Combinatorial Optimization
Homework 3 – Experimental Measurements

Matyáš Skalický
skalimat@fit.cvut.cz

November 21, 2021

Contents

1 Measured Algorithms

Baseline exact method is the **brute-force** algorithm that simply iterates all possible solutions without any speedups.

The **branch&bound** extends the brute-force with 2 speedups. We stop when the current candidate already exceeds the capacity of a bag. Also, we don't recurse further when the cost sum of the items that can be added into the bag is lower than the best found solution.

The dynamic programming approach is based on the **decomposition by cost**. This solution is based on memoization by the recursive function calls by returning the maximum value from the tested branches on return.

The **greedy** heuristic simply adds the items with the highest cost/weight ratio until the capacity of the bag is reached. This is the only algorithm that doesn't always result in the optimal solution.

2 Measured Variables

For the pilot experiments, I've chosen to benchmark all algorithms on the instances of size 24. The generator's maximum weight W is set to 3000 as well as maximum price C.

In the following experiments, we will try to measure how time and the solution quality depend on the generator inputs. Also, we will look how robust each algorithm is against permutations in the algorithm inputs.

- Ratio of bag capacity to summary weight. (cw_ratio $\in [0, 2]$)
- Correlation between price and weight. (pw_corr $\in [0, 1]$)
- Granularity and distribution of weights. (w_dist $\in [0, 2]$)

We will try to put emphasis on the hypothesis below:

3 Experiments

We will measure the effect of each variable separately. We will generate 10 problem instances for each algorithm with each weight.

3.1 Robustness

First, we will try to answer the robustness (invariance to order of input items) of each algorithm. We will use default generator values mentioned above. We will generate 500 instances and calculate 10 permutations for each algorithm. We will calculate the variance in the elapsed time across the 10 permutations and take a mean across all instances.

method	runtime mean variance
greedy_simple	0.00000004
branch&bound	0.00016046
bruteforce	0.00017201
dynamic_cost	0.06896283

Table 1: Mean variance for 10 runs of given method

3.2 Pilot Experiments

We will not measure the permutations in the pilot experiments.

3.3 Detailed Experiments

4 Discussion and Takeoffs