VERKFRÆÐILEGAR BESTUNARAÐFERÐIR

Day 3

T-423-ENOP

Arnar Gylfi Haraldsson
arnarh23@ru.is

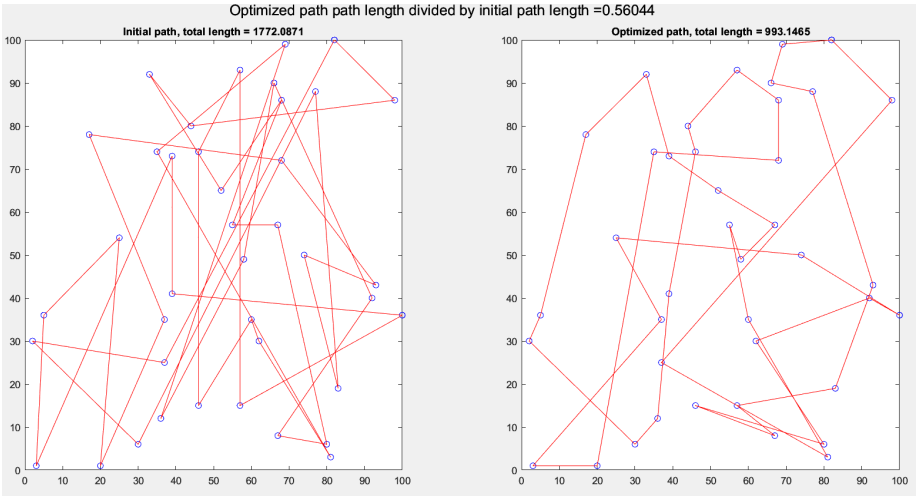
Hafþór Árni Hermannsson
hafthorh20@ru.is

Ragnheiður Gná Gústafsdóttir
ragnheidurg@ru.is

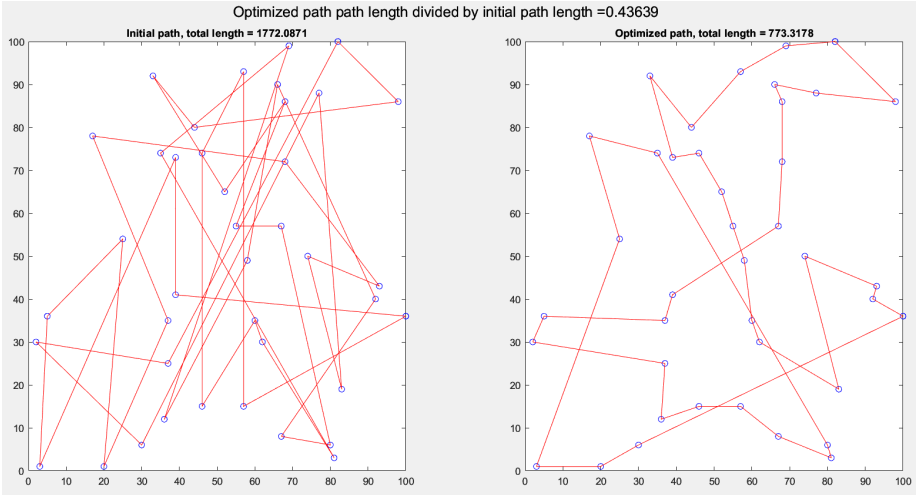
May 2, 2024

Traveling Salesman Problem

Description:
Given an N number of cities, what is the shortest path a salesman can take that goes through all the cities?
Solution:
We randomly generate N points and define a vector $[1,2,3,...,N-1,N,1]$ that describes the city node connections, initially going from city 1 to 2, 2 to 3 and so on, ending on 1 again to ensure a circular path. Then we randomly swap the indices, exluding the first and the last one from swapping, to ensure a circular path. This does not affect the optimization process as changing the start and end point cannot influence the total length of a circular path.
If the total length for the new path definition is shorter, then we accept that path, if it is longer we obviously reject it and keep the original.
Using 40 cities and 1000 iterations we get the following results:



Using the same cities, and increasing the number of iterations to 50000 yields the following plot:



So not a huge improvement from 1000 iterations.
Plotting the path length as a function of iterations we get the following graph: We plateau around 2500 iterations for this specific problem, but different random city placements might influence that result.

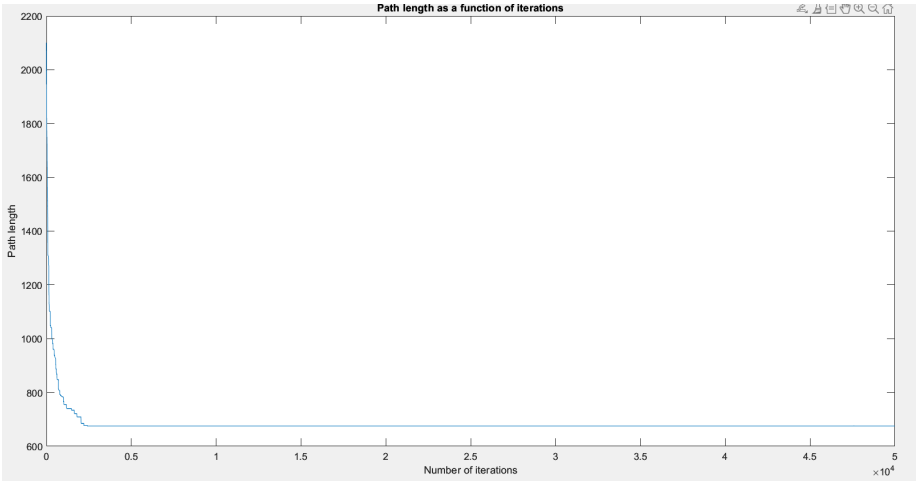


Figure 1

Exercise 2

Description Given a 2D bin of width W, allocate N rectangles of sizes $w_k x h_k$ so that the height of the packed rectangles is minimal.
Write a Matlab function that given an order of the rectangles to be packed and their orientation, puts them from left to right and as low as possible.
Implement a simple algorithm that randomly swaps the order of two rectangles and (randomly changes their orientation (the updated is accepted if it reduced the objective function value). Visualize the process. **Functionality** Firstly the initial parameters of the rectangles, N and W are initialized and a random order of random rectangles are generated. The height and width of these rectangles is also randomized, both parameters lie between 1 and W. Then the packing and optimization is done.

- **Initial Packing:** The rectangles are initially packed using a algorithm that places each rectangle at the leftmost and lowest possible position within the bin. This process uses a function `packRectangles`, which updates a 'skyline' array representing the current height at each position along the bin's width.

- **Optimization:** After the initial packing, the arrangement of the rectangles is optimized through 1000 and 1000 iterations. During each iteration, two rectangles are randomly selected and swapped, and their orientation may also be flipped. Then the new configuration is packed and it is accepted if it does not increase the overall height of the packing.

Results Figures 2 and 3 show the original packing and the optimized packing for 1.000 and 10.000 iterations of optimizing, respectively. Figure 4 is a visualization of the optimization process showing how the total height lowers with each accepted configuration. Included in our code is an animation of the optimization process, just open the code and run it and it will pop up!

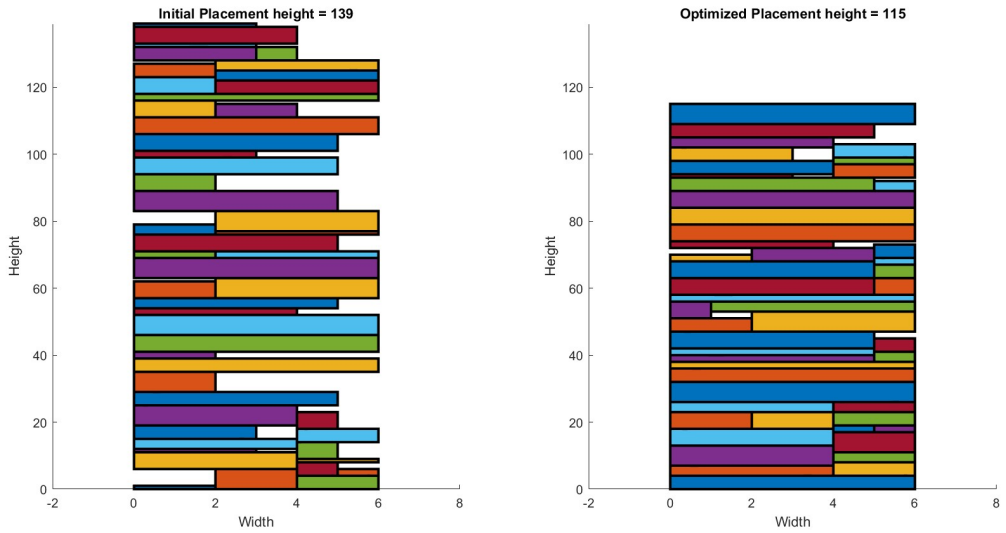


Figure 2: The optimization of N=50 rectangles with a width of W=6 with 1.000 iterations

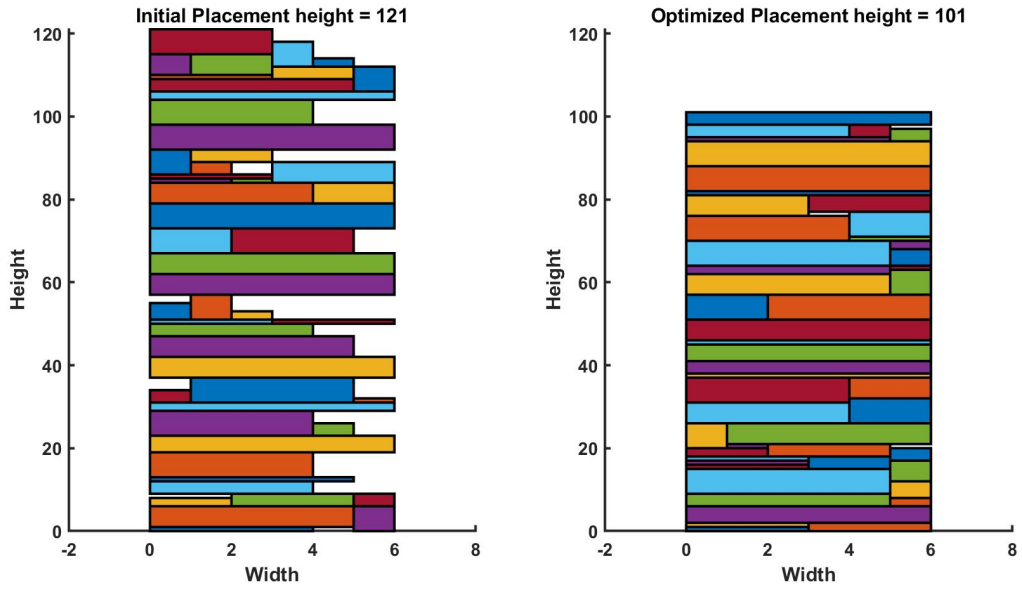


Figure 3: The optimization of N=50 rectangles with a width of W=6 with 10.000 iterations

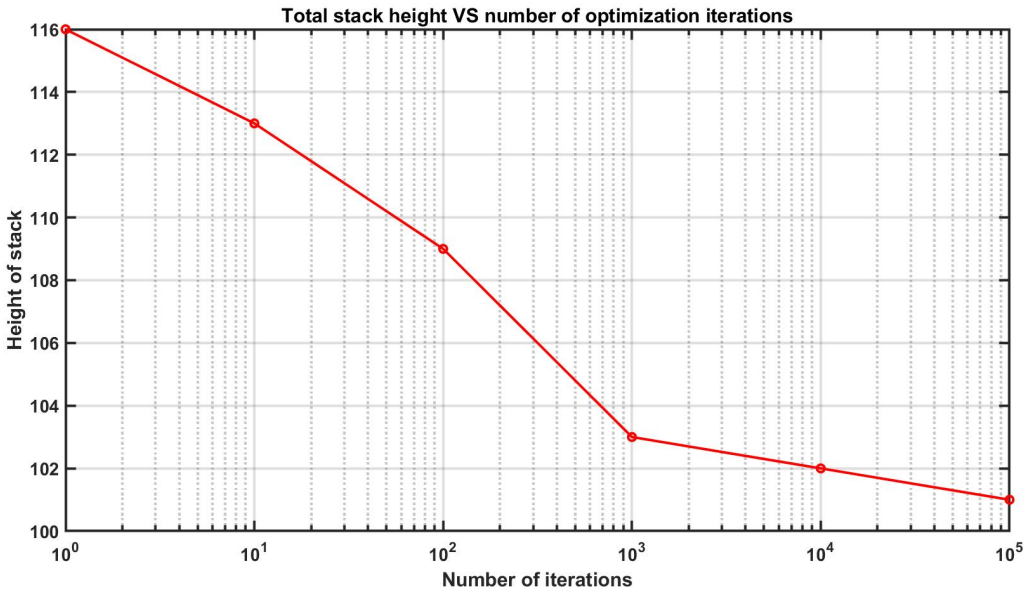


Figure 4: Graph of the height vs log(iterations) during the optimization of N=50 rectangles with a width of W=6 with 100.000 iterations