Part 2

Problem modeling:

The task was crate a model for moving cars in the parking hall. The hall has N lanes and each lane has M parking slots. There are exit and entering point at the begin and at the end of each lane. It is impossible change the lane inside of hall. You can move a car backward and forward in the same lane and you can pull a car out of the hall and replace it in the other lane. The prices of moving a car are following:

|  |  |
| --- | --- |
| Forward in the same lane | 1 |
| Backward in the same lane | 2 |
| Take a car out and add the car at the begin of a lane | 3 |
| Take a car out and add the car at the end of a lane | 4 |

There wasn’t mention about the price about moving a car to empty lane, because it is at the same time the first and the last car in the lane. So we assumed the price be 3. So the task was create an algorithm which solve the problem moving one car set-up to other. The implemented algorithm is A\* which is a heuristic algorithm. Heuristic algorithms don’t check all the possible states instead it uses heuristic function to estimate the states and picks the best one. We have to create admissible function which means the function always return better or equal price than the best real solution. Our heuristic function checks every car and calculates their minimum price to move the goal location assuming that you can move the car throw the other cars and return the a sum of the prices. Algorithm creates states moving each car separately and creates all legal where you can move a car. After that it calculates heuristic price to each one and add the price which has payed to come current state. The sum of these is total cost and algorithm chooses the cheapest one and calculates the child states. Each time we have moved space to space algorithm keeps track the prices and moved cars when we have found the possible solution we have an information about our path.

Model implementation:

The used programing language was python, because it was familiar both of us. We know that python is not the most effective language to write algorithms and for example c++ would have been better choice. We used python standard packages and our choice to implement open list was heap. Heap is a list container which provides fast access to its the smallest or highest value in constant time. Any way we had to always also erase the smallest value so we paid logarithm time, and also the adding of a new value is a logarithm action. The benefit was that we didn’t need to keep the whole list in order, because we needed only access to the smallest element.

Time complexity:

This problems belongs definitely in np category because by increasing the dimension of parking hall or amount of cars consumed time increases exponentially. But the most important fact is the difficulty of getting to goal state. For example if one of your cars is blocked and you have to move the car next to it from its correct location it cause many extra expansions because after the first expansion every state is worse than the initial. So when we inspect time consuming we should look into relationship between time and expansions. The better way to implement the open list in this case would have been bucket/radix sort, because the sum of heuristic function and current cost was always integer. If it’s not heap is usually the best choice. Our test data is showed in picture 1 below.

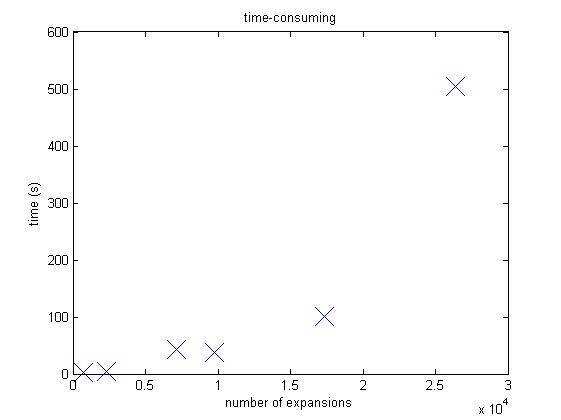


Figure Time consuming

At the first look, we can say that the algorithm is crowing exponentially.