Project 3

Jan van Ruijven 2006698

December 21, 2023

Exercise 1

1

For the generation of Quadratic knapsack instances the following has been applied:

1. Items in total: 300
2. Capacity of the knapsack: 15
3. Weights of the items: Uniformly distributed from 0 to 1
4. Values of the items: Uniformly distributed from 0 to 1
5. Combinatory value of the items: Uniformly distributed from 0 to 1

Solving this kind of instance with gurobi results in long runtimes (over 30 seconds). Setting the maximum execution time of the solver to 30 seconds will result in optimality gaps of around 10%.

2

To explain how finding an optimal solution can take so long, one has to look at the amount of possible solutions. In this case, there are 300 items with an average weight of 0.5. Finding 30 items that will fit in the knapsack is an easy task and calculating the end result as well. However, assuming the we have atleast 100 items with weight < 0.5. There will be atleast 30 items to put in the first position, 29 to put in the second postion etc. This will result in atleast 2.65×10^{32} possible solutions. Therefore, calculating them all and comparing the results is very time consuming and as the amount of items increases, the amount of time needed to calculate increases in a non-polynomial matter. Having many possible solution is not really a problem when can you make use of derivatives to estimate where the optimal location may be. But in this case we have a binary decision variable and are unable to do so. CLIN does help bring some linearity into the original problem in exchange for more constraints. Overall, this does seem to help to help the gurobi solver to find a more optimal solution faster.

3

a

The neural network is going to decide which item will be in the knapsack. It will be trained on quadratic knapsack instances as described in section 1. However, the number of items will be decreased to 200 in total, the rest stays the same. This is due to the fact that we can find optimal solutions in a faster manner this way (these are needed for training) and if we let the neural network decide which items will not be in the solution of an instance with 200 items, the items probably not be in an instance with 300 items. This is due to the fact that the capacity is equal, but there are only more items to choose from. The neural network looks at each item individually and recieves 5 pieces on information:

1. Value of the item
2. Weight of the item
3. 100th highest combinatory value of the item
4. 99th highest combinatory value of the item
5. 98th highest combinatory value of the item

It was found in the training of the neural network that adding the top 100 highest combinary values was adding to much noise to the data. Adding the top combinatory values from 1 to 5, resulted in almost no improvement as well, this was due to the fact that when there are 300 items to have a good combinatory value with, all of the items had a very similar top 5. But as it turned out, adding the 98th, 99th and 100th item would bring better predictions from the network.

b

The overall design of the network looks as follows:

1. Hidder layer 1: 20 nodes, relu activation
2. Hidder layer 2: 20 nodes, relu activation
3. Hidder layer 3: 4 nodes, relu activation
4. Output layer 4: 1 node, sigmoid activation

The model is optimized using adam's optimizer, with a learning rate of 0.003. In the dataset generated there are 20.000 items with their respective values, 15.000 of them were used to train the neural network and 5000 of them are used to test the neural network. The most important aspect to test the neural network on is the number of false negatives. We want to bring better results, so removing an item that would have origianlly in the knapsack is a bad descision from the network. False positives do not really affect the end result at all, estimating an item as positive means the do not remove it (so do nothing).

c

After training the neural network, the amount of true/alse positves and true/false negatives are calculated for a decision making threshold, an overview of these results can be found in the AP-

PENDIX. The over accuracy of the neural network is 0.97. To compare the mathheuristic to the original CLIN, we have done 10 testruns where both the original CLIN and the mathheuristic have a time limit of 30 seconds, and runs where the mathheuristic has a time limit of 25 seconds. This is mainly due to the fact that the mathheuristic is already spending time deciding whether or not we should include an item in the solver or not. The overall results can be found in APPENDIX2 APPENDIX3. The mathheuristic does completely outperform the original problem.

Exercise 2

1

The first greedy strategy is to order the items from high to low and put them in the next fitting bin. The second greedy strategy is to also sort the items from high to low, but this time we divide the items evenly over two bins, if an item does not fit in one of the two bins, we start dividing the items over the second and third bin. An overview of the algorithms can be found in the APPENDIX.

2

In the instance created there are always 5 visible items, this is also the number of actions the neural network has, the network has 5 output nodes, each node representing the predicted reward.The action with the highest predicted reward is taken. Each time the neural network chooses an item, it will be put in the next fitting bin. Each time the neural network chooses an item and this item was already put in a bin, the reward is 0. If the neural network chooses an item and the number of bins stays the same, the reward is 1 and if the number of bins increases, the reward is 0.5. The neural network observes 102 pieces of information in total:

1. The capcity of the bins
2. The number of items seen
3. The weight of each visible item (5 in total)
4. The current weight in each bin (100 in total)

With this information the neural network has to decide which item to take, after taking the action the observation space, the action taken and the reward associated are stored in memory for training.

3

For the Online Bin Packing Problem the folliwng instances are created:

1. Capacity of the bins: 100
2. Number of visible items: 5
3. Number of items in total: interger uniformly distributed from 75 to 100
4. Weight of each item: interger uniformly distributed from 1 to 100

4

After testing multiple neural networks the overall design looked like this:
The greedy strategy is defeated for the first time after 550 games are played. But in the end the greedy still seems a bit better.

5

After comparing t