Programming Project II

Delivery Truck Pallet Packing Optimization

DA 2024/2025 Instructors Team

Departamento de Engenharia Informática (DEI)/Departamento de Ciências de Computadores (DCC)
Faculdade de Eng. da Univ. do Porto (FEUP)/Faculdade de Ciências da Univ. do Porto (FCUP)

Spring 2025

Due Date: May 26, 2025 at 23:59 (PT time)

1. Objectives

This second programming assignment focuses on the **Delivery Truck Pallet Packing Optimization Problem**, a real-world variation of the **0/1 knapsack Problem (0/1 KP)**. The goal is to optimize the packing of pallets in a delivery truck to maximize profit while respecting the truck's maximum weight capacity. The key objectives of this programming project include:

- Developing teamwork skills by working in groups of **2-3 students (3 preferred)**, fostering collaboration and project management skills.
- Implementing and analyzing **multiple algorithms approaches** to solve the problem, including exhaustive search, dynamic programming, and approximation/heuristic algorithms.
- Preparing and delivering a **concise demo presentation** to showcase the functionalities and results of your implementation, emphasizing the trade-offs between optimality and efficiency. In your presentation, you need to be **succinct**, **time-conscious**, and focused on what is essential.

This document describes the motivation of this project, the expected deliverables, the problem statement, and the demo you are expected to present. The problem statement includes a description of each task (alongside the corresponding grading). Lastly, we provide specific turn-in instructions you need to follow.

Recall, the deadline is May 26, 2025 at 23:59.

2. Problem Motivation

In logistics and supply chain management, efficiently packing delivery trucks is a critical challenge. Companies must maximize the value of each shipment while adhering to weight restrictions. This problem can be modeled as a **0/1 Knapsack Problem**, where:

- A **Truck** represents the knapsack.
- **Pallets** represent the items, each with a specific **weight** and **profit** (based on the value of the goods).
- The goal is to select a subset of pallets that maximizes the total profit without exceeding the truck's weight capacity.

In this project, you will explore different algorithmic approaches to solve this problem, including **exhaustive search**, **dynamic programming**, and **approximation and linear integer programming algorithms**. By implementing and evaluating these approaches, you will gain insights into the trade-offs between **optimality**, **efficiency**, and **feasibility** in solving real-world optimization problems.

3. Problem Data and Interface

To make this problem more realistic, you are provided with several dataset representing instances of the **Delivery Truck Pallet Packing Problem**. The datasets include:

- A set of **pallets**, each with a *weight* (w) and a *profit* (p) (based on the value of the goods).
- A *truck capacity* (W) representing the maximum weight the truck can carry.

The datasets are structured as **CSV files**, with the following format:

- **File TruckAndPallets_<X>.csv:** Contains information about the truck capacity and the number of pallets, for each dataset, where **<X>** stands for the identifier number of the dataset (for example, **TruckAndPallets_01.csv** is the file with start information related to first dataset provided). The first line includes the header: **`Capacity`** (truck capacity) and **`Pallets`** (number of pallets available for packing).

```
Capacity,       Pallets
100,            9
```

- **File Pallets_<X>.csv:** Contains the information about each pallets available for packing, for each dataset, where **<X>** stands for the identifier number of the dataset (for example, **Pallets_01.csv** is the file with pallets information related to first dataset provided). The first line includes the header: **`Pallet`** (unique identifiers), **`Weight`** (pallet weight), and **`Profit`** (pallet profit). The next **`Pallets`** lines contains the information for each pallet available for packing.

```
Pallet,         Weight,     Profit
1,              70,         10
2,              60,         5
3,              50,         10
...
```

4. Problem Statement and Organization of Work

To facilitate your work, we have structured the functionalities that you are expected to develop as follows.

4.1. Basic Application Definition and Set-Up

[T1.1: 1.0 point] Develop a Pallet Packing Optimization Tool. The first task will be to create a simple command-line menu that exposes all implemented functionalities in a user-friendly manner. The menu should allow users to select the dataset, choose the algorithmic approach, and view the results. This menu will also serve as a key component for showcasing the work you have developed in a short demo at the end of the project.

[T1.2: 0.5 point] Read and Parse the Input Data. You will need to develop basic functionality (accessible through your menu) to read and parse the provided data set files. Store the data in appropriate data structures (e.g., arrays, lists, …) for use in your algorithms.

[T1.3: 1.5 points] Documentation and Time Complexity Analysis. Include documentation for all implemented code using **Doxygen** (especially for the main algorithms implemented). This should indicate the **time and space complexity analysis** for each of the most relevant implemented algorithms.

4.2. Algorithmic Approaches

For this part of the project, you should implement the algorithms presented during course classes to solve the **0/1 KP** as well as implement more elaborate algorithms:

[T2.1: 3.0 points] Exhaustive (Brute-Force) Approach:

- Implement a brute-force algorithm that explores all possible subsets of pallets to find the optimal solution.
- Analyze the performance of this approach and observe its limitations, particularly for larger datasets.
- Optionally, implement a backtracking algorithmic approach to the 0/1 KP and to try to improve efficiency.

[T2.2: 4.0 points] Dynamic Programming Approach:

- Implement a dynamic programming solution and pay special attention to the data structure used to store the dynamic programming table, as it can become very large for bigger datasets.
- Evaluate different data structure alternatives (e.g., arrays, hash maps) and consider implementing a custom solution if necessary.
- Analyze the performance of this approach and observe its limitations.

[T2.3: 4.0 points] Approximation Algorithms (Greedy Approach):

- Implement the greedy algorithm studied in class that selects pallets based on their **weight-to-profit ratio**, where the last pallet might not be included as one cannot consider a fraction of a pallet.
- Compare the accuracy and performance of this approximation algorithm with the optimal solutions obtained from the exhaustive and dynamic programming approaches (using only small datasets).
- Evaluate the performance and fidelity of the greedy algorithm in the small and large data sets.
- Remember that this problem is not the Fractional Knapsack Problem.

[T2.4: 2.0 points] Integer Linear Programming Algorithm (ILP):

- This task will require some critical thinking and analysis skills. You should develop and analyse your own ILP approach to the 0/1KP.
- You should mathematically formulate the 0/1KP properly as an ILP.
- As an alternative to ILP you are free to pursue other algorithmic approaches such as Genetic Algorithms or a Hybrid Algorithmic approach specifically suited to selected input datasets.
- Critically discuss your justification for the algorithms implemented.

4.3. Performance Evaluation

[T3.1: 2.0 points] Evaluate and Compare Algorithms

- Test your implementations on the provided datasets.
- Measure and report the **run time execution** for each algorithm and dataset. You should perform a graphical representation of the different times obtained.
- Compare the space complexity of the different solutions.
- Compare the accuracy of the different solutions (i.e., how close the greedy algorithm's solution is to the optimal solution).

5. Demo & Presentation

[T4.1: 2.0 points] Giving a presentation that is "short-and-to-the point" is increasingly important. As such, you are required to structure a **short 10-minute** demo of your work, highlighting key aspects of your implementation, specifically:

- Present your different solutions for the provided input datasets demonstrating the results requested in this project;
- Highlight the key aspects of your algorithm solutions, including the conceptual decisions that were made (for example related to the data structures used), and the trade-offs between optimality and efficiency;
- Highlight the most important and challenging aspects of your implementations.

You should also elaborate a PowerPoint presentation to support your presentation. Instructions for preparing the PowerPoint presentation are available on Moodle.

6. Turn-In Instructions & Deadline

Submit a zip file named **DA2025_PRJ2_G<GN>.zip** on Moodle, where **GN** stands for your group number, with the following content:

- **Code** folder (contains program source code)
- Documentation folder (contains html documentation, generated using **Doxygen**)
- Presentation file (**PDF format**) that will serve as a basis for the demonstration.

Late submissions, up to 24 hours and 48 hours, will incur a penalty of 10% and 25% of the grade, respectively. No submissions will be accepted 48 hours after the deadline. Exceptions apply for justified and documented technical submissions issues. There is an **automatic grace period of 59 mins and 59 seconds (1 hour)** where you can submit your code without penalty.

Recall, that the deadline is May 26, 2025, at 23:59 (PT time).