# Combinatorial Optimization
# Homework 1 – Knapsack Problem

Matyáš Skalický
skalimat@fit.cvut.cz

October 10, 2021

## Contents



## 1 Implementation

The brute-force solver of the knapsack problem was implemented using the recursive approach. The solver was implemented in Python. To speed the experiments up, they were ran in parallel to fully utilize the CPU and to make the experiments feasible. The datasets were subsampled from 500 to 250 samples to speed up the experiments.

The naive brute-force search recursively enumerates all possible solutions until a feasible solution is found, or the solution space exhausted.

An optimized branch&bound solver was also implemented and compared to the naïve brute-force approach. It utilizes several speedups:

*exceeded weight* Weight of the bag already exceeds the weight capacity of the bag. Do not recurse further.

*residuals bestcost* Sum the cost of remaining items that can be added into the bag. Do not recurse further if (cost + residual) $<$ bestcost where *bestcost* is the best achieved solution so-far.

*residuals mincost* Sum the cost of remaining items that can be added into the bag. Do not recurse further if (cost + residual) $<$ mincost where *mincost* is the decision version requirement.

## 2 Experiments

### 2.1 Instance Size

The model trained with batches of 2048 samples on all of the (110k) provided data usually reached best validation scores around 30 epochs. A Nadam (Adam with Nesterov momentum) optimizer was used.

### 2.2 Histogram

## 3 Dataset Comparison