Lab 3 Analysis of Knapsack Model Solver Performance

Mateo Saenz Ortiz

November 25, 2024

1. Overview of the Problem

The knapsack problem is the following problem in combinatorial optimization:

Given a set of items, each with a weight and a value, determine which items to include in the collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items. The problem often arises in resource allocation where the decision-makers have to choose from a set of non-divisible projects or tasks under a fixed budget or time constraint, respectively.

Parameters:

- n: Number of items.
- $poidsMax$: Maximum weight capacity of the knapsack.
- $valeur[i]$: Value of item i.
- $poids[i]$: Weight of item i.

Decision Variables:

- $x[i] \in \{0, 1\}$: Binary variable; 1 if item i is selected, 0 otherwise.

Objective Function:

$$\text{Maximize} \quad \sum_{i=1}^{n} valeur[i] \cdot x[i]$$

Constraint:

$$\sum_{i=1}^{n} poids[i] \cdot x[i] \leq poidsMax$$

2. Analysis of Solver Results

In this section, we provide an in-depth analysis of the solver's performance for three different dataset sizes: 500k, 1 million, and 1.5 million items. We discuss the solving time, the quality of the solutions, and the gaps between the upper and lower bounds for each dataset.

2.1. Results for Datasets with 240 or Fewer Instances

For datasets containing up to 240 instances, the solver was able to find optimal solutions with a very short computation time. Since the problem size is smaller, the time required to process each instance was significantly less than one second, which made the 15-second time limit more than sufficient. The gap in these cases was always 0, confirming that the solver found the optimal solution each time.

2.2. Results for 500k Items

The solver was able to find high-quality solutions for the 500k dataset. The gaps observed were very small, indicating that the solver is effectively finding near-optimal solutions.

Table 1: Results for 500k Items

Filename	Upper Bound (UB)	Lower Bound (LB)	Gap	Time (s)
1	12,702,148	12,702,365.8135	0.000017148	30.016
2	12,736,600	12,737,456.5391	0.00006725	26.938
3	12,717,064	12,717,158.8876	0.000007461	27.063
4	12,713,355	12,713,452.5993	0.000007677	29.515
5	12,757,978	12,758,136.0423	0.000012388	28.797

Key Observations:

- The gap values are very small, showing that the solver is performing well and providing near-optimal solutions.
- The average solving time is approximately 28.5 seconds, which is efficient for a dataset of this size.

2.3. Results for 1 Million Items

For the 1 million item dataset, the solver's runtime increased as expected, and the gaps between the upper and lower bounds remained very small, further confirming the solver's efficiency in obtaining near-optimal solutions.

Table 2: Results for 1 Million Items

Filename	Upper Bound (UB)	Lower Bound (LB)	Gap	Time (s)
1	25,464,652	25,465,285.4506	0.000024876	99.063
2	25,487,511	25,488,241.0598	0.000028644	87.968
3	25,473,187	25,473,274.0660	0.000003418	81.719
4	25,431,909	25,433,088.0666	0.000046362	97.281
5	25,517,516	25,519,415.6942	0.000074447	88.734

Key Observations:

- The gap values remain very small, consistently close to zero, indicating that the solutions are almost optimal.
- The average solving time for this dataset is about 88 seconds, which is a reasonable increase compared to the 500k dataset.

2.4. Results for 1.5 Million Items

The solver's runtime for the 1.5 million items dataset increased further, but it continued to provide high-quality solutions with minimal gaps, demonstrating its scalability.

Table 3: Results for 1.5 Million Items

Filename	Upper Bound (UB)	Lower Bound (LB)	Gap	Time (s)
1	38,206,237	38,207,041.6088	0.00002106	148.782
2	38,213,475	38,215,211.4937	0.000045442	126.485
3	38,206,031	38,206,604.9231	0.000015022	154.359
4	38,206,363	38,208,067.2385	0.000044606	130.750
5	38,175,360	38,176,267.5440	0.000023773	98.265

Key Observations:

- The solver consistently achieves near-optimal solutions with gaps that are almost zero.
- The average solving time is about 130 seconds, which is expected due to the increase in problem size.

2.5. Impact of Time Limit on Performance

The time limit initially set at 15 seconds was found to be too short to obtain high-quality solutions for larger datasets. In contrast, increasing the time limit to 300 seconds allowed the solver to reduce the gaps and achieve better solutions. For smaller datasets (up to 240 items), a 15-second limit was sufficient, but for datasets larger than this, a longer time limit was required to ensure optimality.

3. Observations on Solver Performance

- **Gap Analysis:** The gaps for all datasets are close to zero, indicating the solver is highly effective in finding near-optimal solutions.
- **Runtime Analysis:** The runtime scales with dataset size. For 1 million items, the average runtime was 85 seconds, while for 1.5 million items, the average was 130 seconds.
- **Impact of Time Limit:** Increasing the time limit significantly improves solution quality for larger datasets. With the 15-second limit, smaller datasets are solved optimally, but larger ones show higher gaps.
- **Model Efficiency:** The model handles large-scale datasets efficiently, demonstrating the robustness of CPLEX in solving complex integer programming problems.

4. Conclusion

The knapsack model demonstrates impressive performance with the solver, delivering near-optimal solutions even for datasets containing up to 1.5 million items. While smaller

datasets can be solved within shorter time limits, larger datasets achieve optimal results with extended time limits, ensuring higher solution accuracy. These findings highlight the solver's efficiency and its ability to effectively tackle large-scale optimization problems, confirming its robustness and scalability across a wide range of problem sizes.