North Carolina State University

Modelling and Analysis of Supply Chains

ISE 553

Spring 2020

Case Study 2

Sichuan Telecom: Upgrading Supply Chain Management Strategy

Authors:

Ankita Pundir

Chaitanya Rajeev

Monish Reddy Devarapedda

Instructor:

Dr. Reha Uzsoy

April 20, 2020

EDWARD P. FITTS DEPARTMENT OF
INDUSTRIAL AND SYSTEMS ENGINEERING

Table of Contents

1. Executive Summary

Sichuan Telecom, a subsidiary of China Telecom Co. Ltd., savored the top position among the five major telecommunications industries in Sichuan, with respect to total subscribers and business revenue. Out of the services provided, the Internet and related value-added services were of priority as they saw revenue growth of 26.6% and 63% respectively, in 2006. Out of the three types of Internet services provided, ADSL composed of major customers of Sichuan Telecom, and hence, it was a decisive part of the business. However, it was found that the procurement centers were having a hard time managing the inventory of ADSL modems, with a persistent problem of overstock or out-of-stock.

To analyze the problem on hand, the supply chain structure is formed, taking the review period as 1 week. Assumptions like supplier aggregation and transshipment are examined for effects, holding costs calculated, while the service level set at a rate of 99%, to ensure the top position in the market. The assumptions are incorporated into a tree model and based upon Simpson's Guaranteed Service Model approach. Next, optimization of the tree model was done using Dynamic programming, with the algorithms being run in Python 3.6. The optimal parameters and computed values of Base stock levels and Safety stocks for all the 21 districts are presented in tabular form in the Results section.

The optimized model has safety stock placed in both central and district locations. The safety stock values are higher than those estimated previously. The pipeine inventory though will be reduced due to the reduced period length.

2. Company Background

Sichuan Telecom was a subsidiary of China Telecom responsible for providing services like telephones (both fixed and mobile) and data access in Sichuan, the Chinese province. In 1998, the Postal Administration Bureau (former name) divided into different service providers like postal and telecom, mobile and satellite communications becoming a pure enterprise leaving behind the image of a government department. In the year 2000 Sichuan Telecom emerged from the Postal Bureau in Sichuan province and as a result of this change and it was officially registered in 2002. It was later acquired by China Telecom Co. Ltd. In 2006, the total revenue was RMB 12 billion ($ 1.7 billion), with total subscribers reaching 17 million.

Out of the five major telecommunication industries in Sichuan, viz. Sichuan Telecom, China Mobile, Unicom, China Netcom, and China Railcom, Sichuan Telecom and Sichuan Mobile was the market leader in terms of the total number of users and revenue.

While the revenue earned from fixed telephone services was decreasing, revenue from an Internet service provider and other related services was increasing rapidly. In 2006 the growth in internet service revenue was 26.6 % and broadband subscribers increased by 51.8 % from last year.

Sichuan Telecom was providing three types of services: optical fiber, local area network, and ADSL (asymmetric digital subscriber line), out of which the maximum share of subscribers came from ADSL. So, the ADSL business was of great importance for Sichuan Telecom.

2.1 Forecast of Sales and Demand

ADSL sales happen in the following manner – First, a customer requests for ADSL installation. Next, he/she will sign a contract with the branch, account, or area manager. Finally, the community manager finalizes the installation within 24 hours. Under most of the service schemes, there is a monthly subscription fee and the ADSL modem is free of cost. The scheme prices will vary in every district, with facilities available.

The marketing department of Sichuan Telecom predicts demands for the upcoming year and districts are assigned sales targets based on that forecast. Then, every district will decide their monthly target sales based on past performances, from which branches at lower levels decide safety stocks.

2.2 Procurement and Suppliers

At the beginning of each year, Sichuan Telecom would reach an agreement for their modem procurement with several suppliers. There were no specified suppliers for each district. The supply volume of each supplier was within set within predetermined percentage limits. In 2006 an agreement was signed with 5 suppliers with the average price of modem set at RMB 150 ($21)

The procurement process took place in the following manner: The district branch would fill out an application form, specifying type and amount of modem required, get internal approval and send it to the procurement center. The center after making sure the procurement information agrees with the rules of the agreement signed with the supplier and then approves the application and sends it to the supplier. This whole process would usually take 2 days.

3. Original Supply Structure

$L_i \rightarrow$ *Lead time to district warehouse*

$L_i \in \{1,2,3\}$

$i \rightarrow$ *District number*

Exhibit 3

THE SUPPLY PERCENTAGE OF MAJOR SUPPLIERS (2006)

A	34.38
B	13.35
C	20.78
D	7.28
E	24.21

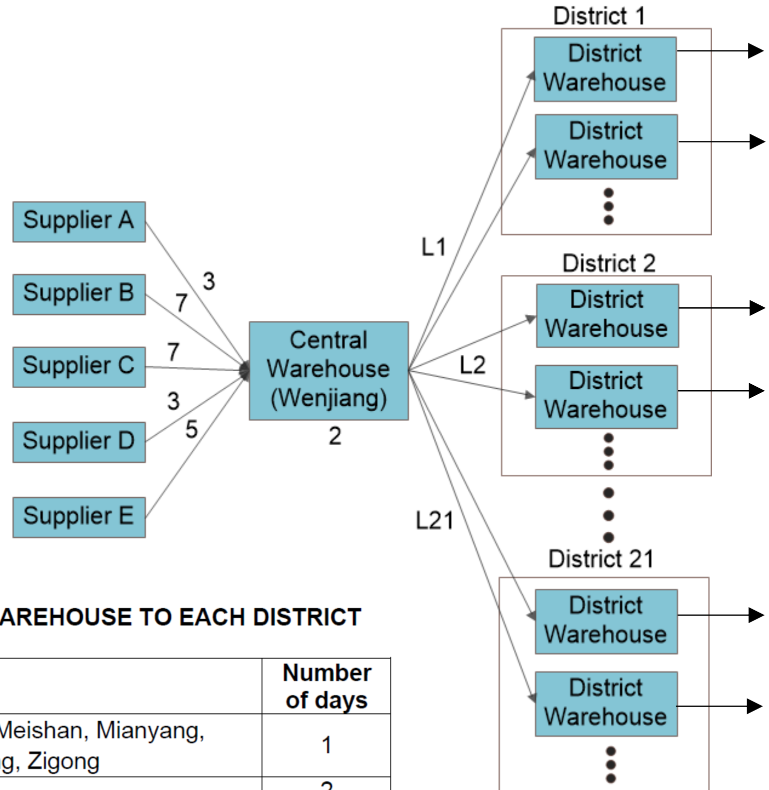


Exhibit 4

THE LEAD TIME OF WENJIANG CENTRAL WAREHOUSE TO EACH DISTRICT

District	Number of days
Chengdu, Deyang, Guang'an, Leshan, Luzhou, Meishan, Mianyang, Neijiang, Nanchong, Suining, Ya'an, Yibin, Ziyang, Zigong	1
Bazhong, Dazhou, Guangyuan	2
Aba, Ganzi, Liangshan, Panzhihua	3

Figure 1

Sichuan Telecom's ADSL modem supply chain structure is depicted in Figure 1. 5 suppliers A, B, C, D and E supply modems to the central warehouse in Wenjiang. The respective supplier lead times are shown above the arrow connecting each supplier to the central warehouse in days. The procurement center (assumed to be at the central warehouse) takes 2 days to process orders which would be its processing time as shown below its stage box. The company operates almost 200 district warehouses clustered in 21 districts which are all supplied from the central warehouse. The lead times to each district are shown in Exhibit 4 from the case (refer Figure 1). The district warehouses serve individual customer demand within their district as depicted by the arrow coming out of the district warehouses.

4. Review Period Discussion

Sichuan Telecom operates almost 200 warehouses in 21 districts which are all served by the central warehouse in Wenjiang. It is mentioned that districts place orders to suppliers via a procurement center presumably located at the site of the central warehouse. The order preparation time (approvals, documentation, etc.) can take up to 2 days. The supplier lead time is assumed to be 5 days. Hence one can expect a total of 7 days to elapse between order placement and material receipt. Since we assume that each district can receive at most a single shipment of modems in a period, it is convenient to choose a period length of 7 days or 1 week.

In selecting the review period, it was noted that the average interval for ADSL modem inventory inspection was seven days in all sub-branch companies in Sichuan. Hence 7 days is selected as the review period which is exactly one period.

5. Assumptions and Implications

> **Assumption 1:**
> All five suppliers can be aggregated into a single supplier with a deterministic lead time (to the central warehouse in Chengdu) of 5 days.

Assumption 1 presents two significant departures from the case. Firstly, the lead times are deterministic instead of random. Secondly, the variability in lead times is pooled from multiple sources into an aggregated source. Both effects will underestimate the actual level of uncertainty in the system. Relevant equations are presented below:

For deterministic Lead Time: *Variance of LTD* $\sigma_{LTD}^2 = \mu_L \sigma_D^2$. *Here L is shown as* μ_L *for comparison*

For random Lead Time: *Variance of LTD* $\sigma_{LTD}^2 = \mu_L \sigma_D^2 + \boxed{\mu_D^2 \sigma_L^2}$ *(V[LTD] is greater)*

For independent suppliers (i = 1 to 5): $SS = \sum_{i=1}^{5}(z^* \sigma_{LTDi})$; $\sigma_{LTDi}^2 = \mu_{Li}\sigma_{Di}^2 + \mu_{Di}^2 \sigma_{Li}^2$

For aggregate supplier: $SS = z^* \sigma_{LTD}$; $\sigma_{LTD}^2 = \mu_L \sigma_D^2 + \mu_D^2 \sum_1^5 \sigma_{Li}^2$

Both lead time pooling and deterministic lead time assumptions reduce uncertainty and hence lead to reduced base stock levels. Hence, our mathematical model will tend to under-estimate safety stock and base stock levels as compared to their realistic requirements for a given service level.

Assumption 2:
If a local warehouse within one of the districts stocks out, demand can be met from another warehouse within the district at negligible additional cost (i.e., reactive transshipment is possible within the district). Transshipment between districts, however, is not permitted.

Exhibit 5 (distance between districts) suggests that transshipment between districts is possible in a real-world scenario. Assumption 2 summarizes that transshipment between districts is not allowed. Transshipments have a similar effect to information pooling wherein one district's demand can partially be served by another district. By nullifying this effect, we would see our average safety stock levels at each district increase. Hence, this assumption will cause our mathematical model to over-estimate the required base stock levels at each district for a given service level.

The assumption also states that Reactive transshipments are possible within districts at negligible cost. This will lead to near-perfect pooling of all demand within each district. This essentially allows us to consider each district as an aggregated inventory location itself. However, this will lead to underestimation of actual total base stock levels at all child warehouses within a district for a given service level.

Assumption 3:
Each district can receive at most a single shipment of modems from the central warehouse in a period, and transportation costs are sufficiently low relative to the shortage costs of a modem to permit this. Assume a similar mode of operation between the central warehouse and the aggregated supplier.

This assumption summarizes that transportation costs are low relative to the shortage costs. This assumption will cause the model to push a lot of inventory downstream to prevent shortages as much as possible. In reality, if the transportation costs are higher than assumed, then the optimal solutions would dictate higher inventory levels to be placed at centralized locations.

6. Reasons for High service-level (0.99)

Even though Sichuan Telecom enjoyed the strongest comprehensive strength and had the best market share, the telecom industry in Sichuan was crowded with 5 major players: Sichuan Telecom, China Mobile (CMCC), Unicom, China Netcom (CNC) and China Railcom(CRNET). Also, more and more people were increasingly choosing mobile phones instead of fixed lines. Hence the revenue from the fixed telephone business was plateauing. Hence the ADSL modem business was of utmost importance to the company.

Also, modems tend to have long-term customers who pay monthly subscriptions. So even one lost or unserved customer could deny the company of years of monthly customer subscription fees. These could be reasons why the management has insisted on a very high service rate of 0.99.

7. Holding Cost Estimation

Exhibit 9 shows the inventory level of each location at the beginning of each month. To estimate the average inventory, one must reasonably estimate the maximum and minimum inventory levels during a period. The average inventory would then be the simple average of the two quantities.

The inventory levels shown to us are counts taken at the beginning of every month. This can reasonably be assumed to be the left-over safety stock of the previous month that remains after the demand of the previous period has been satisfied.

From a continuous cyclic perspective, we assume that inventory is initially observed at the beginning of the month, placed orders arrive and demand gets realized. This means the max inventory value is estimated as inventory level at the beginning of the month + Order quantity for that month.

$$Inventory\ Status\ @\ i^{th} period + Order\ Qty_i - Demand_i$$
$$= Inventory\ Status\ @\ (i+1)^{th} period$$

$$\Rightarrow Order\ Qty_i = Inventory\ Status\ @\ (i+1)^{th} period\ + Demand_i - Inventory\ Status\ @\ i^{th} period$$

Order quantities are hence calculated (**Appendix 10.4**) from which we can estimate the max inventory level. The min inv. level in period i is assumed to be the inventory status of $(i+1)^{th}$ period.

Using this the average inventory per month is calculated (**Appendix 10.6**) as shown:

- Average inventory_i = (Max Inventory level_i + Min Inventory level_i)/2

Now that we have calculated average inventory for a given period for a district, the grand average inventory level is simply the average of the 12 average inventory_i values for the 12 months.

Exhibit 7 and Exhibit 8 gives the total floor space and (sq. m) and rent (RMB/m^2). From these the monthly rent and consequently weekly rent is calculated as shown:

- Rent component = Weekly rent / Average Inventory.

Interest and Material Handling costs of 5% and 6% per annum are added to the rent component (Source: ISE 754, courtesy: Dr. Michael Kay). From these calculations, the holding costs per week are calculated. In the DP these are converted into daily costs because the algorithm requires non-decimal values for processing time (T) and Longest path (M).

Example calculation (For District Aba):

- Average inventory for January = ($\underline{1250+1025}+\underline{2214}$)/2 = 2245.

Similarly, the average inventory for all months is calculated and the average of these 12 inventories. This comes to be 2244.

- Monthly rent = 1027 * 5 = 5135. Weekly rent = 5135 / 4.35 = 1180.46.

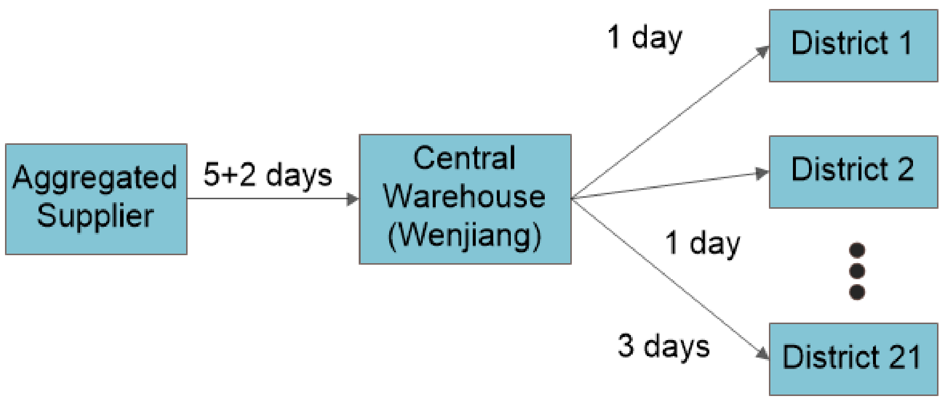
- Rent Component / unit = 1180.46/2244 = 0.53.

The interest and material handling costs of 11% per annum are added weekly to this and the holding cost per unit per week is calculated to be 0.67 weekly. Results are displayed in **Appendix 10.5**

- Weekly Holding cost = 0.53 / (1- (0.11/52)) = 0.67

8. Multi-Echelon Inventory Structure

Sichuan Telecom's supply chain infrastructure can now be modeled as a multi-echelon supply chain depicted in Figure 2. Inexpensive transshipments within districts allow for all warehouses within a district to be considered as one inventory location. The inventory structure is depicted in Figure 2 shown below.



The holding costs for every stage are calculated and shown in **Appendix 10.5**. For the guaranteed service model which shall be discussed later, we also estimate the process times at every stage and inbound and outbound service times (SI and S) wherever applicable.

9. Analysis and Results

9.1 Overstock/Understock Analysis

To make an approximate estimation on whether the demand was overestimated or underestimated during a given period, one must come up with a reasonable approximation of Demand Forecasted by the company for a given period.

To make this approximation we shall assume that the inventory manager ordered a quantity equal to the demand forecast every month.

- Order Quantity = Demand Forecast for a given month (Assumption)

With this assumption, our estimated Forecast Error will be the Order Quantity minus the Demand.

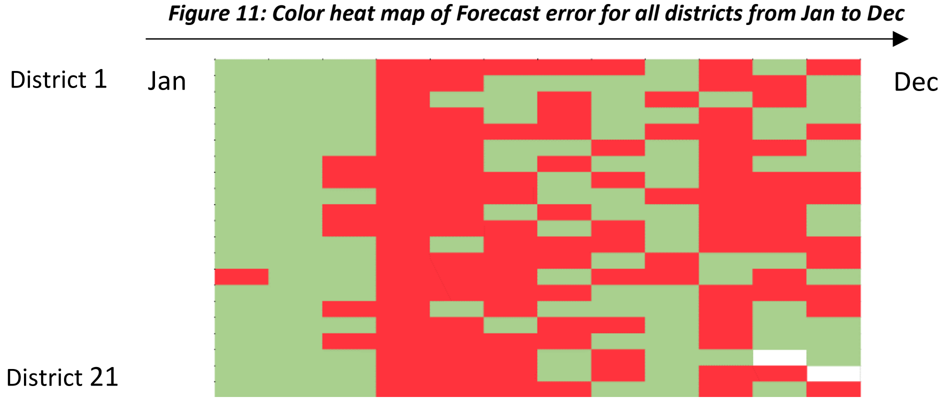
Order quantities have already been calculated in **Appendix 10.4** Using this data, we calculate the estimated forecast error as:

- Forecast Error = Order Quantity – Monthly Demand

The forecast error results are presented in **Appendix 10.7**. Negative errors (marked in red) mean that Monthly Demand exceeded order quantity. Positive errors (marked in green) imply the opposite. This method of analyzing overstocking and understocking does not have a bias due to inventory accumulation from previous periods (if any). It makes a simple comparison between Amount Ordered v/s Demand assuming the forecasted amount was ordered.

From the color map (Figure 11) it is evident that most districts overstocked for the first few months consistently and then began to understock in the middle months. The ending months have equal proportions of red and green suggesting that the forecast was fairly accurate. This could mean some underlying faults with the current forecasting techniques.

Figure 11: Color heat map of Forecast error for all districts from Jan to Dec



9.2 Guaranteed Service Tree Model

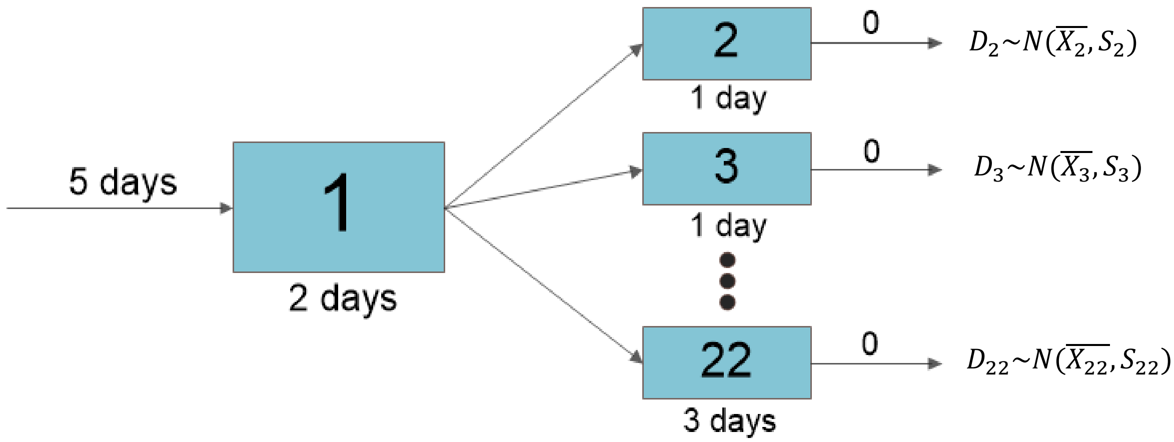


Figure 3

All assumptions made thus far in the case allow us to now build a simple tree model shown in Figure 3 where safety stocks can be optimized using Simpson's Guaranteed Service Model approach.

To briefly describe this model, the stages are labeled from 1 to 22. 1 is our central warehouse and 2 to 22 are our district aggregated inventory locations. Since unlimited transshipments are allowed within districts at negligible transportation cost, this amounts to near-perfect inventory pooling at near-zero additional cost. Hence, we can aggregate all warehouses within a district to 1 single stage.

Each district faces customer demand which is assumed to be normally distributed with no long-term trend. The parameters are $\overline{X_\iota}$ which is an unbiased estimator for μ_i which is the mean of the demand for the ith district. Similarly, S_1 is the estimated variance of demand of the ith district.

Node 1 has an inbound SI value of 5 days which is the lead time from the aggregated supplier. The supplier isn't included as Sichuan Telecom doesn't control inventory decisions at the supplier. The stage processing times are given below each box. Node 1 has a processing time of 2 days as it takes 2 days to process orders. Holding costs and other data for the model are provided in Table 1.

The service times for the district warehouses are assumed to be 0.

In summary, the following assumptions are also made for our model:

- The service time (S) for demand nodes is 0.
- The inbound service time (SI) for the supply node is 5 days.
- Processing time at the central warehouse takes 2 days.
- District warehouses have a processing time equal to the lead time from central WH.
- Customer demand follows a truncated normal distribution with a finite upper bound.

9.3 Optimization

9.3.1 Introduction

The guaranteed service tree model formulated previously can be solved optimally using a dynamic programming algorithm. This approach was introduced by Graves and Willems (2000). The process consists primarily of 2 steps. The first step requires a model with nodes labeled in such a way that each node number is connected to at-most one other node number with a higher index. The second step involves solving the dynamic program with a forward pass of all nodes and a backtracking process to derive optimal service time values.

Both algorithms have been implemented in the Python 3.6 programming language and can be used to solve any tree system. The model can be represented using a graph adjacency binary matrix. An example is illustrated in Figure 9. This is a common representation technique for network models used in network optimization. The labeling functions will be described next.

```
In [35]: x = [1,1,3,4,5,5]
         y = [2,3,5,5,6,7]
         create_adj(x,y,7)

Out[35]: array([[0., 1., 1., 0., 0., 0., 0.],
                [1., 0., 0., 0., 0., 0., 0.],
                [1., 0., 0., 0., 1., 0., 0.],
                [0., 0., 0., 0., 1., 0., 0.],
                [0., 0., 1., 1., 0., 1., 1.],
                [0., 0., 0., 0., 1., 0., 0.],
                [0., 0., 0., 0., 1., 0., 0.]])
```

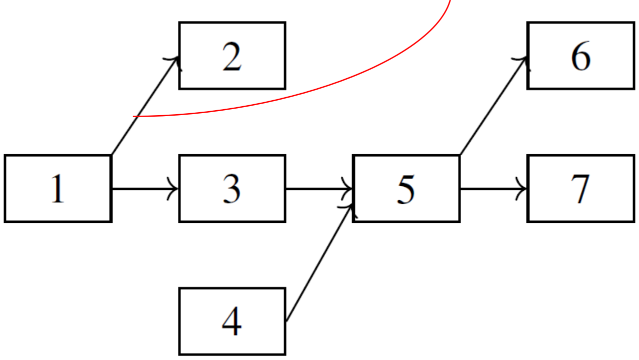


Figure 9

9.3.2 Labeling nodes with the relabel() function

Algorithm 6.1 Relabel stages

$L \leftarrow \emptyset, U \leftarrow \{1, \dots, N\}$ ▷ Initialization
for $k = 1, \dots, N$ **do** ▷ Labeling stages
    choose $i \in U$ such that i is adjacent to at most one other stage in U
    label i with index k
    $L \leftarrow L \cup \{i\}, U \leftarrow U \setminus \{i\}$
end for
return labels

Figure 4. Source: Fundamentals of Supply Chain Theory (Snyder and Shen)

The relabel() function implements the relabel algorithm shown in Figure 4. This helps us relabel our model so that it satisfies the higher node index requirement described in the Introduction section. The code for the relabel() function is provided in **Appendix 5.1**. The function is applied to our network and the output is shown below in Figure 5.

```
x = [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]
y = [2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22]
adj = create_adj(x,y,22) # custom create_adj() function to create adjacency matrix
adj,k = relabel(adj)     # custom relabel() function to relabel nodes
print("relabel vec:",k)  # relabel functions outputs new adjacency matrix and relabel vector

relabel vec: [21, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 22]
```

The red box containing values 1 and 3 mean that there is a connection between node 1 and node 3. The relabel vector output tells us how to relabel our nodes. 21 is in the first position. This tells us that node 1 should be re-labeled as 21, 2 as 1, 3 as 2 and so on. Hence the relabel() function instructs us to relabel our network as shown below.

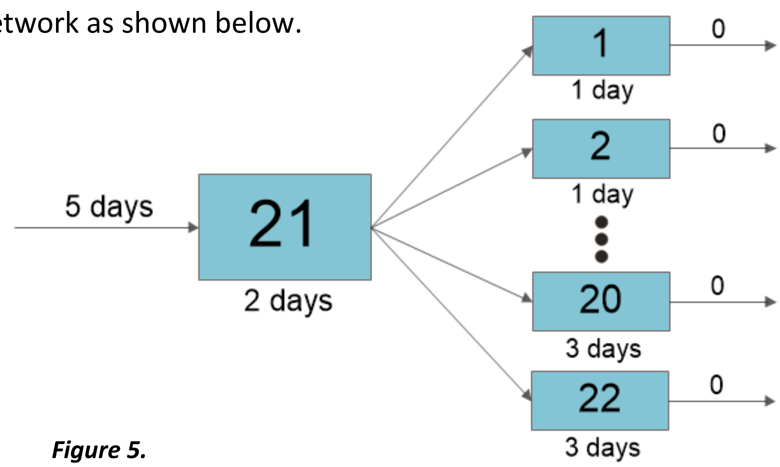


Figure 5.

9.3.3 Solving the network with the dp_tree() function

Algorithm 6.2 DP algorithm for tree SSSPP

for $k = 1, \ldots, N-1$ **do**
  if p_k is downstream from k **then**
    calculate $\theta^o_k(S)$ for $S = 0, 1, \ldots, M_k$
  else
    calculate $\theta^i_k(SI)$ for $SI = 0, 1, \ldots, M_k - T_k$
  end if
end for
$SI^* \leftarrow \text{argmin}_{SI=0,1,\ldots,M_N-T_N}\ \theta^i_N(SI)$
return $\theta^i_N(SI^*)$

Figure 6. Source: Fundamentals of Supply Chain Theory (Snyder and Shen)

The algorithm for solving the tree system is shown above in Figure 6. This algorithm is coded within the dp_tree() function (Code in **Appendix 5.2**.). The dp_tree() function has helper functions theta() and cost() to help with the recursive calculation of theta values at every node. The function requires data input which includes the following at every node: holding costs, processing times, the longest path from supplier node (M), SI(inbound service time) if a node is a supply node, S(service time) if a node is a demand node, echelon values (1,2,3…), Z_{alpha} and sigma values. All input is provided in individual lists.

An example is shown below in Figure 7.

Figure 7

```
echelon = [2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,2]
holding_cost = [9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,4,9]
processing_time = [1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,3,3,3,2,3]
Z = [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]
sigma = [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]
M = [6,6,6,6,6,6,6,6,6,6,6,6,6,6,7,7,7,8,8,8,5,8]

# define supply, demand stages
supply_stage = [21]
supply_stage_SI = [5]
demand_stage = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 22]
demand_stage_S = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

result,bt,io = dp_tree()
```

The input data for our model is summarized in Table 1. This data is input in a similar fashion to the example shown in Figure 8. The dp_tree() function is coded to give out 3 different matrices: the result matrix containing theta values for every stage k, the backtrack matrix which contains the SI or S values that gave the minimum theta value for that stage, and the i/o vector which tells whether theta_inbound("i") or theta_outbound("o") was calculated at every stage.

9.3.4 Data Input

Node No.	Location	Warehouse Type	Holding Cost (daily)	Lead time (from central WH) (days)	Assumed Processing time (daily)	Z_α (0.99)	Sigma (daily)	Longest path (M)	Echelon
1	Chengdu	District	0.22	1	1	2.33	749.11	8	2
2	Deyang	District	0.81	1	1	2.33	73.44	8	2
3	Guang'an	District	0.12	1	1	2.33	96.13	8	2
4	Leshan	District	0.10	1	1	2.33	127.15	8	2
5	Luzhou	District	0.28	1	1	2.33	94.07	8	2
6	Meishan	District	0.27	1	1	2.33	104.46	8	2
7	Mianyang	District	0.13	1	1	2.33	306.95	8	2
8	Neijiang	District	0.66	1	1	2.33	328.92	8	2
9	Nanchong	District	0.22	1	1	2.33	85.31	8	2
10	Suining	District	0.15	1	1	2.33	59.28	8	2
11	Ya'an	District	0.51	1	1	2.33	34.49	8	2
12	Yibin	District	0.37	1	1	2.33	191.08	8	2
13	Ziyang	District	0.28	1	1	2.33	84.82	8	2
14	Zigong	District	0.15	1	1	2.33	62.13	8	2
15	Bazhong	District	0.20	2	2	2.33	74.36	9	2
16	Dazhou	District	0.27	2	2	2.33	84.39	9	2
17	Guangyuan	District	0.16	2	2	2.33	90.56	9	2
18	Aba	District	0.10	3	3	2.33	23.02	10	2
19	Ganzi	District	0.23	3	3	2.33	22.20	10	2
20	Liangshan	District	0.06	3	3	2.33	48.30	10	2
21	Wenjiang	Central	0.37	2	2	2.33	948.30	9	1
22	Panzhihua	District	0.66	3	3	2.33	37.08	10	2

Table 1

The above table summarizes all the data prepared to input into the model. All costs have been calculated daily, and time in days because of modeling requirements.

9.4 Results and Conclusion

The final input code snippet is displayed in Figure 8.

Figure 8

```
#Data Input
echelon = [2]*20+[1,2]
M = [8]*14+[9]*3+[10]*3+[7,10]
Z = [2.33]*22
processing_time = [1]*14+[2]*3+[3]*3+[2,3]

holding_cost = [0.22,0.81,0.12,0.1,0.28,0.27,0.13,0.66,0.22,0.15,
                0.51,0.37,0.28,0.15,0.2,0.27,0.16,0.1,0.23,0.06,0.37,0.66]
sigma = [749.11,73.44,96.13,127.15,94.07,127.15,94.07,104.46,306.95,
         328.92,85.31,59.28,34.49,191.08,84.82,62.13,74.36,84.39,90.56,
         23.02,22.20,48.30,948.30,37.08]

# define supply, demand stages
supply_stage = [21] # supply node number
supply_stage_SI = [5] # supply node SI
demand_stage = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
                15, 16, 17, 18, 19, 20, 22] # demand nodes
demand_stage_S = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

#solve DP
result,bt,io = dp_tree()
```

On backtracking the results using the backtrack matrix, optimal SI values are found to be 0 for the demand nodes and the optimal S value for the supply node is found to be 0. The explanation and the 3 matrices are given in **Appendix 5.3**. Hence one can calculate Safety Stocks and Base Stocks using these values.

The calculated Optimal weekly Safety Stock and Base Stock values are displayed in Table 2 shown below. As evident, the optimal solution isn't exclusively a completely centralized or a completely decentralized system as all stages have some safety stock.

The holding costs and Lead times determine the optimal amount of safety stock at each location. In addition to this, a reduction in the review period will also reduce average cycle stock and hence reduce holding costs.

For the district warehouses, the lead times are considered to be the order processing time plus supplier lead time plus lead time from the central warehouse to the respective district.

Node	Location	T	Net Lead Time (days)	Supplier Lead Time (days)	Net Lead Time (weeks)	Mean Demand (Weekly)	sigma (Weekly)	Z_α (0.99)	Safety Stock (Weekly)	Base Stock (Weekly)
1	Chengdu	1	1	7	2.14286	8052	5243.80	2.33	17886	25938
2	Deyang	1	1	7	2.14286	1300	514.10	2.33	1754	3054
3	Guang'an	1	1	7	2.14286	1038	672.92	2.33	2296	3334
4	Leshan	1	1	7	2.14286	1633	890.03	2.33	3036	4669
5	Luzhou	1	1	7	2.14286	1540	658.46	2.33	2246	3786
6	Meishan	1	1	7	2.14286	1080	731.21	2.33	2495	3575
7	Mianyang	1	1	7	2.14286	1842	2148.66	2.33	7329	9171
8	Neijiang	1	1	7	2.14286	1075	2302.43	2.33	7854	8929
9	Nanchong	1	1	7	2.14286	1974	597.18	2.33	2037	4011
10	Suining	1	1	7	2.14286	1076	414.95	2.33	1416	2492
11	Ya'an	1	1	7	2.14286	606	241.44	2.33	824	1430
12	Yibin	1	1	7	2.14286	1652	1337.57	2.33	4563	6215
13	Ziyang	1	1	7	2.14286	783	593.74	2.33	2026	2809
14	Zigong	1	1	7	2.14286	1188	434.92	2.33	1484	2672
15	Bazhong	2	2	7	2.28571	844	520.52	2.33	1834	2678
16	Dazhou	2	2	7	2.28571	1474	590.75	2.33	2082	3556
17	Guangyuan	2	2	7	2.28571	899	633.90	2.33	2234	3133
18	Aba	3	3	7	2.42857	265	161.11	2.33	585	850
19	Ganzi	3	3	7	2.42857	223	155.38	2.33	565	788
20	Liangshan	3	3	7	2.42857	885	338.07	2.33	1228	2113
21	Wenjiang	2	7	0	2	29915	6638.09	2.33	21874	51789
22	Panzhihua	3	3	7	2.42857	486	259.57	2.33	943	1429
								Total	70705	122483

Table 2

Since the review period is of 1 week, a value of 1 period is added to the net lead time. The safety stocks and base stocks obtained from calculations are displayed in Table 2.

Because of the higher required service level and our assumed review period, the safety stock is observed to higher than the previous average inventory levels. But the lower review period will lead to reduced pipeline inventory as well. This should allow total holding costs to come down as well. To achieve the 0.99 service level as well as minimize costs even further, a few recommendations are provided later.

10. Recommendations

Through the course of the analysis, several potential issues were uncovered and presented. A few recommendations are presented below to partially alleviate some of these issues:

- This report assumes that the suppliers are aggregated for the analysis. In reality, though, aggregation of suppliers would reduce lead time uncertainty and hence variance of lead time demand. This would reduce safety stock requirements to achieve a 0.99 service level.
- The marketing division of Sichuan Telecom generates the forecasts for each of the districts for the following year based on their targets and performance. As shown in the overstock/understock analysis, there seems to be severe over-ordering in the first few months of the year. This could be due to systematic errors in forecasting. Hence it is recommended to the management revisit forecasting methodologies and probably use reliable methods such as Exponential Smoothing, Winter's Method or ARIMA methods.
- A cost-effective 3rd party logistics provider could be roped in to provide or enhance transshipment capability between districts. This would further reduce safety stocks. However, one would need to re-model the systems by adding logistics costs to the total cost function and optimize the number and quantity of transshipments.
- The procurement system could be overhauled to reduce order processing times from the current 2 days. Approvals could be done digitally using IT systems. ERP systems could implement a centralized procurement module where district managers could access and place orders and upper-level management could approve them with digital signatures. This will reduce processing times in the system.

11. Appendix

11.1 relabel() and create_adj() function code and example

```
def relabel(adj): # implementes the graph relabelling
                  # algorithm in Snyder and Shen (2019)
    adj2 = adj.copy()
    U = list(range(1,adj.shape[0]+1))
    L =[]
    label = [0] * adj.shape[0]
    for k in range(1,adj.shape[0]+1):
        for i in range(adj2.shape[0]):
            if sum(adj2[i,:])<=1 :
                adj2 = reduced_adj(adj2,i+1)
                label[U[i]-1] = k
                L.append(U[i])
                del U[i]
                break
    adj2 = adj.copy()
    adj3 = np.zeros(adj2.shape)
    U = list(range(adj2.shape[0]+1))
    for i in range(adj2.shape[0]):
        for j in range(adj2.shape[0]):
            if adj2[i,j]==1:
                adj3[label[i]-1,label[j]-1]=1

    return adj3,label
```

```
def create_adj(x,y,dim): # creates a graph adjacency matrix
                         # of the supply chain from link vectors
                         # x and y
    adj = np.zeros((dim,dim))
    for k in range(len(x)):
        adj[x[k]-1,y[k]-1] = 1
        adj[y[k]-1,x[k]-1] = 1
    return adj
```

Re-labelling example of Synder and Shen example network Page 212, Fig. 6.12

```
x = [1,1,3,4,5,5]
y = [2,3,5,5,6,7]
adj = create_adj(x,y,7)
adj
```

```
array([[0., 1., 1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 1., 1., 0., 1., 1.],
       [0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0.]])
```

```
adj2,k = relabel(adj)
adj2
```

```
array([[0., 1., 0., 0., 0., 0., 0.],
       [1., 0., 1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 0., 1., 0.],
       [0., 0., 1., 1., 1., 0., 1.],
       [0., 0., 0., 0., 0., 1., 0.]])
```

11.2 dp_tree() function code

```
def dp_tree():
    result = np.zeros((max(M),max(k_new)))
    result[:] = np.nan
    backtrack = np.zeros((max(M),max(k_new)))
    backtrack[:] = np.nan
    theta_type = []
    for k in range(1,len(k_new)):                     # iterating over all stages
        if higher_linked(k,adj)[0]:                   # checking if there is a linked stage with higher index u
            p = higher_linked(k,adj)[1]               # setting p to the index number of linked stage (p(k))

        if downstream(p,k):                           # checking if p is downstream of k in supply chain
            range_S = list(range(M[k-1]+1))
            for i in range_S:
                result[i,k-1] = theta(k,"o",i,i)[0]
                backtrack[i,k-1] = theta(k,"o",i,i)[1]
            theta_type.append('o')


        else:
            range_SI = list(range(M[k-1]-processing_time[k-1]+1))
            for i in range_SI:
                result[i,k-1] = theta(k,"i",i,i)[0]
                backtrack[i,k-1] = i
            theta_type.append('i')
    k = max(k_new)
    if supply_node(k): # checking if node N (last node in set K) is a supply node. The SI is fixed in this case
        range_SI = list(compress(supply_stage_SI, list(map(lambda x:x==k,supply_stage))))
    else:
        range_SI = list(range(M[k-1]-processing_time[k-1]+1))
    for i in range_SI:
        result[i,k-1] = theta(k,"i",i,i)[0]
        backtrack[i,k-1] = i
    theta_type.append('i')
    return result,backtrack,theta_type
```

Numerous other helper functions help with the coding of the main functions. These are given in the last section of the Appendix.

11.2.1theta() and cost() functions

```
def theta(k,direction,S,SI):
    if direction == "o":
        if supply_node(k):
            range_SI = list(compress(supply_stage_SI, list(map(lambda x:x==k,supply_stage))))
        else:
            range_SI = list(range(max(0,S-processing_time[k-1]),M[k-1]-processing_time[k-1]+1))
        cost_list = []
        for i in range_SI:
            cost_list.append(cost(k,S,i,adj,holding_cost,processing_time,sigma,Z))
        return min(cost_list),range_SI[cost_list.index(min(cost_list))]
    elif direction == "i":
        if demand_node(k):
            range_S = list(compress(demand_stage_S, list(map(lambda x:x==k,demand_stage))))
        else:
            range_S = list(range(0,SI + processing_time[k-1]+1))
        cost_list = []
        for i in range_S:
            cost_list.append(cost(k,i,SI,adj,holding_cost,processing_time,sigma,Z))
        return min(cost_list),np.nan
```

```
def cost(k,S,SI,adj,holding_cost,processing_time,sigma,Z):
    term1 = holding_cost[k-1]*Z[k-1]*sigma[k-1]*math.sqrt(SI+processing_time[k-1]-S)
    if not lower_linked(k,adj):
         return term1
    elif lower_linked(k,adj)[0]:
         links = lower_linked(k,adj)[1]
         result = term1
         for j in links:
             if downstream(j,k):
                 inter = []
                 for l in range(S,M[j-1] - processing_time[j-1]+1):
                     inter.append(theta(j,"i",l,l)[0])
                 if len(inter)!=0:
                     inter_min = min(inter)
                 else:
                     inter_min = 0
                 result = result + inter_min
             if upstream(j,k):
                 inter=[]
                 for l in range(0,SI+1):
                     inter.append(theta(j,"o",l,l)[0])
                 if len(inter)!=0:
                     inter_min = min(inter)
                 else:
                     inter_min = 0
                 result = result + inter_min
         return result
```

11.3 Output matrices (result, backtrack, i/o)

Result:

(S,SI)	theta_ 1	theta_ 2	theta_ 3	theta_ 4	theta_ 5	theta_ 6	theta_ 7	theta_ 8	theta_ 9	theta_ 10	theta_ 11
0	384	138.6	26.88	29.63	61.37	80	28.49	161	157.34	114.96	101.37
1	543	196	38.01	41.9	86.79	113	40.3	227	222.52	162.57	143.36
2	665.1	240.1	46.55	51.31	106.3	139	49.35	278	272.53	199.11	175.58
3	768	277.2	53.76	59.25	122.7	160	56.99	321	314.69	229.92	202.75
4	858.6	309.9	60.1	66.25	137.2	179	63.71	359	351.83	257.05	226.68
5	940.6	339.5	65.84	72.57	150.3	196	69.8	393	385.41	281.59	248.31
6	1016	366.7	71.11	78.38	162.4	212	75.39	425	416.29	304.15	268.21
7	1086	392	76.02	83.79	173.6	226	80.59	454	445.03	325.15	286.73
I/O	**i**	**i**	**i**	**i**	**i**	**i**	**i**	**i**	**i**	**i**	**i**
(S,SI)	**theta_ 12**	**theta_ 13**	**theta_ 14**	**theta_ 15**	**theta_ 16**	**theta_ 17**	**theta_ 18**	**theta_ 19**	**theta_ 20**	**theta_ 21**	**theta_ 22**
0	51.11	22.5	66.78	55.9	55.28	39.2	34.06	84.1	5.5741	1748.4	1877
1	72.27	31.82	94.44	68.46	67.7	48	39.33	97.1	6.4364	2387.2	1896.9
2	88.52	38.97	115.7	79.05	78.17	55.4	43.97	109	7.1961	2881	1914.4
3	102.2	45	133.6	88.38	87.4	62	48.16	119	7.8829	3298.3	1930.3
4	114.3	50.31	149.3	96.82	95.74	67.9	52.02	128	8.5145	3665.9	1944.9
5	125.2	55.12	163.6	104.6	103.4	73.3	55.61	137	9.1024	3997.6	1958.4
6	135.2	59.53	176.7	111.8	110.6	78.4	58.99	146	9.6546	4300.8	1971.2
7	144.5	63.64	188.9	118.6	117.3	83.2	62.18	153	10.177	4571.5	1983.2
I/O	**i**	**i**	**i**	**i**	**i**	**i**	**i**	**i**	**i**	**o**	**i**

Backtrack:

1	2	3	4	5	6	7	8	9	10	11
0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7	7

12	13	14	15	16	17	18	19	20	21	22
0	0	0	0	0	0	0	0	0	5	0
1	1	1	1	1	1	1	1	1	5	1
2	2	2	2	2	2	2	2	2	5	2
3	3	3	3	3	3	3	3	3	5	3
4	4	4	4	4	4	4	4	4	5	4
5	5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	5	6
7	7	7	7	7	7	7	7	7	5	7

The green cells in the backtrack highlight optimal Ss for supply nodes (i/o = "o") and optimal Sis for demand nodes (i/o = "i")

Output snippet for the result:

```
print(result)
print(bt)
print(io)
```

```
[[ 383.993786    138.603312     26.877948     29.62595      61.371268
    79.990065    28.493803    160.638588    157.34257     114.95754
   101.373873    51.105288     22.501276     66.78246      55.89837497
    55.27592726   39.20399116   34.05708986   84.05833788    5.57407898
  1748.36155607 1877.01091152]
 [ 543.04922003  196.01468362   38.01115859   41.89742029   86.79207955
   113.12303478   40.29632265  227.17726979  222.51599643  162.57451217
   143.36430607   72.2737914    31.82160969   94.44466066   68.46124807
    67.69890842   48.01488711   39.32574      97.062208       6.436392
  2387.23530268 1896.91303607]
 [ 665.09674714  240.06797848   46.55397154   51.31365062  106.2981543
   138.54685668   49.3527145   278.23419607  272.52532543  199.11229999
   175.5846986    88.51695535   38.97335327  115.67061377   79.05224
    78.171966     55.442816     43.96751395  108.51884757    7.19610502
  2880.99225961 1914.44715979]
 [ 767.987572    277.206624     53.755896     59.2519      122.742536
   159.98013     56.987606    321.277176    314.68514     229.91508
   202.747746    102.210576     45.002552    133.56492      88.38309121
    87.39891496   61.98695272   48.16399838  118.87644145    7.88293809
  3298.28503081 1930.29921933]
```

Backtrack output:

```
[[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
   0.  0.  5.  0.]
 [ 1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
   1.  1.  5.  1.]
 [ 2.  2.  2.  2.  2.  2.  2.  2.  2.  2.  2.  2.  2.  2.  2.  2.  2.  2.
   2.  2.  5.  2.]
 [ 3.  3.  3.  3.  3.  3.  3.  3.  3.  3.  3.  3.  3.  3.  3.  3.  3.  3.
```

11.4 Order Quantity Estimation

Order Quantity												
	Jan.	Feb.	Mar.	Apr.	May	June	July	Aug.	Sept.	Oct.	Nov.	Dec.
Aba	1025	1025	1965	240	800	818	1612	1318	3067	180	672	566
Bazhong	5140	10630	6650	118	50	2802	4335	3720	3030	2120	2855	3901
Chengdu	19929	38896	63816	49464	2000	44676	44744	52490	50664	48200	13	7604
Dazhou	5063	9094	12380	4515	550	7051	6746	8180	7350	4530	5600	3830
Deyang	3523	3903	10586	7073	0	4694	6651	5975	5680	3922	6519	5488
Ganzi	1861	1862	1058	565	0	630	350	940	900	140	820	1800
Guang'an	4087	8218	8930	7176	240	3560	5770	4952	2699	2870	3390	3270
Guangyuan	5806	10883	8255	67	1345	4124	4628	2660	1720	1101	1135	798
Leshan	9401	18602	14072	3293	0	6451	10441	8499	7948	5725	1351	2
Liangshan	2790	6675	4325	1172	0	4639	5660	4379	4772	3366	2559	4739
Luzhou	7833	13033	10549	5351	0	6746	7890	6680	8282	3949	3730	5513
Meishan	170	12165	11994	1599	7473	2137	4180	5108	5670	2667	3439	213
Mianyang	2844	41846	18763	-1002	18424	281	1701	8440	291	325	305	0
Neijiang	2368	12997	12129	-988	0	3359	4970	5300	3300	5110	1429	3048
Nanchong	3101	23312	37465	-738	552	6587	9699	8363	7307	387	1339	0
Panzhihua	1730	3334	1586	-28	2012	2499	4079	3226	3203	2350	620	492
Suining	6006	12402	8043	-2438	0	4500	5540	4192	5189	2560	3234	2728
Ya'an	3641	5421	4110	1089	410	2379	3081	2460	2360	1910	2250	1960
Yibin	9872	20212	15775	4912	0	7601	11553	8346	5620	1701	191	630
Ziyang	4859	5001	7223	2392	0	2697	4491	4168	4456	1255	92	1
Zigong	5178	14106	11323	-606	0	3647	6046	5330	5552	3645	3645	1416

Order Quantity example:

$$Order\ Qty_i\ = Inventory\ Status\ @\ (i+1)^{th}period\ + Demand_i - \ Inventory\ Status\ @\ i^{th}period$$

Hence for District Aba during Jan:

Order_Qty(Aba) = Exhibit9 [row 1,col 1] + Exhibit2 [row 1,col 1] – Exhibit9 [row 1,col 2]

11.5 Holding Cost Estimation

Location	Avg inventory level (monthly)	Floor space (m^2)	Rent / month / m^2	Monthly rent	Weekly rent	Rent component (/unit)	Holding cost (¥/unit/week)	Holding cost (¥/unit/day)
Aba	2244	1027	5	¥ 5,135.00	¥ 1,180.46	¥ 0.53	¥ 0.67	¥ 0.10
Bazhong	3456	1640	10	¥ 16,400.00	¥ 3,770.11	¥ 1.09	¥ 1.38	¥ 0.20
Chengdu	26734	9600	15	¥ 144,000.00	¥ 33,103.45	¥ 1.24	¥ 1.57	¥ 0.22
Dazhou	7441	4772	10	¥ 47,720.00	¥ 10,970.11	¥ 1.47	¥ 1.87	¥ 0.27
Deyang	5554	10792	10	¥ 107,920.00	¥ 24,809.20	¥ 4.47	¥ 5.67	¥ 0.81
Ganzi	1732	1,907	5	¥ 9,535.00	¥ 2,191.95	¥ 1.27	¥ 1.60	¥ 0.23
Guang'an	4047	1174	10	¥ 11,740.00	¥ 2,698.85	¥ 0.67	¥ 0.85	¥ 0.12
Guangyuan	5353	2040	10	¥ 20,400.00	¥ 4,689.66	¥ 0.88	¥ 1.11	¥ 0.16
Leshan	7171	1717	10	¥ 17,170.00	¥ 3,947.13	¥ 0.55	¥ 0.70	¥ 0.10
Liangshan	4337	1296	5	¥ 6,480.00	¥ 1,489.66	¥ 0.34	¥ 0.44	¥ 0.06
Luzhou	6119	4154	10	¥ 41,540.00	¥ 9,549.43	¥ 1.56	¥ 1.98	¥ 0.28
Meishan	4114	2672	10	¥ 26,720.00	¥ 6,142.53	¥ 1.49	¥ 1.89	¥ 0.27
Mianyang	11923	3840	10	¥ 38,400.00	¥ 8,827.59	¥ 0.74	¥ 0.94	¥ 0.13
Neijiang	4957	7802	10	¥ 78,020.00	¥ 17,935.63	¥ 3.62	¥ 4.59	¥ 0.66
Nanchong	12515	6742	10	¥ 67,420.00	¥ 15,498.85	¥ 1.24	¥ 1.57	¥ 0.22
Panzhihua	2223	3534	10	¥ 35,340.00	¥ 8,124.14	¥ 3.65	¥ 4.63	¥ 0.66
Suining	5577	1999	10	¥ 19,990.00	¥ 4,595.40	¥ 0.82	¥ 1.05	¥ 0.15
Ya'an	3105	3770	10	¥ 37,700.00	¥ 8,666.67	¥ 2.79	¥ 3.54	¥ 0.51
Yibin	6743	5932	10	¥ 59,320.00	¥ 13,636.78	¥ 2.02	¥ 2.57	¥ 0.37
Ziyang	4845	3272	10	¥ 32,720.00	¥ 7,521.84	¥ 1.55	¥ 1.97	¥ 0.28
Zigong	7267	2600	10	¥ 26,000.00	¥ 5,977.01	¥ 0.82	¥ 1.04	¥ 0.15
Central warehouse	137456	82282	15	¥ 1,234,230.00	¥283,731.03	¥ 2.06	¥ 2.62	¥ 0.37

Rent component (RMB/unit/day) = ((Monthly_rent*7)/(Floor_space*30.5))/(grand_avg_inventory*(30.5/7))

Material Handling component = 0.06 * Holding cost ; Interest component = 0.05*Holding cost

11.6 Average Inventory Estimation

	Average Inventory											
Month	**Jan.**	**Feb.**	**Mar.**	**Apr.**	**May**	**June**	**July**	**Aug.**	**Sept.**	**Oct.**	**Nov.**	**Dec.**
Aba	2245	3151	4256	2867	2162	1822	2050	1690	2739	1360	1406	1178
Bazhong	4092	7697	6503	2479	1095	2065	3064	3035	2832	2362	2534	3715
Chengdu	17944	33180	51047	38795	9186	31213	29741	32650	32006	31890	5068	8095
Dazhou	9316	13111	17422	11470	3784	4768	4681	5452	6136	4398	3969	4786
Deyang	6749	7421	11508	9054	2808	3722	4557	4338	4422	3322	4742	4004
Ganzi	2777	3865	4045	2966	907	510	713	1090	1025	635	880	1375
Guang'an	3817	8411	8828	6027	1230	2323	3535	3448	2810	2522	2608	3013
Guangyuan	9390	14223	13420	6857	3611	3068	3344	2477	2287	2171	2000	1390
Leshan	6596	15631	17523	10010	3676	5295	7229	6342	6037	4668	2029	1014
Liangshan	4760	7982	7717	4170	1662	3950	4487	3853	4173	3186	2312	3797
Luzhou	7890	11419	10523	6661	2145	4752	5421	4881	6568	4824	3670	4681
Meishan	511	7476	9070	3692	6887	4385	3752	3515	4035	2568	2604	879
Mianyang	8032	36499	34067	16683	16669	6693	5245	6991	3258	3262	3404	2273
Neijiang	4718	10995	13382	5705	2864	3003	3661	4194	2868	3837	1882	2372
Nanchong	11269	30226	46534	18133	5504	4892	5901	5323	7340	5713	5392	3960
Panzhihua	2292	3292	2089	487	1614	2224	3014	2876	3201	2670	1548	1377
Suining	8861	14603	14612	6449	2580	3069	3666	2895	3430	2083	2405	2271
Ya'an	4326	6357	5865	3236	1716	2351	2662	2357	2356	1905	2015	2121
Yibin	8585	15140	14544	7391	2027	4891	7193	5517	4693	3936	3308	3690
Ziyang	7933	9799	12866	9523	3787	2315	3148	3189	3440	1502	377	260
Zigong	7659	17994	20904	10312	3888	3474	4251	3950	4707	4087	3954	2022

11.7 Order Quantity – Monthly Demand

Order Qty(Appendix 10.4) - Monthly Demand(Exhibit 2)												
	Jan.	**Feb.**	**Mar.**	**Apr.**	**May**	**June**	**July**	**Aug.**	**Sept.**	**Oct.**	**Nov.**	**Dec.**
Aba	964	849	421	-1474	-497	-200	-138	-289	639	-510	109	-459
Bazhong	484	1235	358	-1875	-824	12	452	106	177	-206	-185	1501
Chengdu	3569	7935	2880	-13032	1277	102	-3115	1188	-651	2883	-8340	6803
Dazhou	466	3094	2241	-6279	-5129	597	-466	573	1625	-2281	354	3050
Deyang	682	282	1210	-2605	-2814	-52	-235	473	-11	-430	672	-1117
Ganzi	1087	1088	75	-1739	-1814	390	295	-130	40	-60	-130	140
Guang'an	2236	2822	-2700	-1149	-1509	375	-160	804	173	-921	573	358
Guangyuan	2870	1718	-695	-4244	-3526	-339	388	-154	713	-326	-50	-833
Leshan	2723	6146	2167	-6413	-2963	-249	126	42	-101	-413	-491	-190
Liangshan	461	2098	-277	-3665	-178	114	-61	74	174	-743	-197	986
Luzhou	1102	756	-64	-2461	-1220	-312	505	-374	2146	-1302	-787	1026
Meishan	170	1765	1594	-1955	2471	-2140	-1168	-234	711	-642	-58	-165
Mianyang	2644	15288	2930	-17932	-1522	-287	-4029	782	-100	74	230	-2187
Neijiang	-1006	2931	2711	-4949	-1721	-1359	1064	-328	-324	452	-682	43
Nanchong	709	16993	1470	-20068	-6480	-780	-313	492	4599	-934	-660	-865
Panzhihua	255	142	-800	-791	1006	-274	274	303	370	-579	65	-278
Suining	1280	3807	571	-6416	-3761	239	-85	-109	182	-247	218	20
Ya'an	1446	837	-510	-1728	-632	-67	-14	25	74	-527	407	95
Yibin	301	2470	775	-4219	-1597	-276	929	-1074	2152	253	0	325
Ziyang	1849	1740	2173	-4029	-5051	-589	461	-56	269	-943	-144	0
Zigong	3628	8114	488	-9742	-3712	-763	-82	196	1096	-429	163	-1799