

**Master Studies**

**Final Project. Delivery Logistics**

Field of Study: Advanced Analytics-Big Data

Advanced Simulation Modelling

|  |  |
| --- | --- |
| **Student names:** | **Album nr:** |
| Oleksandr Bachynsky | 83457 |
| Oleksandr Romanchenko | 83459 |
| Nazarii Melnyk | 83471 |

----------------------------------------

Instructor: Anna Szczurek

----------------------------------------

Warsaw, 2020

**Contents**

[**1.** **Introduction** 3](#_Toc41744420)

[**2.** **Problem statement** 5](#_Toc41744421)

[**3.** **Key Findings** 11](#_Toc41744422)

[**4.** **Sensitivity analysis** 14](#_Toc41744423)

[**5.** **Conclusions and recommendations** 18](#_Toc41744424)

[**6.** **References** 19](#_Toc41744425)

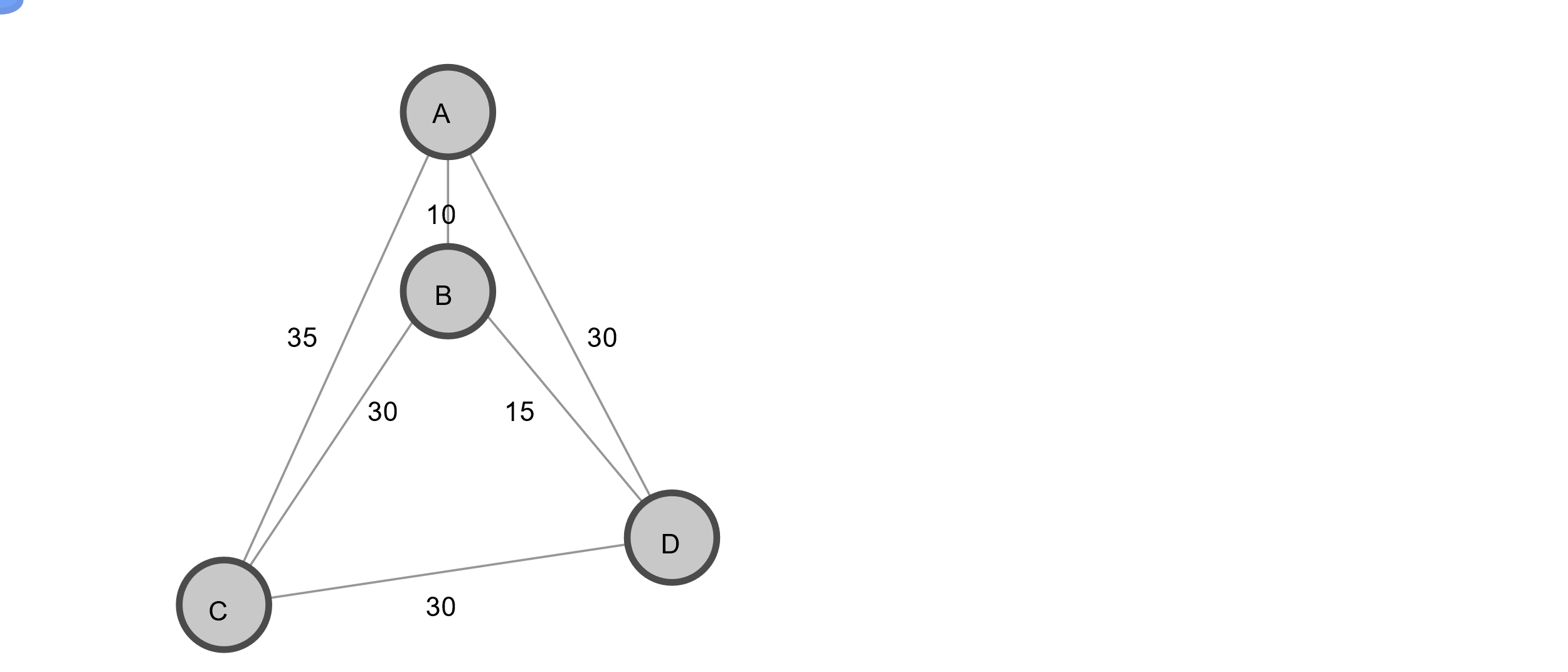
# **Introduction**

1. **The travelling salesman problem (TSP)**

The **travelling salesman problem (TSP)** is a classical dilemma used for testing the effectiveness of various optimization algorithms. The problem is stated as: *"Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?"*

The problem was first formulated in 1930 and is one of the most intensively studied problems in optimization. It is used as a benchmark for many optimization methods. Even though the problem is computationally difficult, many heuristics and exact algorithms are known, so that some instances with tens of thousands of cities can be solved completely and even problems with millions of cities can be approximated within a small fraction of 1%.

A TSP can be represented by a graph, in which the nodes correspond to the locations, and the edges (or arcs) denote direct travel between locations. For example, the graph below shows a TSP with just four locations, labelled A, B, C, and D. The distance between any two locations is given by the number next to the edge joining them. By calculating the distances of all possible routes, you can see that the shortest route is ACDBA, for which the total distance is 35 + 30 + 15 + 10 = 90.



*Figure 1.1. Travelling salesman problem.*

The TSP has several applications even in its purest formulation, such as planning, logistics, and the manufacture of microchips. Slightly modified, it appears as a sub-problem in many areas, such as DNA sequencing. In these applications, the concept city represents, for example, customers, soldering points, or DNA fragments, and the concept distance represents travelling times or cost, or a similarity measure between DNA fragments. The TSP also appears in astronomy, as astronomers observing many sources will want to minimize the time spent moving the telescope between the sources. In many applications, additional constraints such as limited resources or time windows may be imposed.

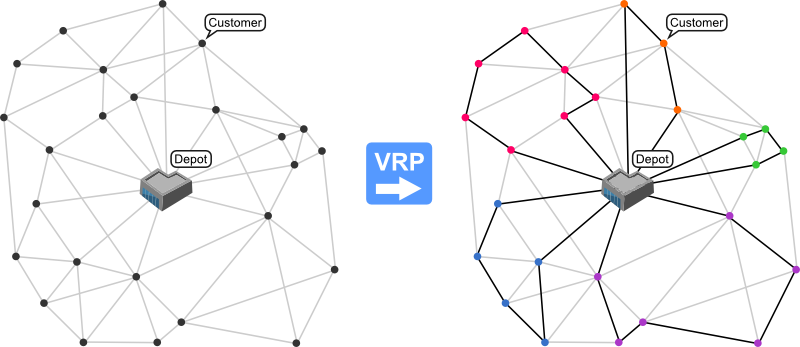
1. **The vehicle routing problem (VRP)**

A more general and complex version of the TSP is the **vehicle routing problem (VRP),** in which there are multiple vehicles. In most cases, VRPs have constraints: for example, vehicles might have capacities for the maximum weight or volume of items they can carry, or drivers might be required to visit locations during specified time windows requested by customers.

It first appeared in a paper by George Dantzig and John Ramser in 1959, in which first algorithmic approach was written and was applied to petrol deliveries. Often, the context is that of delivering goods located at a central depot to customers who have placed orders for such goods. The objective function of a VRP can be very different depending on the particular application of the result but a few of the more common objectives are:

* Minimize the global transportation cost based on the global distance travelled as well as the fixed costs associated with the used vehicles and drivers
* Minimize the number of vehicles needed to serve all customers
* Least variation in travel time and vehicle load
* Minimize penalties for low quality service

Vehicle routing problems are inherently intractable: the length of time it takes to solve them grows exponentially with the size of the problem. For sufficiently large problems, it could take years for any routing software to find the optimal solution.



*Figure 1.2. An instance of a VRP (left) and its solution (right)*

# **Problem statement**

As was mentioned above, there are several approaches to the VRP. Since we want to make this project as close to real life as possible, we will try to find an optimal route solution for a delivery company by combining elements from various models. In particular, we will use elements from the following models:

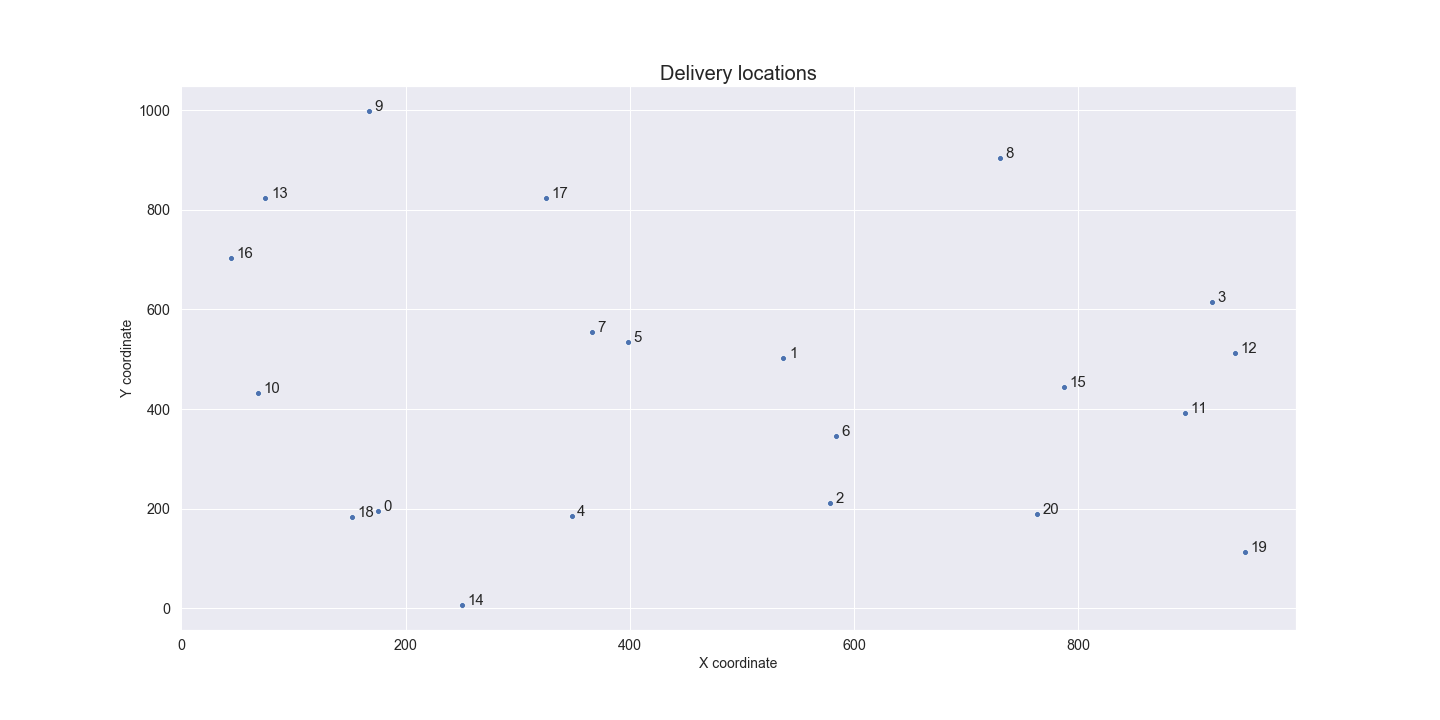
* capacitated vehicle routing problem (CVRP)
* vehicle routing with pickups and deliveries
* vehicle routing with time windows (VRPTWs)

Let`s go through the elements separately.

1. **Basic Goal of the VRP**

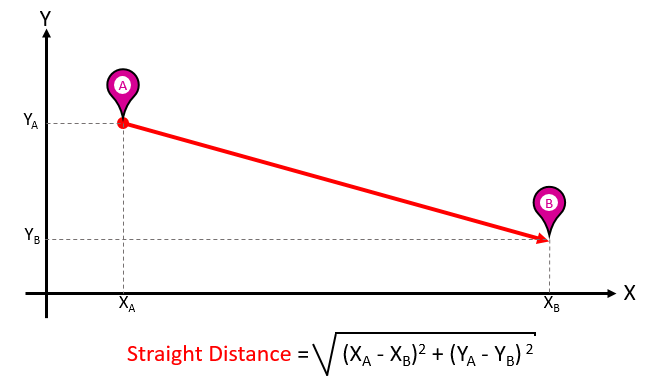
In general, the optimal solution for a VRP problem is defined as a solution for all elements simultaneously, not each element separately. Typically, the goal is to minimize the length of the longest single route among all vehicles. Since we are planning to implement time delays on our routes, we are going to minimise the total delivery time among all cars.

In order to do so, we will first determine the number of delivery locations, as well as their coordinates. The figure below shows the city grid with marked locations. The Depot is marked as 0 and is located in the southwest part of the city. As can be seen from the figure, the location coordinates are generated randomly - we have locations with close proximity as well as locations that are on the other side of the town.



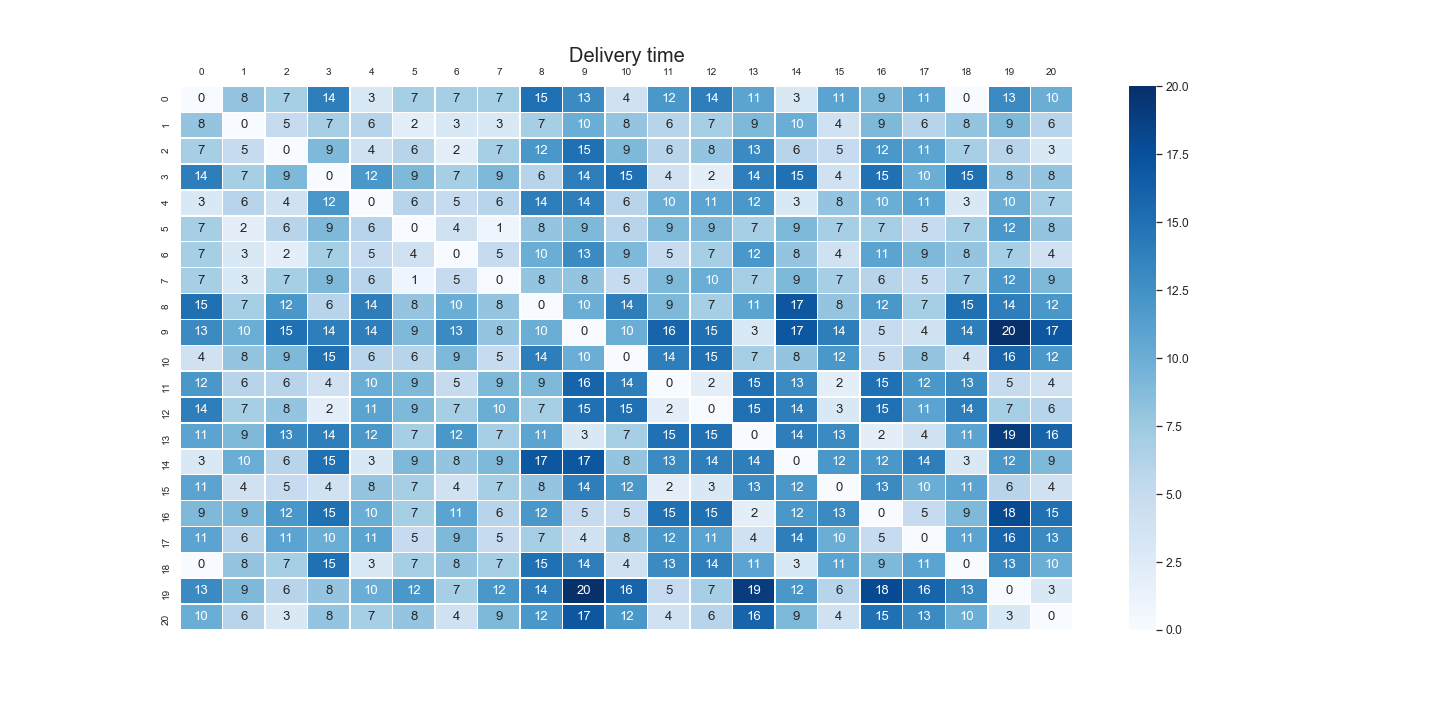
*Figure 2.1. Delivery locations.*

The next step is to determine the minimum time needed to reach each location. We are assuming that each location is connected via a straight line. In order to calculate the distance, we are using the Euclidian distance which is shown in the figure below.



*Figure 2.2. Example of calculating the Euclidian distance.*

Assuming that each car travels with the speed of 60 km/h, we can calculate the minimum time needed to reach each location. The heatmap below shows these time requirements. In general, a delivery between nearby locations requires a couple of minutes, while for distant locations the delivery can last up to 20 minutes (for example, between locations 9 and 19).

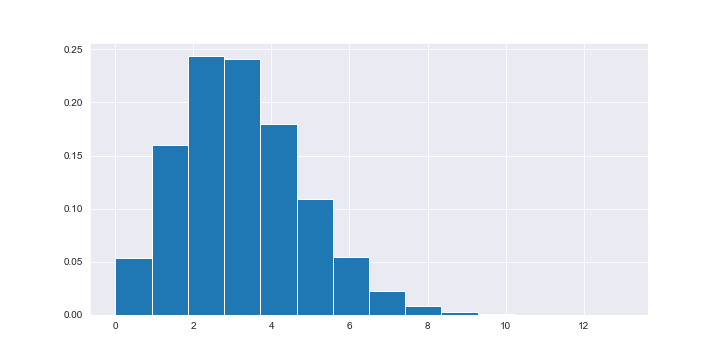


*Figure 2.3. Delivery times.*

1. **Capacity constraints**

The capacitated vehicle routing problem (CVRP) is a VRP in which vehicles with limited carrying capacity need to pick up or deliver items at various locations. The items have a quantity, such as weight or volume, and the vehicles have a maximum capacity that they can carry. The problem is to pick up or deliver the items for the least cost, while never exceeding the capacity of the vehicles.

In our model we are assuming that each week the locations place orders on a delivery (measured in tonnes). The delivery volumes follow a Poisson distribution – the average delivery size is between 2 and 5 tonnes, but sometimes large orders can occur. These orders are then delivered by the cars given their weight limit – 15 tonnes.

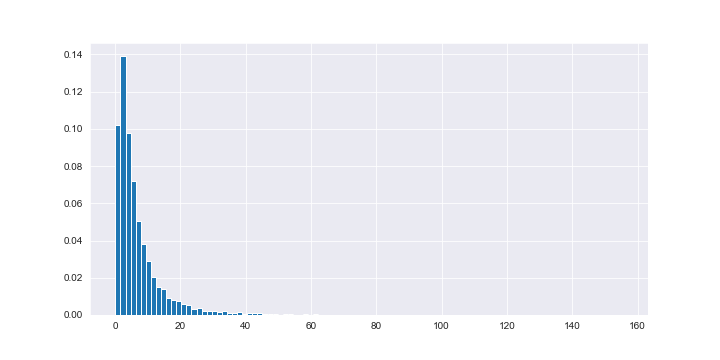


*Figure 2.4. Delivery size distribution.*

1. **Time constraints**

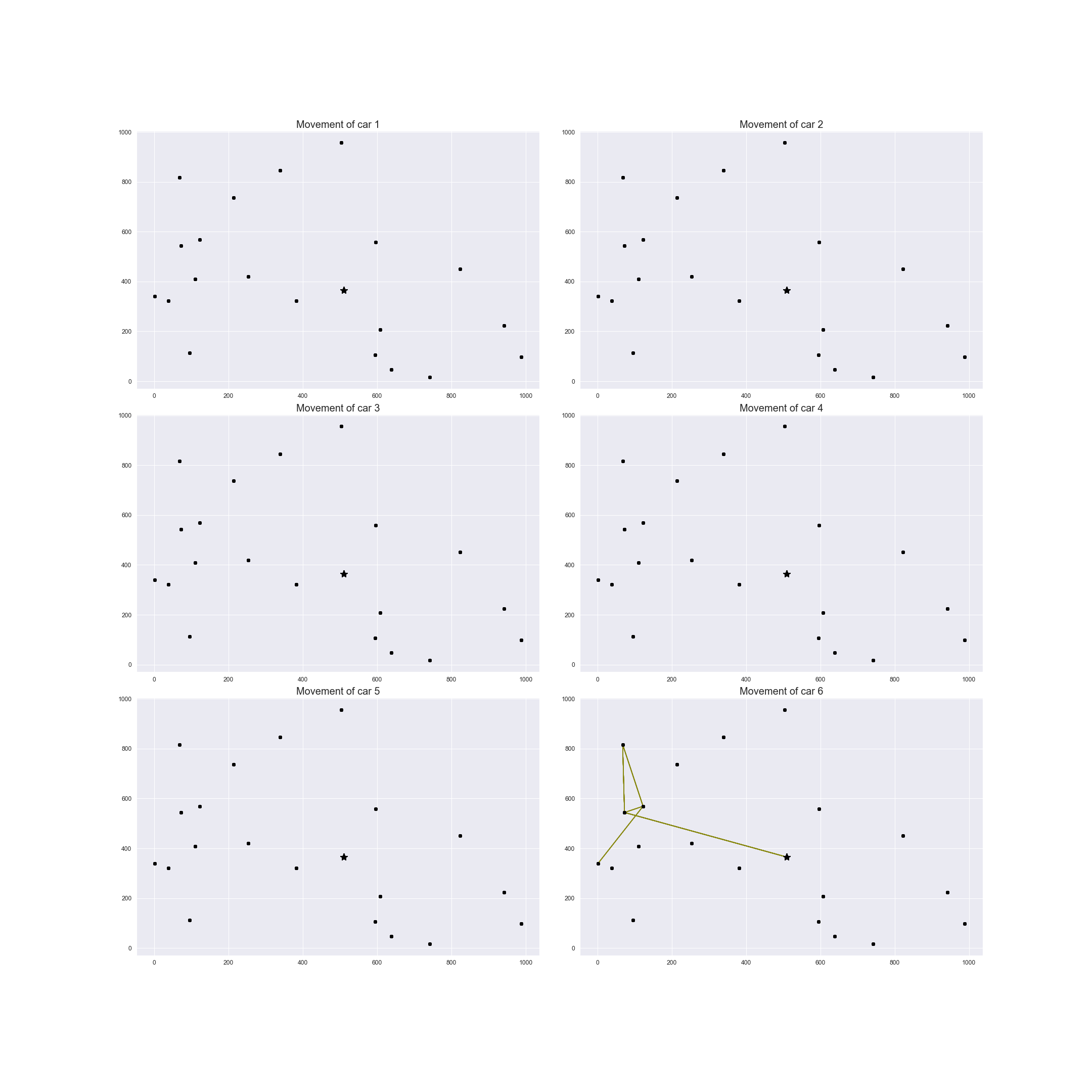
Many vehicle routing problems involve scheduling visits to customers who are only available during specific time windows. These problems are known as vehicle routing problems with time windows (VRPTWs). Similarly, to the delivery volumes, the locations are defining certain time windows during which they can accept a delivery. This way the company must plan a route that will allow the car to reach the location at a specified time.

Another issue influencing the success of the delivery is the time required to take the route. The delivery times, as shown in figure 3 above can change due to various events – for example, car accidents or detours. Such events increase the route time by some margin – normally from 0 minutes to 1 hour. Such time increase follows a lognormal distribution – typically the accidents are minor and do not cause a large delay, but sometimes serious accidents cause large traffic jams.



*Figure 2.5. Distribution of time delays due to car accidents.*

If the accident occurs, it can affect one or many routes. Based on the available information, the car will then readjust its trajectory by selecting the optimal route. In the example below, the car was supposed to travel through 4 locations – A, B, C and D. But due to an accident on route BC, the car was forced to change its trajectory by visiting location E.



Car accident occurred

A

C

B

D

E

*Figure 2.6. Detour taken by car 6.*

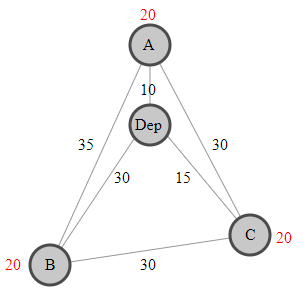
Apart from constraints that apply during vehicle travel, we will also apply constraints at the depot: all vehicles need to be loaded before departing the depot and unloaded upon return. Since there are only two available loading docks, at most two vehicles can be loaded or unloaded at the same time. As a result, some vehicles must wait for others to be loaded, delaying their departure from the depot.

1. **Penalties and Dropping Visits**

Given all constraints described above, there could be no feasible solution present. For example, if you are given a VRP with capacity constraints in which the total demand at all locations exceeds the total capacity of the vehicles, no solution is possible. In order to address such issue, we will allow the vehicles to drop visits to some locations. The problem is how to decide which visits to drop.

To solve the problem, we introduce new costs—called penalties—at all locations. Whenever a visit to a location is dropped, the penalty is added to the total time travelled. The solver then finds a route that minimizes the total time plus the sum of the penalties for all dropped locations.

As an example, consider the simple VRP with capacity constraints given by the graph below, in which the red numbers next to the three locations (other than the depot) are demands.



*Figure 2.7. Delivery size distribution.*

Suppose there is just one vehicle with capacity 50. It can't visit all three locations, A, B, and C, because the total demand is 60. To solve the problem, you assign a large penalty—say 100—to each location. After detecting that the problem is infeasible, the solver drops location B. Then the shortest route will be: “Depot → A → C → Depot”, which visits two of the three locations (the distance is 55).

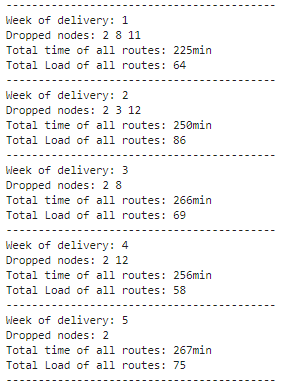
In our case we are imposing a penalty of 20 minutes. As a result, after dropping one location to make the problem feasible, the solver doesn't drop any additional locations, because the penalty for doing so would exceed any further reduction in travel time. Assuming the company wants to make as many deliveries as possible, this gives a satisfactory solution to the problem.

To sum it up, the model will have the following properties:

* A company operates within a city of size of 1000\*1000 km.
* The company has one depot located in the city and *n* number of delivery locations. Each location has a set of coordinates and is connected with all other locations.
* The company has *m* cars travelling throughout the city. Each car has a certain carry capacity. The average speed of each car is 60 km/h.
* The company has to deliver a certain amount of goods to each location at a specified time window.
* If a delivery is not possible to a certain location, the company can refuse a delivery but incurs a time penalty.
* The depot can only accept a limited number of cars at a time.
* Loading/unloading goods requires time.
* Car accidents can occur on routes, causing a delay in deliveries. Such accidents occur randomly and can affect any number of roads.

# **Key Findings**

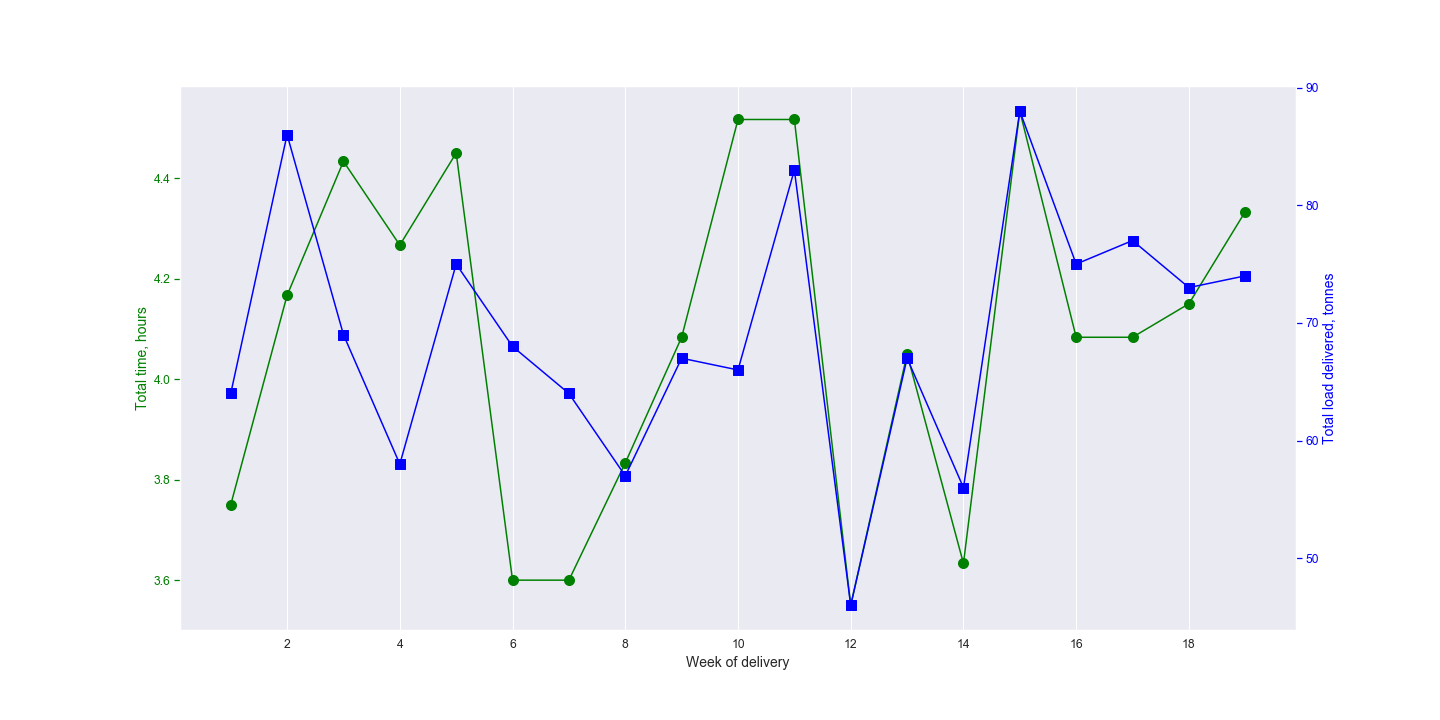
After running the algorithm, we obtain information regarding the total time needed to deliver the goods, the total load of the delivery and the locations excluded due to certain issues – either the order was too big, the time window was not suitable or there was some accident on the road leading to that location.



*Figure 3.1. Example output of first 5 weeks.*

After analysing the excluded nodes, we can conclude that out of 21 exclusion cases, 14 (66%) were at location 2. This means that there is an issue with reaching that location and the advice to the company would be to reconsider the feasibility of servicing it.

The plot below shows the total time and total delivered load for each of the weeks. We can see serious fluctuations in time (from 3.6 to 4.5 hours) and delivery volumes (from 46 to 88 tonnes). The time differences can be explained by two things – available time windows and accidents on the road. They both affect the total delivery time. As for the delivery volumes, they are affected by the placed order, car capacity and feasibility of a delivery to a certain location.



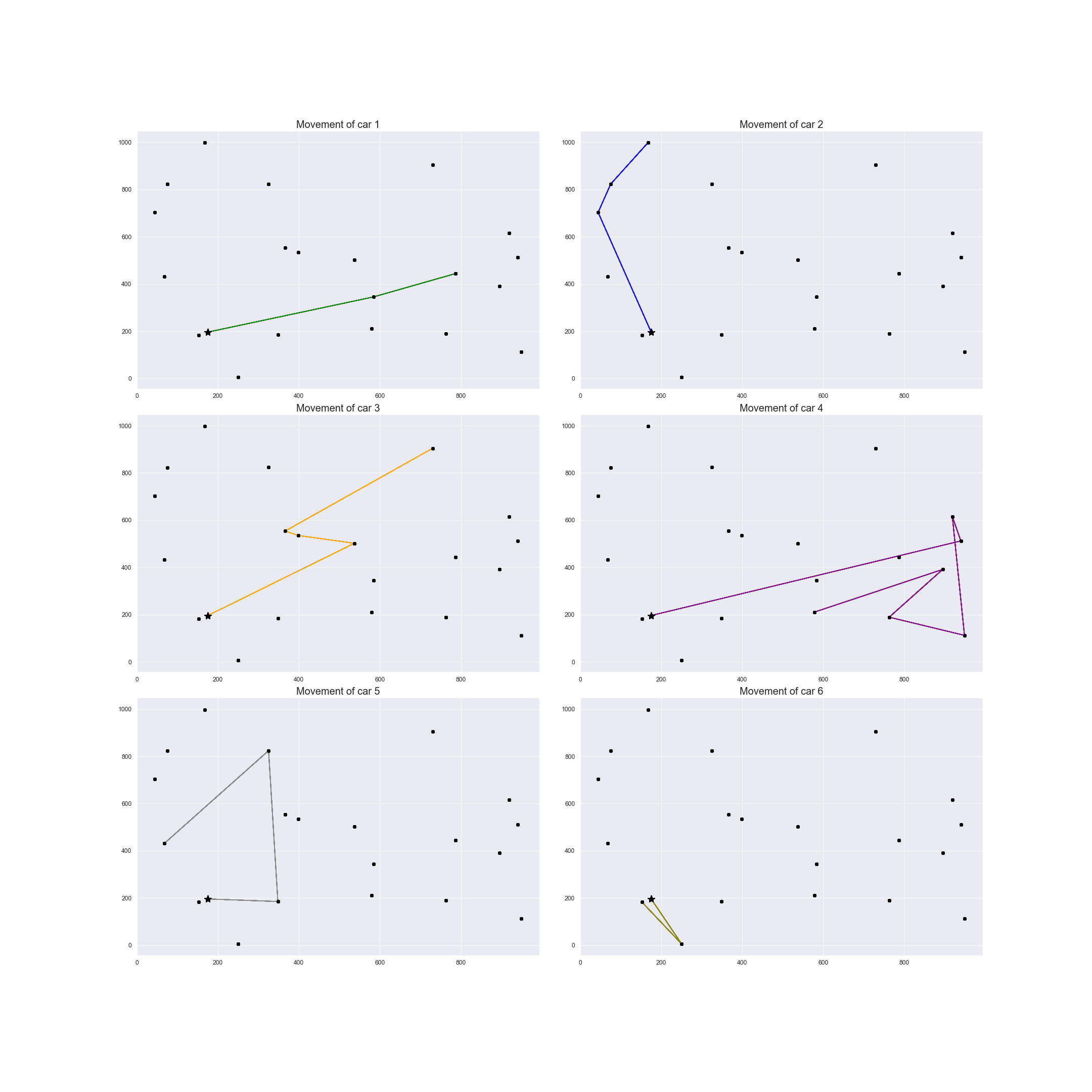
*Figure 3.2. Delivery time and tonnage.*

We can also analyse the car travel path. Figures 3.3 and 3.4 show movement of all 6 cars during week 16, as well as the time and loads of each delivery. Each car starts its route from the Depot which is marked with a star. During their travel, each car visits between 2 and 6 locations. Such difference can be explained by the load delivered by each car – some cars deliver small loads and can visit a lot of locations, while others have to deliver large orders.

At the same time, we can see that their path patterns also differ. While car 1 travels in a straight line, car 4 has to return to certain locations. Even though this is not an optimal configuration for the car, it is the best solution for the company due to available time windows – instead of waiting for a location to accept a delivery, the car simply circles around to deliver orders to other locations.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Car id | Destination id | Arrival time | Departure time | Load capacity |
| 1 | 0 | 0 | 0 | 3 |
| 6 | 7 | 7 | 6 |
| 15 | 11 | 11 | 11 |
| 2 | 0 | 0 | 0 | 3 |
| 16 | 9 | 10 | 7 |
| 13 | 19 | 19 | 10 |
| 9 | 25 | 25 | 13 |
| 3 | 0 | 0 | 0 | 3 |
| 8 | 15 | 19 | 4 |
| 1 | 34 | 43 | 5 |
| 7 | 46 | 46 | 7 |
| 5 | 47 | 47 | 14 |
| 4 | 0 | 0 | 0 | 3 |
| 11 | 12 | 13 | 5 |
| 12 | 14 | 17 | 5 |
| 3 | 16 | 19 | 7 |
| 19 | 31 | 36 | 11 |
| 20 | 34 | 39 | 11 |
| 2 | 56 | 56 | 15 |
| 5 | 0 | 5 | 5 | 3 |
| 4 | 14 | 24 | 5 |
| 17 | 16 | 26 | 7 |
| 10 | 35 | 35 | 11 |
| 6 | 0 | 5 | 5 | 3 |
| 14 | 48 | 51 | 7 |
| 18 | 54 | 54 | 11 |

*Figure 3.3. Locations, loads and delivery times of each car during week 16.*



*Figure 3.4. Car movement trajectory during week 16.*

# **Sensitivity analysis**

In this section we are going to have a look at how situation changes if some key simulation parameters are changed.

There are several parameters that can be changed:

* Number of locations
* Number of cars and their capacity
* Load/unload times
* Depot capacity

Increasing the number of locations would simply result in problems with converging the optimisation algorithm. More locations would increase delivery times and loads. In order to minimise the time, the algorithm would drop certain deliveries, but this would increase time (due to penalties). As a result, the algorithm would struggle with finding a proper solutions and sometimes would drop all locations as this would reduce time.

Instead, we will focus on the other three parameters. In our case we will use the following starting values: location\_number=21, car\_park=6, car\_capacity=[15, 15, 15, 15, 15, 15], load\_time=5, unload\_time=5, depot\_capacity=4.

1. **Experiment 1.**

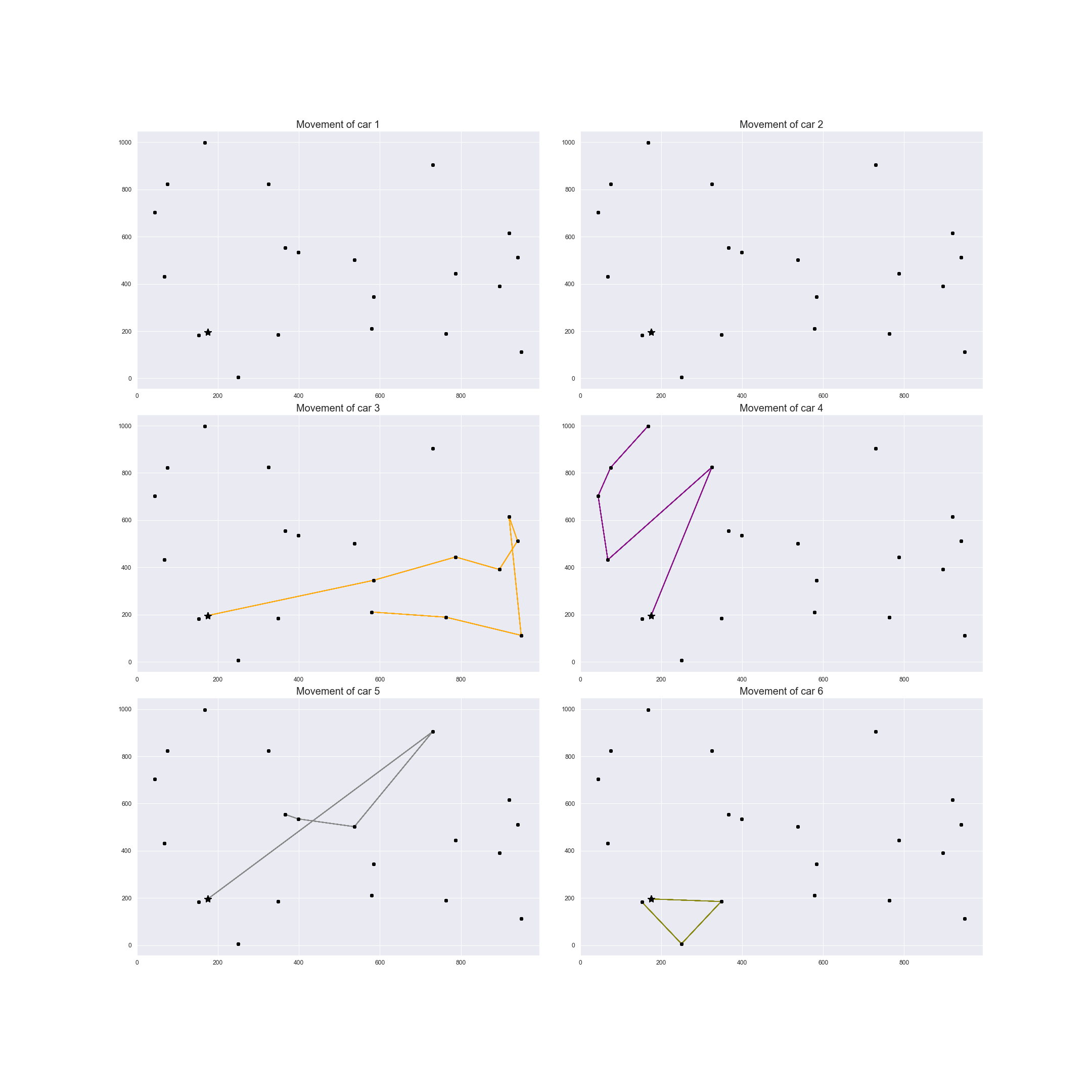
**Changes:** increasing car\_capacity of all 6 cars to 35

**Assumption:** since the same workload will be distributed between cars which have more than twice of initial capacity this would result in shorter routes for some cars or even unused cars.

A group of people in a room

Description automatically generated

*Figure 4.1. Delivery time and tonnage. Experiment 1*

 *Figure 4.2. Car movement trajectory during week 16. Experiment 1.*

As expected, having cars which are twice as capable as initials ones leads to a situation that cars 1 and 2 weren’t used at all because 4 other cars have managed to deal with the workload by themselves!

1. **Experiment 2.**

**Changes:** decreasing car capacity to 10 for all cars, decreasing load/unload time to 2 and increasing depot capacity to 12. In such a case we have light-weight cars which need less time for load/unload + increased capacity of depo which should at least solve this issue with depo queues

A picture containing indoor, table, computer, desk

Description automatically generated

*Figure 4.3. Delivery time and tonnage. Experiment 2*

A close up of a map

Description automatically generated

*Figure 4.4. Car movement trajectory during week 16. Experiment 2.*

Here, looking at delivery time and load capacity chart we can see that the total number of tones delivered is significantly lower than in Experiment 1 while the time spent is a bit higher with all cars being busy all the time.

1. **Experiment 3.**

**Changes:** as in Experiment 1 we are going to increase a capacity of all cars to 35 but now decrease the number of cars from 6 to 4. Thus, we can confirm that 2 cars are redundant in case of heavy-weight cars. Let’s see if cars will manage to deal with a workload.

A picture containing indoor, table, computer, desk

Description automatically generated

*Figure 4.5. Delivery time and tonnage. Experiment 3*

A picture containing text, computer

Description automatically generated

*Figure 4.6. Car movement trajectory during week 16. Experiment 3.*

Here we can see that Time and load capacity plot is almost the same as in Experiment 1 with the same fluctuation range of tons delivered but with 4 cars.

# **Conclusions and recommendations**

In our simulation we have tried different parameters: car capacity, load/unload time, depo capacity. The parameter we didn’t try to play with is the number of locations (was set to 21 in all experiments). We left it for future analysis because it would be much harder to concentrate on optimization of car capacity, car number and depo capacity given different number of locations (which can change perception of the problem completely).

**Note, that since we didn’t try changing number of locations, our conclusions are true only for case with constant number of delivery locations. For cases where this number is dynamic the inference may be different.**

Looking at the results we can make a recommendation to any company having similar logistics problem with heavy weight transportation involved: it’s better to have few heavy-weight cars instead of many light-weight cars for a constant number of locations. The main reason is that the heavy-weight cars have the same workload as lightweights. Also, from the business point of view it’s much more convenient to maintain a working condition with fewer number of cars – reduced fuel consumption and fewer number of car drivers.

On the other hand, if the company has to deliver goods to many locations in a short period of time, they can face the same issue as mentioned in the sensitivity analysis – the cars aren`t able to visit all locations and some are dropped. In this case it is reasonable to increase the number of cars as then multiple routes can be planned.

As for further improvements to the model, the following can be implemented to make the model more close to reality:

1. **In our model we assume that all cars travel with the same speed.** However, depending on the type and carry capacity, their speeds may differ. One way to deal with this is to include a coefficient decreasing the speed of trucks with higher capacity so that the model represents the reality better.
2. **In our case we are assuming that any location may be reach from anywhere.** In a city it is sometimes impossible due to the city’s topography (e.g. a bridge limits possible routes over the river). Further improvements will be required to address this.
3. Similarly to 2), sometimes there are several routes that can be used to reach a certain location. So if there is a constraint on one road, another can be used. This should also be accounted for. In the model

# **References**

1. https://www.researchgate.net/publication/313005083\_Vehicle\_routing\_problem\_Models\_and\_solutions
2. https://www.optaplanner.org/learn/useCases/vehicleRoutingProblem.html
3. https://www.mdpi.com/2073-8994/11/4/546/htm
4. https://pl.wikipedia.org/wiki/Problem\_marszrutyzacji
5. https://developers.google.com/optimization
6. https://pypi.org/project/ortools/
7. https://github.com/google/or-tools/issues/1038
8. https://github.com/google/or-tools/issues/1291
9. https://github.com/google/or-tools/issues/78