LINMA2450 — Project Part 1
Combinatorial Optimization

Brieuc Dallemagne `77122100`   Alois Tavier `58242100`

Problem 1 — Optimal Boards Cutting

(a) **(1.1) Knapsack.**

A cutting pattern is feasible if the total sum of the lengths l_j of the pieces cut from a large board does not exceed the board length L. So we can express X, the set of all feasible patterns, as a knapsack-type constraint:

$$\sum_{j=1}^{n} l_j\, x_j \le L, \quad x_j \in \mathbb{Z}_+, \quad x \in P \quad \Rightarrow \quad x \in X.$$

We can formulate this as follow:

$$X = \left\{ x \in \mathbb{Z}_+^n : \sum_{j=1}^{n} l_j\, x_j \le L \right\}$$

(b) **(1.2) Model P1.**

Let X denote the set of feasible cutting patterns $x^t \in P$, where x_j^t is the number of items of type j in pattern t and $y_t \in \mathbb{Z}_+$ represents the number of boards cut according to pattern t.

$$\begin{aligned}
\text{(P1)} \quad & \min_{y_t \in \mathbb{Z}_+} \sum_{t \in X} y_t \\
& \text{s.t.} \sum_{t \in X} x_j^t\, y_t \;\ge\; d_j, \quad j = 1, \ldots, n, \text{ and } x^t \in X.
\end{aligned}$$

or in matrix form with $y \in Z_+^T$, $A \in Z_+^{n \times T}$ such that $a_{jt} = x_j^t$, $|X| = T$ and $d = \;[d_1, \cdots, d_n]$ we obtain:

$$\begin{aligned}
\text{(P1)} \quad & \min_{y \in \mathbb{Z}_+^T} \mathbb{1}^\top y \\
& \text{s.t. } A \cdot y \;\ge\; d, \quad x^t \in X.
\end{aligned}$$

(c) **(1.3) Size and Tractability.**

To generate new cutting patterns, we can solve a knapsack problem where the objective is to maximize the total used length without exceeding the board length L:

$$\max_{x \in \mathbb{Z}_+^n} \sum_{j=1}^{n} l_j x_j \quad \text{s.t.} \quad \sum_{j=1}^{n} l_j x_j \le L.$$

The resulting integer vector x defines a new feasible cutting pattern that can be added to the master problem (P1) to improve its primal bound.

The full model (P1) has one integer variable per feasible pattern, so $|P| = |X|$, where

$$X = \Big\{x \in \mathbb{Z}_+^n \mid \sum_{j=1}^{n} l_j x_j \le L\Big\}.$$

An upper bound on the number of variables is

$$|P| \le \prod_{j=1}^{n} \big(\lfloor L/l_j \rfloor + 1\big),$$

which grows combinatorially with n and L. For each item type j, the integer variable x_j can take any value between 0 and the maximum number of pieces of length l_j that fit in a board, given by $\lfloor L/l_j \rfloor$. Considering all item types simultaneously, the total number of combinations equals the product of these possible ranges. This exponential growth in feasible patterns makes enumeration computationally in-tractable, confirming that the cutting stock problem is NP-hard.

Proposition
Let's take an example where $L = n$ and $l_j = 1$ for $j = 1, \ldots, n$:
The constraints is the following:

$$X = \left\{x \in \mathbb{Z}_+^n \middle| \sum_{j=1}^{n} x_j \le n, \quad x_j \in \mathbb{Z}_+\right\}$$

Every binary vector $x \in \{0,1\}^n$ is a valid solution but $|\{0,1\}^n|$ is smaller than $|X|$ because this doesn't include the cases where $x_j > 1$.Let's construct a lower bound:

$$|X| \ge |\{0,1\}^n| = 2^n$$

We know that the number of variable in the model is $T = |X|$ and that X can grow exponentially in certain cases with the number of customer's demands. This exponential growth in feasible patterns makes enumeration computationally in-tractable, confirming that the cutting stock problem is NP-hard.

(d) **(2.1) Implementation.**

Code : See notebook

(e) **(2.2) Implementation.**

Code : See notebook

(f) **(2.3) Implementation.**

Code : See notebook

(g) **(3) Implementation.**

Code : See notebook

Problem 2 — Optimal Delivery Assignment

(a) **(1.1) Model — Optimal Delivery Assignment**

First we precompute

$$c_{ij} = \mathrm{dist}_G(i,j) \quad \text{(shortest-path length from } i \in D \text{ to } j \in C\text{; set } c_{ij} = +\infty \text{ if unreachable).}$$

We create binary variables x_{ij} s.t for each $i \in D$, $j \in C$,

$$x_{ij} \in \{0,1\} \quad \text{equals 1 if client } j \text{ is served by warehouse } i.$$

the we create our MILP formulation as.

$$\min_x \sum_{i \in D} \sum_{j \in C} c_{ij}\, x_{ij}$$

$$\sum_{i \in D} x_{ij} = 1 \quad \forall\, j \in C \qquad \text{(each client is assigned)}$$

$$\sum_{j \in C} x_{ij} \le 1 \quad \forall\, i \in D \qquad \text{(at most one client per warehouse)}$$

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

(b) **(1.2) The Hungarian Method**

The Hungarian method is a combinatorial optimization technique used to solve the assignment problem for 2 sets of the same size (finding a minimum-cost one-to-one matching between two sets).

Main idea: The method transforms the cost matrix step by step:

1. Subtract the smallest element in each row and column so that every row and column contains at least one zero.
2. Select a maximum set of independent zeros (one per row and column).
3. If all rows or columns are covered, the corresponding assignment is optimal. Otherwise, adjust the uncovered elements and repeat until a complete matching is obtained.

In our case the matrix might be rectangular. (when $|D| > |C|$) So, we added dummy rows with a cost of zero to obtain a square matrix.

Complexity. The algorithm runs in polynomial time $O(n^3)$ and guarantees an optimal solution for any finite cost matrix.

(c) **(1.3) Optimality and Tractability**

The Hungarian method always finds an optimal solution because it is based on the duality principle of linear programming. At each iteration, the algorithm maintains feasible dual variables (row and column reductions) and ensures complementary slackness with respect to zero-cost entries.

When all tasks are assigned, the primal (assignment) and dual (reduction) conditions are both satisfied, proving optimality.

Tractability. The assignment problem is efficiently solvable in polynomial time. Its linear relaxation is integral since the constraint matrix is totally unimodular, ensuring that the optimal solution of the relaxed problem is also integer.

(d) **(2.1) Mathematical Modeling.**

(e) **(2.2) Hungarian Method.**

(f) **(2.3) Implementation and Comparison.**

References

[1] Wikipedia contributors, *Assignment problem*, *Wikipedia, The Free Encyclopedia*, available at: `https://en.wikipedia.org/wiki/Assignment_problem`, accessed November 2025.

[2] BYJU'S, *Assignment Problem – Definition, Methods, and Solved Examples*, available at: `https://byjus.com/maths/hungarian-method/`, accessed November 2025.

[3] GeeksforGeeks, *Hungarian Algorithm for Assignment Problem (Set 1 – Introduction)*, available at: `https://www.geeksforgeeks.org/dsa/hungarian-algorithm-assignment-problem-set-1-introduction/`, accessed November 2025.