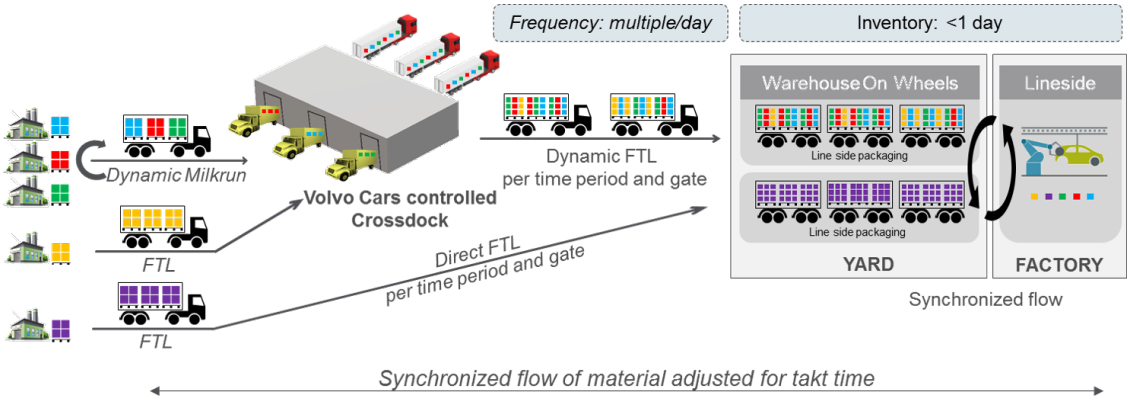
CHALMERS
UNIVERSITY OF TECHNOLOGY



AI & ML in warehousing and milk-run problem

Implementation of reinforcement learning and supervised learning

Systems, Control and Mechatronics

Huanyu Yu
Yanda Shen

Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Master's thesis 2019:NN

AI and ML in logistics

Implementation in warehousing and milk-run problem

Huanyu Yu, Yanda Shen

CHALMERS
UNIVERSITY OF TECHNOLOGY

Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2019

AI and ML in logistics
Implementation in warehousing and milk-run problem
Huanyu Yu
Yanda Shen

Supervisor: Ericsson Niclas, M&L department, Volvo Cars
Examiner: Balázs Adam Kulcsár, Chalmers Technology University

Master's Thesis Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Printed by Chalmers Reproservice
Gothenburg, Sweden 2019

AI & ML in logistics
Implementation in warehousing and milk-run problem
Huanyu Yu, Yanda Shen
Electrical Engineering
Chalmers University of Technology

Abstract

Nowadays, Artificial Intelligence(AI) and Machine Learning(ML) is a growing technique in lots of areas, which are simplifying and shaping the way we live. At the meantime, a fast-increasing product ordering and subscription needs to be secured in logistics. However, there is few studies and literature on implementation of AI and ML in the field of logistics management. It is possible and has potential to improve the performance in this field by applying AI and ML. For example, they can be implemented in forecasting and balancing warehousing problems, optimizing the transport routes or handling supplier selection scenarios. All of them can improve the performance in fast-increasing product ordering and subscriptions.
The main goal of this thesis work is to investigate the possibility and implement AI and ML approaches in automotive logistics. A model of logistical system was built for further investigation. In our logistics model, two factories were receiving orders and processing products by using the supplies both from the Cross dock(X-doc) and from suppliers directly. The X-doc was replenished by a truck running around and collecting supplies from several different suppliers. Data prediction, which is supplies consumption and replenishment in warehousing problem, was solved by applying Recurrent Neural Network(RNN) and regression analysis methods. The storage optimization in factories was solved by implementing an approach called Reinforcement Learning(RL). The selection of suppliers and path optimisation were solved with the help of RL as well, which was compared with several stochastic optimisation algorithms. It is figured out there is potential to using ML in logistics. Also ML gave acceptable and reasonable results, but its performance can be improved surely. Although the stochastic methods gave a better results in some occasions, compared with ML approaches, and ML cost more time to find the optimal, ML was very useful to handle with high volume, unstructured and complex data.
This thesis work has been an basic idea and inspiration of investigation and implementation on AI and ML at M&L department at Volvo Cars. In order to solve this project more efficient, it has been divided into two parts. The first part was to handle the balance in warehousing management and prediction the shipment of X-doc. Recurrent Neural Network(RNN), a supervised learning method, has been used for solving the prediction problem in factories shipment. And a RL, similar to Jack's Car Rental(JCR) case, has been applied to solve the warehousing management. The other was to select suppliers according to the parts reserves in the X-doc. Then the optimal route among them was found and generated by implementing optimization methods and ML.
Keywords: Logistics, machine learning, artificial intelligence, Travelling Salesman Problem (TSP), Loading Problem, Reinforcement Learning (RL), Markov Decision Process (MDP), Dynamic Programming (DP) and Jack's Car Rental (JCR) problem.

Acknowledgements

I would like to express my very great appreciation to Volvo Cars and Niclas Ericsson, one of the managers in $M\&L$ department at Volvo Cars, for giving this opportunity to work with his team. And I also am appreciate the help, valuable information, the Volvo Cars has shared. The IT department are very kind to give us supports on both hardware and software as well. I would also like to extend my thanks to Balazs Adam Kulcsár who has acted as our supervisor, as well as our examiner, at CTH. He provided important suggestions and advices when we met difficult problems. Finally I would like to thank our family and friecnds for their support during the thesis work.

Huanyu Yu and Yanda Shen, Gothenburg, February 2019

Contents

List of Figures

List of Tables

1
Introduction

This master thesis is carried out at Volvo Cars Corporation to figure out if it is possible to apply Artificial Intelligence(AI) & Machine Learning(ML) in logistics. This chapter begins with the background information of Volvo Cars company and literature studies about logistics and ML, followed by aim and purpose. In order to get main and sub research questions, the problems and limitations of warehousing management and milk-run system are described and analyzed. After that, ML methods and optimisation approaches for solving the problems above were presented. Finally, the thesis ended with a brief conclusion of comparison between basic optimisation methods and ML approaches. And future work for further studies was stated as well.

1.1 Background

Volvo Cars is a global car manufacture, they have different kinds of factories in Sweden, China, Belgium and USA. So they have to face to the transportation of large quantities of raw materials, half-finished products, parts and complete cars everyday which could spend a lot of time and cost huge money. How to improve the automotive production and logistics to make sure the transportation in Volvo Cars could handle the increased orders in the future is already a challenge for them.

The Manufacturing and Logistics department (M&L) is responsible to solve the problems caused by the fast-increasing product orderings · They are dedicated to improve and digitalize the current Manufacturing and Logistics processes. Their goals are aligned at highest management level and the “beating heart” of M&L is seen as one of the most important foundations to secure the fast-increasing product ordering and subscriptions.

In the M&L department, they have a new idea about the connecting of the suppliers and factories is called Milk-run. The main idea of the Milk-run is replacing the traditional warehouse by the the containers with wheels, so the idea could also be called as "Warehouse on Wheels". The trucks will travel between the several suppliers and a X-doc (Cross Dock), consolidator stations, for collecting supplies. Then the parts needed by the factories will be transited both from X-doc and suppliers. The mode that parts transported directly from suppliers is called Full Truck load (FTL), the quantities of parts for FTL will normally less than the consumption in factories. Then the remaining parts needed for the factories will be transported from X-doc.

So there are several types of parts will be transited from X-doc to the factories at once to reduce the cost. The idea of Milk-run is shown in figure 1.1 as below.

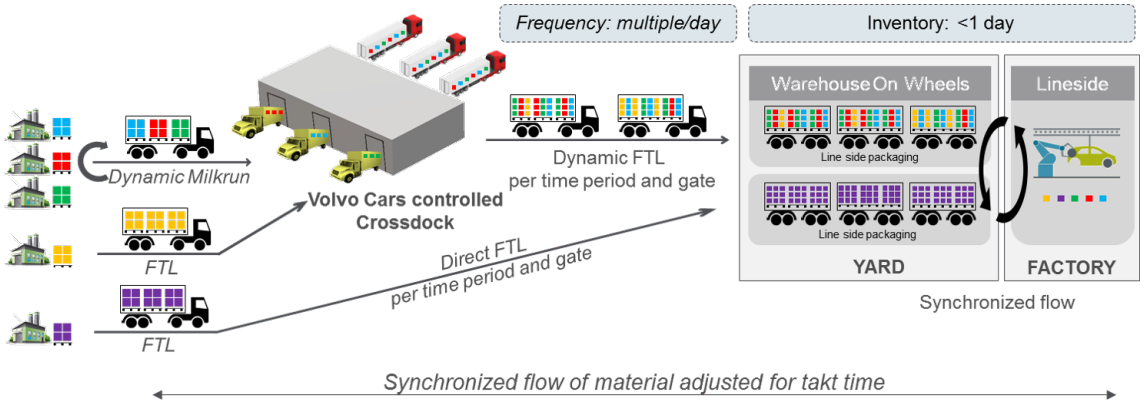


Figure 1.1: Planned 'milkrun' setup

1.2 Aim and Purpose

The initial aim of this thesis work is to find how could AI and ML be used in the field of logistics. In order to make this thesis work more specific, but could be widely referred for the ML department, the problem is related to the Milk-run and separated into two parts. The first problem focuses on warehousing management in factories which includes: prediction problem of parts consumption and replenishment, and optimization problem of transit between the X-doc and factories. The second problem focus on the optimization problem of loading parts for a truck and route planning problem among suppliers and X-doc.

1.3 Problem Description

Volvo Cars company has two factories, one is located in Torslanda, Sweden, and the other one is in Ghent, Belgium. A consolidator, it is called Cross-dock(X-doc), is located in East Europe. Both of the factories have their 'warehouse on wheels' in their yard. It found that it is better if they can use the storage in X-doc more than the warehouse in factories. The factories send Call-off contracts to their suppliers which is a purchase order that enables bulk orders last for months.

Trucks tavel among suppliers in order to pick up supplies for factories. If the shipment of one supplier takes up an entire truck by itself, it is called Full Truck-Load(FTL), the truck goes and delivers the supplies to the factory directly. At the same time, some vehicles go to suppliers one by one and collect supplies, then they return supplies to the X-doc. Finally, the X-doc decides which type of supplies should deliver to which factory, here to recall the 'Milk-run' process.

1.3.1 Management in factories

The problem for management in factories can also be separated into two areas which are prediction problem and optimization problem. The prediction problem includes using previous data to predict the future consumption and replenishment in factories. Time series prediction has been a significant problem in the machine learning field, because it is always related with big data and machine learning has been proved to have great advantage to deal with large quantity of data. So in this report, several machine learning methods will be used to predict the future consumption and replenishment in factories and compare between each other.

How many quantities of parts should be stored in the factories is always a problem for companies, since if there are too many or lack of parts stored in the factories could both be a problem. Too many parts might cost a lot of money for a company, however, lack of parts could effect the capacity of factories. So the second purpose for management in factories is to use machine learning method to optimize the quantity of parts in factories and how many parts should be transported from X-doc based on the result of the prediction problem.

1.3.2 Milk-run problem

The X-doc needs to be replenished, since supplies stored in it were sent to the factories, However, it can't be all filled due to the limitation of time, and the capacity of transport trucks. An assumption was made that only one truck is travelling around collecting parts from suppliers, hence only a few suppliers would be selected to be visited. The final goal for Milk-run problem is to optimal the path among the chosen suppliers. A Traveling Salesman Problem(TSP) is formed, since the truck with a limited capacity is required to collect supplies from several suppliers then deliver them to the other warehouse. The optimal solution of the path selection needs to be carried out by using existing optimal methods and machine learning. The pros and cons of these methods will be discussed.

1.4 Limitations

- Each supplier will only provide one type of part.
- There is only one truck traveling among suppliers collecting supplies.
- The thesis work is carried out by two students within 20 weeks.

1.5 Research Questions

How AI and available data set can be used to optimize the logistic process

- How many supplies should be hold in the factories?
- Which is the best method for solving route planning in Traveling Salesman Problem?

Comparing with the other deterministic optimization approaches, how the performance of reinforcement learning in TSP?

1.6 Thesis Setup

This thesis was carried out at the Manufacture and Logistics Digital department in Volvo Cars. Two students Huanyu Yu and Yanda Shen are paired to finish this thesis together. Yanda was mainly researching the implementation of machine learning on prediction, and application of reinforcement learning on warehouse management. The selection of suppliers and the travelling salesman problem was studied by Huanyu Yu.

2

Literature Study

In this chapter, the books, articles and papers used for this thesis project are briefly described and introduced. Starting with a basic introduction of logistics, followed by optimisation methods from previous study. In the end, different literature and papers about implementation of machine learning algorithms in logistics problems has been presented.

2.1 Introduction to Logistics

Logistics activity management, or supply chain management, can be a whole process that managing the flow of products from suppliers to final customers. The companies in supply chain act as the suppliers as well as the customers. Nowadays, a efficient and effective logistics management playing an important role in companies' future success and competitive advantages. There are different kinds of problems in logistics management, such as, Facility locations, Distribution systems (inventory distribution), Lot size (optimal replenishment strategy), Bin packing and Vehicle routing problems.[10] The logistics problems in our thesis are included in those problems, like products predicting, inventory management and optimizing the journey length. It is easy to figure it out that if any part in the process of logistics management is broken, the entire supply chain will stop. In order to avoid that situation, a warehouse, as a buffer, is needed to make the process smoother. However, the size of the buffer is a question mark. It can't be too large since it will cost a lot, and it won't play its role when the warehouse is too small. One problem in our thesis is to balance the storage of warehousing in order to minimize the cost. The other one is to optimize the journey routes, as known as Traveling Salesman Problem(TSP). A Bin packing-like problem is considered in order to load products in a limited space more efficiently. Besides that, time and path length are the key factors of an efficiency logistics during the transportation process. To meet customers' needs, the process of transportation service should not take too long. Normally, the path optimization problem can be solved by using optimization methods. But one goal of our thesis is to make attempts to solve these optimization problems by implementing machine learning algorithms instead of solving it by optimization approaches manually.

2.2 AI and ML in logistics

In recent years, more and more companies seize and are positioning to embrace AI in different aspects of supply chain. One of the reason is the growing unstructured data and complexity which is larger than ever. People can't handle these harmonious problems amid sensitive deadlines, high volumes and asset allocation. AI and ML can help logistics industries to predict and process autonomously.[16] There are few literature had done researches about the implementation of AI and ML in logistics. However, there are plenty of literature and studies on solving logistics problems by using optimization approaches. Several literature and papers of optimisation methods and machine learning in logistics have been presented below.

2.2.1 Optimisation methods

In fact, almost all machine learning algorithm, no matter supervised learning, unsupervised learning or reinforcement learning, will lead to a type of optimisation problem finally. There are lots of optimisation methods like, Gradient Descent(GD), Newton's Method are often used in machine learning. And there are also heuristic algorithms can be found in ML methods, for example, Genetic Algorithm(GA), Ant Colony Optimization(ACO) and Particle Swarm Optimization(PSO). In 'Supply chain inventory optimisation with multiple objectives: an industrial case study', they implemented a model of GA together with Petri net in an inventory optimisation problem, and obtained less inventory cost and improved service in practice.[14] The ACO algorithm has also been implemented in solving a vehicle routing problem of hospital blood delivery in the literature 'stochastic local search procedures for the probabilistic two-day vehicle routing problem'. Their approaches performed very well which reduced the cost of delivery significantly by up to 4% of total driving time.[15] In the second part of the thesis 'Solving strategic and tactical optimisation problems in city logistics', Mixed Integer Linear Programming(MILP), Branch & Cut and Branch & Price, has been applied to solve Vehicle Routing optimisation problem.[17]
According to these researches, it can be seen that the field of using optimisation methods to solve logistics problems has been well-studied. And this is one the reason that we believe it is possible to implement machine learning in the field of logistics since machine learning has very close connection with optimisation methods and it can handle more complex situation and solve problems with high volume data.

2.2.2 Reinforcement Learning(RL)

The book 'Reinforcement Learning An Introduction'[8] were also referred in the project. The book introduces how to build a model for reinforcement learning, for instance, Markov decision process is one of the most widely used models for reinforcement learning. The book describe the elements, principle and structure of Markov decision process in detail. Then several solutions are used to compute optimal policies of a Markov decision process model such as dynamic programming, Monte Carlo methods, Temporal-Difference learning, and etc. But in this project,

the dynamic programming introduced in the book is used to solve specific problems. In addition, one of the examples of dynamic programming explained in the book which is 'Jack Cars' Rental Problem' is similar to the problem management in factories in this thesis project, so 'Jack Cars' Rental Problem' will explain and implement in detail in the chapter 4.1.2.

The other article that we've considered is 'Reinforcement Learning for Solving the Vehicle Routing Problem'[3]. They have developed a framework with Reinforcement Learning(RL) and implemented it in the Vehicle Routing Problems(VRP). This VRP problem is delivering items to multiple customers and picking up additional items when it runs out. In order to find the best route of a set of routes, the objective is to attain the maximum reward. With the implementation of RL, they considered this optimisation problem as a sequence of decisions, Markov Decision Process(MDP). Comparison with exist optimisation algorithms, the advantage of their method is that once the trained model is available, it can be used many time for the new problems. In addition, RL doesn't need any matrix calculation which would be computationally difficult. Similarly, there is also an article about using Reinforcement learning to solve the travelling salesman problem(TSP) called 'Ant-Q: A Reinforcement Learning approach to the traveling salesman problem'[6]. In this report, they've developed Ant-Q algorithm which strengthen the connection between Reinforcement Learning, in particular Q-learning and Ant algorithm.

2.2.3 Supervised Learning

Forecasting for the demand of product has been one of the hottest topics in recent years because of the development of Machine learning. In logistic area, the demands are normally related with the seasons, month, or even hours. So the demand forecasting in logistic is specific called time series forecasting. There are two main methods for time series forecasting, which are regression methods and Long-short Term Memory(LSTM) based on Recurrent Neural Network(RNN). One article called 'Predicting Logistics Delivery Demand with Deep Neural Networks'[1] which is relevant to the thesis project and used LSTM method. In this article, they developed a method to predict delivery demand based on machine-learning in order to improve the performance of their logistics in e-commerce service. The method is first to form an input matrix with required data information, then go though a Long-Short Term Memory(LSTM), a special kind of Recurrent Neural Network(RNN), in order to achieve a good prediction of the delivery demand.

In addition to the LSTM Recurrent Neural Network method for time series forecasting, the other method is regression. Based on different principles, many methods have been developed for regression analytic, such as linear regression, random forest, gradient boost regression tree, and so on. In order to study linear regression and implement linear regression for solving time series problem, a paper "forecast: Forecasting functions for time series and linear models"[12] was referred. One of the most used nonlinear regression methods is Random Forest. Article "Classifica-

tion and Regression by Random Forest"[11] is studied to understand the principle of Random Forest Regression. However, in order to implement the code of Random Forest, another article "Comparison of ARIMA and Random Forest time series models for prediction of avian influenza H5N1 outbreaks"[9] was also studied. What is more, one more paper "Comparison of stochastic and machine learning methods for multi-step ahead forecasting of hydro-logical processes"[13] implements the comparison between 11 stochastic and 9 machine learning methods. The performance of 9 machine learning methods solving time series forecasting problem were used to refer for this thesis project.

3
Methods

In order to be more efficient and clear, this thesis project has been divided into two main sections. One is focusing on managing the storage of two factories, at the mean time, arrange the supplementation from the X-doc to the factories. It will send out an order of the insufficient parts type. The other one is solving the milk run problem. According to the order sent by the X-doc, the suppliers on the list will be filtered, because the limitations of both the time and truck capacity. A stochastic method has been used to solve this problem. After the valid suppliers have been selected, an optimal route among them needs to be generated. Reinforcement learning, a simple stochastic method and Ant-Q system has been implemented in order to find the best route.

3.1 Development Process for Management in factories

This segment focuses on introducing the methods are used to development the algorithm for solving two problems in management in factories field.

3.1.1 Preparation

Before actually starting specific works, desk research method is used to map how the current artificial intelligent worked in logistic area. After having some abstract ideas for this thesis work, interviews were carried out with professors from Chalmers Technology University and managers, experts in logistic area from Volvo Cars. With the help of interviews, two specific research directions in management in factories were decided for this thesis work, which are: forecasting the consumption and demands in the factories, and optimization the storage in factories.

Then requirements were carried out to determine the specific needs between the current knowledge and expected result for the Manufacture and Logistic Digital Department. The desired purpose was to use previous 'Call-off' and 'Requirements' data to predict future consumption and demands. Then using the predicted data to optimize the quantity of parts in factories and how many parts should be transported from X-doc.

3.1.2 Data acquisition

In order to implementing optimization and prediction methods, a large number of real data need to be collected. After negotiating with managers in Volvo Cars, Volvo Cars will provide required data from their PLUS system. Data kept updating everyday so the number of data was increased continuously, finally the data includes 'Call-off' and 'Requirements' of each day for nearly a year. The content in provided data includes: supplier, part number, call time, quantity, last arrival time and created time. The actual form of PLUS data is shown in table 3.1 as below.

Table 3.1: PLUS data

Planning	Supplier	Part Number	Call Time	Quantity	Last Arrival Time	Create Time
1003	C7CUL	6906543	2019/5/9 6:30	42	2019/5/9 11:30	2019/5/8 2:25
1003	AEGFW	31690924	2019/5/9 8:00	4224	2019/5/23 6:30	2019/5/8 0:00

3.1.3 Machine learning library

There are a lot of machine learning libraries for python language. With the help of these libraries, some algorithms are already implemented and they just need to be called when it is necessary to use them.

In this thesis work, Panda, Numpy, Keras, Scikit-learn and SciPy are also used to simplify the programming work. For example, Panda is used to read psv files from the path. Instead of programming all the regression algorithms, they can be called from Sikit-learn library. What is more, using SciPy can calculate the Poisson distribution by calling its Poisson model.

3.2 Milk-run system

A truck starts from the X-doc, which is assigned to collect supplies from the suppliers. In order to make this process efficient, limited number of suppliers will be chosen instead of visiting them all. The truck has to be loaded as much parts as possible when it returns to the Cross dock. Also, the shortest route needs to be found for the efficiency. The order list from the previous step will be used for determining how many suppliers and which supplier needs to be visited. After that, the suppliers' access sequence of the optimal route will be generated, with the help of optimization algorithm and machine learning. Some useful techniques are stated as below, like data management and analysis.

3.2.1 Data acquisition

Before implementing optimization method, more data information needs to be gathered for our model, for example, the distance between each suppliers. There are total 15 suppliers are used in this project. The supplier code and suppliers' address are

given by the Volvo Cars company, as shown in table A.1 with their new numerical order. With the help of Google maps, the information about the shortest distance among these fifteen suppliers can be obtained, and they are recorded and formed well in an Excel table, as shown in table A.2 and table A.3, for easy of use in the future.

3.2.2 the use of order form

An order form has been decided and generated from the X-doc. The order form looks like table 3.2. In the first column of the order list, it is showing the desired type of supplies. It is assumed that each supplier only provide their one unique type of parts. Hence, the type of supplies is corresponding to the suppliers' numerical order which is shown in table A.1, for example, In the first row, the number '3' means the supplies from supplier 3, which is 'WITTE NejDeck(D0RYB)' according to the supplier list. In the second column of the order list, it shows the demanded amount of that type of supplies from the Cross dock. This quantity combine with their volume will be used later to identify that whether they can be loaded into the truck. If none of some type of supplies can't be loaded, that supplier will be removed from suppliers' visit list. This process will be introduced more in detail at section 'loading problem' of chapter 'Milk run'.

In the third column of table 3.2, priority is used for selecting suppliers that need to be visited. That priority list is generated due to the amount of parts in the inventory of the Cross dock. That is, the less amount of one type of supplies is, the higher its priority is. Hence, the supplies type with high priority won't fail to be elected normally.

Table 3.2: Order Form

type	amount	priority
3	5	1
5	10	2
8	2	3

3.2.3 Supplies' volume list

The volume of supplies plays an important role in loading problem when calculating the reward. The maximum amount of each type of supplies the truck can load, listed in column 2 of table 3.3, has been used in this method. With a default capacity of the truck, the volume list can be generated easily, as shown in column 3 of table 3.3. The default capacity of the truck is 1. The first row is information data about the X-doc which we don't care in here. Also the supplier 6, 7, 9 and 16 are abandoned for some reason.

numerical order	supplies amount in FTL	Supplies volume
1	N/A	N/A
2	200	1/200
3	2000	1/2000
4	2000	1/2000
5	200	1/200
6	N/A	N/A
7	N/A	N/A
8	200	1/200
9	N/A	N/A
10	2000	1/2000
11	20	1/20
12	200	1/200
13	200	1/200
14	2000	1/2000
15	200	1/200
16	N/A	N/A

Table 3.3: Supplies volume list

4

Pre-knowledge for machine learning in logistic area

In this chapter, the knowledge acquired in preparation for this thesis work is introduced. The investigated pre-knowledge of machine learning is more relevant to the forecasting problem or optimization problem which could be used to solve the problem in this thesis work. The knowledge includes reinforcement learning, recurrent neural network, stochastic optimization and regression analysis.

4.1 Reinforcement learning(RL)

Reinforcement learning is one of the three area in machine learning. It differs from supervised learning in that input and output need not to be presented and paired. Instead it focus on taking actions in an environment so as to maximize reward for the actions.

Reinforcement learning *is learning what to do—how to map situations to actions—so as to maximize a numerical reward signal. The learner is not told which actions to take, but instead must discover which actions yield the most reward* in the environment by trying them. *In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards.* The structure of reinforcement learning is shown in figure 4.1.

learning.png

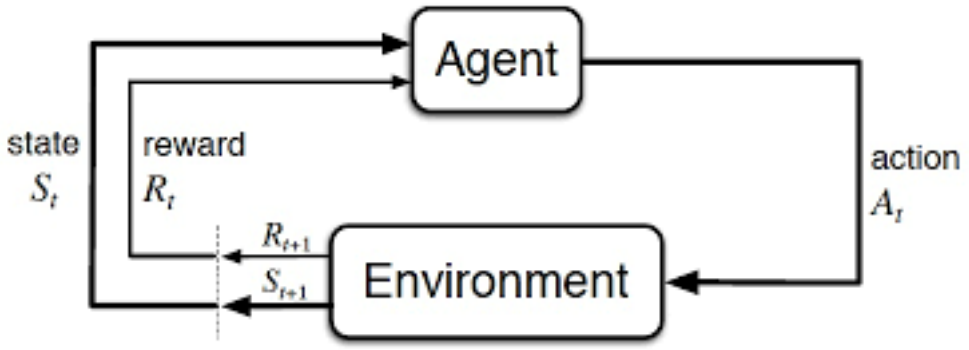


Figure 4.1: Reinforcement learning

The environment is typically formulated as a Markov decision process (MDP), and as

many reinforcement learning algorithms for solving MDP use Dynamic programming method. The detail concept of Markov decision process and Dynamic programming is explained together with two examples: Grid world problem and Jacks' car rental problem in the next two subsections below.

4.1.1 Dynamic programming-Grid world problem

Grid world is an implementation example of reinforcement learning. This problem is solved by using Dynamic Programming(DP) according to the Bellman equation, as shown in equation 4.1.

$$v_\pi(s) = \mathbb{E}_\pi(R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s) \tag{4.1}$$

Where γ is the discount factor between 0 and 1, S_{t+1} is the state at next time step after the agent took an action, and R_{t+1} is the next time step reward. This Bellman equation is derived from Markov value function as shown in 4.2, which is an Markov assumption illustrated that the probability of an action at state s only depends on the current state.

$$v_\pi(s) = \mathbb{E}_\pi(G_t|S_t = s) = \mathbb{E}_\pi(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots|S_t = s) \tag{4.2}$$

From Bellman equation 4.1, it can be found that this is a recurrence equation, which means current state value function can be updated by using the state value function from previous iteration. Hence, it is quite nature to implement DP to solve reinforcement learning problem.

The grid world is a 4-by-4 world, 16 grids, as shown in figure 4.2.

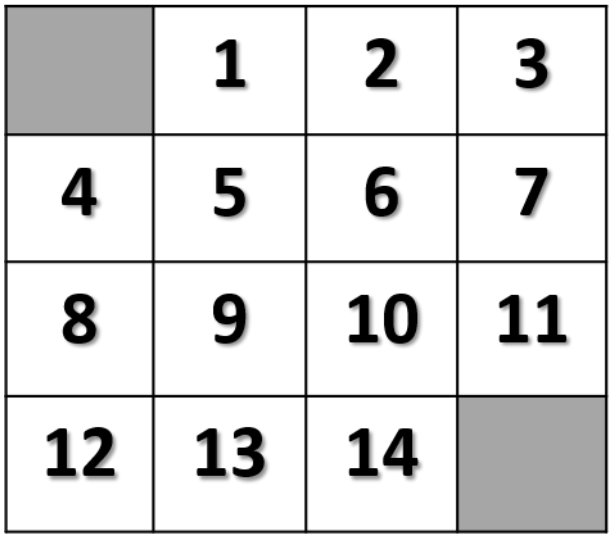


Figure 4.2: Grid World

The grids on the top-left and bottom-right are the terminal grid, which is if any agents arrive these grids, they will stop moving and gain zero reward. Every time,

when an agent moves, it obtained '-1' reward. The actions each agent can take are $a = [' up',' down',' left',' right']$. If one agent moves outside the boundary, it will return to the previous grid. The discount factor γ in bellman equation is set to $\gamma = 1$. The agent takes action randomly, which is, the probability for agent choosing different actions is one quarter since there are 4 kinds of actions.

The state value function will be iterated again and again until the error between each iterations is converged, less than a small positive number. This process is called 'policy evaluation'.

The next step is to improve the policy to find a better policy using greedy algorithm. Agent selects the action which gives the highest reward. After the policy is improved, go back to the first step and the optimal policy(π_*) and state value function(v_*) will be obtained.

4.1.2 Jacks' car rental problem

The purpose of Jacks' car rental problem[8] is to decide a particular number of cars at each parking lot to maximize profit return. There problem setting is that there are two rental car locations. Two locations have varying levels of demand and return rate. Since one of locations has more demand than return rates, some cars are expected to move from one location to the other in the night to make sure there are enough cars at each parking lot to maximize return.

The solution of Jacks' car rental problem is quite similar to the Grid world problem which also uses dynamic programming, where the problem break up into smaller subsets and iteratively solve them. The reason of using dynamic programming is because it has a perfect model of the environment in the form of an Markov Decision Process. Then the goal of Jacks' car rental problem is to setup the environment of the Markov decision process, namely defining our states, actions, probabilities, and rewards. And then use Policy Iteration to solve for the problems.

The difference between the Jacks' car rental problem and Grid world problem is states, actions and rewards. The states are listed in a list which contains a paired values. Each of these paired values correspond to the number of cars in each location. For example, $[4, 5]$ means the state where there are 4 cars at lot 1 and 5 cars at lot 2. Next, the actions are the number of moving cars:

$$action = \begin{bmatrix} -5 & -4 & -3 & ... & 4 & 5 \end{bmatrix}$$

Where positive numbers indicate moving cars from lot 1 to lot 2, and negative numbers indicate moving cars from lot 2 to lot 1. Finally the rewards are depends on costing of moving cars and profit from renting cars.

After determining the states, actions and rewards, the process is the same as Grid world problem which uses policy iteration method: keep updating the state value function by using bellman equation and improve policy until achieve maximum

returns. The result of Jacks' car rental problem is shown in figure 4.3, where x and y label are number of cars in the first and second location, different colors indicate how many cars should be moved at corresponded states. The first five figures are results after each iteration and the last figure shows the optimal value.

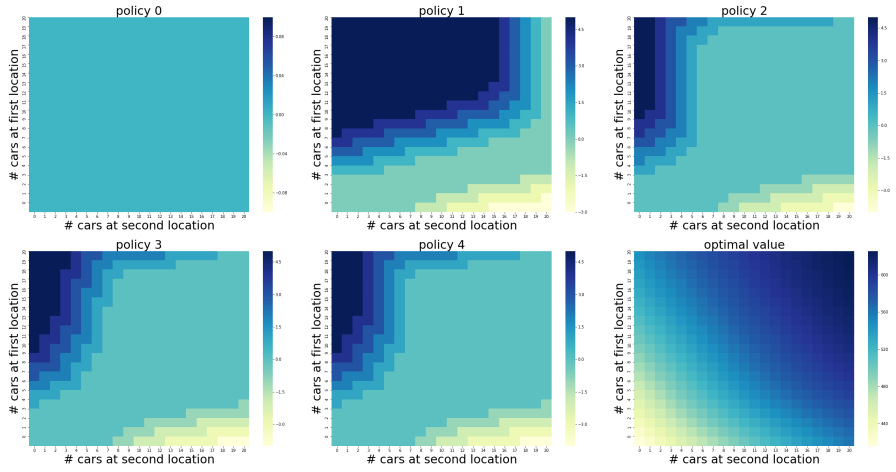


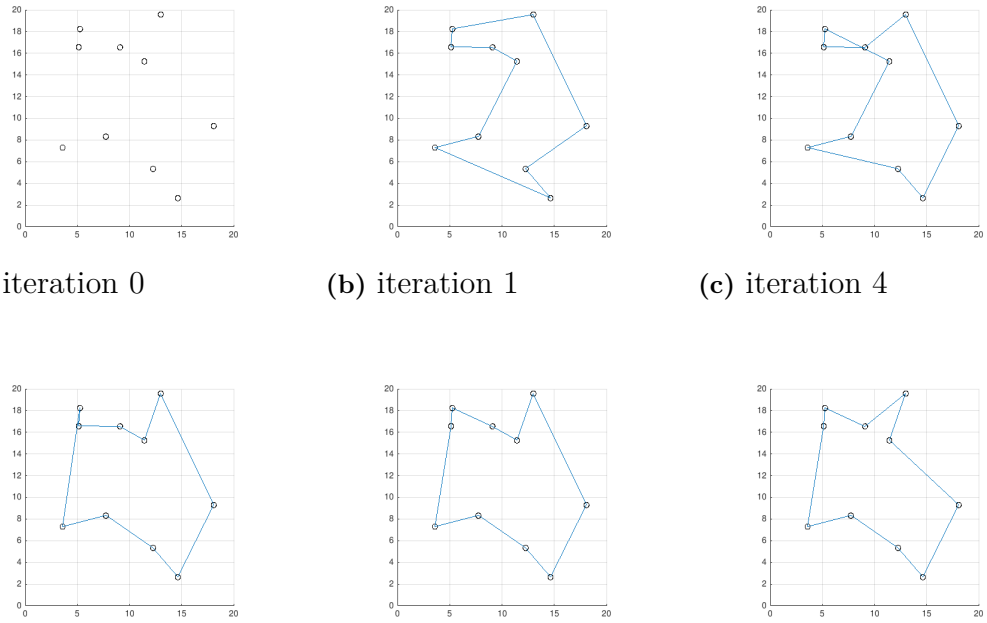
Figure 4.3: Result of Jacks' car rental problem

4.2 Ant Colony Optimization(ACO)-TSP

Traveling Salesman problem(TSP) is an optimal problem that can be solved by many optimal method, like Genetic Algorithm(GA), Particle Swarm Optimization and Ant System. An instance of Ant Colony Optimization(ACO) method is stated in this section.

The TSP states like that, given n cities, a salesman will travel all of them and back to the city where he starts, the question is to find the shortest route among them.

ACO is inspired by the behavior of that ants selecting paths when they search food. The paths selection that ants made are depending on the pheromones generated by previous ants. The stronger the pheromones are, the higher probability is. Hence, ants communicate in this method to find the best way to their food. According to this feature, although the algorithm may start with making many wrong selections, it will converge to the optimal one after several iterations. An instance of ACO on TSP has been shown in figure 4.4.



(a) iteration 0 **(b)** iteration 1 **(c)** iteration 4

(d) iteration 20 **(e)** iteration 21 **(f)** iteration 200

Figure 4.4: Optimal route found in each iteration

ACO is more like an heuristic algorithm, it's a frame, it can be improved by adding other algorithms. The Ant-Q method, which combines the ant system with Q-learning, is used to solve TSP in this project will be introduced later.

4.3 Long-short Term Memory Recurrent Neural Network

The Recurrent Neural Network(RNN) belongs to Deep Neural Network of deep machine learning which could be worked on sequence data such as a series of time. RNNs have a vector of several activation for each time step[19]. However RNNS could only learn the history of last step, then in order to learn more histories, long-short Term Memory solution is needer.

Long-short Term Memory(LSTM) Recurrent Neural Network is a special form of recurrent neural network. The working principle of LSTMs is similar to the random access memory of the computer. It contains the information is not included in the recurrent neural network. The information can be stored in, written to, or read from a cell. The cell can make decision about when to allow recurrent neural network to read and write by gates that open and close. Even about what to store is also decided by the cell. With the help of LSTMs, the error can be preserved and reused through time and layer. By maintaining a more constant error, they allow recurrent neural networks to continue to learn histories over many time steps, thereby opening

a channel to link causes and effects remotely[18]. The structure of LSTM Recurrent Neural Network is shown in figure 4.5.

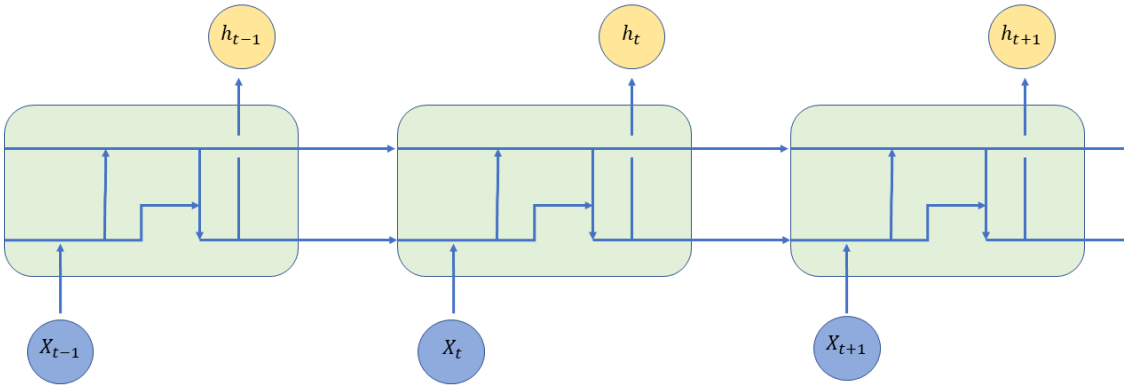


Figure 4.5: Structure of LSTM Recurrent Neural Network, where X_t is input and h_t is output.

4.4 Regression

Regression analysis method is a set of statistical processes for estimating and modelling the continuous variables. Regression analysis is one of supervised machine learning methods, and is widely used for prediction and forecasting, for instance, forecasting the tendency of stock, future temperature, housing price and so on. Since regression analysis belongs to supervised machine learning, it has to have outputs which normally are dependent variables Y. And corresponded independent variables X are also needed. Besides the independent and dependent variables, there are always unknown parameters, denoted as β, which may represent as a scalar or a vector. So a regression model relates Y to a function of X and β normally represents as[12]:

$$Y \approx f(X, \beta)$$

The approximation is usually formalized as:

$$E(Y|X) = f(X, \beta)$$

In order to find out the model function f, many methods have been developed, such as linear regression and regression tree. Linear regression is a most simple and common method. A linear regression model assumes that the relationship between dependent variables Y and corresponded independent variables X is linear, the model will also be added with disturbance or error.

Except linear regression, regression tree which could also called decision tree model is also a widely used regression model. Differ from linear regression, regression tree can fit nonlinear characters in the data set. The main idea of regression tress is to separate a model function into several small functions. Adding all small functions together, the model will be acquired. And the implementation of regression tree is shown in chapter 6.2.2.

4.5 Poisson Distribution

Poisson distribution is a discrete probability distribution that express the probability of a given number of events occurring in a fixed interval time. The stochastic variables fit Poisson distribution only if they are independent positive and appear randomly. Poisson distribution is normally used to describe the probability of a number occurs in real world, such as the number of people at bus station, the number of consumption part in a factory. The probability function of Poisson distribution is normally described as:

$$P(k) = \frac{e^{-\lambda}\lambda^k}{k!}$$

Where k is the number of occurrences and λ is the expected number of occurrences which could be the average number among all k. The function is defined only at integer values of k. Figure 4.6 is an example of Poisson distribution.

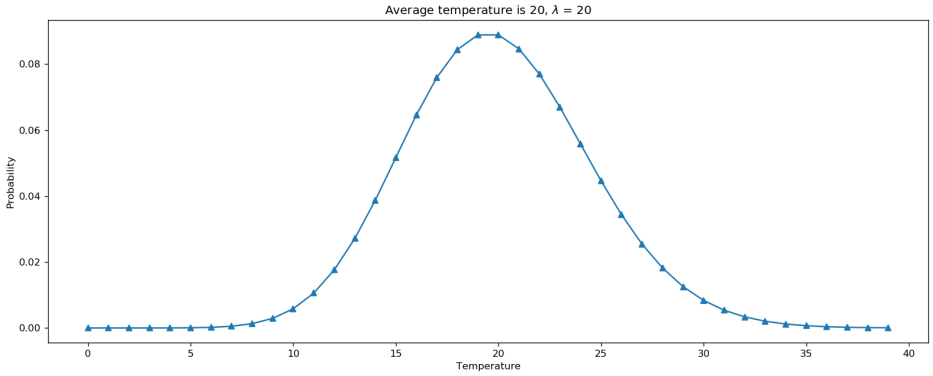


Figure 4.6: An example of Poisson distribution

Notice that the horizontal axis is the index k which is the number of temperature in the figure, and the vertical axis is the probability of temperature k occurrences given average temperature $\lambda = 20$.

5

Milk run problem

Despite the Full Truckload, Volvo Cars used another logistics technique called milk-run or milk-round. Instead of letting suppliers sending their products to warehouse, Volvo decided to pick up supplies by itself. The advantage of this transportation is to maximum the load of the truck. In addition, it does not only reduce the cost of transportation expense, but also enables the delivery scheduled and efficient. In this chapter, it is presented solving a Travelling Salesman Problem(TSP) by implementing several algorithms.
Besides, because of the limitation of time and truck's capacity, some of the suppliers won't be visited in that tour. In the section of 'loading problem', it is presented how suppliers are selected.

5.1 Loading problem

This is a process of selecting which supplier needs to be visited. It also ensures all the supplies from selected suppliers can be loaded. The problem states like, a truck from X-doc will collect supplies from the suppliers after the order list, shown in table 3.2, from the X-doc is received. However, if there are too many suppliers on that order list to travel, it is not possible for the truck to visit all of them, due to the time and capacity limitations. In that case, some of the suppliers on that order list will be selected. According to the priority on the order form, supplier with high priority will be selected to visit, if the total volume of the supplies is not exceed the maximum capacity of the truck. In order to be more efficiency, the truck should be loaded as much as possible, so the proportion of truck's inventory will also be considered as a factor. Supplies can't be loaded this time will be assigned in the next turn. One limitation of this loading problem is each supplier provide their own specific supplies, which means supply type is corresponding to their suppliers.

5.1.1 Dynamic Programming (DP)

It is easy to implement reinforcement learning instead of other supervised learning, since the Markov Decision Process(MDP) model is easy to build in this loading problem, and no need to input previous data information for training process. The pseudo-code is shown as algorithm 1.
In order to form a MDP model, the first step is to find states, actions, rewards and state value function. The states are chosen as the loading sequence of the supplies. And the actions are choices of the types of supplies. For example, an agent starts at

Algorithm 1: Dynamic Programming

Input: The order list from the X-doc(including supplies' type, amount and priority), $Order$, n by 3 matrix as shown in table 3.2, where n is the number of suppliers; All supplies' volume set, $volumeset_n$; Selected supplies' volume set, $volume_n$; Future discount factor, γ; Bonus reward for fully-load, R_b; Small positive number, ρ; Iteration times is a large positive number, $threshold$

Output: Amount and types of the supplies(suppliers, since every supplier has their own supplies), N_i and $Type_i$, where i is the index code of selected suppliers; Abandoned supplies, $Left_i$;

```
Policy initialization policy using supplies' type and amount, Order(:, 1 : 2);
Generating priority coefficients, coefficient_n using priorities list on Order(:, 3);
while k < threshold do
    policy = policy';
    m = length(policy);
    v = v';
    while error > ρ do
        v = zeros(m, 1);
        for i = 1 : m do
            Calculate the state value function, v;
        error = |v − v'|;
        v' = v;
    inventory = Order(:, 2);
    capacity = 0;
    for i = 1 : sum(Order(:, 2)) do
        Qarray = zeros(n, 1);
        for j = 1 : n do
            Check if the supply fits perfectly;
            Calculate each action value function Q and save them into Qarray;
        The best index will be saved and the corresponding supply type will be
         saved to the new policy;
        if there is that type of supply left in inventory then
            remove 1 of them, since it has been loaded;
        capacity = capacity − volumeset(policy_i);
        if capacity exceeds the maximum capacity of the truck then
            break the loop;
    policy' = policy;
    k = k + 1;
Count the types of loaded supplies, Types;
Count the amount of each loaded supply's type, N;
return Types and N;
```

the initial state, after selecting which supply to be loaded first, the agent comes to the second state and needs to decide which supply will be loaded next. Each action gives a reward '-1'. When an 'agent' arrives that state it gains state reward, which can be calculated by supplies' volume and priority. Only when the truck is fully loaded, the action reward is zero.
The state value function is the sum of action reward, following an initial policy, and product of chosen supply's volume and priority coefficient. The initial policy is generated randomly corresponding to supplies' type and amount on the order list. The limitation of truck capacity is considered, which means the truck might not load all the supplies.
In the policy evaluation step, state value function will be updated using Bellman equation shown in 5.1.

$$v_\pi(s) = \sum_{a \in A} \pi(a|s)(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s')) \tag{5.1}$$

Where s is current state, s' is next state, π is the policy, the discount factor is set to $\gamma = 0.9$, R_s^a is the reward at state s after taking action a, and $P_{ss'}^a$ is state transition probability which is the probability of taking action a at current state s transfer to the next state s'. In this case, the probability is related to the number of left supplies' types. More specifically, if there are three types of supplies left to be loaded at current state, the transition probability is one third since every time the agent can only choose one supply, to load it to the truck. As the remaining supplies are running out, transition probability is increasing. For example, if there is only one type left to be loaded, the agent has to choose that one, which means that transition probability is 100%.
After assigning the rewards and actions in this equation, the state value function will keep updating until it is converged. The iteration stop condition is when the error Δ, between the current and previous value function, less than a small positive number, which means the value function is stable and can't be updated any more.
In policy improvement step, greedy algorithm is used for proposing a new policy. The left room of the truck is calculated and applied to check whether there is any supplies is able to fit in the left space perfectly. If there is one, a bonus reward R_b will be added to the action value function. After the total action reward at each state is computed, policy will be updated by the action with the highest reward. The action value function is given by 5.2.

$$Q_\pi(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \tag{5.2}$$

In which it is shown the action value function Q is based on the state value function v. The new policy will then be brought back to policy evaluation to be updated. After that, another new policy will be proposed in policy improvement step. The stop condition is a counter k larger than a pre-set positive large number. This iteration process is called policy iteration in order to obtain an optimal policy v^*.

5.1.2 Priority first policy proposal

Comparing with the dynamic programming method, this one is easier to understand. The idea is to load the supplies with high priority first. The pseudo-code is as shown

in algorithm 2. This method is only considering supplies' priority given by the order

Algorithm 2: Priority first

Input: The order list from the X-doc(including supplies' type, amount and priority), $Order$, n by 3 matrix as shown in table 3.2, where n is the number of suppliers; All supplies' volume set, $volumeset_n$; The remaining space in the truck, $storage$;

Output: Loading sequence of desired supplies, $policy'$

```
Policy initialization policy using Order;
Split the initial policy by different types of supplies, rank_1, rank_2, ..., and rank_n;
for i = 1 : length(policy) do
    These sets of supply type are listed as a new policy, policy', ordered by priority
     Order(:, 3);
    storage = storage − volumeset(policy(i));
    if storage < 0 then
        policy'(i : end) = []
return policy';
```

list from X-doc. Higher priority supplies come first, and will keep loading until the total volume meets truck's maximum capacity. The remaining space, $leftState$, in the truck will be watched. And if there is no room left, no more supplies can be loaded and the $policy'$ will also stop updating. This algorithm is simple and fast, however, the truck's inventory might not be used efficiently.

5.1.3 Stochastic method with 'Priority first' policy proposal

'Stochastic' means policies are proposed randomly, which is aimed for propose different policies, as well as for avoiding the stuck in a local optimal solution, which always happens in heuristic optimization algorithm. It will increase the speed of finding optimal that combining priority first algorithm with stochastic method. It's pseudo code is as algorithm 3.
In this loading case, desired supplies are loaded randomly to the truck until it is full. Supplies that can't be loaded will be saved and wait for the next truck. The total rewards based on supplies' volume and priorities will be computed. The action with the best reward will be saved for policy updates. However, it is an inefficient and time-consuming process if the algorithm is only based on that random method. Hence, the priority greedy method is inserted into this algorithm in order to increase the searching speed. As it is mentioned in last section, the priority greedy is a method which is finding an action given the highest reward on each state.

5.2 Travelling Salesman Problem(TSP)

TSP has been stated in section 'ACO-TSP' of chapter 'pre-knowledge'. In our milk-run model, the salesman who is traveling around becomes a truck from X-doc. It needs to travel among several specific suppliers in order to collect demanded supplies.

Algorithm 3: Stochastic algorithm with Priority first method

Input: The order list from the X-doc(including supplies' type, amount and priority), $Order$, n by 3 matrix as shown in table 3.2, where n is the number of suppliers; All supplies' volume set, $volumeset_n$; Probability of selecting priority first method, ρ; the remaining space in the truck, $storage$; iteration times, $threshold$;

Output: Loading sequence of desired supplies, $policy'$

```
1  Policy initialization policy using Order;
2  Split the initial policy by different types of supplies, rank_1, rank_2, ..., and rank_n;
3  Generate random parameter, p = rand;
4  while iteration < threshold do
5      if p < ρ then
6          policy' = policy(randperm(length(policy)));
7          else
8              for i = 1 : length(policy) do
9                  These sets of supply type are listed as a new policy, policy', ordered
                    by priority, Order(:, 3);
10     for i = 1 : length(policy') do
11         storage = storage − volumeset(policy(i));
12         if storage < 0 then
13             policy'(i : end) = [ ];
14     Calculate the total reward, R. Find the best and save the corresponding policy';
15     iteration = iteration + 1;
16 return policy';
```

The main problem is to obtain the best visiting sequence of the suppliers, which will be the shortest path among them. The truck starts from the X-doc and gets back to it after taken the delivery of supplies. The truck will only visit the chosen suppliers, since this is a process after the suppliers selection of loading problem. One assumption is made that there is only one truck can travel at a time.

5.2.1 Dynamic programming

Dynamic Programming has been used again to solve another Reinforcement learning problem on TSP. In the MDP model, it is considered the suppliers that need to be visited are the states. And the actions are to select one of the remaining suppliers. The reward is inversely proportional to the distance between each two locations including suppliers and X-doc. In state value function, the transition probability $P^a_{ss'}$ depends on the number of remaining suppliers. As a reciprocal of remaining suppliers, $P^a_{ss'}$ will increase as the iteration goes. And it will become one in the end, since the truck will visit each suppliers only once. The pseudo code is as shown in algorithm 4. All the suppliers are represented as numerical order according to the A.1, for example, the X-doc is number 1. In policy initialization, the sequence of suppliers are generated randomly. However, the first and last one is the X-doc, since the truck starts from X-doc and it needs to get back to the X-doc as well after it finishes the tour. The distance matrix is shown as A.2 and A.3, the reward matrix is their transposition with an negative sign which is used for computing the state value function. In policy evaluation, state value keeps updating until the error, Δ, between v and v' is almost zero. The greedy algorithm has been implemented again in policy improvement step, in order to find the shortest path between each two locations. In every step, action value function Q is calculated for every taken action using state value function v, and its value is saved to a list $Qarray$. The action with the highest Q is added to a new policy, $policy'$. This $policy'$ is brought back to policy evaluation step and generate another converged state value function. The policy iteration loops a large number of times in order to find the optimal result.

5.2.2 Ant-Q in milk run problem

Stochastic optimization algorithm, like ant system is suitable for solving TSP problem. Luca Gambardella and Marco Dorigo has proposed an algorithm called Ant-Q in 1995 [6], which combine ant system with Q-learning. Q-learning has been implemented to update pheromones in ant system. The pseudo code is shown as algorithm 5. Ant system has been introduced in 'Ant Colony Optimization(ACO)-TSP' of chapter 4, which the implementation is similar to this problem. It is considered the salesman is a truck, and the cities are suppliers.
Starting from the Cross dock, the next state is selected by using an action choice rule according to [6] as shown in equation 5.3.

$$argmax\{[AQ(r,u)]^{\delta} \cdot [HE(r,u)]^{\beta}\}, u \in J_k(r) \tag{5.3}$$

However, in order to be stochastic, there is a probability $\rho = 1 - q_0$ to select the next supplier randomly from the remaining suppliers. From the equation 5.3, it

Algorithm 4: Dynamic Programming in TSP

Input: List of suppliers to visit, S_n; The distance reward matrix, R; Future discount factor, γ;

Output: suppliers visit order, *policy*;

```
Policy initialization, policy;
policy' = ones(length(policy));
while policy ≠policy' do
    policy = policy';
    Δ = 1;
    while max(Δ) > 10^-6 do
        v' = v;
        v = zeros(n, 1);
        leftState, which is the remaining suppliers;
        for i = 1 : (n − 1) do
            Remove one supplier from leftState according to the policy(i);
            for j = 1 : length(leftState) do
                v(i) = v(i) + (1/length(leftState)) × (R(policy(i), leftState(j)) +
                 γ ∗ v'(i + 1));
            if leftState < 1 then
                break;
        v(n) = R(policy(n), 1) + γ × v'(1);
        for i = 1 : n do
            Δ(i) = |v'(i) − v(i)|;
    Initialize policy';
    Reset leftState;
    for i = 1 : n − 1 do
        Qarray = zeros(m, 1);
        for j = 1 : m do
            Q(i) = R(policy'(i), leftSuppliers(j))+...gamma∗valueFunction(i+1);
            Qarray(j) = Q(i)
        BestIndex = find(Qarray == max(Qarray));
        Q(i) = Qarray(BestIndex);
        bestAction = leftState(BestIndex);
        leftState(BestIndex) = [];
        policy'(i + 1) = bestAction;
return policy';
```

Algorithm 5: Ant-Q algorithm

Input: supplies and X-doc, *allstates*; distance matrix, *costmatrix*; selecting optimal path stability, q_0; learning step, α; discount factor, γ; the relative weight of importance of the learned AQ-values AQ and the heuristic values HE, δ and β; iteration times, *iterationnumber*; number of agents, m;

Output: Optimal visiting order of suppliers, *Bestpolicy*

```
n = length(allstates);
Initialization of HE = -R', AQ = zeros(n, n), J(:, n) = allState which is the
  memory matrix of remaining suppliers; the current supplier r = zeros(n + 1, m);
the initial supplier is X-doc, r(1, :) = 1;
iter = 0;
while iter < iterationnumber do
    tour = zeros(n, 2m);
    for j = 1 : m do
        J_k = J(:, j);
        for i =1:n do
            Select the next supplier s(i, j);
            Calculate the cost between current supplier r(i, j) to every remaining
              supplier;
            Save the order of them to tour(i, 2 * j - 1 : 2 * j) = [r(i, j), s(i, j)];
            AQarray = zeros(length(Jk), 1);
            for k = 1 : length(Jk) do
                AQarray(k) = AQ(s(i, j), Jk(k));
            AQ(r(i, j), s(i, j)) =
              (1 - alpha) × AQ(r(i, j), s(i, j)) + α × γ * ×max(AQarray);
            r(i + 1, j) = s(i, j);
    for k = 1 : m do
        for i = 1 : size(tour, 1) do
            L(k) = L(k) + costmatrix(tour(i, 2 * k - 1), tour(i, 2 * k));
    ΔAQ = W/min(L);
    Update AQ value using ΔAQ,
      AQ(r, s) ⟵ (1 − α) · AQ(r, s) + α · (ΔAQ(r, s) + γ · Max(AQ(s, z))), z ∈ J_k(s);
    Save the best visit order and save it to Bestpolicy;
    iter = iter + 1;
return Bestpolicy;
```

is found the selection depending on two matrix AQ and HE. HE here is the cost matrix, and AQ is the Ant-Q value keeps updating during the iteration. The update equation of AQ is shown on line 22 in algorithm 5.

6
Management in factories

In this chapter, the theory and knowledge introduced in chapter 4 are implemented. The implementation of the algorithms will be used to solve specific problem in Volvo Cars. The implementation tools are introduced in section 6.1. Then the prediction problem is shown in section 6.2 and the optimization the storage can be found in section 6.3.

6.1 Implementation tools

There are many programming language could be chosen for machine learning, but since Python has more useful libraries, such as Scikit-learn, Keras, SciPy, all of them were used in this thesis work. So the programming language Python was chosen. Spyder in Anaconda was chosen as integrated development environment for programming because it is a open source cross-platform integrated development environment for scientific programming in the Python language. What is more, visible consoles and variable explorer, excellent debugging function is also the good reason for choosing Spyder. In addition, in order to build neural network and make sure algorithms could work, Tensorflow is added in python environment. Except tensorflow, several machine learning libraries such as Panda, Numpy, Keras, Scikit-learn and SciPy are also used to simplify the programming work.

6.2 The prediction of demand and consumption

According to the table 3.1, the information about consumption and call off data from PLUS system of Volvo cars is about date and quantities of each part. So the prediction problem is defined as a time series problem. And the most popular solution for time series prediction problem is using regression analysis methods and Recurrent Neural Network with Long Short Term Memory (LSTM). In this section, the implementations of regression analysis methods and Recurrent Neural Network are presented base on the data provided by PLUS system.

6.2.1 Recurrent Neural Network

In this section, Recurrent Neural Network is introduced and implemented in Python using the Keras machine learning library to address a demonstration time-series pre-

diction problem.

The most important process in solving time series prediction problem by LSTM Recurrent Neural Network is to build features for input data as training set to fit the model. The solution is to look back of previous information, which is the number of previous time steps to use as input variables to predict the next time period. For example, if the value of look back is 5, which means the quantities of part for 5 days before will all become features for current date. The Algorithm 6 presents the complete persuade code for creating features for training sets.

Algorithm 6: Building features for data set

Input: dataset, look back
Output: featureX, labelY

```
featureX = [ ];
labelY = [ ];
for i in range (length of dataset - look back -1) do
    temporary value = dataset[i:(i+look back),0];
    featureX.append(temporary value);
    labelY.append(dataset[i+look back,0]);
return featureX, labelY
```

After building features for the data, it is necessary to split data into train and test data sets. The train data sets is used to fit model, and test data sets is to test the model to check how the model fit new unseen data. The Algorithm 7 presents the complete persuade code for splitting data sets into test and train sets.

Algorithm 7: Splitting data set into test and train set

Input: dataset, train size, look back
Output: TrainX, TrainY, TestX, TestY

```
Train set = dataset[0:train size, :];
Test set = dataset[train size:, :];
TrainX, TrainY = create feature(Train set, look back);
TestX, TestY = create feature(Test set, look back);
TrainX = np.reshape(TrainX, (TrainX.shape[0], 1, TrainX.shape[1]));
TestX = np.reshape(TestX, (TestX.shape[0], 1, TestX.shape[1]));
return TrainX, TrainY, TestX, TestY
```

Notice that the create feature in Algorithm 7 is function of the Algorithm 6.

The data sets for training the model is done, LSTM network is ready to be designed and fitted for the problem. It's simple to build a LSTM Recurrent Neural Network with the help of Keras library. The Algorithm 8 presents the complete persuade code for building LSTM model, fitting the data sets and predicting the future information.

Algorithm 8: Train model and predict data

Input: TrainX, TrainY, TestX, TestY
Output: Model

```
model = Sequential();
model.add(LSTM(64, input shape=(1, look back)));
model.add(Dense(32));
model.add(Dense(16));
model.add(Dense(8));
model.add(Dense(1));
model.compile(loss='mean squared error', optimizer='adam');
model.fit(trainx, trainy, epochs=500, batch size=1, verbose=2);
trainPredict = model.predict(trainx);
testPredict = model.predict(testx);
trainPredict = scaler.inverse transform(trainPredict);
trainy = scaler.inverse transform([trainy]);
testPredict = scaler.inverse transform(testPredict);
testy = scaler.inverse transform([testy]);
```

Notice that since LSTM are sensitive to the scale of the input data, the data sets are normalized using the MinMaxScarler preprocessing class from the Scikit-learn library. So the predictions should do inverse normalize to revert back to quantity unit.

6.2.2 Regression

In this section, regression analysis methods are introduced and implemented in Python using the SciKit-learn machine learning library to address a demonstration time-series prediction problem.

Similar to the create feature function of Algorithm 6 for LSTM Recurrent Neural Network (RNN), features of data sets are also needed for training regression models. The same idea as using look back method for building features for RNN, it is called shift feature for regression analysis. However, differ from look back method, regression analysis shift all the data at the same time. As an example in table 6.1, lag 1 means move all data one grid down and lag 4 means move all data four grid down. If there is no data, it will replace by Not a Number (NaN). And when lag start from 2 and end at 4, the features for 2019/3/25 will be [720, 1080, 1080]. The benefit for using lag start and lag end than look back method is the created features are more feasible.

Time	Quantity	Lag 1	Lag 2	...	Lag 4
2019/3/19	1080	NaN	NaN	...	NaN
2019/3/20	1080	1080	NaN	...	NaN
2019/3/21	720	1080	1080	...	NaN
2019/3/22	2520	720	1080	...	1080
2019/3/25	1080	2520	720	...	1080
...	...	...	...	...	...

Table 6.1: An example for time shifting features

Except shift features for regression method, there is also time features for regression method. The time features could be weekday, holiday and season. For instant, when the date is 2019/3/19, the feature 'Tuesday' could be added. Algorithm 9 presents the complete code for building time and shift features.

Algorithm 9: Building features for data set

Input: dataset, lag start, lag end, target encoding
Output: featureX, labelY

```
last date = data["timestamp"].max();
dataset.set index("timestamp",drop = True, inplace = True);
for i in range (lag start, lag end) do
    dataset["lag[ ]".format(i)] = dataset.y.shift(i);
dataset["month"] = dataset.index.month;
dataset["weekday"] = dataset.index.weekday;
dataset["weekend"] = dataset.weekday.isin([5, 6]) * 1;
dataset["quarter"] = dataset.index.quarter;
if target encoding then
    dataset["weekday avg"] = calculate the average quantities of all weekday;
    dataset["month avg"] = calculate the average quantities in the same month;
    dataset["weekend avg"] = calculate the average quantities of all weekend;
    dataset["quarter avg"] = calculate the average quantities in the same quarter;
    dataset = dataset.drop(["month","weekday", "weekend","quarter"], axis = 1);
laberlY = dataset.dropna().y;
featureX = dataset.dropna().drop("y", axis=1);
return featureX, labelY;
```

Notice that if target encoding is true, the time features will be replaced the average quantities of corresponded feature.

There is one more function needed, which is predict function. The predict function is to build features for future date. So with the help of future features, future information could be predicted by fitted model. Algorithm 10 presents the complete code for building features for future date.

Algorithm 10: Predict future information

Input: model, predictX, lag start, lag end, predictY
Output: PredictY

```
predictY[0:lag start] = model.predict(scaler.transform(predictX[0:lag start]));
for i in range(lag start, length(predictX)) do
    lastline = predictX.iloc[i-1];
    index = predictX.index[i];
    predictX.at[index, "lag".format(lag start)] = predictY[i-1];
    for j in range(lag start + 1, lag end) do
        predictX.at[index, "lag[ ]".format(j)] = lastline["lag[ ]".format(j-1)];
    predictY[i] = model.predict(scaler.transform([predictX.iloc[i]]))[0];
return predictY
```

Finally, it is able to use created features to fit the model, and then using fitted model to predict future information by using prediction function. The regression model is easy to be called by using SciKit-learn library. Algorithm 11 presents the complete code for fitting regression models and predicting future information.

Algorithm 11: Train model and predict data

Input: TrainX, TrainY, TestX, TestY
Output: Model

```
scaler = StandardScaler();
x train scaled = scaler.fit transform(x train);
x test scaled = scaler.transform(x test);
lr = GradientBoostingRegressor(n estimators=100);
lr.fit(x train scaled, y train);
yfuture = predict future(lr, scaler, xpred, ypred, lag start, lag end);
yfit = lr.predict(np.concatenate((x train scaled, x test scaled)));
```

Notice that the regression model showed in Algorithm 11 is only Gradient Boosting Regressor, but there are more regressors could be called by SciKit-learn library, such as Linear Regressor, Adamtree Regressor, and etc.

6.3 Optimizing storage in factories

In this section, the solution for optimizing the quantity of parts in factories is presented. At last the pseudo code for the algorithm is also presented.

This optimization problem is derived from Jacks' car rental problem which has described in section 4.1.2. Two parking lots took place by two factories of Volvo Cars, at the mean time the transportation goods will be the consumed parts in factories. The rental and return cars are replaced by Consumption parts in factories and corresponded Call-off parts from suppliers. In Jacks' car rental problem, the number of cars will be transferred between each parking lots to satisfy the rental requirements of each lot for maximum returns. How ever in this problem, there is no

transit between each factory, instead the X-doc will supply factories for insufficient parts because of the parts from Call-off will normally less than consumption in factories. But if storing too many parts in factories and transferring too many parts could cost too much money, so the problem will be to find the best quantities of parts should store in factories and the best quantities should be transferred from X-doc to factories, and finally the result will have maximum returns.

6.3.1 Initialization

There are some parameters need to be defined for particular problem. Table 6.2 shows the parameters are used in the algorithm.

1	Number of Factory
2	Max storage in Factory
3	Max move parts
4	Capacity in X-doc
5	Discount rate
6	Factory profit
7	Supplier cost
6	X-doc cost

Table 6.2: Parameters for optimizing storage in factories

Since the number of states are related with the quantity of each part could be stored in the factories, states can be defined with help of Max storage in factory in table 6.2. The same to the initial policy, initial state value and action can be defined with Max move parts and number of factory in table 6.2. The algorithm 12 presents the definition of policy, state values and actions.

Algorithm 12: Algorithm define action, policy and state value

Input: Number of Factory, Max storage in Factory, Max move parts
Output: State value, Policy, Actions

```
1 Policy = np.zeros((Number of Factory, Max storage in Factory + 1));
2 State value = np.zeros((Number of Factory, Max storage in Factory + 1));
3 Actions = np.arrange(0, Max move parts + 1);
```

Notice that the max range of Actions should be increased by 1, because in np.arrange, the end of interval is not included. The Policy and State value should include the state when there is no part left in factory, so the range of Policy and State value also add one more.

6.3.2 Expected Return

The expected return actually is the result of Bellman equation which can also be summarized as the probability of being in some subsequent state given a particulate action, multiplied by the reward for being in this position plus the discounted value of being some subsequent state. The state here is related with the quantities of

consumed and replenished parts. And the total rewards includes the profit from factory and also should minus the cost from suppliers and X-doc. Since the problem uses dynamic programming as a solution, the bellman equation is used to update state value function. So the expected return will update the state value function, it also can help to find best policy.

In order to find the maximum returns, it's necessary to look through all the states, then add all the values of every states together to get total return. So two loops are needed for look through all the possibility of both consumption and replenishment in the factory. Since the probability of being in some quantities of parts is different, a model for probability distribution is needed. And for the quantities of parts is an independent, positive and stochastic number in real life, Poisson Distribution fits the probability model in this problem perfectly. The detailed introduction of Poisson Distribution is described in section 4.6. Then the total probability should multiply the probability of number of parts of consumption and replenishment. The algorithm 13 presents the completed persuade code of expected return.

Algorithm 13: Expected return

Input: Number of part in factory, Number of transported part, state value
Output: Returns

```
1  Returns -= X-doc cost * Number of transported part;
2  for consumption in factory from Poisson lower bound to Poisson upper bound do
3      Number of parts in factory = min(Initial number of part in factory, number of
         transported part);
4      Real Consumption in factory = min(Number of parts in factory, Consumption
         in factory);
5      reward = Real Consumption in factory * Factory profit;
6      Number of parts in factory -= Number of parts in factory - Real Consumption
         in factory;
7      for replenishment in factory from Poisson lower bound to Poisson upper bound
         do
8          Current Number of parts in factory = min(Number of parts in factory +
             replenishment in factory, Max storage in factory);
9          Returns -= replenishment in factory * Supplier cost;
10         Probability = Poisson(consumption in factory, Predicted consumption in
             factory) * Poisson(replenishment in factory, Predicted replenishment in
             factory);
11         returns += Probability * (reward + Discount rate * state value[Current
             Number of parts in factory]);
```

Notice that the reason for states are only looked through from Poisson lower bound to Poisson upper bound is if the number of state is not in the range of Poisson bound, the probability will be zero. So the number of quantities for both consumption and replenishment outside the Poisson bound is not necessary to be concerned. What is more, the Poisson function is called from SciPy libraries. The first variable as input for Poisson function is the variable waited to calculate probability. The second input

is the average number of the variable, so in this case, the value is the predicted data of consumption or replenishment in factories.

6.3.3 Policy Iteration

Policy in this problem is the quantities of each part moving from X-doc to factories. So the problem now becomes to find best policy in order to maximize returns. Since dynamic programming is used, policy and state value will be updated step by step together with Bellman equation in Expected returns. The process of updating policy and state value is called policy iteration. Policy iteration can be separated into two parts depends on updating the policy or state value, which is policy evaluation and policy improvement. Policy iteration will keep running until there is no change in policy.

6.3.3.1 Policy Evaluation

Policy evaluation is the process for updating the state value according to current policy. State value is calculated by algorithm 13 Expected return. Since the policy of every state need to be updated, the for loop is needed to look through all the states in two factories. In this problem states are the number of quantities of part in factory. Policy evaluation will keep running until the change of state value less than 1×10^{-4}. The algorithm 14 presents complete persuade code of policy evaluation.

Algorithm 14: Algorithm for evaluating policy

Input: Number of factory, Max storage in factory, state value, policy
Output: New state value

```
for State value change > 1 * 10^-4 do
    New state value = np.copy(state value);
    for j in range of Number of factory do
        for i in range of Max storage in factory + 1 do
            New state value = Expected return(i, policy[j,i], New state value[j, :]);
    State value change = np.abs((New state value - state value)).sum();
    State value = New state value;
```

6.3.3.2 Policy Improvement

After finishing the process of policy evaluation, the new state value is ready for updating policy. Policy improvement is the act of looking at all actions could take from a given state, acting greedily by choosing the action that gives the highest return with respect to new state value. Then the action which gives the highest return will be replace old action in the policy. In this problem, actions are the all possible number of quantities of parts moving from X-doc to factories, and the initial actions are all zero for the corresponded states. The algorithm 15 presents complete persuade code of policy improvement.

Algorithm 15: Algorithm for improving policy

Input: Number of factory, Max storage in factory, State value, Policy, Actions
Output: New policy

```
for policy change ≠ 0 do
    New policy = np.copy(Policy);
    for j in range of Number of factory do
        for i in range of Max storage in factory + 1 do
            action returns = [ ];
            for action in range of Actions do
                action returns.append(Expected return(i, action, State value[j, :]));
            New policy = Actions[j,np.argmax(action returns)];
    Policy change = (New policy ≠ Policy).sum();
    Policy = New policy;
```

7
Results

This section states the valid results of the solutions of the loading and TSP problem in milk-run, also the demand prediction and storage optimization of management in factories, which can be found in chapter 5 and 6 respectively.

7.1 Management in factories

There are mainly two important aspects to check the quality of machine learning algorithms, which are time and accuracy. The results of two problems will be discussed more under these two aspects. Section 7.1.1 presents the results of prediction problem and results of optimization of storage in factories are introduced.

Normally, machine learning is used to deal with huge number of data, so running time is an important fact. Besides, the running time also depends on the performance of computer. The algorithms implemented in this thesis work were tested on an Dell XPS 15 laptop with specification shown in Table 7.1. The other aspect is the accuracy, especially for prediction problem. So mean square error was used to present the error between predicted data and original data.

Component	Specification
Processor	i7-7700HQ, up to 2.8[GHz]
RAM	8[GB],2400[HZ]
GPU	GTX1050 4[GB]
Hard Drive	NVMe 250[GB] SSD

Table 7.1: Specifications of computer

The requirements and consumption data of 11 parts from 11 corresponded suppliers are selected for presenting the algorithms more specifically and the performance of algorithms more intuitively. All 11 parts will be firstly predicted for future information by forecasting algorithms and then be optimized their best quantities in factories to achieve maximum profit for factories. In table 7.2 presents the suppliers' code and parts number of selected 11 parts.

Supplier	Part number
0347A	31694057
A426K	32130197
AD9P4	31386348
AD883	31699307
AEA3C	31377912
AEGFW	31690924
AEHPD	31658471
BA7ZA	31362330
CK06U	31386236
D0RYB	31349975
DXQLA	31455915

Table 7.2: Suppliers and corresponded parts

Notice that the name and location of suppliers' code are mentioned in Table 7.6 in section 7.2.1.

7.1.1 The prediction of demand and consumption

In this section, the results of Recurrent Neural Networks algorithm and regression analysis algorithms are presented and compared. The first step is to compare the results between Recurrent Neural Networks and all Regressors. Then the best algorithm will be used for prediction based on real data.

The data of part 31353931 randomly selected from PLUS system is used for comparing all algorithms. Figure 7.1 shows the quantity tendency of part 3135391, where the horizontal axis is the Call-off date of part, and vertical axis is the amount of part for Call-off. The implementation of regression method and Recurrent Neural Network will both use this data for comparison. Running time and mean absolute error were used to judge the performance of algorithms.

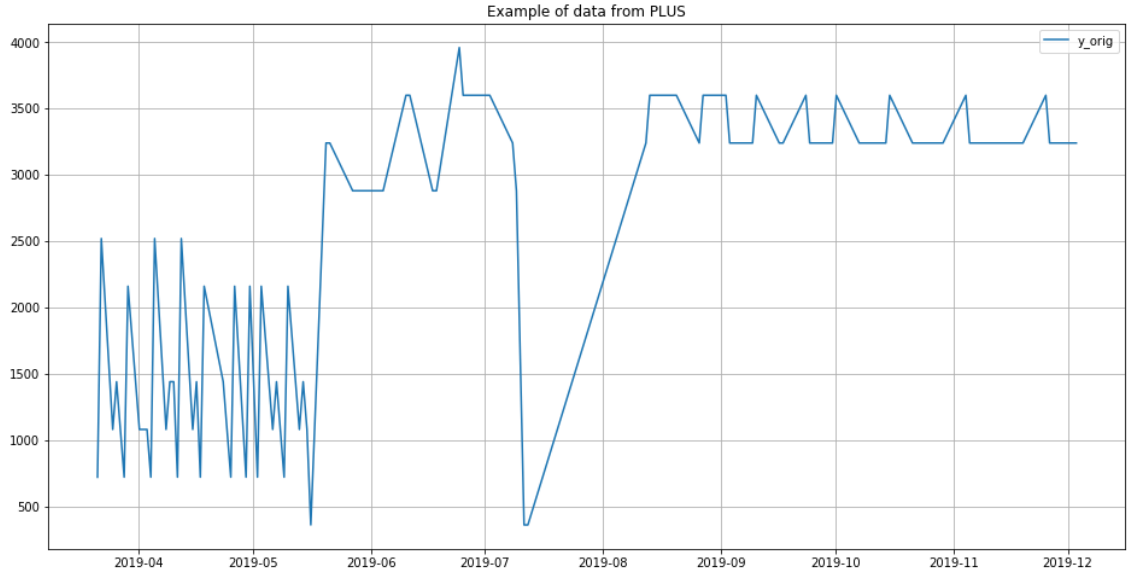


Figure 7.1: An example of presented data of consumption

Firstly, linear regression was used. Linear regression needs fitted data has linear feature, but from figure 7.1, the data set does not show any linear feature, so the result is quite bad. The mean absolute error for train set is 45%, and test set has 15.54 % mean absolute error. What is more, the running time of linear regression was 0.298 seconds. The result of linear regression is presented in figure 7.2.

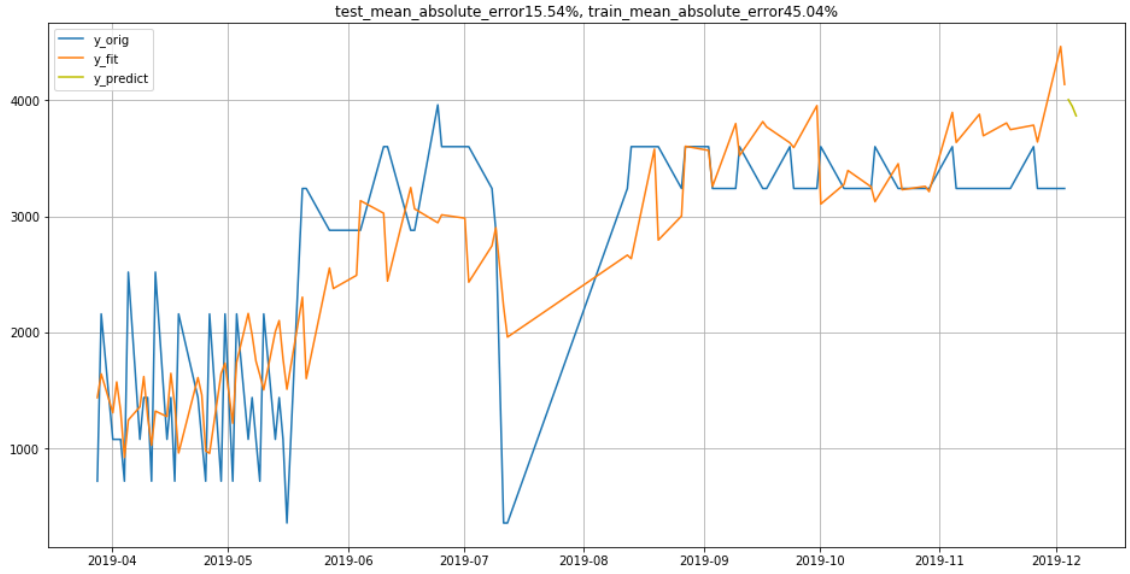


Figure 7.2: Prediction by Linear regression

The next is Random Forest regression. Random Forest regression is able to fit nonlinear data so the result is much better than linear regression. For Random Forest regression, the mean absolute error for train set is 15.56%, and test set has 4.86 % mean absolute error. What is more, it costed 0.382 seconds to finish running the code. The result of Random Forest regression is presented in figure 7.3.

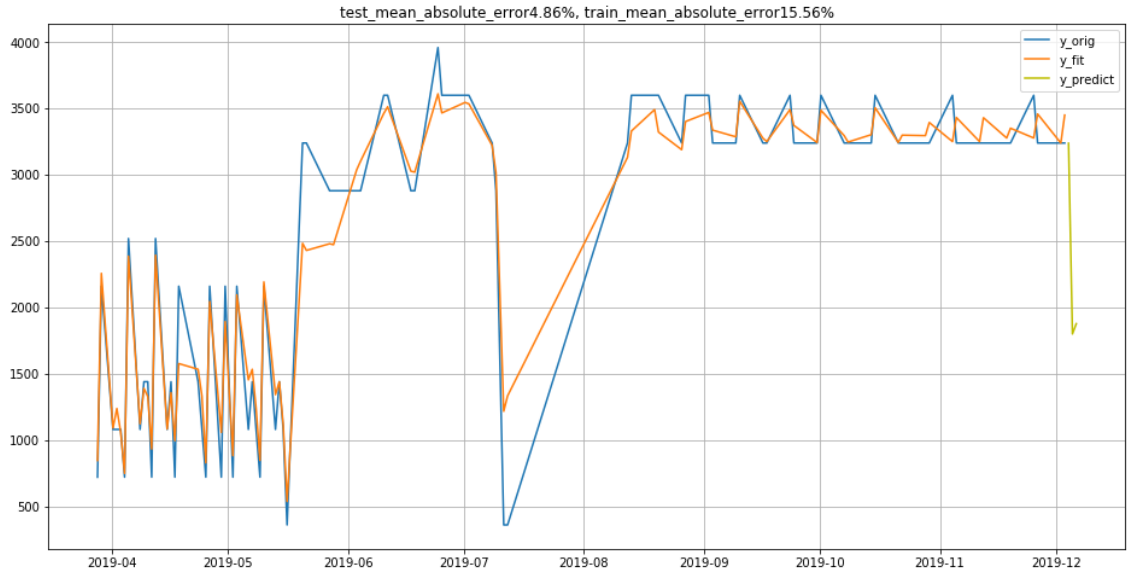


Figure 7.3: Prediction by Random Forest regression

What is more, Gradient Boost Regression Tree is also used for prediction. Gradient Boost Regression Tree is similar to Random Forest regression, both of them can fit nonlinear data set. The mean absolute error for train set is only 7.14%, and test set has 3.70 % mean absolute error. The running time is still a little shorter than Random Forest which is only 0.346 seconds, so Gradient Boost Regression Tress is better than Random Forest regression. The result of Gradient Boost Regression Tree is presented in figure 7.4.

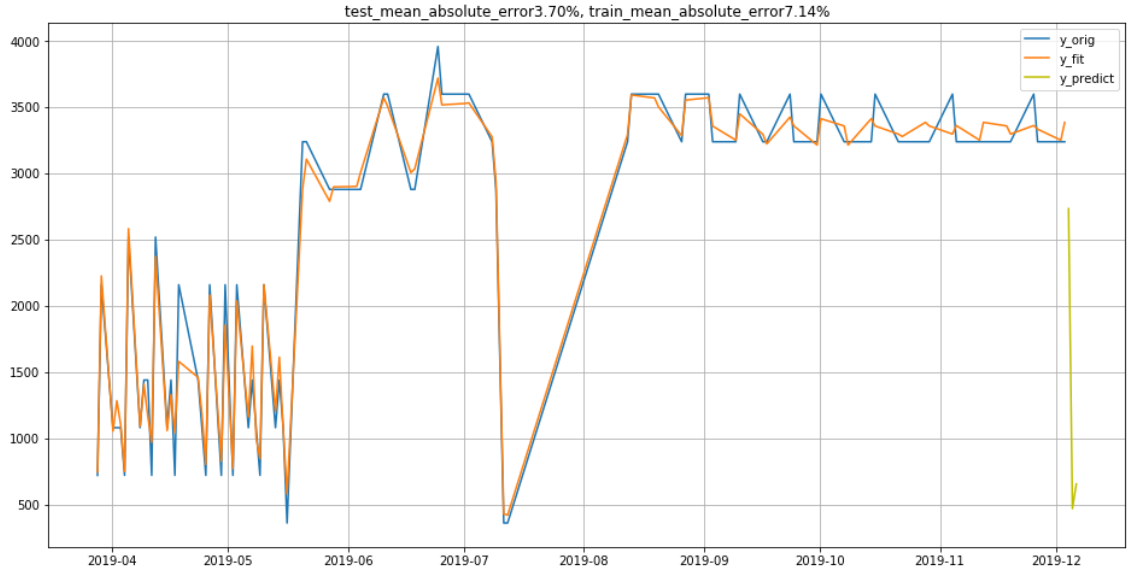


Figure 7.4: Prediction by Gradient Boost Regression Tress

Finally, LSTM Recurrent Neural Networks was implemented for predicting testing data. LSTM Recurrent Neural Networks belongs to supervised learnign which is different with regression analysis algorithms. But in this problem, the performance

of LSTM Recurrent Neural Networks is not good. The mean absolute error for train set is 30.06%, and test set has 9.50 % mean absolute error. The mean error for test set is not higher, this is because the model is over fitting. From figure 7.5, it can be easily seen that there is lag for test sets compared with original data. The reason for this situation is the model directly use current value as prediction data which is also caused by over fitting. Except accuracy, the running time was also not satisfied. It took 198.98s to finish running complete algorithm which is much longer than regression analysis algorithms.

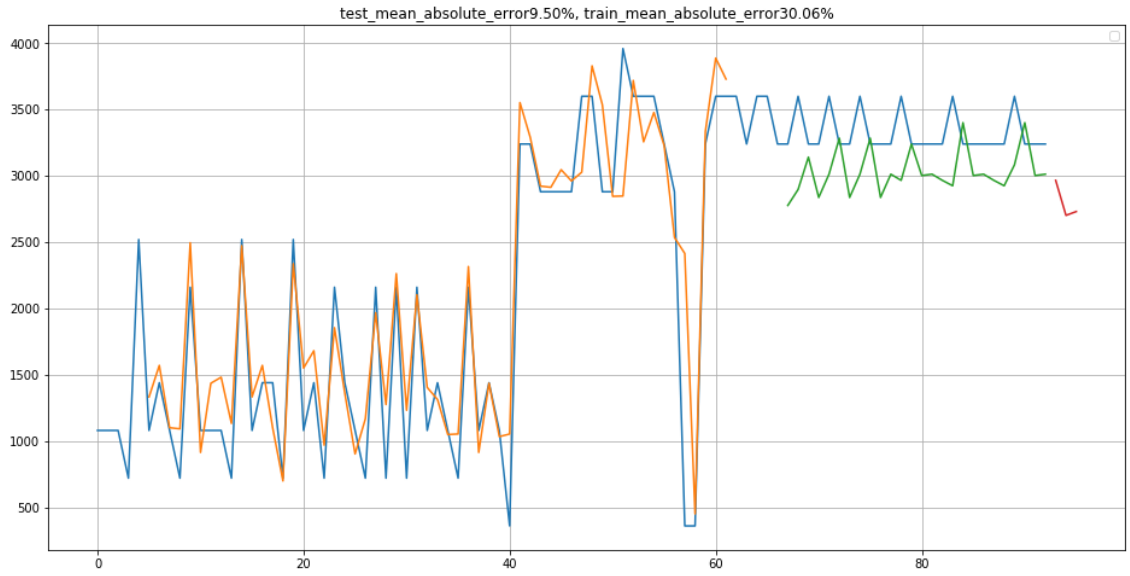


Figure 7.5: Prediction by LSTM Recurrent Neural Networks

All in all, Gradient Boost Regression Trees is selected for predicting selected 11 parts of 11 different suppliers. The figure 7.6 and figure 7.7 shows the result of one part 31377912 from supplier AEA3C as an example. The rest results are shown in Appendix.

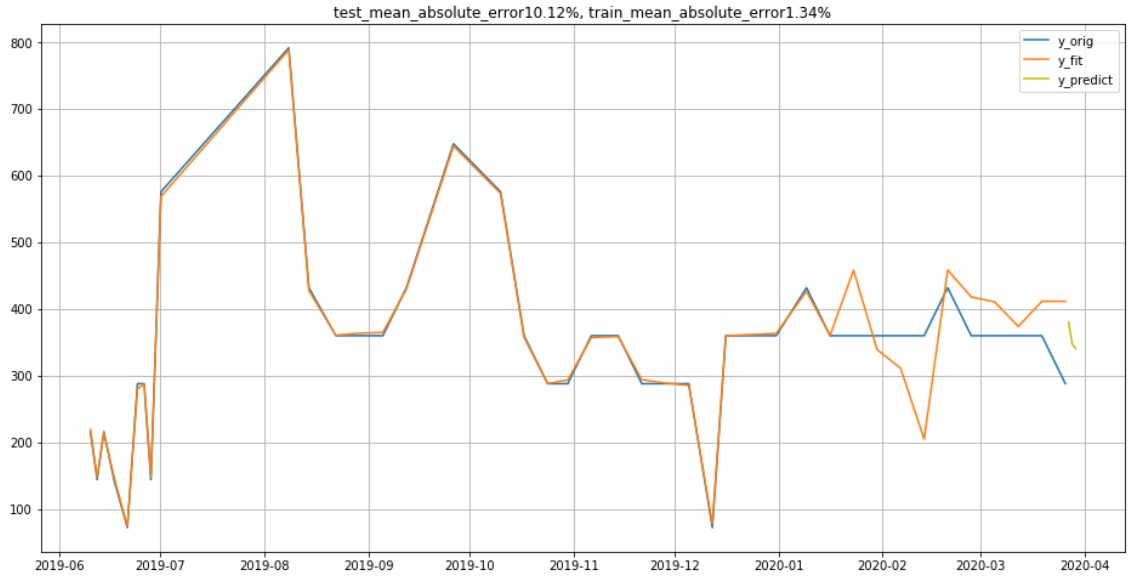


Figure 7.6: An example of prediction for the call-off of part 31377912

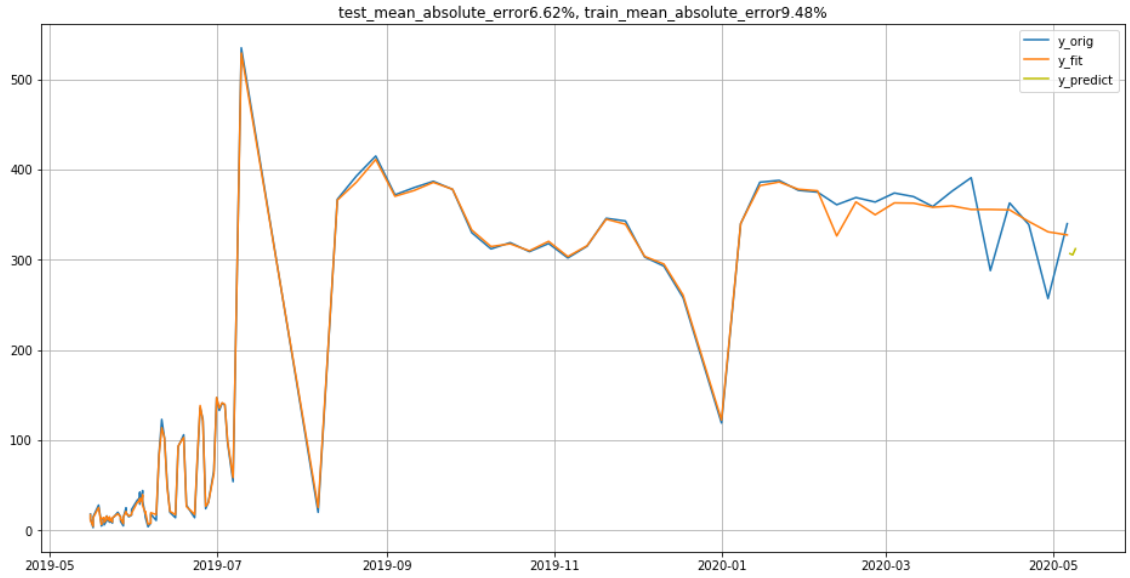


Figure 7.7: An example of prediction for the call-off of part 31377912

The algorithm also predicted the next 3 days' consumption and requirements in the factory which is listed in the table 7.3. The rest predicted results of the rest 10 parts are also listed in the Appendix.

Future date	Predicted Call-off	Predicted requirements	Inventory
2020/3/27 8:00	260.12	303.65	164
2020/3/27 8:00	251.38	301.48	
2020/3/27 8:00	251.68	309.61	

Table 7.3: An example of prediction result

The results will keep using for optimization problem as an example for the next section 7.1.2, and the value of results will round up to the closest multiple.

7.1.2 Optimization of storage in factories

According to the predicted data in table 7.3, parameters are specific defined in this particular problem. Volvo Cars has two factories and a X-doc, one is in Torslanda and the other is in Ghent, the X-doc locate at Czech. One of suppliers for 'Full Truck Load' (FTL) is also locate in east Europe. Base on the distance between supplier and X-doc to two factories, the cost for transporting from suppliers to factories is 5 credits and the transportation from X-doc to factories will cost 3 credits. Since the consumed parts are used for assembling cars, it should be concerned as profits for the factories, and defined as 10 credits.

Discount rate γ is also an important parameters in Markov Decision Process which presents value of future rewards. When γ close to zero, it leads to "myopic" evaluation. And when γ close to one, it leads to "far-sighted" evaluation. The discount rate in this problem was defined as 0.9, because the future best rewards could effect current returns were wanted. In this case, the result can be converged to the best rewards much faster.

What is more, based on the data predicted by regression and Recurrent Neural Network method, the quantities of eleven parts are arrange from two digits to four digits. Since states mean can present how many parts are left in factories and the number of state based on the maximum storage of parts in factories, during the test, the maximum storage was set to 700, there will be 701 states altogether. The running time for 701 states was 6.5 hours which was unworthy. So if the quantities of parts are larger than 1000, the quantities would be counted every 100 parts. In this way, the states will reduce to 11 states. Doing the same process to the quantities of parts larger than 100, but the quantities would be counted every 10 parts. Finally, the states, maximum storage in factories, maximum moving parts from X-doc to supplier can be united in two digits. The maximum storage of parts in factories are 90 unit and maximum moving parts are 20 unit. For example, 20 unit for quantities of moving parts larger than 100 means the maximum moving parts is 200 parts, for quantities of parts larger than 1000 means the maximum moving parts are 2000. With this operation, the running time can be reduced to 0.5 hours, but still be reasonable and accurate.

Parameters	Value
Max storage in Factory	90/900/9000
Max move parts	20/200/2000
Discount rate	0.9
Factory profit	10
Supplier cost	5
X-doc cost	3

Table 7.4: The values set for parameters in maximum storage in factories problem

After the parameters for code were defined, the results were finally get. The total time for running the code is 5000s which is quite satisfied. One of 11 parts 31377912 are selected as as an example. The parts are from supplier AEA3C. The table 7.5 presents the optimal policy for parts 31377912. The left column of the table is the number of parts in factory which is also the states for dynamic programming. The right column is the number of parts should be transported from X-doc to the factory which is also the actions in dynamic programming. For example, the table shows that if there are only 10 parts left in the factory, 200 parts should be transported since the maximum transported parts is 200. And it is easy to find that when there are 390 parts left in the factory, no parts need to be transported, so the best quantity of part 31377912 should be stored in the factory is 390, which could achieve maximum rewards for the factory.

Noticed that the rest results of 10 parts from 10 suppliers are listed in Appendix. With all the results for transporting, the order will be listed for the Milk-run problem.

Number of parts left in factory	Number of parts need to be transited from X-doc
0	200
10	200
...	...
190	200
200	190
...	...
380	10
390	0
400	0
...	...
900	0

Table 7.5: An example of result of storage optimization problem

7.2 Milk-run

The results of this section will be divided in two parts. The first one is the type and amount of the supplies that need to be collected, which can represent which suppliers need to be visited. The other is the optimal visiting sequence of the chosen suppliers. The best path can be found easily with the help of Google Maps.

7.2.1 Selected Suppliers

According to the order form from the X-doc, shown as table 7.6, the result of loading problem has been computed using both reinforcement learning, stochastic method and priority greedy method.

order	Supplier	Quantity	Priority
12	Hanon System Autopal(0347A)	200	1
14	Continental Automotive Czech Republic(A426K)	500	8
4	Benteler SK Automotive Ltd(AD9P4)	1300	9
13	MAHLE Behr Mnichovo Hradiste(AD883)	200	3
5	YanFeng Slovakia Automotive Interior Systems(AEA3C)	200	6
10	Brose Prievidaza, spol. Ltd(AEGFW)	700	10
11	BWI Czech Repblic(AEHPD)	20	4
8	Cooper-standard Automotive Ceska republika(BA7ZA)	190	7
15	Benteler automotive rumburk s.r.o(CK06U)	200	2
3	WITTE NejDeck(D0RYB)	2000	11
2	Autoneum CZ(DXQLA)	90	5

Table 7.6: An example of order form

In stochastic method combine with priority greedy, the selected suppliers and amount of supplies need to be collected are shown as table 7.7. From the output of our MATLAB code, it is found supplier 2, 5, 11, 12, 13 and 15 are selected to be visit and supplier 14, 4, 10, 8 and 3 are abandoned due to the limitations. The truck's capacity is set to 5.0005 units here. Where the 0.0005 is volume of the smallest supplies. The purpose of adding that is to figure out if the algorithm is able to fill the free space of the truck with small size supplies. The probability of proposing priority first policy is set to 0.2 and the iteration time is 100 to make sure the generation of different kinds of policy combinations and the priority first proposal has been active. The code runtime is about 2.8.

Table 7.7: Results of Stochastic with priority greedy method in loading problem

supplier order	amount of supplies	visit or not
12	200	✓
14	0	×
4	0	×
13	200	✓
5	110	✓
10	0	×
11	20	✓
8	0	×
15	200	✓
3	0	×
2	90	✓

The result of reinforcement learning method has been listed in the table 7.8. As it is shown, supplier 2, 5, 11, 12, 13, 14 and 15 has been selected. The discount factor γ has been set to $\gamma = 0.9$ here. Maximum capacity of the truck is 5.0005 units. And the bonus discount for fully loading is 0.3. Iteration time is set to 100 as the one in stochastic method in order to do the comparison.

Table 7.8: Result of reinforcement learning in loading problem

supplier order	amount of supplies	visit or not
12	200	✓
14	1	×
4	0	×
13	200	✓
5	110	✓
10	0	×
11	20	✓
8	0	×
15	200	✓
3	0	×
2	90	✓

7.2.2 Optimal path

Since the suppliers has been selected, the optimal path among them will be generated using Ant-Q, reinforcement learning and stochastic method.
The result of Ant-Q is shown in a visit sequence of suppliers. After that, bring them into the Google Maps and obtain the overview of the optimal route. Using the selected supplier list from stochastic method, which is supplier 2, 5, 11, 12, 13 and 15. Bring them into our Ant system, we obtain the optimal visit sequence(in numerical order): 1, 11, 15, 13, 2, 14, 12, 5 and 1, where 1 is the X-doc. The optimal route overview is shown as figure 7.8. The total path length is 1383km according to it. The runtime of Ant-Q in MATLAB is about 0.1 second.

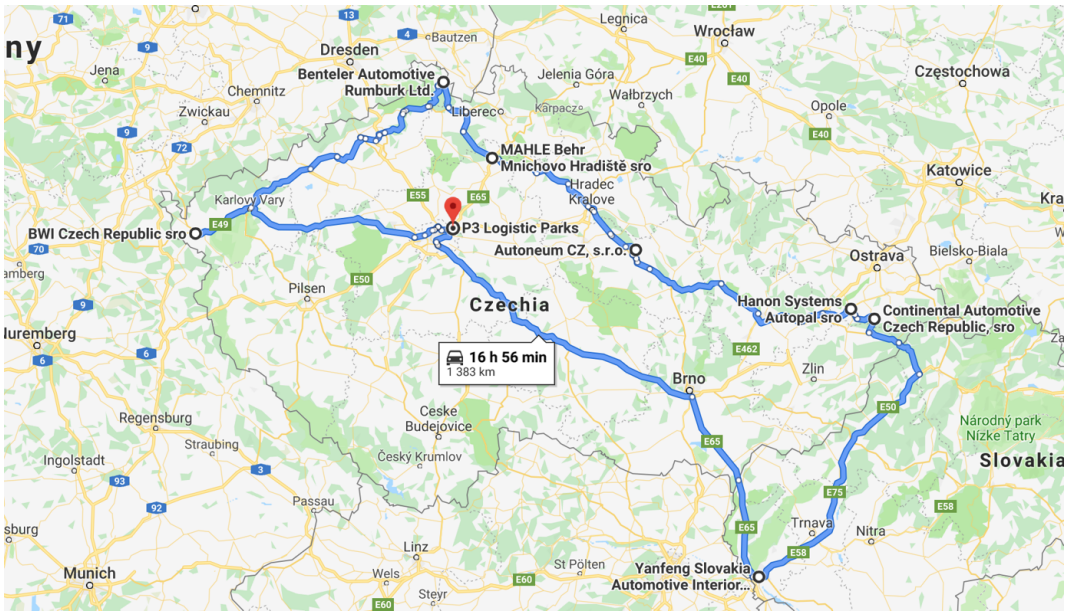


Figure 7.8: Optimal route of Ant-Q algorithm

In ant-Q, the number of agents has been set to 7, which is the amount sum of

suppliers and the X-doc. The learning step $\alpha = 0.1$, optimal path stability q_0 is 0.9, two weight factors $\delta = 1$ and $\beta = 2$, the parameter W, that used to calculate the ΔAQ, is 10. Iteration number has been set to 300 to make sure that the Ant-Q obtain the best result.
According to the reinforcement learning method, the sequence is: 1, 13, 15, 2, 12, 14, 5, 11 and 1. The route in Google Maps is shown as figure 7.9. The discount factor is $\gamma = 0.9$.

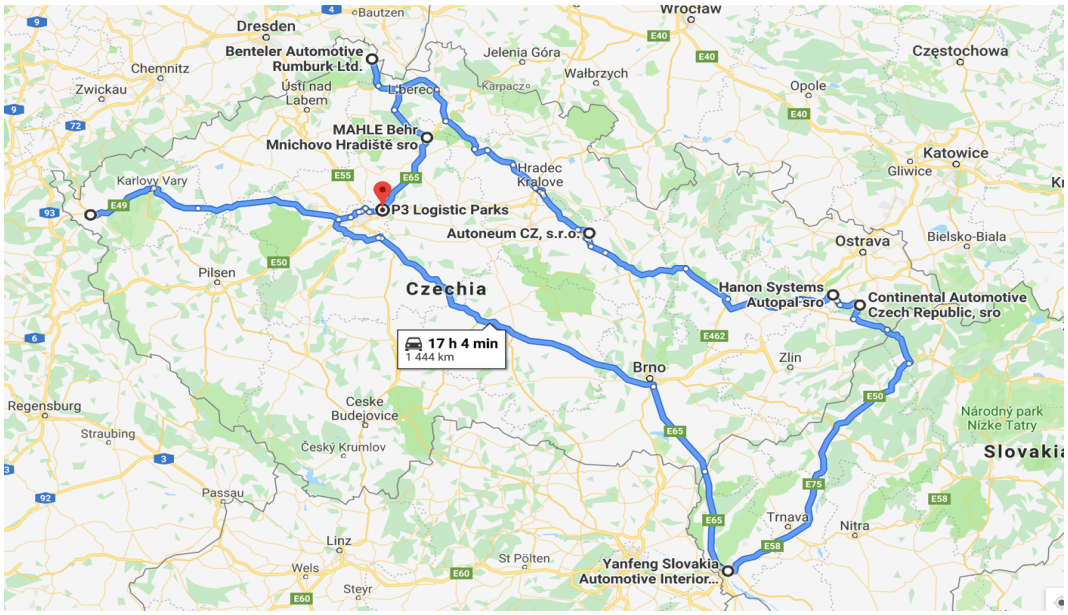


Figure 7.9: Optimal route of Reinforcement learning

As it is shown, the total path length here is $1444km$. RL took 11.899 seconds to get the best result when the iteration number is set to 300, as the same as the one in Ant-Q learning. However, RL do not need many iterations, it converged very fast. The best result can be obtained within 0.048 second.

8

Discussion and future work

This chapter discussed the differences among the methods used for solving X-doc warehousing and milk-run problems. Pros and cons of AI and ML have been discussed compared with traditional optimization methods.

8.1 Management in factories

As results shown in section 7.1, the outcome of algorithms for two problems satisfy the expectations, but there are still some insufficient area need to be improved. So in this section, the results of algorithms will be analyzed firstly, and then future work will be discussed.

8.1.1 Prediction method

In prediction problem for warehouse management, the purpose was to use a time series data set to predict the output value for future date. Normally, the output is just a numerical variable like quantity, temperature, population and so on, so the difficulty for time series prediction problem is to build features as input to find connection between the output. In a word, the performance of predicted result is highly depending on the data sets.

In this thesis work, the information in data sets that can be used is only a series of date and the corresponded value of quantities of parts in factories. So the features built for fitting models are only previous quantities of parts, and regression analysis methods can fit simple, periodic numerical variable perfectly, this is the reason why regression analysis algorithms are much better than LSTM Recurrent Neural Networks. However if the data set has large concussion or does not show any periodic feature, the prediction with regression analysis will occur errors. Especially when the test set has different tendency with training set, it will occur very large error.

However, compared with regression analysis methods in this thesis project, the results of LSTM Recurrent Neural Networks are worse. The reason for this situation still might because the simple data sets with single feature. From figure 7.5, it could be easy to find that the predicted amounts of test set is lag than the original amounts, which means the predicted amounts are over relying on the amounts one day before, that equal to the previous day. The solution for the lag is to add more features in the feature, for instance, the colour of interior according to the seasons.

The gender of customers could also be a feature for training. In one word, adding more features could increase the accuracy of the model. Differing from regression analysis methods, LSTM Recurrent Neural Network can recognize not only the numerical variable as features but also the characteristic like gender, colour, comments for cutomers and so on. Then in the future, the results of LSTM Recurrent Neural Network could be much better and more accuracy.

8.1.2 Optimization method

In optimization problem, the solution for finding the best amounts of parts should stored in the factories is to solve Markov decision process as model by dynamic programming based on reinforcement learning. The results of this solution is good enough, but as normal problem for reinforcement learning is that the time for running code is quite long when there are huge states and actions. There are a few methods for reducing the running time in the future. Firstly, the initial value for the actions were set to all zero in this thesis project, this could increase steps for iteration process. So the initial value for the actions could be changed to the amounts of parts related with the predicted value which might closer to the best value for the actions. What is more, except the dynamic programming to solve Markov decision process, there are still some more solutions for dynamic programming such as Monte-Carlo methods, Temporal-Difference Learning methods, and so on. All these methods can be tried to reduce the running time of code and compare accuracy between each other in the future.

8.2 Milk-run Problem

From the result in the last chapter, it was found that both optimization method and machine learning method can both solve the loading and Travelling Salesman problems. In this section, the differences between them were analyzed.

8.2.1 Loading problem

In loading problem, the aim was to select suppliers which is most wanted. The method depended on the amount, volume of supplies and the maximum capacity of the truck which was used to transport. As shown in table 7.7 and table 7.8, it is found that the reinforcement learning method can use small supplies to fill the free space in the truck while the stochastic method can not. In reinforcement learning method, there was a bonus, which was activated when the truck is loaded without any free space. Once the bonus was active, the reward was remembered in state value function which was able to have effect on the next iteration. That was a learning process. However, the bonus point was not considered in stochastic method, because the policy was generated randomly every time instead of depending on previous factors. Due to the priority first and random proposed policy, supplies were loaded type by type, until the truck can't load any more. After all, the reinforcement learning method made the truck more efficiency. If the ρ, which weigh the relative

probability of random proposed policy and priority greedy policy, was very large, it needed time to make lots of attempts on the different combinations of policies. While ρ was small, the order of suppliers was mainly based on the priority from the order form.
Instead of adjusting manually, reinforcement learning made the machine learned itself, which was the advantage of machine learning in this problem. The future work will be that more suppliers will be considered in this loading problem. Other optimization methods can be implemented in this problem and to see if they can obtain a better result.

8.2.2 Traveling Salesman Problem

After selecting suppliers, the next task is to optimize the route among them. According to the results in the last chapter, both RL and Ant-Q algorithm were able to generate optimal visiting sequence among the selected suppliers. As it is shown in the 7.8 and 7.9, Ant-Q learning approach gave a better result than the reinforcement learning, since this is the first attempt of implementing RL in TSP, as Ant-Q has been studied before. However, as it is mentioned before, RL method didn't need many iterations to get the optimal result. This is because in the step of policy improvement of RL, the action value function, Q, which is just to compute the total distance among the selected suppliers. It is proved that there is potential to implement RL to solve this kind of problems and functions, like value functions, can be improved for a better result.

9
Conclusion

The work of this thesis project has covered implementation and discussion of the AI and ML in field of logistics. Forecasting and optimizing problems of a supply chain system have been modeled, and solved with different optimization methods and ML approaches. The results proved that it is possible to implemented AI and machine learning in the field of logistics. It has been shown that the potential of implementing reinforcement learning in loading problem, travelling salesman problem and warehouse optimization. The implementation of supervised learning in inventory prediction of factories. Thesis work also resulted in comparison of stochastic method and machine learning on optimization problem.

Especially for the time prediction problem. Time series forecasting problem are a difficult type of predictive modeling problem. It is already a popular research direction in economy, warehouse management, weather forecasting, and so on. In this thesis project, a regression analysis method which belongs to supervised learning approach was chosen because the data sets for learning are simple and only time series sets with corresponded value of parts' amount. However, forecasting by deep neural network is still at a very beginning phase, it should have a huge prospect in the future.

In a conclusion that this thesis project proved that AI and machine learning could be a good tool for solving problems in logistic, so it is necessary to have more future work to dig deeper in this area. It could bring a lot of profits for the company.

Bibliography

[1] "Predicting Logistics Delivery Demand with Deep Neural Networks", Yao-san Lin, Yaofeng Zhang, I-Ching Lin, Che-Jung Chang, International Conference on Industrial Technology and Management, 2018

[2] "Can Deep Reinforcement Learning Improve Inventory Management? Performance and Implementation of Dual Sourcing-Mode Problems", Joren Gijsbrechts, Robert N. Boute, Jan A. Van Mieghem, Dennis J. Zhang, KU Leuven, Northwestern University, Washington University in St. Louis, December 17, 2018

[3] "Reinforcement Learning for Solving the Vehicle Routing Problem", Mohammadreza Nazari, Afshin Oroojlooy, Martin Takac, Lawrence, Lehigh University, 21 May, 2018

[4] "Vehicle Routing Problem: Simultaneous Deliveries and Pickups with Split Loads and Time Windows", Yong Wang, Xiaolei Ma, Yunteng Lao, Yinhai Wang, Haijun Mao, University of Washington, 1st November, 2012

[5] "Mastering the game of Go with deep neural networks and tree search", Google DeepMind, 2016

[6] “Ant-Q: A Reinforcement Learning approach to the traveling salesman problem”, Luca M. Gambardella, Marco Dorigo, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, July 9–12, 1995

[7] “Logistics management and strategy competing through the supply chain, fifth edition”, Alan Harrison, Remko van hoek, Heather Skipworth, Pearson Education Limited 2002, Saffron house 6-10, Kirby street, London, 2014

[8] "Reinforcement Learning: An Introduction", Richard S.Sutton, Andrew G.Barto, Second edition, The MIT Press, Cambridge, England, 2018

[9] "Comparison of ARIMA and Random Forest time series models for prediction of avian influenza H5N1 outbreaks", Michael J Kane, Natalie Price, Matthew Scotch, Peter Rabinowitz, BMC Bioinformatics 15, 2014

[10] "Simplified Modeling and Solving of Logistics Optimization Problems", Hanno Friedrich, Jonathan Gumpp, Technical university of Darmstadt, 2014

[11] "Classification and Regression by Random Forest", Andy Liaw, Matthew Wiener, R News, 2012, [http://CRAN.R-project.org/doc/Rnews/]

[12] "Forecasting functions for time series and linear models", Rob J. Hyndman, George Athanasopoulos, Christoph Bergmeir, Gabriel Caceres, Leanne Chhay, Mitchell O'Hara-Wild, Fotios Petropoulos, Slava Razbash, Earo Wang, Farah Yasmeen, R package, April 2018

[13] "Comparison of stochastic and machine learning methods for multi-step ahead forecasting of hydrological processes", Georgia Papacharalampous, Hristos Tyralis, Demetris Koutsoyiannis, Stochastic Environmental Research and Risk Assessment, 2019

[14] "Supply chain inventory optimisation with multiple objectives: an industrial case study", Lionel Amodeo, Haoxun Chen, Aboubacar El Hadji, university of technology of Troyes, France, 2008

[15] "Stochastic local search procedures for the probabilistic two-day vehicle routing problem", Karl F.Doerner, Walter J.Gutjahr, Richard F.Hartl and Guglielmo Lulli, university of Vienna, University of Milano Bicocca, Austria, 2008

[16] "Artifical intelligence in logistics", DHL Custuomer Solutions & Innovation, Represented by Matthias Heutger, DHL CSI, Germany, 2018

[17] "Solving strategic and tactical optimisation problems in city logistics", Paolo Gianessi, University of Paris 13 Nord, Paris, 2014

[18] "Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling", Hasim Sak, Andrew Senior, Francoise Beaufays, Google, USA, 2014

[19] "An Empirical Exploration of Recurrent Network Architectures", Rafal Jozefowicz, Wojciech Zaremba, Ilya Sutskever, Google, USA, 2015

A

Appendix 1

A.1 Tables

Table A.1: Suppliers list

X-doc: P3 Logistic Park	1
Autoneum CZ(DXQLA)	2
WITTE NejDeck(D0RYB)	3
Benteler SK Automotive Ltd(AD9P4)	4
YanFeng Slovakia Automotive Interior Systems(AEA3C)	5
Hella Slovakia Signal - Lighting(AD0UM)	6
Volvo Car Corporation Engine Skovde(C7CUL)	7
Cooper-standard Automotive Ceska republika(BA7ZA)	8
Promens, Inc (AEJFK)	9
Brose Prievidaza, spol. Ltd(AEGFW)	10
BWI Czech Republic(AEHPD)	11
Hanon System Autopal(0347A)	12
MAHLE Behr Mnichovo Hradiste(AD883)	13
Continental Automotive Czech Republic(A426K)	14
Benteler automotive rumburk s.r.o.(CK06U)	15
Rochling Automotive Koprivnice S.r.(AECQX)	16

Table A.2: Shortest Distance among suppliers corresponding to the supplier code(units: km) part 1

	X-doc	DXQLA	D0RYB	AD9P4	AEA3C	AD0UM	C7CUL	BA7ZA
X-doc	0	141	161	308	342	343	1195	146
DXQLA	138	0	299	212	246	236	1285	75.9
D0RYB	162	304	0	444	478	478	1163	303
AD9P4	308	212	442	0	39	125	1487	166
AEA3C	343	246	477	38.2	0	123	1523	202
AD0UM	344	246	478	125	121	0	1446	202
C7CUL	1195	1331	1160	1488	1523	1447	0	1348
BA7ZA	146	76.1	303	167	201	202	1348	0
AEJFK	315	178	449	132	180	68.1	1392	174
AEGFW	406	309	540	177	165	66.3	1428	264
AEHPD	185	328	46.4	467	502	502	1191	327
AB6RA	355	175	489	224	222	105	1332	213
AD883	54.4	124	215	325	359	359	1210	160
A426K	378	199	512	254	242	111	1332	237
CK06U	120	193	190	388	464	422	1081	223
AECQX	320	189	503	244	283	115	1330	205

Table A.3: Shortest Distance among suppliers corresponding to the supplier code(units: km) part 2

	AEJFK	AEGFW	AEHPD	AB6RA	AD883	A426K	CK06U	AECQX
X-doc	312	404	184	354	56.9	329	123	325
DXQLA	170	298	322	175	128	195	198	191
D0RYB	450	540	46.5	490	217	509	182	505
AD9P4	132	176	465	224	321	206	388	246
AEA3C	170	181	501	270	357	230	465	280
AD0UM	68.1	65.7	501	105	357	112	423	115
C7CUL	1397	1431	1189	1332	1203	1333	1077	1331
BA7ZA	173	263	326	191	157	210	223	199
AEJFK	0	120	472	61.2	329	71.3	395	69.2
AEGFW	120	0	563	155	419	144	485	154
AEHPD	473	564	0	513	241	533	226	529
AB6RA	62.1	155	512	0	296	21.2	485	15.5
AD883	294	421	243	292	0	311	71.9	307
A426K	71.4	144	535	21.2	320	0	485	10.8
CK06U	363	484	224	361	69.2	487	0	376
AECQX	75	153	526	15.6	310	10.9	380	0