London School of Economics and Political Science

Department of Statistics 2021/2022

Reorder Point Optimization

Group 2
Candidate Number:
34500
34929
37792

Submitted for the Master of Science, London School of Economics, University of London

Contents

List of Figures

List of Tables

1 Executive Summary

Reorder point optimization is crucial in modern industry. The reorder point optimization focuses on inventory management. The inventory level should be kept at an optimal level, which can guarantee seamless product production and keep the holding cost low. In this project, we focus on a specific reorder point optimization problem that happened in some of Siemens' production factories.

Before we go into our project, there are several concepts we need to address. Below is a visualization of the relationship between reorder point and inventory level. At the start of the production, we have a certain amount of inventory materials. When the production goes on, the material inventory gradually drops. When the inventory drops to a certain point which we called reorder point, we reorder a certain amount of materials from exterior suppliers. However, it will take several days to fulfil the purchase, this time interval is called lead time. There will be times when the materials can not fulfil the production when the material is dropped to 0. During this period, production must halt. The percentage of time that the inventory is above 0 is called service level, which is usually denoted by a certain percentage.

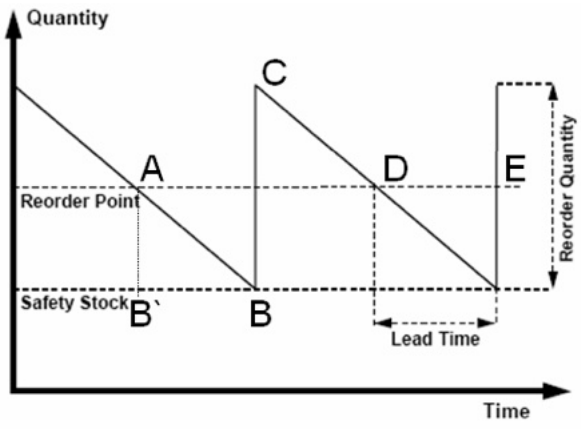

Figure 1: Illustration of Inventory Levels

In this project, there are two questions we want to answer. Firstly, we want to understand and predict the future order demand. By predicting future order demand, we will know what level of inventory we would need in the coming future. In our case, orders are usually coming in on weekdays, so we need to build a prediction method that can capture such characters. We explore the use of Prophet (Facebook Developed

Tool), long short term memory and ARIMA methods. We determine that ARIMA fits best in this case situation.

Secondly, we want to optimize the costs. There are three types of costs we want to minimize here. Firstly is the inventory cost. Inventory cost relates to those costs that happened during storage. When you put materials into storage, they will incur inventory costs. When factories store necessary raw materials, such as plastics, and machine parts, there are costs related to that storage. They occupy spaces and require certain maintenance. This type of cost is called inventory cost. Another type of cost is called stock out the cost. If you are running a shop, and a customer wants to buy a product that has empty stock, you can not offer such a service. Thus, you lose revenue because you would earn if you have that product. The same applies to the factory, when certain materials are running out, the whole assembly line would stop. And the revenue that could have been earned, if those products are produced, would lose. This type of cost is called stock out cost. Thirdly, there is cost related to the order and delivery of the materials, the material costs money and the material delivery also costs money per delivery.

We use two methods for the optimization problem. The first approach is reinforcement learning. In the reinforcement learning method. We solve the optimization problem by giving an optimal action, which is the quantity of the reorder delivery. We develop an agent, where the action refers to the quantity, reward refers to the total cost, and state refers to the current inventory level. Through Q-learning, the agent will try to select the optimal action given the current state.

The second approach we use is mixed integer programming. Using the branch and bound method, integer programming can be optimized rapidly. A mixed integer formulation considering fixed ordering cost, unit ordering cost, inventory cost and stock out penalty is established. Through the MIP method, we succeed in providing a prompt and accurate stock order solution given the forecast future demand.

We evaluate the two methods and determine that MIP method is better than the reinforcement learning method in terms of its optimization capacity and running speed. The MIP method can achieve a service level of 99% while performing much better than the baseline.

2 Abstract

In the supply chain system, safety stock is one of the most significant roles that enable the whole system to run well. Differ from classical safety stock optimization models which only focus on safety stock level while order quantity is predetermined based on simple rules, our model built by reinforcement learning and mixed integer programming optimizes both safety stock level and demand parameters. This allows the models to no longer be constrained to the existing data and static situation but can dynamically optimize and adjust the safety stock.

Keyword: Supply chain; Safety stock; Machine learning; Reinforcement learning; Dynamically

3 Introduction

Supply chain management can be viewed as a demand propagation problem, and its final goal is to meet the needs of customers. During this process, there are various factors causes uncertainty and finally influence customer experiences. Coping with these emerging uncertainties, therefore, is one of the major issues in supply chain management. Based on a number of study, safety stocks are considered to be the most robust strategies to prevent inventory shortage and to handle supply and demand variability. However, with the advancement of globalization and the improvement of transportation and storage technology, the supply chain has become increasingly complex. This makes the traditional inventory optimization method which mainly based on personal experience and simple calculation become no longer applicable. In this case, the industry introduced newly emerging technologies in the field of artificial intelligence, offering promise in finding optimal solution.

To minimize operating costs outside of production, companies have to manage the operating plan of raw materials and products. How to define a method to maintain effective supply chain management can be a significant issue. In supply chain optimization, enterprises are assumed to have a specific number of material suppliers, material storage and production plant. The optimization task refers to a lead time allocation plan. In this plan, materials needed in the following weeks or months are scheduled in supply in order to ensure sufficiency, and control storage costs at the same time. The allocation states the future reorder point, which is the time to call suppliers for new material delivery. In this process, a demand pattern is assumed, and multiple kinds of costs should be considered, for instance, holding cost and delivery cost. The problem being faced at this point is the uncertainty, which can be generated by several factors: service level, lead time and demand volatility. To overcome these uncertainties, researchers suggest a safety stock level guarantee stability for normal production.

The goal of this study is to develop a comprehensive supply chain plan including demand forecasting and a safety stock solver is modelled. To understand future demand patterns by historical orders, a time series prediction method is proposed. To solve the optimal supply based on future demand, a mixed integer programming and reinforcement learning model are constructed.

4 Literature review

Safety stock optimization has taken the attention of researchers in recent 30 years. Some of the studies related to this problem are summarized as follows.

Traditional dimensioning safety stock strategy has a common assumption on demand pattern distribution, typically modelled as normally distributed or other stationary patterns. Variation of demand is being considered. The early paper usually discusses the probabilistic aspects of safety stocks. For example, Kelle[1] figured out a solution of an approximate formula. Taking into consideration of random delivery, she proposes a safety stock formula to ensure the service level of the product. She also proposes a more superior and efficient model which takes both randomized demand and delivery into account. Dar-El and Malmborg[2] constructed a service level-based model which discussed the inventory cycle replenishment. Their approach abandons the traditional probabilistic view on the inventory safety stock. They propose that the inventory can drop below the safety stock, as long as the replenishment arrives before the stock empties. Then, in the aspect of uncertainty factors caused by stochastic demand, Lu et al.[3] focus on solving stochastic inventory control problems by investigating deeply and developing an OR-based method for non-stationary stochastic demand and random supply yield. Their methods are able to generate a safety stock level under a predefined service level. Besides the stochastic view method on safety stock, mixed integer programming can be a classical method in safety stock operational modelling. The classical incapacitated lot-sizing problem was first introduced by Wagner and Whitin[4], assuming that the demand over a specific time length is known with certainty. They developed a minimum total cost inventory scheme in a time series environment. On this basis, later, Küçükyavuz and Pochet [5] added the uncertainty of backlogging, the probability that delivery delays, and also the cause of customer loss. Further on, service level constraint was taken into account by Gade, D. and Küçükyavuz [6]. In the light of reviewed studies, a mixed-integer programming model is proposed, considering stochastic pattern, backlogging, and service level constraints.

5 Data Overview

5.1 Supply Chain Data Format

In this research, we have access to Siemens's data for the five years, from the year 2015 - 2020. We truncate data to remove the last year of the data, the year 2020. It is because the coronavirus outbreak impacts Siemens's supply chain and production. The data contains four columns. And their meanings are listed below.

Column Name	Description
DATE	The date of record
MATERIAL_NUMBER	The type of material
PLANT	The plant which requires the material
QUANTITY	The amount of the material needed

Table 1: Data Description

The DATE column is the date of the record, ranging from 2015 to 2020. We do not use the year 2020 data as a result of the coronavirus outbreak. The whole data we use is from 2015 to 2019. The first 3 years and 9 months of data is used as training data and the last three months of data is used as testing. The MATERIAL_NUMBER column is the type of material, and the PLANT column is the plant which requires the material. Due to information security policies, we are unable to acquire the specific material and plant name. There are in total 8 types of materials, and we name them A, B, C, D, E, F, G, H. There are 2 plants which require materials, and we name them X1 and X2. Below is a table showing the example data.

DATE	MATERIAL_NUMBER	PLANT	QUANTITY
01.02.2016	A	X1	179
01.02.2016	A	X2	747
01.02.2016	B	X2	4588
01.02.2016	B	X1	1781
01.02.2016	C	X1	0
01.02.2016	D	X1	630
01.02.2016	D	X2	2586
01.02.2016	E	X2	10006

Table 2: Sample Data

5.2 Parameter Data Format

In our research, several materials are preset. For each material, we have different supply cost parameters. And for each delivery, we have delivery cost parameters. Each material has unique lead time, inventory costs, stock out costs and unit ordering costs. Lead time describes how long it takes from the order to the delivery of the material. Inventory costs, stock out costs, and unit ordering costs reflect the costs to store the material, the lack of inventory costs incur when customers come, and the costs associated with each unit material purchase. Aside from the costs related to material, there is a fixed ordering cost incurred when ordering one delivery of certain materials.

MATERIAL_NUMBER	Lead_Time_Days	Stock_Out_Costs	Inventory_Costs
A	12	0.05	0.05
B	12	0.05	0.1
C	19	0.05	0.05
D	12	0.05	0.1
E	12	0.05	0.05
F	12	0.05	0.1
G	12	0.05	0.05
H	19	0.05	0.1

Table 3: Material Parameter Values

Cost	Value
Unit Ordering Cost	0.02
Fixed Delivery Cost	200

Table 4: Fixed Material Costs Values

6 Methodology: Order Demand Prediction

In this project, the whole process can be divided into three steps: 1) order demand prediction; 2) Material clustering and stocking strategy; 3) reorder point optimization, and this paper only focuses on the first and third steps. The first step is grounded on statistical methods and machine learning technologies like neural networks. While the third is based on reinforcement learning and mixed integer programming.

In this research, we explore three different ways of order demand prediction. We explore ARIMA-related methods, Prophet, and LSTM. ARIMA is a statistical method focusing on the statistical prediction of time series data. The prophet is a package developed by Facebook focusing on the rapid computation of time series data prediction. LSTM is a machine learning algorithm that can predict the future of time series data. The result shows that the ARIMA-related methods perform best in our research environment.

6.1 Prophet

Prophet is a Facebook-developed time series data forecasting tool. It will produce results automatically given the input time series data but allows some level of human intervention. Compared to LSTM and ARIMA methods, it can predict large-scale data quickly and automatedly. However, Prophet does not possess the same level of accuracy that LSTM and ARIMA have. Prophet uses an additive regression model. Compared to traditional models, its novelty is that it can provide different components in a regression model automatically. Its algorithm includes the year, month, and week components. In addition to that, it also provides an important component - the holiday. Since factories usually close on holidays, this analytic advantage is important. Generally speaking, Prophet is a good tool to produce forecasts quickly, but its lack of human manipulation factors limits its prediction accuracy.

6.2 LSTM

Long short-term memory is a special type of recurrent neural network(RNN). Compared to normal RNN network, the LSTM network can possess information for a longer period of time. Therefore, LSTM is more advantageous in predicting time series data.

Compared to RNN, LSTM solve the problem of vanishing or exploding gradients. Normal machine learning algorithms have fully connected layers. However, there are two separate states within its training. The LSTM models has two states, cell states (above) and hidden states (below). Besides the cell states that carries short term memory forward, the hidden states are designed in such a way that they could carry memory for a longer period of time. There are predefined parameters that can control the speed of memory by selectively memorizing relevant information. These two layers communicate with each other every time when the training move forward. And the hidden memory will provide additional knowledge compared to cell state memory when providing predictions.

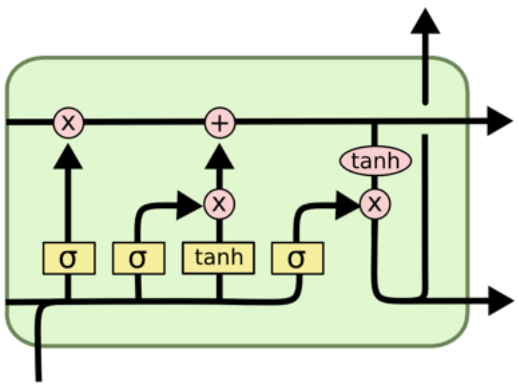

Figure 2: LSTM Algorithm Illustration

We use time series for the past 14 days to train and predict the demand for 1-day forward. LSTM fits best when the time series data fluctuate in certain patterns, but the order demand has sudden spikes some times and does not follow a regular pattern. LSTM does not perform very well in this case.

6.3 ARIMA

ARIMA is a type of autoregressive integrated moving average model. We use statsmodels package in python to analyze the time series data. In ARIMA, there are three terms we need to estimate, p, d, q, which refer to autogressive, differencing and moving average terms accordingly. We use the grid search method to determine the optimal p, d, q values. We explore different combinations of p, d, q, where p is from 0-6, d is from 0-3, q is from 0-6. The different combinations of p, d, q are tested and the output model is assessed using residual sum of square(RMSE). The model with the lowest RMSE is adopted.

We compare the three methods above and find that the ARIMA method is generally the best in terms of RMSE. In the given 13 different combinations of product demand forecast, ARIMA achieves the best in 6 of them. In the rest of the combinations, the ARIMA method does not perform much inferior compared to the other 2 methods. So, the ARIMA model can generally represent a good prediction model for the time series data we encounter in this project.

Product	**Plant**	**LSTM**	**Prophet**	**ARIMA**
A	X1	301.99	137.14	199.10
A	X2	420.258	875.81	921.44
B	X1	726.253	911.98	590.12
B	X2	1838.10	1251.21	608.45
C	X1	12.10	6.50	8.20
D	X1	433.03	230.61	227.15
D	X2	2679.11	3581.20	3472.55
E	X1	1348.255	1163.94	903.84
E	X2	4801.64	6191.50	2821.48
F	X1	1.81	30.14	41.61
F	X2	1.14	108.58	149.258
G	X2	359.57	220.70	218.15
H	X2	19.81	101.44	55.89

We then explore deeper into the ARIMA method. For the time series method, we apply ARIMA and ARIMAX models and process the original data in two ways: 1)do log first and then make difference, and 2)only do difference. After the preprocessing, plot the series and find they are still unstationary. Thus, we use the seasonal decompose function to decompose the series into three parts: seasonality, trend and residuals, and use time series models to simulate trend and residuals separately. For the ARIMA model, we use the first 4 years' data as the training dataset and separately use week and month as a unit step to do a rolling forecast. After simulating the test sequence, we calculate the RMSE of the test dataset and do a gird search to adjust the parameters. And for the ARIMAX model, except for introducing local GDP and workday as the exogenous variable respectively, the other steps are consistent with the ARIMA model. The details of the results of each model are presented in the following table. The native method refers to the original ARIMA method, log method refers to the ARIMA method that uses the log of data as input. Workday refers to the ARIMA method that takes holidays into account, and GDP refers to the ARIMA method that takes GDP into account.

Series	**native**	**log**	**workday**	**workday log**	**GDP**	**GDP log**
(A,X1)	208.698	239.116	199.454	233.280	199.101	235.810
(A,X2)	913.975	377.523	917.008	261.551	921.437	746.2516
(B,X1)	608.801	800.653	667.810	830.2549	590.116	863.732
(B,X2)	639.231	1126.226	675.2599	950.021	608.447	1513.652
(C,X1)	7.608	3.677	8.852	3.669	8.200	4.134
(D,X1)	228.825	292.945	234.950	326.484	227.153	301.470
(D,X2)	3115.661	2013.013	3237.2536	1881.618	3472.551	1949.773
(E,X1)	977.827	1403.525	955.989	1325.411	903.836	1338.816
(E,X2)	2724.199	5182.895	2980.538	5409.258	2821.476	4537.135
(F,X1)	42.798	36.911	43.206	37.141	41.608	38.084
(F,X2)	144.861	195.297	140.063	198.2599	149.2577	200.702
(G,X2)	232.410	76.944	227.521	76.854	218.152	73.603
(H,X2)	53.666	33.086	53.627	33.805	55.893	31.001

Table 5: Result RMSE for Time Series Models

From the table by calculating the average RMSE of each model we find out that ARIMAX whose exogenous variable is local GDP performs best. And combining the plots of each series, we find that when data volatility is low, the log process can significantly improve the fitting rate, while data with high fluctuation making difference is a better choice.

7 Methodology: Safety Stock Optimization

Supply chain management(SCM) is a problem of programming production batches, inventory levels, and delivery methods so that market demands can be satisfied at the lowest cost. Stock optimization is one of the most important components of SCM problems and supply chain risk management(SCRM) issues. In the supply chain, we start with raw materials produced by suppliers, sent to material storage, made into products as scheduled, delivered to a distribution centre, and finally allocated to stores, e-shops, and other client-side. When dealing with supply chain issues, as a 'function of the cycle service level, demand uncertainty, the replenishment lead time, and the lead time uncertainty, the biggest problem is the hysteresis and uncertainty in each process. For each material or product, at the time of shipment, a certain delivery time is required, which is called the 'lead time'. Lead time means the latency between the initiation and completion of a process. When the inventory is about to fail to meet the expected demand, an application for goods transfer should be made in time, which is called the 'reorder point'. For uncertainty, we set the 'service level', which gives the percentage of the supply chain that should be achieved. The service level is expected to ensure the flexibility of the planned supply chain. Dealing with uncertainty is an important issue in business production: even a small amount of uncertainty in the supply chain can lead to early or late delivery. If the delivery time is too early, the goods will face the problem of expiration, invalidation, and excessive storage costs; If the delivery time is too late, the consequences will be more serious: compensation for non-performance of the contract, loss of customers and other problems. Consequently, to avoid the problem of late delivery, we apply the concept of 'safety stock'. Safety stock means the stock threshold value that manages to meet the emergency caused by any of the uncertain factors, by reserving a buffer to satisfy the temporary situation. Of course, too large a value of safety stock is also bad, which will greatly increase the storage cost.

7.1 Reinforcement Learning

7.1.1 Markov Decision Process

In this project, the problem of inventory optimization can be modelled as a Markov Decision Process (MDP). As shown in the Figure 2, we call the learner and decision maker as agent who is presented in state S_t at any given time t. And the thing it interacts with, comprising everything outside the agent is called the environment. Each step, the agent takes an action A_{t+1}, receiving a numerical reward R_{t+1} and moves to the new state S_{t+1}.

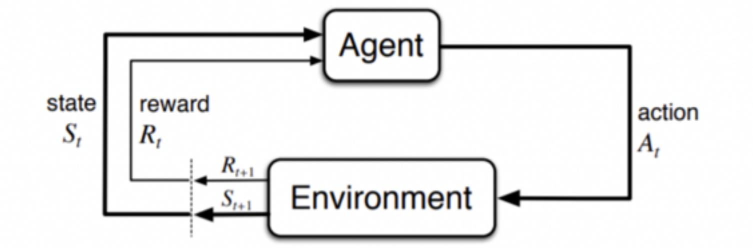

Figure 3: The Agent–environment Interaction in a Markov Decision Process

7.1.2 Q learning

Typically, people focus on the value function $V(s)$ which represents the expected values of the reward for agent in given situation s, trying to choose action maximum $V(s)$.

$$V(s) = max_a E(R_s^a + \gamma V(s'))$$

Here R_s^a is the reward for state s that take action a and γ represents a discounted rate we defined. However, since this inventory optimization process is too complicated, it will be time consuming for us to calculate the value function through a whole map of the environment. To solve this, we suppose to use model free and value-based algorithms which can avoid directly calculate the $V(s)$ function while estimating the rewards, and Q learning is one the most well-known ones. For Q learning we use state-action function, $Q(s, a)$ which called Q value instead. The meaning of this function is the expected cumulative reward we have got when choose an action a in state s and following the optimal policy in the rest steps. We write optimal Q function in state s

by taking action a as $Q^*(s, a)$:

$$Q^*(s, a) = \max_{\pi} E[r_0 + \gamma r_1 + \gamma^2 r_2 + ...] = \max_{\pi} E[\sum_{t=0}^{\infty} \gamma' r_t]$$

Then based on the Bellman Optimality Equation, we can transfer above formula into:

$$Q^*(s, a) = E[r_t + \gamma \max_{a'}(Q^*(S_{t+1,}, a')|A_t = a, S_t = s]$$

To get the optimal policy in Q-learning, we first start at a random policy update the Q function value repeatedly. After several iterations, the value of Q function will finally converge to the optimal Q^*.

The process of Q-learning algorithm can be summarized in three steps. First, determine the learning rate α and exploring rate ϵ (greedy) and randomly initialize $Q(s, a)$ in the Q-table. Second, generate lots of random number which are between 0 to 1 and compared them to the $\epsilon(greedy)$. If it large than $\epsilon(greedy)$, then it use exploration to generate $Q(s, a, r, s')$. If less, it use exploitation to generate another $Q(s, a, r, s')$. Third, calculate $Q(s, a)$=$Q(s, a) + \alpha(targetQ(s, a))$ and continuously repeat last two step until the $Q(s, a)$ function converge to the real value $Q^*(s, a)$.

7.1.3 Deep Q Learning (DQN)

Based on the above concept, we know that the main drawback of Q learning is the expensive learning process for the agent as every state-action pair should be frequently visited during the process of convergence. However, in this inventory optimization process, the combinations of states and actions can be extremely large which makes the learning process unacceptably slow. To solve this, we switch to deep network Q learning (DQN), using deep learning to evaluate the Q value. Instead of using Q-table to processing the Q learning, DQN algorithm using a neural network. Mnih et al. (2015) mentioned that using DQN can make RL agents achieve a high level of performance. In general, DQN uses two key technologies to improve the stability of Q-learning, one is the replay buffer and the other is the target network. Because the Q-table will update for each state-changing which may lead to disability in the Q-learning algorithm. Then the reply buffer is introduced to reduce the frequency of updates. We update the Q-value by iteration and make it converge to the target (real value) in Q-learning, however, with each update of the Q function, the Q function used for target calculation also changes, i.e. the "criterion" for target calculation is always changing. Therefore, we set up a separate target network, independent of the Q table to reduce the variability of the target calculation method. Algorithm 7.1 describe the Deep Q learning.

Algorithm 1 Algorithm 7.1 DQN with Expression Replay

```
Algorithm Parameters:
   Learning rate $\alpha$
   Exploration rate $\epsilon > 0$
   Initialize replay memory D to capacity N
   Initialize action-value function Q with random weights
Begin:
Loop for each episode:
Initialize sequence s$_1 = x_1 andpreprocessedsequence \phi_1 = \phi(s_1)$
    Loop for each step of episode:
       With probability step $\epsilon selectarandomaction$a$_t$, or otherwise select $a_t = argmax_a Q(\phi(s_t), a)$
       Choose action $a_t$ and get reward $r_t$
       Set $s_{t+1}, a_t$ and preprocess $\phi_t + 1 = \phi(s_{t+1})$
       Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
       Sample batch of transitions randomly from D
       Set $y_j = r_j + \gamma \max_{a'} Q(\phi_{j+1}, a')$
       Perform gradient descent on $(y_j - Q(\phi_j, a_j))^2$ with respect to the network parameters S ¡- S'
    Until S is terminal
 End
```

7.1.4 Dueling DQN

However, using $Q(s, a)$ always over-estimate the real value of Q*(s, a) which prevent the Q function to converge. Therefore, the biases between the estimator and real value become larger and we need to do some improvement on this model. Dueling DQN is a technique used to solve this. Having minor changes on DQN algorithm, it can improve the performance significantly. So, in this case, we defined the advantage function A(s,a) as the difference of Q function and value function V, which can be written as A(s,a)=Q(s,a)-V(s). In the same state s, the sum of advantage function value for all visible actions is 0, because the expectation of rewards of all actions at state s is exactly the value function of s. Hence, in Dueling DQN, the Q value could be written as:

$$Q_{\eta}, \alpha, \beta(s,a) = V_{\eta}, \alpha(s) + A_{\eta}, \beta(s,a)$$

Where $V_{\eta}, \alpha(s)$ is value function of state s, and $A_{\eta}, \beta(s,a)$ is advantage function for each action a at state s, indicating the difference in taking various actions. η is a network parameter shared by value function and advantage function. It's generally used in the neural network to extract the first few layers of features. α and β are the parameters of the state value function and the advantage function, respectively. Under such a model, we no longer let the neural network directly output the Q value, but train the branches of the last several layers of the neural network to output the value function and advantage function separately, and then add them together to obtain the value. The structure of Dueling DQN is shown as figure 3:

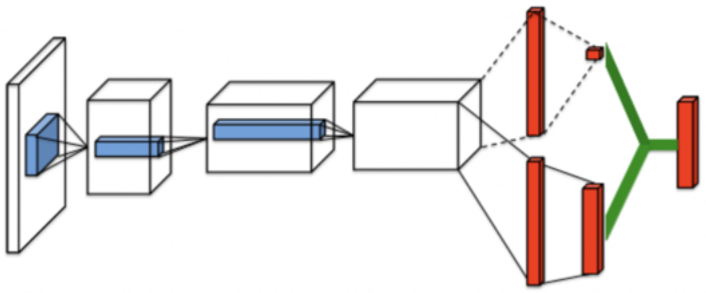

Figure 4: The Agent–environment Interaction in a Markov Decision Process

The advantage of splitting Q values into two functions is that in some situations, the agent only focuses on the value of the state and ignore the differences caused by different actions. At this time, modeling these two separately enable agent to handle the states that are less related to the actions much better.

7.1.5 Experiment

The dataset is generated from the demand forecast part above, consisting of the order demand for 13 series. In the optimization part, we use the same dataset, the demand forecast for next 90 days, for training and testing. The details of the data are shown in the following table.

DATE	MATERIAL_NUMBER	PLANT	QUANTITY
01.02.2019	A	X1	188
01.02.2019	A	X2	634
01.02.2019	B	X1	3256
01.02.2019	B	X2	3356
01.02.2019	C	X1	15
01.02.2019	D	X1	865
01.02.2019	D	X2	6914
01.02.2019	E	X1	2028
01.02.2019	E	X2	9933
01.02.2019	F	X1	0
01.02.2019	F	X2	0
01.02.2019	G	X2	0
01.02.2019	H	X2	202

Table 6: Sample Data

To learning the inventory process, we need to build an environment to simulate the process first. Here, we set current stock level and sums of orders in the queue as state, the action will be one of the possible quantities which generated by the 20 quantiles of the sum of the demand value in the lead time. What's more, the reward is defined as the cost for the action we choose. The cost is split into 3 parts, holding cost, penalty for stock out and transport fee which includes a fix part and a flexible part for each unit. Besides, to better simulate real-life scenarios, we set a parameter called emergency rate, represent the cases that unpredicted demand occur.

To better evaluated the model, we set two benchmarks before the training. One is called basic, got by the policy that always choose the action in the action space that is closest to the average of demand list, and the other is called goal, obtained by the policy that always choose the action in the action space that closest to the demand after lead time. We defined the models having higher score than the goal as good models, while scores lower than basic as failed ones.

In the training part, the agent was trained for 10 epochs over 500 steps. During the training, we tried to get the total reward and compared it with the goal benchmark.

If the value is higher than that, we save the model and load it in the test part. After each training, we tried to print the graphs describing the relationship between the total reward and iteration times and figures reflect the changes of Q values.

After getting great model in the training part, we load these models and run them 500 times, outputting the total reward, sum of unit transport fee, sum of fixed transport fee, sum of holding cost and sum of penalty, and the inventory list. Then, we plot the inventory list and demand together to evaluate the model and also draw a graph to compare the total return with goal and basic benchmarks.

7.2 Mixed Integer Programming

Our data includes only the part of supplier to production plant. As a result, we consider a material delivery process with multiple suppliers and multiple plants. Mixed-integer Programming (MIP) model is proposed to calibrate the factors including material supply and stochastic inventory process. In MIP problems, it is very important to define the equation. For our data scenario, we start with the simplest single product to find the balance of the supply chain.

Mixed integer programming is a optimization problem in which some decision variables are constrained to integer values. The use of integer variables greatly expands the range of useful optimization problems that one can define and solve.

In integer programming, the problem is successively divided into smaller sub-problems (branching) that are solved recursively, using LP relaxations and cutting planes to provide strong dual bounds. General branch and bound(BB) method can be seen in the pseudo code below.

Algorithm 2 Branch and Bound ($\mathcal{D}, f$)

```
  Algorithm Parameters:
    Search space $\mathcal{D}$
    Objective function $f$

  Begin:
  Set $\mathcal{L} = \mathcal{D}$ and initialize the feasible solution $\hat{x}$
  While $\mathcal{L} \neq \emptyset$:
    Select a sub-problem $\mathcal{S}$:
      if a solution $\hat{x}'$ can be found in $\mathcal{S}$ then:
        Set $\hat{x} = \hat{x}'$
      end if
      if $\mathcal{S}$ cannot be pruned then:
        Partition $\mathcal{S}$ into r sub-problems and insert into $\mathcal{L}$
      end if
      Remove $\mathcal{S}$ from $\mathcal{L}$
  end while
  return $\hat{x}$
  End
```

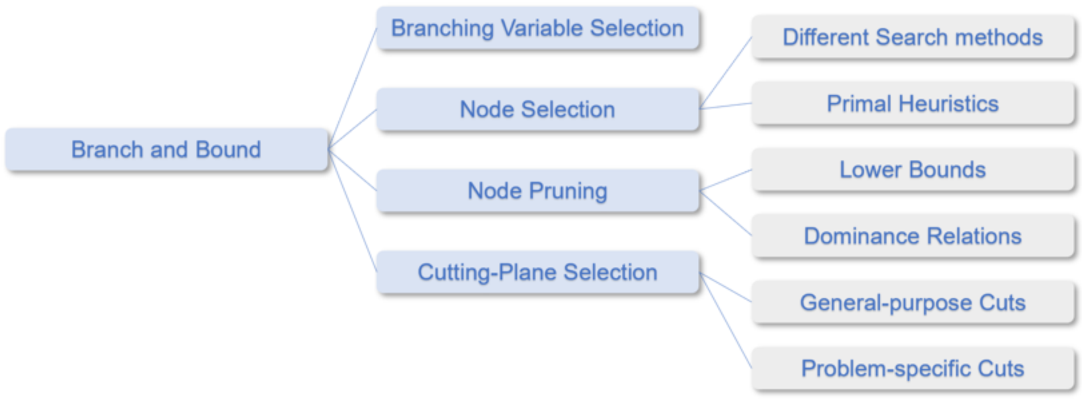

Figure 5: Overview of Branch and Bound methods

Branch and Bound methods can be divided into four categories, which can be seen in figure 5.[7] Our work uses a MPsolver, SCIP. SCIP is a framework for Constraint Integer Programming oriented towards the needs of mathematical programming experts who want to have total control of the solution process and access detailed information down to the guts of the solver. It can also be used as a pure MIP solver or as a framework for branch-cut-and-price.[8]

Consider the simplest case that if only one material is shipped to one plant. The overall cost can be divided into three items: ordering cost, inventory cost and stock out penalty. Ordering cost can be separated as delivery cost and unit ordering cost. The former means the fixed ordering cost each delivery time, including labor costs and vehicle costs; the latter includes the fee related to the amount of each order. When the amount increases, the total ordering cost would increase accordingly.

Let the time horizon be t, which is known with certainty. f_t is the fixed ordering cost, g_t is the unit ordering cost in time period t. The unit holding cost is given by c_t. Let x_t be the amount ordered and s_t be the amount of ending inventory in period t, respectively. Set a dummy variable ω_t to judge the fixed ordering cost: if there exists delivery at day t, $\omega_t = 1$, if not, $\omega_t = 0$. Finally, add a stock-out penalty for each material failed to meet the demand. The amount of stock-out material is z_t, and the unit penalty cost is p_t. Consequently, our goal to minimize the total cost can be represented as

$$\min \sum_{t=1}^{T} \left(f_t \omega_t + g_t x_t + c_t s_t + p_t z_t \right),$$

given the demand in future time. [9]

Supply chain has its supply-inventory balance. As can be seen in figure 6, let d_t be the forecast demand at time t, the stock amount at time horizon t equals to the inventory amount at t-1 plus the amount delivered at the day, and subtract the quantity demanded at time t. Such flow balance equations can be set up continuously.

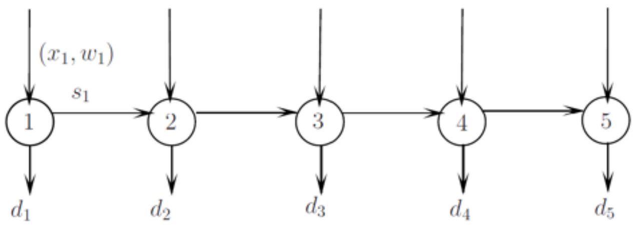

Figure 6: Flow Balance in Supply-Inventory Relation

In addition, set a upper bound of storage capacity for the material, since the warehouse has limited space. x_t also has a upper bound: the amount of each order cannot be more than the sum of demand from the beginning to time t.

Hence, the constraints for single material can be shown as:

$$\begin{cases} s_{t-1} + x_t = d_t + s_t, \forall t \\ s_t \leq cap \\ x_t \leq (\sum_{t=1}^{T} d_t)\omega_t, \forall t \\ s_0 = 0 \\ \omega \in \{0,1\}^n \\ (x,s) \in R_+^{2n+1} \end{cases} \tag{1}$$

Apply the Supply-Inventory-Stockout Relations to multiple materials and multiple plants.[10] Considering $M = 13$ materials and $N = 2$ plants in total, the objective function is:

$$\min(\sum_{t=1}^{T}\sum_{m=1}^{M}\sum_{s=1}^{S} f_{mst}\omega_{mst} + \sum_{t=1}^{T}\sum_{m=1}^{M}\sum_{s=1}^{S} g_{mst}x_{mst} + \sum_{t=1}^{T}\sum_{m=1}^{M} c_{mt}s_{mt} + \sum_{t=1}^{T}\sum_{m=1}^{M}\sum_{i=1}^{N} p_{mit}z_{mit})$$

where

f_{mst}: fixed ordering cost of material m from supplier s in period t

g_{mst}: unit ordering cost of material m from supplier s in period t

c_{mt}: unit holding cost of material m in period t

p_{mit}: stock-out penalty of material m product i in period t

ω_{mst}: $\{0, 1\}$, whether an order takes place of material m from supplier s in period t

x_{mst}: amount ordered of material m from supplier s in period t

s_{mt}: amount of inventory of material m at the end of period t

z_{mit}: stock-out amount of material m product i in period t

MIP problem is programmed by Python ortools. For the boolean variables, the boundary needs to transform into linear formulation. For instance, fixed ordering indicator ω_t:

$$\omega_t = \begin{cases} 1, & x_t > 0 \\ 0, & x_t = 0. \end{cases}$$

should be transformed into:

$$\begin{cases} x_t \leq M_x \cdot \omega_t \\ \omega_t \leq M_x \cdot x_t \end{cases}$$

where M_x refers to the amount that exceeds the upper bound of x_t.

As for the stock-out amount z_t, $z_t = max{-s_t, 0}$. We represent in formulation in this way:

$$\begin{cases} -s_t \leq M_s \cdot y_t \\ s_t \leq M_s \cdot (1 - y_t) \\ z_t \geq 0 \\ z_t \geq -s_t \\ z_t \leq -s_t + M_s \cdot (1 - y_t) \\ z_t \leq M_s \cdot y_t. \end{cases}$$

8 Numerical Studies

8.1 Reinforcement Learning

For the reinforcement learning, we generate the inventory level and return plot for every combinations of the materials and factories. There are in total 13 combinations. In the first plot, we plot each material's inventory versus the demand. The blue line denote the current inventory level while the orange line denote the current material demand. In the second plot, we plot each trained model's return versus basic benchmark and goal benchmark, where basic benchmark is a basic model that reorders on a fixed pattern regardless of the future demand, and goal benchmark is when we order exactly on the lead time if there is incoming demand. We can see that the model return is better than the basic benchmark, and sometimes better than the goal benchmark.

Batch	('A', 'X1')	('A', 'X2')	('B', 'X1')	('B', 'X2')	('C', 'X1')	('D', 'X1')	('D', 'X2')
Benchmark	56793.08	157736.08	509210.00	1042390.00	19297.92	94782.68	972863.84
RL	29471.37	70601.32	99524.16	591551.01	2687.52	51769.17	707903.72
Batch	('E', 'X1')	('E', 'X2')	('F', 'X1')	('F', 'X2')	('G', 'X2')	('H', 'X2')	
Benchmark	710783.40	4032825.00	7502.01	52482.96	61575.02	22543.04	
RL	122664.92	710474.99	5977.72	24034.02	29309.62	4901.84	

Table 7: Total Cost Trained by Reinforcement Learning

material combination	total cost	inventory cost	unit ordering cost	fixed ordering cost	stock out penalty
('A', 'X1')	29471.4	2012.1	1209.1	17800.0	8450.1
('A', 'X2')	70601.3	10772.2	6374.1	17800.0	35655.0
('B', 'X1')	99524.2	19753.5	6605.2	17800.0	55365.5
('B', 'X2')	591551.0	331834.4	84511.6	17800.0	158405.0
('C', 'X1')	2687.5	0.0	0.0	17800.0	2687.5
('D', 'X1')	51769.2	12732.0	3652.2	17800.0	17585.0
('D', 'X2')	707903.7	147119.4	39909.3	17800.0	503075.0
('E', 'X1')	122664.9	26963.2	15374.6	17800.0	62527.1
('E', 'X2')	710475.0	121355.7	68864.3	17800.0	502655.0
('F', 'X1')	5977.7	0.0	0.0	17800.0	5977.7
('F', 'X2')	24034.0	2470.1	563.9	17800.0	18400.0
('G', 'X2')	29309.6	7736.0	3773.6	17800.0	0.0
('H', 'X2')	4901.8	11.8	0.0	17800.0	4890.0

Table 8: Split of Total Cost Trained by Reinforcement Learning

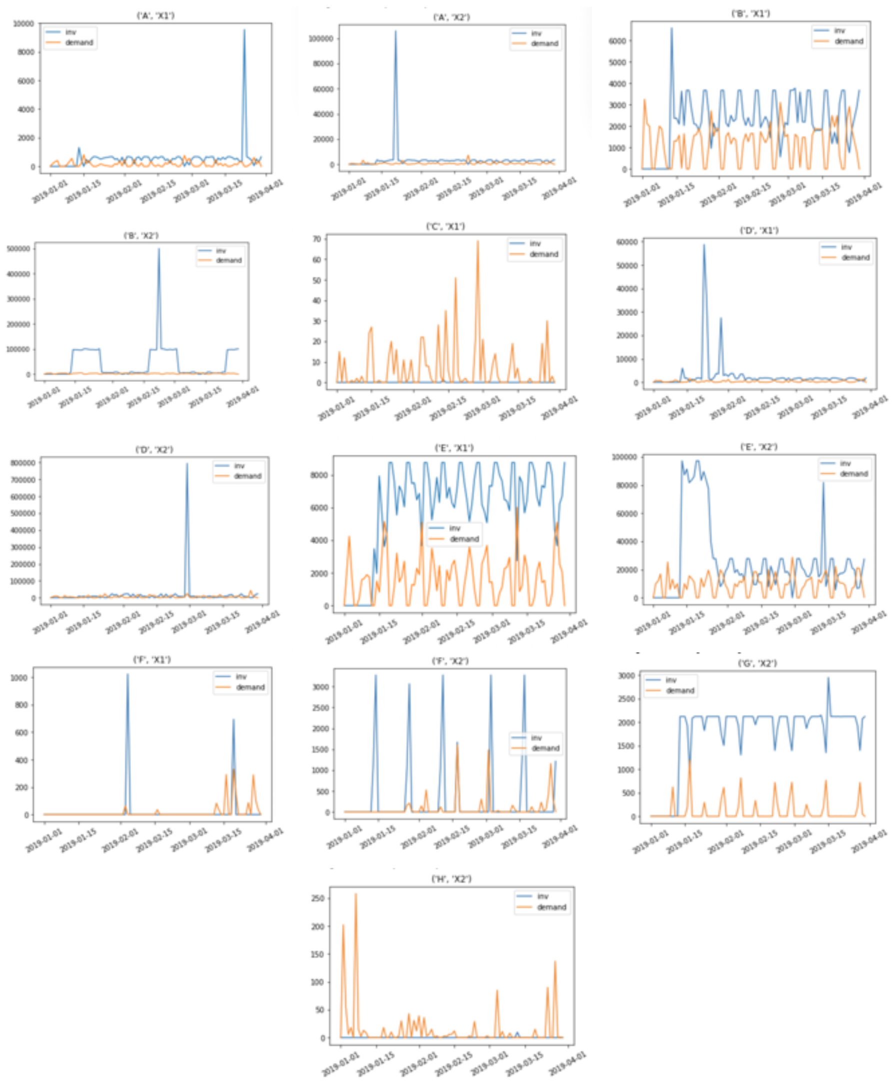

Figure 7: Inventory Level for Every Material

The figure above compares the changes of inventory level and demand list. From the graph, we find that on some series, the inventory curves (blue line) are always slightly above or even quite similar with the demand list (orange line) and their values hardly go to zero, making sure we always have enough stock to deal with demand and maintain a safety stock level, while do not have to pay a huge amount of holding fee. But in some series the inventory curves have an obvious difference with the demand lists. For example, (E,X1), the inventory level is much larger than the demand, which avoids the stock out but has to pay extra holding cost. What's worse, for (C,X1), the demand curve is not only totally different from inventory list, but the inventory list keep around zero. Although we cannot judge whether this situation causes stock out from this graph, this is not recommended. Above all, from this figure, reinforcement learning's performance seems great, but it does not consider the importance of safety stock as for most of the series the inventory level keeps around zero. To fully evaluate this model, more details are needed.

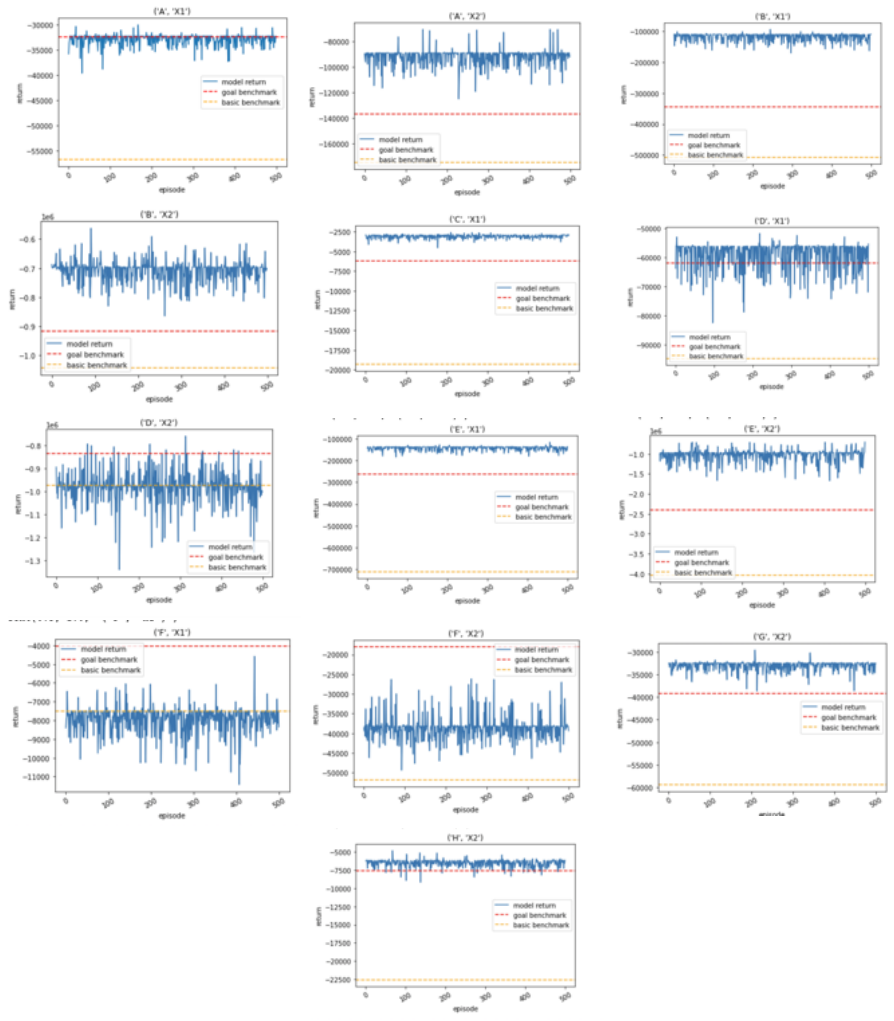

Figure 8: Return for Every Material

The Figure 8 plots the total cost for 13 series. In each graph, we plot the total reward we obtain form the model and use two dotted lines to mark the goal and basic benchmarks. First, from the figure, we could know all of the total rewards seems stable. Then, for six of the materials, nearly a half, the return is obvious higher than goal benchmark. For four of them, the returns are around the goal benchmark, but they can give a better policy. And for the rest, them cannot reach the goal benchmark in testing part but having higher score than basic benchmark.

8.2 Mixed Integer Programming

Use Python Ortools (Google) and SCIP solver to find the optimal solution. There's obvious improve in MIP model in comparison to the goal benchmark. Of these, it was the fixed ordering cost's decline that had the biggest impact on the total optimization. With the high cost per shipment, workers tend to decrease the times of delivery while order much more amount each time. As can be seen in Figure 9 later, the delivery times changes from nearly every day to 2 to 10 times in our final solution.

Batch	(A, X1)	(A, X2)	(B, X1)	(B, X2)	(C, X1)	(D, X1)	(D, X2)
Benchmark	56793.08	157736.0799	509210	1042390	19297.92	94782.68	972863.84
MIP	6315.49	66685.63	53855.54	93803.04	3272.1	19465.64	557712.8
Batch	(E, X1)	(E, X2)	(F, X1)	(F, X2)	(G, X2)	(H, X2)	
Benchmark	710783.4	4032825	7502.010128	52482.95785	61575.02322	22543.04	
MIP	49495.44	255691.67	5999.96	9853.04	16234.82	34775.9	

Table 9: Total Cost Trained by Mixed Integer Programming

Table 10 below shows the value of four objective parts in each material combination's solution. The lowest total cost is 6315.5 from material A with plant X1, while the highest comes from material D with plant X2. As a upper bound of ordering amount is set, all materials tend to make an delivery order with the amount of the possibly biggest. This is because the fixed ordering cost of each delivery is high enough compared with inventory cost. Hence, the materials are more preferable to be stocked in the warehouse, but rather than increasing delivery frequency.

material combination	total cost	inventory cost	unit ordering cost	fixed ordering cost	stock out penalty
('A', 'X1')	6315.49	4341.95	288.54	1400.00	285.00
('A', 'X2')	66685.63	64004.55	1666.08	600.00	415.00
('B', 'X1')	53855.54	49667.50	1643.04	1600.00	945.00
('B', 'X2')	93803.04	61465.30	3652.74	26000.00	2685.00
('C', 'X1')	3272.10	2810.90	46.20	400.00	15.00
('D', 'X1')	19465.64	16282.20	748.44	1800.00	635.00
('D', 'X2')	557712.80	544642.30	12070.50	1000.00	0.00
('E', 'X1')	49495.44	45471.70	2623.74	1400.00	0.00
('E', 'X2')	255691.67	220823.15	16868.52	18000.00	0.00
('F', 'X1')	5999.96	4838.00	66.96	600.00	495.00
('F', 'X2')	9853.04	8295.00	248.04	1200.00	110.00
('G', 'X2')	16234.82	15266.90	367.92	600.00	0.00
('H', 'X2')	34775.90	33895.10	220.80	400.00	260.00

Table 10: Split of the Total Cost Trained by Mixed Integer Programming

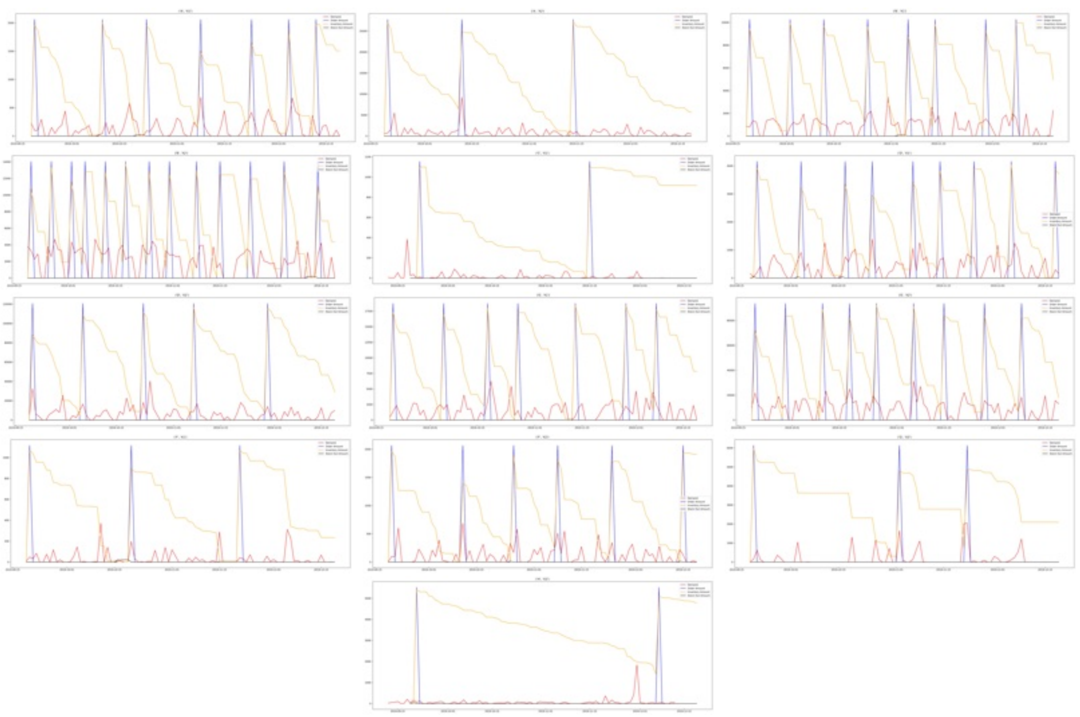

Figure 9: Supply-Inventory-Stockout Patterns of Each Material

As shown in figure 9, the red line represents the demand, blue line represents order amount, yellow line represents inventory amount and black line represents the stock out amount. It's clear that all materials tend to make the order with the amount of the possibly biggest, with subtle sacrifice on stock out amount.

8.3 Comparison of two methods

We compare there two methods, and the results are listed below. The red color refers to the optimal solution. For the total cost, we see that the mixed integer programming performs better than the reinforcement learning. To divide it into parts. We discover that reinforcement learning deals better in inventory cost, while the mixed integer deals better in unit ordering cost, fixed ordering cost and stock out penalty. Generally speaking, the mixed integer programming does better compared to reinforcement learning. Another advantage is that the mixed integer programming does not require training and can solve the equations quickly, while the reinforcement learning usually takes time to train and test.

	total cost		inventory cost		unit ordering cost		fixed ordering cost		stock out penalty	
material	RL	MIP	RL	MIP	RL	MIP	RL	MIP	RL	MIP
('A', 'X1')	29471.4	6315.5	2012.1	4342.0	1209.1	288.5	17800.0	1400.0	8450.1	285.0
('A', 'X2')	70601.3	66685.6	10772.2	64004.6	6374.1	1666.1	17800.0	600.0	35655.0	415.0
('B', 'X1')	99524.2	53855.5	19753.5	49667.5	6605.2	1643.0	17800.0	1600.0	55365.5	945.0
('B', 'X2')	591551.0	93803.0	331834.4	61465.3	84511.6	3652.7	17800.0	26000.0	158405.0	2685.0
('C', 'X1')	2687.5	3272.1	0.0	2810.9	0.0	46.2	17800.0	400.0	2687.5	15.0
('D', 'X1')	51769.2	19465.6	12732.0	16282.2	3652.2	748.4	17800.0	1800.0	17585.0	635.0
('D', 'X2')	707903.7	557712.8	147119.4	544642.3	39909.3	12070.5	17800.0	1000.0	503075.0	0.0
('E', 'X1')	122664.9	49495.4	26963.2	45471.7	15374.6	2623.7	17800.0	1400.0	62527.1	0.0
('E', 'X2')	710475.0	255691.7	121355.7	220823.2	68864.3	16868.5	17800.0	18000.0	502655.0	0.0
('F', 'X1')	5977.7	6000.0	0.0	4838.0	0.0	67.0	17800.0	600.0	5977.7	495.0
('F', 'X2')	24034.0	9853.0	2470.1	8295.0	563.9	248.0	17800.0	1200.0	18400.0	110.0
('G', 'X2')	29309.6	16234.8	7736.0	15266.9	3773.6	367.9	17800.0	600.0	0.0	0.0
('H', 'X2')	4901.8	34775.9	11.8	33895.1	0.0	220.8	17800.0	400.0	4890.0	260.0

Table 11: Comparison of Reinforcement Learning and Mixed Integer Programming Result

Finally, we choose the better model of MIP and calculate the service level. By table 12 we see that each material manages to reach a service level higher than 99%. As the stock out price is serious, our supply chain did well in satisfying strict production plan.

Batch	(A, X1)	(A, X2)	(B, X1)	(B, X2)	(C, X1)	(D, X1)	(D, X2)
Service level	99.57%	99.89%	99.76%	99.71%	99.79%	99.62%	100.00%
Batch	(E, X1)	(E, X2)	(F, X1)	(F, X2)	(G, X2)	(H, X2)	
Service level	100.00%	100.00%	96.83%	99.79%	100.00%	99.17%	

Table 12: Service Level of Each Material

9 Conclusion

In this project, we test and choose the best prediction method for future demand. Among Prophet, LSTM, ARIMA, ARIMA gives the best test result while maintaining relevantly quick training time. We then design two approaches to solve the reorder point optimization problem by using reinforcement learning and mixed integer programming. The reinforcement learning method uses Q-learning with dueling network agent. Mixed integer programming tries to solve an optimal result by predefined equations. By testing reinforcement learning and mixed integer programming methods, we determine that mixed integer programming usually performs better than reinforcement learning. Another advantage of mixed integer programming is that it can solve equations within a short time, while reinforcement learning will take many hours to train. The optimized solution can achieve lower total costs compared to baseline and can achieve a service level of over 99%.

10 Bibliography

[1] P. Kelle. *On the safety stock problem for random delivery processes.* 17(2). 1984, pp. 191–200. DOI: https://doi.org/10.1016/0377-2217(84)90233-9.

[2] Dar-El EM, Malmborg CJ. *Improved strategy for service level-based safety stock determination.* 2(2). 1991, pp. 116–21.

[3] Lu H, Wang H, Xie Y, Li H. *Construction material safety-stock determination under nonstationary stochastic demand and random supply yield.* 63(2). 2016, pp. 201–12.

[4] Wagner, H. and Whitin, T. *Dynamic Version of the Economic Lot Size Problem.* 5(1). 1958, p. 89.

[5] Küçükyavuz S, Pochet Y. *Uncapacitated lot sizing with backlogging: the convex hull.* 118(1). 2016, pp. 151–175.

[6] Gade, D. and Küçükyavuz, S. *Deterministic lot sizing with service levels.* URL: http://www.optimization-online.org/DB_HTML/2010/12/2844.html.

[7] Lingying Huang et al. *Branch and Bound in Mixed Integer Linear Programming Problems: A Survey of Techniques and Trends.* 2021. DOI: 10.48550/ARXIV.2111.06257. URL: https://arxiv.org/abs/2111.06257.

[8] URL: https://www.scipopt.org/index.php#about.

[9] Simge Küçükyavuz. *Mixed-Integer Optimization Approaches for Deterministic and Stochastic Inventory Management.* URL: https://pubsonline.informs.org/doi/10.1287/educ.1110.0085.

[10] Aktas Ahmet and Temiz Izzettin. *Multi-Period Mixed Integer Programming Model for Supply Chain Planning Under Safety Stock.* DOI: 10.47512/meujmaf.816402.