Coursework: Optimizing Inventory Management with 1D Bin Packing

Xinan Chen
xinan.chen@nottingham.edu.cn
University of Nottingham Ningbo China - Artificial Intelligence Methods (COMP2051)

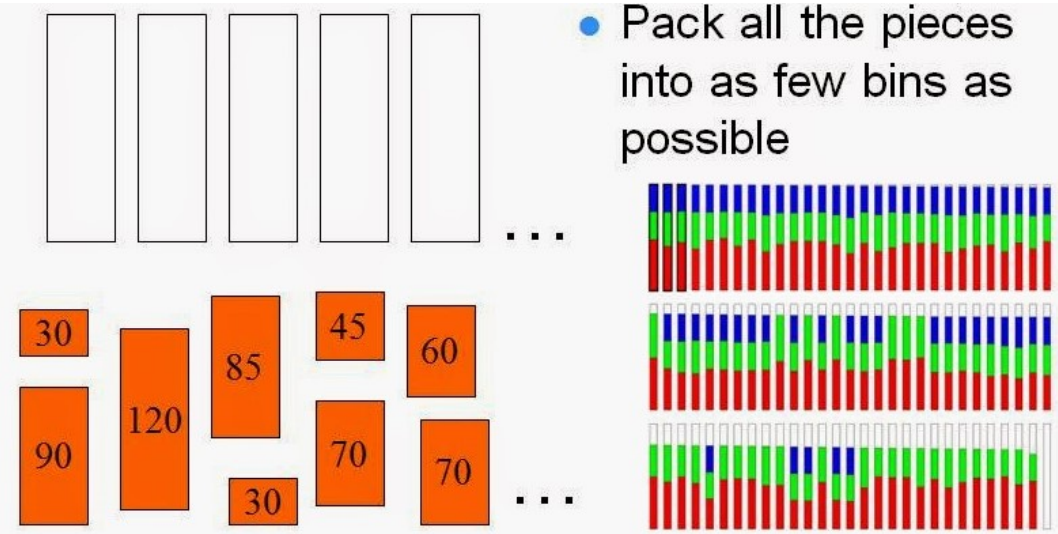
1 Overview

This coursework focuses on the application of the 1D bin packing problem to inventory management in a retail or manufacturing setting. Students will be tasked with developing an algorithm to efficiently allocate items of various sizes into fixed-size bins (representing storage units or shipping containers) to minimize space wastage and optimize inventory storage or distribution costs. This real-world application helps in understanding how computational algorithms can solve operational challenges in inventory management.

This coursework carries **45% of the module marks**. The rest of the module marks come from the final written exam and lab quizs.

2 Background

In inventory management, space optimization is crucial for reducing storage and transportation costs. The 1D bin packing problem, a classic problem in artificial intelligence and management, involves packing objects of different volumes into a finite number of bins with a certain capacity in the most space-efficient way. Applying this concept to inventory management can significantly enhance operational efficiency and cost-effectiveness.



3 Bin Packing Problem (BPP)

Given a set of n items, each item j has a volume of a_j, BPP aims to pack all items in the minimum number of identical-sized bins without violating the capacity of bins (V). The problem can be mathematically formulated as follows:

$$\text{minimize } B = \sum_{i=1}^{n} y_i$$

$$\text{subject to } B \geq 1,$$

$$\sum_{j=1}^{n} a_j x_{ij} \leq V y_i, \forall i \in \{1, \ldots, n\}$$

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

$$y_i \in \{0, 1\}, \qquad \forall i \in \{1, \ldots, n\}$$

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

where $y_i = 1$ if bin i is used and $x_{ij} = 1$ if item j is put into bin i.

This mathematical formulation is generally NOT solvable by existing integer programming solvers like CPlex, Gurobi, and LPSolve, especially when the number of items n is large. To consistently solve the problem with good quality solutions, metaheuristics and hyper-heuristics are used, which is the task of this coursework.

4 Objectives

- To understand the principles and applications of the 1D bin packing problem in inventory management.
- To develop and implement an algorithm that optimizes the packing of items into bins.
- To evaluate the algorithm's efficiency and effectiveness in reducing unused space and potential cost savings.

5 Materials

Students will be provided materials that include:

- A list of items to be stored or shipped, each with its volume.
- The volume capacity of the bins (assumed to be uniform for simplicity)
- An example python code can output the solution file.

JSON File: https://moodle.nottingham.ac.uk/pluginfile.php/11653384/mod_folder/content/0/CW_ins.json?forcedownload=1
Example Code: https://moodle.nottingham.ac.uk/pluginfile.php/11653384/mod_folder/content/0/CW_exp.py?forcedownload=1

6 Tasks

- Literature Review:
  Study the 1D bin packing problem and existing algorithms used to solve it, such as Next-Fit First-Fit, Best-Fit, and other advanced methods.
- Algorithm Selection and Development:
  Choose an appropriate algorithm or develop a new strategy for solving the 1D bin packing problem in the context of inventory management. Implement the algorithm using a programming language of choice (e.g., Python, Java).
- Simulation and Optimization:
  Run simulations using the provided dataset to allocate items into bins based on the developed algorithm. Optimize the algorithm to improve the space utilization ratio.
- Analysis and Reporting:
  Analyze the algorithm's performance, comparing it with other strategies studied during the literature review. Prepare a detailed report documenting the project's methodology, implementation, results, and conclusions.

7 Deliverables

- A no more 3 pages **report** includes a declaration of AI usage, introduction, description of the chosen algorithm, implementation details, results, analysis (compare with other algorithms), and conclusions (a PDF file).
- **Source code** for the implemented algorithm (a Python/C/Cpp/Java file program, program should take no more than 5 mins on the CS Linux to produce output). Hint: Using **delta evaluation** or other methods to save evaulation time, and adapt your approach based on dataset size.
- **Solutions** for the provided problem sets (a JSON file output by example code).

Note:

- Submit your lab report via the dedicated Moodle coursework link (https://moodle.nottingham.ac.uk/mod/assign/view.php?id=8081091).
- Essay only accepts .pdf format.
- Code only accepts .py/.c/.cpp/.java format.
- Solution only accepts .json format.
- Please name your submission file as ID_FirstName_LastName.xxx (e.g., 1234567_Xinan_Chen.pdf).
- Late submission is subject to normal late submission penalties. Three extra hours are provided on the moodle submission portal in case of internet connection issues. The extra hours should not be relied on for submission. Submitting your essay a few hours before the deadline would always be good practice.
- You can ask your classmates for help, but you cannot completely copy their code, solutions, or essays.
- The CS Linux is used to test your program. If your runtime exceeds 5 minutes, it may result in penalties, or you may even receive a 0 for the results part.

- Your solution should match your program. If your program does not match your solution, you may receive a 0 mark in the results part.

8 Mark Criteria

1. The quality of the experimental results (30%). Your algorithm shall be tested for a file containing 10 instances. The performance of your algorithm is evaluated by computing the absolute gap with the best-known results using. If your result is better than the best-known result, you will receive a **3-mark bonus per instance**, but the total marks for this coursework cannot exceed 100:

$$abs\_gap = your\_bin\_used - best\_known\_bin\_used \tag{1}$$

Criteria	Mark
$abs\_gap < 0$	3 marks + 3 marks bonus per instance
$abs\_gap = 0$	3 marks per instance
$0 < abs\_gap <= 1$	2 marks per instance
$1 < abs\_gap <= 2$	1 marks per instance
$2 < abs\_gap <= 3$	0.5 marks per instance
$abs\_gap > 4$ or errors	0 marks

Best-known results for all instances:

- Instance: instance_1 Best known of bins used: 52
- Instance: instance_2 Best known of bins used: 59
- Instance: instance_3 Best known of bins used: 24
- Instance: instance_4 Best known of bins used: 27
- Instance: instance_5 Best known of bins used: 47
- Instance: instance_6 Best known of bins used: 49
- Instance: instance_7 Best known of bins used: 36
- Instance: instance_8 Best known of bins used: 52
- Instance: instance_large_9 Best known of bins used: 417
- Instance: instance_large_10 Best known of bins used: 375
- Total: 1138

Note: Use the solution marker to check the mark and feasibility for your solution.
`https://moodle.nottingham.ac.uk/pluginfile.php/11653384/mod_folder/content/0/CW_marker.py?forcedownload=1`

2. The quality of codes (30%) The code quality evaluation will be based on the following criteria, each contributing to the total mark of 30. The objective is to assess not only the functionality of the code but also its efficiency, readability, and adherence to good programming practices.

   (a) Readability (10 Marks)

   - Clarity: Code is easy to read and understand (5 Marks).
   - Comments: Adequate comments are provided to explain the logic and functionality (5 Marks).

(b) Structure and Organization (10 Marks)

- Modularity: Code is well-structured and divided into functions or modules where appropriate (6 Marks).
- Consistency: Consistent naming conventions and code layout (4 Marks).

(c) Efficiency and Performance (10 Marks)

- Algorithm Efficiency: The chosen algorithm or approach efficiently solves the problem (6 Marks).
- Resource Management: Efficient use of computational resources, minimizing unnecessary computations (4 Marks).

3. Report (40%) The report submitted for evaluation will be assessed based on the following criteria, totaling 40 marks:

(a) Introduction and Background (7 Marks)

- Clarity of Problem Statement (4 Marks): Clear presentation of the problem being addressed.
- Context and Relevance (3 Marks): Explanation of the importance and applications of the problem.

(b) Methodology (10 Marks)

- Approach and Justification (5 Marks): Detailed description of the approach taken to solve the problem, including justification for the chosen methods.
- Implementation Details (5 Marks): Comprehensive explanation of how the solution was implemented, including any algorithms, models, or frameworks used.

(c) Results and Analysis (16 Marks)

- Presentation of Results (6 Marks): Clear and accurate presentation of the findings, supported by appropriate figures, tables, or graphs.
- **Critical Analysis (10 Marks)**: In-depth analysis of the results, including comparing their performance with other methods and the pros and cons of each method.

(d) Conclusion and Recommendations (7 Marks)

- Summary of Key Findings (4 Marks): Concise summary of the main outcomes of the project.
- Future Work and Recommendations (3 Marks): Insightful suggestions for future research or improvements to the project based on the findings.

9 AI Usage

This coursework permits using AI to assist in programming or essay writing. However, you must **declare the use of AI at the beginning of your essay** (e.g., used AI for programming). Failure to do so may result in a loss of marks for any part where AI assistance is detected.

10 Repetition Check

If a repetition rate higher than **20% is found (or 5% from a single source)**, your submission will be checked by lecturers. You may lose a certain number of marks for improper citations, plagiarism, etc. Severe cases will be reported to the exam officer for further action.

11 Submission Deadline

6th May 2025, 11:59pm Beijing Time