MAST 90014 - Optimisation for Industry Group Project 2025

Student Name 1, Student Name 2, Student Name 3
Student ID 1, Student ID 2, Student ID 3
Course Name

May 18, 2025

1 Introduction

2 Problem definition

3 Data

4 Model

5 Solution strategy

As described in the above section, we propose an IP model to optimize the rental and scheduling plan. Such a formulation is usually solved exactly using branch-and-bound algorithm. In this project, We solve the proposed formulation using Gurobi, which solves IP or MIP model using branch-and-cut algorithm. Compared with standard branch-and-bound algorithm, this solver integrates some heuristics policy and valid cuts into the branch-and-bound algorithm, significantly accelerating the solving process.

However, this solver cannot handle our formulation when the size is large. Specifically, we observe that the solver needs more than 100 seconds to solve the size of 5 warehouses, 30 types of goods, 15 retailers and 5 periods using our synthetic data. To overcome this issue, some techniques proposed in this section are applied to accelerate the solving process.

5.1 Synthetic Data

Our collected dataset isn't enough to support the computational evaluation for large-scale instance, we thus extend it with some modifications. While some parameters are used directly, other parameters are generated using the random number API provided by Numpy.

In addition, we adjust the maximum number of trucks using the following expression:

$$maximum\ number\ of\ trucks = \frac{total\ demands}{number\ of\ periods}.$$

Table 1: Example of a two-column table with three lines

Column A	Column B
holding costs	$U(20, 100)$
transportation costs	$U(1, 4)$
penalty costs	$U(1, 1.2)\times$ holding costs
demands	$U(40, 100)$

5.2 Branching policy

The branch priority is significant to the solving efficiency. As the branch-and-bound is far more larger, especially when variables are integer, compared with binary case. To mitigate this issue, we require the solver to branch significant variables with higher priority.

In our case, the priority of the number of trucks rented is assigned with the highest priority, followed by the scheduling variables, and then the shortage calculation. The above process is implemented by setting the parameter ***BranchPriority*** of each variables in Gurobi.

5.3 Active cuts search

We note that the best bound of the problem provided by the solver is hard to be improved, while the best incumbent is relatively easy to derive with the help of heuristics integrated into the solver. Based on this observation, we attempt to add more valid cuts into the formulation, so as to tighten the relaxation bound. To implement this, we set the parameter ***Cuts*** to **3** in Gurobi, so as to encourage the solver to find more valid cuts.

5.4 Computational test

In this section, we compare the efficiency between our method and the benchmark (using Gurobi directly without any modifications). The maximum runtime of Gurobi is limited to 100 seconds to meet the efficiency requirements of modern business operations.

We run the same instance for 10 times to eliminate the randomness introduced by the solver behaviors. The computational result is listed in Table 2.

Table 2: Performance comparison

Scale	**Improved**		**Benchmark**	
	Time (s)	Gap (%)	Time (s)	Gap (%)
(5, 25, 15, 5)	**37.01**	0.00	76.92	0.00
(5, 25, 15, 10)	100.08	0.00	100.21	0.00
(5, 30, 15, 5)	**37.43**	0.00	100.15	0.00
(5, 35, 15, 5)	100.14	0.00	100.14	0.00
(10, 25, 25, 5)	100.24	0.00	100.07	0.00
(20, 60, 50, 5)	100.32	0.00	100.24	0.00

We observe that most instances can be solved to near optimality within 100 seconds. However, our method performs better than the benchmark method in larger instances in solving efficiency, while similarly in small cases.

6 Results and analysis

7 Conclusion and recommendations

8 Individual contributions

Yitian Wang: Synthetic data creation, solution strategy design and computational experiments design in Section 5.