad i, j = 1, \ldots, n,$

if t_j = start time of job j, $j = 1, \ldots, n$ (measured from time 0)

let δ_j be the processing time of job j, $j = 1, \ldots, n$

the ILP model of the **Sequencing Problem** has the following constraints

$$\begin{aligned} t_j &\geq t_i + \delta_i - M\,(1 - x_{ij}), \\ t_i &\geq t_j + \delta_j - M\,x_{ij} \end{aligned}$$

that model the sequence disjunction:

$$t_j \geq t_i + \delta_i \qquad \text{or} \qquad t_i \geq t_j + \delta_j$$

either job j is after job i or job i is after job j

Sequencing Problem

let d_j be the due date of job j, $j = 1, \ldots, n$

the job j is late if $\quad t_j + \delta_j > d_j$

thus the following constraints

$$\begin{aligned} & t_j + \delta_j - (s_j^+ - s_j^-) = d_j, \\ & s_j^+, s_j^- \geq 0 \end{aligned}$$

define that

job j is ahead of schedule if $s_j^- > 0$

job j is behind schedule if $s_j^+ > 0$

Facility Location Problem

Example of a Facility Location Problem

Jobco wants to build a new warehouse to serve its factories located in three cities: City A, City B and City C.
Transportation costs between the potential warehouse location (in one of the three cities) and each city are shown in the following table

City	A	B	C
A	-	25	19
B	25	-	12
C	19	12	-

The goal is to minimize the total transportation cost from the warehouse to these cities.

Facility Location Problem

This problem involves determining the best location for a facility (like a warehouse, factory, or service center) to serve all the demand and minimize costs

$$y_j = \begin{cases} 1, & \text{if } j \text{ is selected,} \\ 0, & \text{otherwise,} \end{cases} \quad j = 1, \ldots, n,$$

$$x_{ij} = \begin{cases} 1, & \text{if } j \text{ is served by facility in } i, \\ 0, & \text{otherwise,} \end{cases} \quad i, j = 1, \ldots, n,$$

the ILP model of the **Facility Location Problem** is

$$\begin{array}{ll} \min & \sum_{i,j=1}^{n} c_{ij} x_{ij} \\ \text{s. to:} & \sum_{i=1}^{n} x_{ij} + y_j = 1, \qquad j = 1, \ldots, n \\ & x_{ij} \leq y_j, \qquad i, j = 1, \ldots, n \\ & x_{ij} \geq 0, i, j = 1, \ldots, n \\ & y_j \in \{0, 1\}, j = 1, \ldots, n \end{array}$$

Traveling Salesperson Problem (TSP)

Example of a Travelling Salesperson Problem

A courier service needs to deliver packages to several locations in a city. The goal is to find the shortest possible route that allows the courier to visit each location exactly once and return to the starting point.

Location	A	B	C
A	5	25	19
B	20	22	12
C	15	35	34

Travelling Salesperson Problem (TSP)

TSP deals with finding the shortest (closed) tour in an n-city situation, where each city is visited exactly once before returning back to the starting point.

The associated TSP model is defined by two pieces of data:

1. the number n of cities,
2. the distances d_{ij} between cities i and j ($d_{ij} = \infty$ if cities i and j are not linked).

The maximum number of tours in an n-city situation is $(n-1)!$ if the network is directed ($d_{ij} \neq d_{ji}$) and half that much if it is not.
Note that $10! = 3\ 638\ 800$

Example of a TSP

The daily production schedule at the Rainbow Company includes batches of white (W), yellow (Y), red (R), and black (B) paints. The production facility must be cleaned between successive batches.

Inter-batch Cleanup Times (in minutes)				
Paint	White	Yellow	Black	Red
White	∞	10	17	15
Yellow	20	∞	19	18
Black	50	44	∞	22
Red	45	40	20	∞

The objective is to determine the sequencing of colors that minimizes the total cleanup time.

Example of a TSP

Solution of the Paint Sequencing Problem by Exhaustive Enumeration

No. of feasible solutions (production loops): $(n-1)! = 3! = 6$

Production loop	Total cleanup time (min)
W → Y → B → R → W	$10 + 19 + 22 + 45 = 96$
W → Y → R → B → W	$10 + 18 + 20 + 50 = 98$
W → B → Y → R → W	$17 + 44 + 18 + 45 = 124$
W → B → R → Y → W	$17 + 22 + 40 + 20 = 99$
W → R → B → Y → W	$15 + 20 + 44 + 20 = 99$
W → R→ Y→ B→ W	$15 + 40 + 19 + 50 = 124$

ILP formulation for the TSP

$$x_{ij} = \begin{cases} 1 & \text{if paint j follows paint i} \\ 0 & \text{otherwise} \end{cases} \qquad i, j = W, Y, B, R; \quad i \neq j$$

$$\min \sum_{i,j=W,Y,B,R;i\neq j} c_{ij} x_{ij}$$

$$\sum_{i=W,Y,B,R} x_{ij} = 1, \qquad j = W, Y, B, R,$$

$$\sum_{i=W,Y,B,R} x_{ji} = 1, \qquad j = W, Y, B, R,$$

$$x_{ij} \in \{0, 1\}, \qquad i, j = W, Y, B, R, i \neq j$$

????

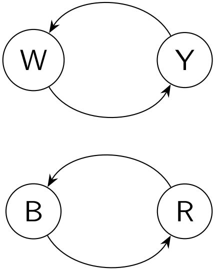
Example of a TSP

The solution

$$x_{WY} = x_{YW} = x_{BR} = x_{RB} = 1$$

satisfies all the previous constraints

$$\begin{aligned} x_{WY} + x_{WB} + x_{WR} &= 1 \\ x_{YW} + x_{YB} + x_{YR} &= 1 \\ x_{BY} + x_{BR} + x_{RW} &= 1 \\ x_{RW} + x_{RY} + x_{RB} &= 1 \\ x_{YW} + x_{BW} + x_{RW} &= 1 \\ x_{WY} + x_{BY} + x_{RY} &= 1 \\ x_{WB} + x_{YB} + x_{RB} &= 1 \\ x_{WR} + x_{YR} + x_{BR} &= 1 \\ x_{ij} \in \{0,1\} \quad \forall i,j \quad & i \neq j \end{aligned}$$



and $\min 10x_{WY} + 17x_{WB} + 15x_{WR} + 20x_{YW} + 19x_{YB} + 18x_{YR} + 50x_{BW} + 44x_{BY} + 22x_{BR} + 45x_{RW} + 40x_{RY} + 20x_{RB}$

ILP formulation for the TSP

Subtour elimination constraints are missing

for example

$$x_{WB} + x_{WR} + x_{YB} + x_{YR} + x_{BW} + x_{BY} + x_{RW} + x_{RY} \geq 1$$

how to establish such constraints?

Dantzig, Fulkerson, Johnson, 1954:

For every set S of cities, add a constraint saying that the tour leaves S at least once. For every $S \subseteq \{1, 2, \ldots, n\}$ with $1 \leq |S| \leq n - 1$:

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1$$

This will happen for any tour: eventually, we must go from a city in S to a city not in S. In a solution to the local constraints with subtours, this is violated if we take S to be the set of cities in a subtour

- The formulation with the subtour elimination constraints describe TSP.
- Number of constraints increase exponentially: for n cities, there are $2^n - 2$ subtour elimination constraints! $2^{n-1} - 1$ if we assume $1 \in S$.

Dantzig, Fulkerson, Johnson, 1954:

The complete model is:

$$\min \sum_{i,j\in\{1,2,\ldots,n\};i\neq j} c_{ij}x_{ij}$$

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

$$\sum_{i\in\{1,2,\ldots,n\}} x_{ji} = 1, \qquad j \in \{1,2,\ldots,n\},$$

$$\sum_{i\in S}\sum_{j\notin S} x_{ij} \geq 1 \qquad S \subseteq \{1,2,\ldots,n\},\ 1 \leq |S| \leq n-1$$

$$x_{ij} \in \{0,1\}, \qquad i,j \in \{1,2,\ldots,n\}, i \neq j$$

Miller, Tucker, Zemlin, 1960:

Add variables representing the time at which a city is visited.
For $i = 1, \ldots, n,$, let t_i denote the time at which we visit city i, with $1 \leq t_i \leq n-1$. We leave t_1 undefined.
We want an inequality to encode the logical implication

if $x_{ij} = 1$, then $t_j \geq t_i + 1$ for every pair of cities $i, j \neq 1$.

How do we know that the timing constraints get rid of subtours?

1. For any tour, we can satisfy the timing constraints. If we visit cities $i_1, i_2, \ldots, i_{(n-1)}$, in that order from city 1, set $i_1 = 1, \;\; i_2 = 2, \;\; \ldots, \;\; i_{(n-1)} = n-1.$
2. If there is a subtour, then we can't satisfy the timing constraints.
3. Suppose $x_{ab} = x_{bc} = x_{ca} = 1$ and none of a, b, c are 1. Then we can't satisfy the three constraints $t_b \geq t_a + 1, \quad t_c \geq t_b + 1 \quad t_a \geq t_c + 1$

Miller, Tucker, Zemlin, 1960:

If $x_{ij} = 1$, then $t_j \geq t_i + 1$.
Using the big number M:

$$t_j \geq t_i + 1 - M(1 - x_{ij}) \text{ for some large } M.$$

When $x_{ij} = 1$, this simplifies to $t_j \geq t_i + 1$.
When $x_{ij} = 0$, we get $t_j \geq t_i + 1 - M$, which has no effect on the value of t_i, t_j.

We can check: if we take $M = n$, then any actual tour can satisfy these constraints. The times $t_2, \ldots, t_n$ can be chosen between 1 and $n - 1$, so $t_j \geq t_i + 1 - n$ always holds.

The inequality is

$$t_j \geq t_i + 1 - n(1 - x_{ij})$$

Miller, Tucker, Zemlin, 1960:

The complete model is:

$$\min \sum_{i,j \in \{1,2,\ldots,n\}; i \neq j} c_{ij} x_{ij}$$

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

$$\sum_{i \in \{1,2,\ldots,n\}} x_{ji} = 1, \qquad j \in \{1,2,\ldots,n\},$$

$$t_j \geq t_i + 1 - n(1 - x_{ij}) \qquad i,j \in \{1,2,\ldots,n\},\ i \neq j$$

$$x_{ij} \in \{0,1\}, \qquad i,j \in \{1,2,\ldots,n\}, i \neq j$$

DFJ versus MTZ

On the one hand:

- DFJ's formulation has $2^{(n-1)} - 1$ extra constraints, plus the $2n$ local constraints.
- MTZ's formulation has only n^2 extra constraints. There are $n - 1$ extra variables, which can be integer variables, but don't need to be.

On the other hand:

- DFJ's formulation has an efficient branch-and-cut approach.
- MTZ's formulation is weaker: the feasible region has the same integer points, but includes more fractional points.

Relaxations for the TSP

The Assigment relaxation:

$$\min \sum_{i,j\in\{1,2,\ldots,n\};i\neq j} c_{ij}x_{ij}$$

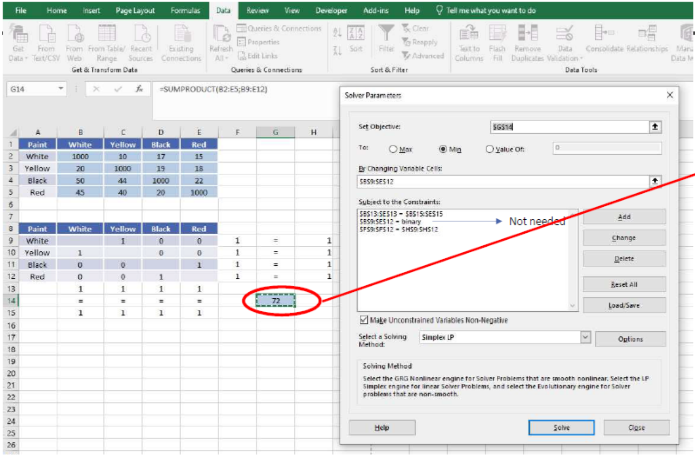
$$\sum_{i\in\{1,2,\ldots,n\}} x_{ij} = 1, \qquad j \in \{1,2,\ldots,n\},$$

$$\sum_{i\in\{1,2,\ldots,n\}} x_{ji} = 1, \qquad j \in \{1,2,\ldots,n\},$$

$$x_{ij} \in \{0,1\}, \qquad i,j \in \{1,2,\ldots,n\}, i \neq j$$

Relaxations for the TSP: the paints example

Solving the Assigment Relaxation using the Solver of the Excel



Paint	White	Yellow	Black	Red
White	1000	10	17	15
Yellow	20	1000	19	18
Black	50	44	1000	22
Red	45	40	20	1000

Paint	White	Yellow	Black	Red			
White		1	0	0	1	=	1
Yellow	1		0	0	1	=	1
Black	0	0		1	1	=	1
Red	0	0	1		1	=	1
	1	1	1	1			
	=	=	=	=		72	
	1	1	1	1			

Assignment problem

Lower bound TSP **72**

Optimal solution by complete enumeratior

W -> Y-> B-> R-> W

10 + 19 + 22 + 45 = **96**

Constructive heuristics for the TSP: nearest neighbor

Input: $G = (V, A), V = \{1, 2, \ldots, n\}, |V| = n, \text{cost} c_{ij} \to (i, j) \in A$

Initialization

Arbitrarily choose a city $i \in V$

$L = \{1, 2, \ldots, n\} - \{i\}$ (L set of cities not yet visited)

Iteration

REPEAT

Select in L city j closest to i

Insert the city j immediately after i in the route

Update $i = j$

$L := L - \{i\}$

UNTIL $L = \emptyset$ OR no city can be selected

If possible complete the cycle by going back to the beginning

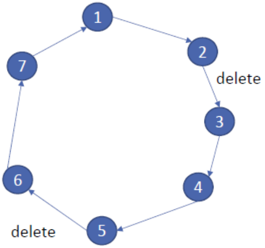
and calculate the total distance

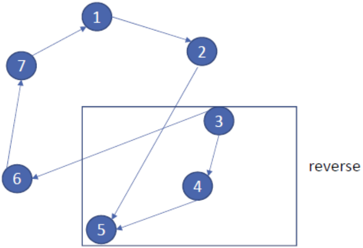
STOP

Improvement Heuristics for the TSP

- Starting from a feasible circuit, try edge swapping that can lead to new lower-cost circuits.
- The algorithm consists of starting with a feasible circuit and swapping r edges until it is no longer possible to improve the solution.

Swapping 2 edges: delete, reverse, connect





Improvement Heuristics for the TSP: 2-optimal

Consider the case of heuristics that perform 2 edge swaps to improve the solution already obtained.

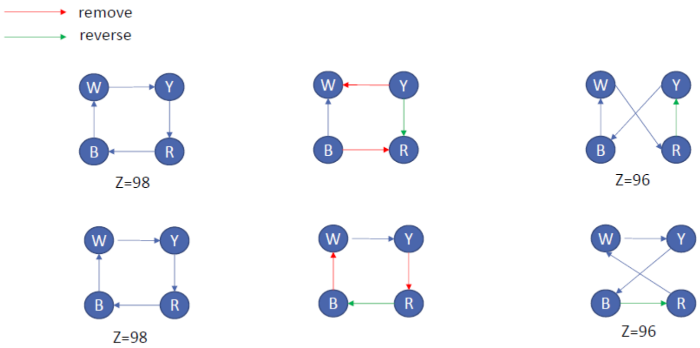
- If you have a circuit and you swap 2 edges that are not consecutive, how many different circuits can you get?

$$\frac{n[(n-1)-2]}{2}$$

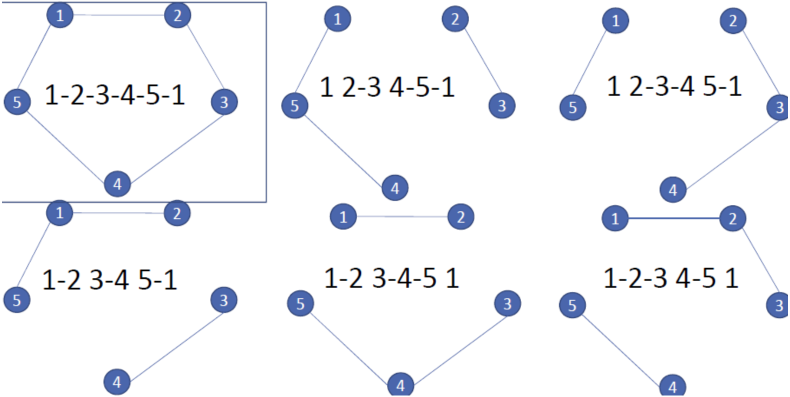
- Let $N_2(T)$ be the neighborhood of the circuit T, i.e. $N_2(T)$ is the set of circuits that differ from circuit T on a maximum of 2 (non-consecutive) edges.

- $|N_2(T)| = \frac{n(n-3)}{2} + 1$

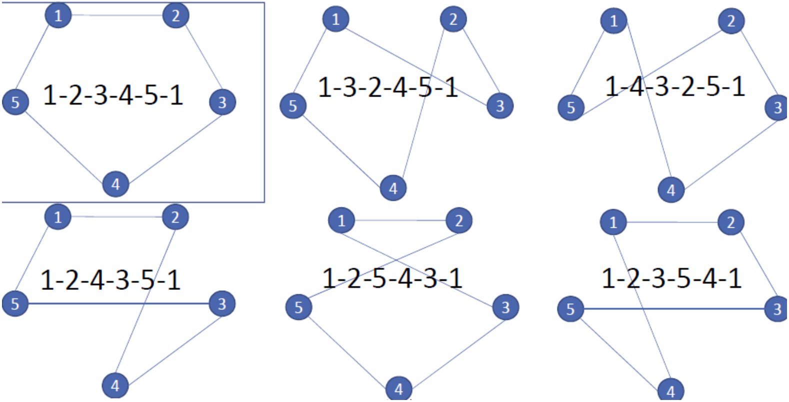
Paints example of swapping 2 edges



2-opt neighborhood - remove



2-opt neighborhood - reverse



Improvement Heuristics for the TSP

1st version

1. Determine a circuit T.
2. Determine $N_r(T)$ (the set of all possible swaps of r edges) and the cost of all its circuits.
3. Determine a circuit $Q \neq T$ such that Q is the circuit with the minimum cost in $N_r(T) \backslash \{T\}$.
4. If the cost Q is less than the cost T, then do $T := Q$ and return to step 2, otherwise $STOP$, it is not possible to improve the current circuit.

Improvement Heuristics for the TSP

2nd version

1. Determine a circuit T.
2. Sequentially examine the elements $Q \neq T$ of $N_r(T)$ and determine its cost.
3. If cost Q is less than cost T, then do $T := Q$. Return to step 2. If there is no more element to search in $N_r(T)$ then $STOP$ (it is not possible to improve the current circuit in the considered neighborhood).

TSP: Exercise

$$[c_{ij}] = \begin{bmatrix} - & 10 & 22 & 12 & 10 \\ & - & 12 & 8 & 13 \\ & & - & 15 & 15 \\ & & & - & 9 \\ & & & & - \end{bmatrix}$$

Knapsack Problem

Knapsack Problem

References:

P. Toth, S. Martello. Knapsack problems: algorithms and computer implementations. Wiley, 1990.
H. Kellerer, U. Pferschy, D. Pisinger. Knapsack Problems. Springer, 2004

Given

C – capacity of the knapsack,

n – number of different objects,

for $j = 1, \ldots, n$

u_j – utility or cost of object j,

v_j – volume or weight of object j

ILP models:

Binary decision variables:

$$x_j = \begin{cases} 1, & \text{if object } j \text{ is selected,} \\ 0, & \text{otherwise,} \end{cases} \quad j = 1, \ldots, n,$$

Binary Knapsack

$$\max \sum_{j=1}^{n} u_j x_j$$

$$s.t. \sum_{j=1}^{n} v_j x_j \leq C$$

$$x_j \in \{0, 1\}, j = 1, \ldots, n$$

Subset-sum

$$\max \sum_{j=1}^{n} v_j x_j$$

$$s.t. \sum_{j=1}^{n} v_j x_j \leq C$$

$$x_j \in \{0, 1\}, j = 1, \ldots, n$$

ILP models:

Integer decision variables:

$x_j \in \mathbb{N}_0$ number of objects type j selected, $j = 1, \ldots, n$,

Limited Knapsack

$$\max \sum_{j=1}^{n} u_j x_j$$

$$s.t. \sum_{j=1}^{n} v_j x_j \leq C,$$

$$x_j \in \{0, 1, \ldots, \ell_j\}, \quad j = 1, \ldots, n$$

Change Machine

$$\max \sum_{j=1}^{n} x_j$$

$$s.t. \sum_{j=1}^{n} v_j x_j = C,$$

$$x_j \in \{0, 1, \ldots, \ell_j\}, \quad j = 1, \ldots, n$$

$\ell_j \in \mathbb{N}_0$

Multiple Knapsack

m – number of different knapsacks,
C_i – capacity of knapsack i, $i = 1, \ldots, m$
Binary decision variables:

$$x_{ij} = \begin{cases} 1, & \text{if object } j \text{ is selected for knapsack } i, \\ 0, & \text{otherwise,} \end{cases}$$

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} u_j x_{ij}$$

$$s.t. \sum_{j=1}^{n} v_j x_{ij} \leq C_i, i = 1, \ldots, m$$

$$\sum_{i=1}^{m} x_{ij} \leq 1, j = 1, \ldots, n$$

$$x_{ij} \in \{0, 1\}, i = 1, \ldots, m, j = 1, \ldots, n$$

Generalized Assignment

u_{ij} – utility obtained from assigning task j to machine i,
v_{ij} – consumption of resource (machine) i by task j,

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} u_{ij} x_{ij}$$

$$s.t. \ \sum_{j=1}^{n} v_{ij} x_{ij} \leq C_i, i = 1, \ldots, m$$

$$\sum_{i=1}^{m} x_{ij} \leq 1, j = 1, \ldots, n$$

$$x_{ij} \in \{0, 1\}, i = 1, \ldots, m, j = 1, \ldots, n$$

different utility and weights depending on the knapsack selected

Binary Knapsack

Binary decision variables:

$$x_j = \begin{cases} 1, & \text{if object } j \text{ is selected}, \\ 0, & \text{otherwise}, \end{cases} \qquad j = 1, \dots, n,$$

Binary Knapsack

$$\max \sum_{j=1}^{n} u_j x_j$$

$$s.t. \sum_{j=1}^{n} v_j x_j \leq C$$

$$x_j \in \{0,1\}, j = 1, \dots, n$$

Example:
$n = 6$, $C = 12$,
$u = (2, 5, 3, 4, 5, 4)$,
$v = (6, 8, 4, 6, 7, 2)$.

$$\max\ z = 2x_1 + 5x_2 + 3x_3 + 4x_4 + 5x_5 + 4x_6$$

$$s.t.\ 6x_1 + 8x_2 + 4x_3 + 6x_4 + 7x_5 + 2x_6 \leq 12$$

$$x_j \in \{0,1\}, j = 1, \dots, 6$$

Binary Knapsack: the Critical index

Assumptions:

$u_j > 0, \ j = 1, \ldots, n,$
$0 < v_j \leq C, \ j = 1, \ldots, n,$
$\sum_{j=1}^{n} v_j > C > 0$

$$\frac{u_1}{v_1} \geq \frac{u_2}{v_2} \geq \cdots \geq \frac{u_k}{v_k} \geq \cdots \geq \frac{u_n}{v_n}$$

Critical index:

k such that:

$$\sum_{j=1}^{k-1} v_j \leq C$$

$$\& \sum_{j=1}^{k} v_j > C$$

Example: $n = 6, \ C = 12, \ \bar{u} = (2, 5, 3, 4, 5, 4), \ \bar{v} = (6, 8, 4, 6, 7, 2).$

1	2	3	4	5	6
$\frac{2}{6}$	$\frac{5}{8}$	$\frac{3}{4}$	$\frac{4}{6}$	$\frac{5}{7}$	$\frac{4}{2}$
0.(3)	0.625	0.75	0.(6)	0.7	2

1	2	3	4	5	6
2	0.75	0.7	0.(6)	0.625	0.(3)
6	3	5	4	2	1

reorder: $u = (4, 3, 5, 4, 5, 2), \ v = (2, 4, 7, 6, 8, 6) \rightarrow k = 3$

initial items position $(6, 3, 5, 4, 2, 1)$

Upper bound with the Linear Relaxation:

Linear Relaxation:

$$\max \sum_{j=1}^{n} u_j x_j$$
$$s.t. \sum_{j=1}^{n} v_j x_j \leq C$$
$$0 \leq x_j \leq 1, j = 1, \ldots, n$$

Algorithm:

- Obtain the critical index k
- The optimal LR solution is

$$x_j^* = \begin{cases} 1, & 1 \leq j \leq k-1; \\ \frac{C - \sum_{j=1}^{k-1} v_j}{v_k}, & j = k; \\ 0, & k+1 \leq j \leq n. \end{cases}$$

Example (reordered): $u = (4,3,5,4,5,2)$, $v = (2,4,7,6,8,6) \rightarrow k = 3$
the LR optimal solution is $x_{LR}^* = (1, 1, \frac{12-6}{7}, 0, 0, 0) = (1, 1, 0.85, 0, 0, 0)$
with value $z_{LR}^* = 4 + 3 + 5 \times 0.85 = 11.28$

Lower Bounds by feasible solutions:

Greedy Algorithm:

- Obtain the critical index k
- Let

$$x_j = 1, j = 1, \dots, k-1;$$
$$x_j = 0, j = k, \dots, n;$$
$$Z' = \sum_{j=1}^{k-1} u_j.$$

- Consider $\underline{Z} = \max\{Z', u_k\}$

Example (reordered):

$u = (4, 3, 5, 5, 4, 2)$,
$v = (2, 4, 7, 8, 6, 6) \rightarrow k = 3$
the greedy feasible solution is
$x_G = (1, 1, 0, 0, 0, 0)$
with value $Z' = 4 + 3 = 7$
The lower bound is
$\underline{Z} = \max\{Z', u_k\} = \max\{7, 7\} = 7$

Lower Bounds by feasible solutions:

Greedy utility Algorithm:

- Order objects by non increasing order of utility
- For $j = 1, \dots, n$ take object j
  $\rightarrow$ if $\sum_{i=1}^{j} v_i \leq C$
  then $x_j = 1$,
  otherwise $x_j = 0$,
  $$\rightarrow Z' = \sum_{j=1}^{n} u_j x_j.$$
- Obtain $\underline{Z} = Z'$

Example: $u = (4, 3, 5, 4, 5, 2)$,
$v = (2, 4, 7, 6, 8, 6)$
the greedy feasible solution is
$x_G = (1, 0, 0, 1, 0, 0)$
with value $Z' = 5 + 4 = 9$
and capacity 9
The lower bound is $\underline{Z} = 9$

Set Covering Problem

Set Covering Problem

Given a matrix of zeros and ones and a cost associated with each column, determine the subset of columns that covers all the rows, i.e. such that for each row there is at least one in one of the selected columns. The set covering problem is NP-hard.

Sets:

- $N = \{1, \ldots, n\}-$ set of columns
- $M = \{1, \ldots, m\}-$ set of rows
- $N_i = \{j \in N : a_{ij} = 1\}, i \in M.$
- $M_j = \{i \in M : a_{ij} = 1\}, j \in N.$

Parameters:

1. c_j cost of the column j, $j \in N$.
2. $a_{ij} = 1$ if column j covers row i and $a_{ij} = 0$ otherwise, for all $i \in M$, $j \in N$.

ILP formulation for the Set Covering Problem

Variables:

$$x_j = \begin{cases} 1, & \text{if column } j \text{ is selected} \\ 0, & \text{otherwise;} \end{cases} \quad j \in J$$

$$min \sum_{j \in N} c_j x_j$$

$$s.a: \sum_{j \in N} a_{ij} x_j \geq 1, \forall i \in M$$

$$x_j \in \{0,1\}, j \in N$$

If constraints $\sum_{j \in N} a_{ij} x_j \geq 1, \forall i \in M$ are replaced by constraints

$$\sum_{j \in N} a_{ij} x_j = 1, \forall i \in M$$

we get the partition problem

Set Covering Problem: Variants

Multiple Set Covering problem

$$min \sum_{j \in N} c_j x_j$$

$$s.a : \sum_{j \in N} a_{ij} x_j \geq b_i, \forall i \in M$$

$$x_j \in \{0, 1\}, j \in N$$

Generalized Set Covering problem

$$min \sum_{j \in N} c_j x_j$$

$$s.a : \sum_{j \in N} a_{ij} x_j \geq b_i, \forall i \in M$$

$$x_j \geq 0 \text{ e inteiro}, j \in N$$

b_i is an integer greater than or equal to 1. For example, it could represent the minimum number of workers on the shift i.

Set Covering Problem: Preprocessing

Reductions:

1. If there exists $i \in M$ such that $N_i = \emptyset$ then the problem is impossible.
2. If $i \in M$ is such that $N_i = \{j(i)\}$ (i is covered by only one column) then $j(i)$ is in the solution.
3. (Dominance between rows) If $i, \ell \in M$ are such that $N_i \subseteq N_\ell$ then the row ℓ can be eliminated.
4. (Dominance between single columns) If $k, j \in N$ are such that $M_k \subseteq M_j$ and $c_k \geq c_j$ then column k can be removed.
5. (Dominance between columns) If $k, j_1, \ldots, j_s \in N$ are such that $M_k \subseteq \bigcup_{t=1}^{s} M_t$ and $c_k \geq \sum_{t=1}^{s} c_t$ then column k can be removed.
6. (Weak dominance between columns) Let $d_i = min_{j \in N_i} c_j$ and $k \in N$ such that $c_k \geq \sum_{i \in M_k} d_i$ then column k can be removed.

Set Covering Problem: Greedy Algorithm

Greedy Algorithm
Initialise $R \leftarrow M, S \leftarrow \emptyset, t \leftarrow 1$
While $R \neq \emptyset$ do
  Let $i^* \in R$ be such that $\mid N_{i^*} \mid = \min_{i \in R} \mid N_i \mid$
  Choose $j(t)$ such that $f(c_{j(t)}, k_{j(t)}) = \min\{f(c_j, k_j) : j \in N_{i^*} \wedge k_j > 0\}$
   where $k_j = \mid M_j \cap R \mid, \forall j \in N_{i^*}$
  Make $R \leftarrow R \setminus M_{j(t)}$, $S \leftarrow S \cup \{j(t)\}$, $t \leftarrow t+1$.
Sort the S cover in non-increasing order of costs: $S = \{j_1, \ldots, j_t\}$.
For $i = 1$ to t do:
  If $S \setminus \{j_i\}$ is cover then $S \leftarrow S \setminus \{j_i\}$

There are several alternatives to $f(c_j, k_j)$. For example $f(c_j, k_j) = c_j / k_j$.

Set Covering Problem: example

$m = 5, \quad n = 6, \quad [c_j] = [2\ 2\ 3\ 3\ 5\ 7]$

$$[a_{ij}] = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Exercise

Formulate the dual of the linear relaxation of the Set Covering problem. What is the relationship between the optimal value of the Set Covering problem and the value of a feasible solution to the dual?

Location Problem

ILP Formulation

Sets:

1. $I = \{1, \ldots, m\}-$ customers
2. $J = \{1, \ldots, n\}-$ services

Parameters:

1. f_j = cost of installing a service in j, $j \in J$
2. c_{ij} = cost of customer i being served by the service installed in j, $j \in J$

Variables:

$$y_j = \begin{cases} 1, & \text{if a service is installed on } j; \\ 0, & \text{otherwise;} \end{cases} \quad j \in J$$

$$x_{ij} = \begin{cases} 1, & \text{if customer } i \text{ is served by service } j, \\ 0, & \text{otherwise} \end{cases} \quad , i \in I, j \in J$$

ILP Formulation

$$\min \sum_{j \in J} f_j y_j + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}$$

$$s.t. : \sum_{j \in J} x_{ij} = 1, \forall i \in I$$

linking constraints

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

$$y_j \in \{0, 1\}, j \in J$$

ILP Formulation

Linking constraints - to ensure that a center j can only serve a customer i if a center is installed in j :

$$x_{ij} = 1 \Rightarrow y_j = 1$$

Alternative 1

$$x_{ij} \leq y_j, \forall i \in I, j \in J$$

Alternative 2

$$\sum_{i \in I} x_{ij} \leq m y_j, \forall j \in J$$

Greedy Heuristic

Step 0: **Initialisation**

- calculate $z_j = f_j + \sum_{i=1}^{m} c_{ij}$, for all $j = 1, \ldots, n$
- determine j^* such that $z_{j^*} = \min_{j=1,\ldots,n} z_j$
- $S := \{j^*\}$ (solution)
- $C(S) := z_{j^*}$ (solution cost)
- $u_i = c_{ij^*}$ for all $i = 1, \ldots, m$

Step 1: **Selecting a new center**

- for each $j \notin S$ calculate $\rho_j = f_j + \sum_{i=1}^{m} \min(0, c_{ij} - u_i)$
- determine j^* such that $\rho_{j^*} = \min_{j \notin S} \rho_j$
- if $\rho_{j^*} \geq 0$, STOP S contains the obtained solution of cost $C(S)$
- else **(Update)**
  - $S := S \cup \{j^*\}$
  - $C(S) := C(S) + \rho_{j^*}$
  - $u_i := min(u_i, c_{ij^*})$ for all $i = 1, \ldots, m$
  - if $|S| < n$, repeat this step

Example

$$m = 4, \quad n = 6, \quad [f_j] = [3\ 2\ 2\ 2\ 3\ 3]$$

$$[c_{ij}] = \begin{bmatrix} 6 & 6 & 8 & 6 & 0 & 6 \\ 6 & 8 & 6 & 0 & 6 & 6 \\ 5 & 0 & 3 & 6 & 3 & 0 \\ 2 & 3 & 0 & 2 & 4 & 4 \end{bmatrix}$$