MASTER

The development of transportation policy for excess inventory pullback operations at Cisco

Qian, Y.

Award date:
2009

Eindhoven, April 2009

The Development of Transportation Policy for Excess Inventory Pullback Operation at Cisco

by
Yong Qian

BSc Electrical Engineering
Student Identity Number: 0620945

in partial fulfillment of the requirements for the degree of

Master of Science
in Operations Management and Logistics

Supervisors:

Dr. Tarkan Tan	TU/e
Prof. A. G. Ton de Kok	TU/e
Klaas van de Meent	Cisco System
Marco van Duijnhoven	Cisco System

Abstract

This master thesis describes the development of an optimized model and solution for excess pullback management in European region for Cisco System. Excess pullback management is the term for management of pulling excess inventory from European local depots back to Cisco central warehouse. This optimal solution describes transportation policy on the basis of total cost consideration in excess pullback procedure, which stating when shipments must take place in which depot for each shipment, in such a way that the sum of transportation cost, handling cost and management cost are minimized. The model has been incorporated in a program that enables Cisco System to use optimal solution in excess pullback management.

Executive Summary

Cisco System, Inc is the worldwide leader in networking for the Internet. Today, networks are an essential part of business, education, government, and home communications. Cisco hardware, software, and service offerings are used to create the Internet solutions that make these networks possible, giving individuals, companies, and countries easy access to information anywhere, at any time. In addition, Cisco has pioneered the use of the Internet in its own business practice and offers consulting services based on its experience to help other organizations around the world.

Cisco SSCD (Service Supply Chain Delivery) is responsible for the service of Cisco products in European market. Advance Hardware Replacement is one of most important components of technical services. In order to supply high-quality service to European customers, Cisco locates one central warehouse (NL1) in Helmond (The Netherlands) and 160 local depots (RMA) around European region.

Cisco offers four Advance Hardware Replacement (AHR) service for customers, which are:

1. Same Day Delivery
2. Next Business Day Delivery,
3. Two-hour Delivery
4. Four-hour Delivery

The first two services are operated by central warehouse in Helmond, which means Cisco ships the spare parts in central warehouse to customers directly. The last two contracts are operated by local depots which are close to the local customers, because the delivery time is very short. If customers want to make changes on their product contracts, for example, one customer wants to change his service contract from Four-hour Delivery to Same Day Ship, which results excess spare parts in that local depot, because the service for the customer is switch from local depot to central warehouse. Cisco needs to pull back the excess spare parts to central warehouse to fulfill its service demand. In the mean time, if the customer updates his contract from Same Day Ship to Four-hour Delivery who is near another depot, Cisco needs to ship more spare parts from central warehouse to that depot in order to keep high service level. Sometimes, customers want to update their Cisco products, and then old version of products can not be used anymore, which causes previous spare parts obsolete in this nearby local warehouse. Cisco needs to pull back these obsolete spare parts to central warehouse in Helmond as well. While central warehouse needs these excess spare parts to supply its customers directly. The customer service contracts are changing frequently and the new service will be started immediately after customers' changing request. Currently, all local depots in European region face the problem of excess stock. Cisco faces difficulty to deliver high service level with low cost, Because Cisco can not utilize excess stocks, and has to deliver more spare parts from central warehouse to the local depots when they have shortage problem.

In order to solve above problem, currently Cisco pulls these excess stocks back to its central warehouse (NL1) periodically. The purpose of excess pull back is to redistribute the excess stock among all local depots including central warehouse. Cisco believes current excess pull back procedure should be reviewed because it has possibility to be improved on the basis of total cost. That is the point where this project initiates.

Because the inventory policies in central warehouse (NL1) and local depots are given by Cisco planning team in UK, this project is mainly focused on optimizing transportation policy during excess pull back procedure, analyze current excess pullback procedure in Cisco and develop models for improvement of transportation policy. Specifically:

> Develop model of optimal transportation policy on the basis of total cost consideration in excess pullback procedure, which states when shipments must take place in which depot for each shipment, in such a way that the sum of transportation cost, handling cost and management are minimized.

In this project, the scope is only within European region (geographic regions defined by Cisco System: UK & Ireland, Germany, Italy, France, Nordics, Benelux, Alpine, Mediterranean). Third-part logistics provider Kuehne+Nagel are operating excess pullback procedure. Because Cisco planning team in United Kingdom has decision on inventory policy, optimization of inventory policy is not considered in this project. All the parameters related to inventory policy is treated as given.

During this project, a proposed excess pullback plan has been drawn. The model is set on the basis of min-knapsack problem. Csirik, etc (1991) proposed some very useful algorithms, which is used for solving min-knapsack problem. A proposed program is made based on proposed model and three algorithms. After comparison between original plan and proposed plan, it can be seen that the proposed plan has significant improvement on the basis of total cost. Also, the proposed model has improved on the value of cost per pullback part.

Three conclusions can be drawn as follows:

1. The original plan has been improved significantly by proposed plan. The original plan was mainly drawn based on amount of excess parts in each depot; the depots which have most excess spare parts are selected as pullback depots. This plan is very rough and imprecise. The original plan has a lot of space to be improved. The proposed plan is modeled on basis of min-knapsack problem, and the transportation problem is solved by algorithm, which makes transportation plan more efficient and less cost.

2. The result of comparison between original plan and proposed plan shows the significant improvement of proposed plan, both on total involved cost and cost per pullback part. The saving percentages of two criteria are 27.8% and 28.7% respectively.

3. The outcome of proposed plan is not only reducing cost and but also improving service quality. In proposed plan, the value of cost per part has significantly decreased as total cost, which also means Cisco System can pull back much more excess parts with same amount of total cost as before. Under this circumstance, Cisco System can fulfill more demands from different customers, which means the improvement on service quality.

Recommendations for further research are stated as below:

1. Transshipment in excess pullback procedure

Tagaras (1999) pointed that lateral transshipment is an effective means of improving customer service and reducing total system costs. Lateral transshipment in our project means distributing excess parts among all local depots. If transshipment is involved in this project, not all excess parts need to be pulled back to central warehouse, and then sent to another depot; some excess parts can be distributed among depots. Excess parts are delivered directly from one depot to another depot which needs these excess parts. This integrated plan is more efficient and more complicated, which can be one of research areas in the future.

2. Maximize pallet usage

In the proposed plan, calculation for pallet loading volume is based on Cisco historical data, which is imprecise. In the literatures, there exists a research area which is about pallet loading problem. The pallet loading problem is generally tackled by attempting to maximize the number of rectangles that can be fitted orthogonally within a larger containing rectangle. The main issue is about maximizing the pallet space to load as many parts as possible. Several methods have been studied in recent years. But these methods may produce solutions which do not satisfy such real-life problems as load stability and transportability. Carpenter (1985) indicates that the criteria to cater for above problems are not only depending on the type of layout proposed and the way in which the pallet stack is built up. The physical characteristics of the boxes being loaded and the requirements of individual packaging staff also need to be considered. Besides that, in real life, no one would like to spend much time to load products with proposed pallet loading solution, which may only save a little cost. All above constraints show the difficulties of making ideal pallet loading solution. But from other point, because of its difficulty, pallet loading problem with practical consideration would be interesting to investigate in future research.

3. Different values of excess parts

In this project, Cisco treats all excess parts as same value, which means no priority of parts when they are pulled back. But in some situations, excess parts do have priority. So putting excess inventory pullback priority into consideration would be future research area.

Considering the significant improvement which proposed plan has made, it is strongly suggest Cisco starting to use this proposed plan immediately, especially during this recession time.

Acknowledgement

You are reading the final thesis for the Master of Operations Management and Logistics (sub program in Operations, Planning, Accounting, and Control) from Eindhoven University of Technology by Yong Qian.

My project takes place at Cisco System, a worldwide leader in networking for Internet. The project is focused on drawing an optimal transportation policy on the basis of total cost consideration in excess pullback procedure, which stating when shipments must take place in which depot for each shipment, in such a way that the sum of transportation cost, handling cost and management are minimized. I developed an optimal model which fulfills the above requirements and the corresponding program. The program can be used to draw optimal solution for excess pullback procedure on the basis of excess parts information. This model will be implemented in Cisco System soon.

The final thesis could not be finished without the help of a number of people, who I would like to thank.

First of all, I would like to thank my university supervisors. Dr Tarkan Tan, who is my first university supervisor, supported me during entire project with his expertise in operations management. As an international student, I met more problems than Dutch students, not only in study but also in life, which made Dr Tarkan Tan put more energy on me. Hereby I give my sincere regards to him. I also would like to thank Prof. Ton de Kok. His helpful suggestions are very useful to make this project success.

Second, I would like to thank my supervisors in Cisco System. Klaas van de Meent, my first supervisor in Cisco System, who helped me to find the way to work in Cisco System and provided me all the useful data for this project. Furthermore, I also would like to thank Marco van Duijnhoven, Hans-Richard, and all colleagues in Cisco System and Kuehne+Nagel. During this project, they gave me a lot of suggestions which are very useful for project.

Final I would like to thank my dear wife Ying for supporting me all the time whenever I am down or up.

Yong Qian
Eindhoven, April 2009

1. Company Description

1.1 Cisco System

Cisco System, Inc is the worldwide leader in networking for the Internet. Today, networks are an essential part of business, education, government, and home communications. Cisco hardware, software, and service offerings are used to create the Internet solutions that make these networks possible, giving individuals, companies, and countries easy access to information anywhere, at any time. In addition, Cisco has pioneered the use of the Internet in its own business practice and offers consulting services based on its experience to help other organizations around the world.

Cisco was founded by a group of computer scientists from Stanford University in 1984, headquartered in San Jose, California. After 24 years development, Cisco System has become a multinational corporation with more than 66,000 employees and annual revenue of US$39 billion in fiscal year of 2008. Figure 1.1 shows overview of income and revenue from 2003 to 2007. The Cisco name has become synonymous with the Internet, as well as with the productivity improvements that Internet business solutions provider. Cisco designs and sells networking and communications technology and services under five brands, namely Cisco, Linksys, WebEx, IronPort, and Scientific Atlanta. At Cisco, the vision is to change the way people work, live, plan and learn.

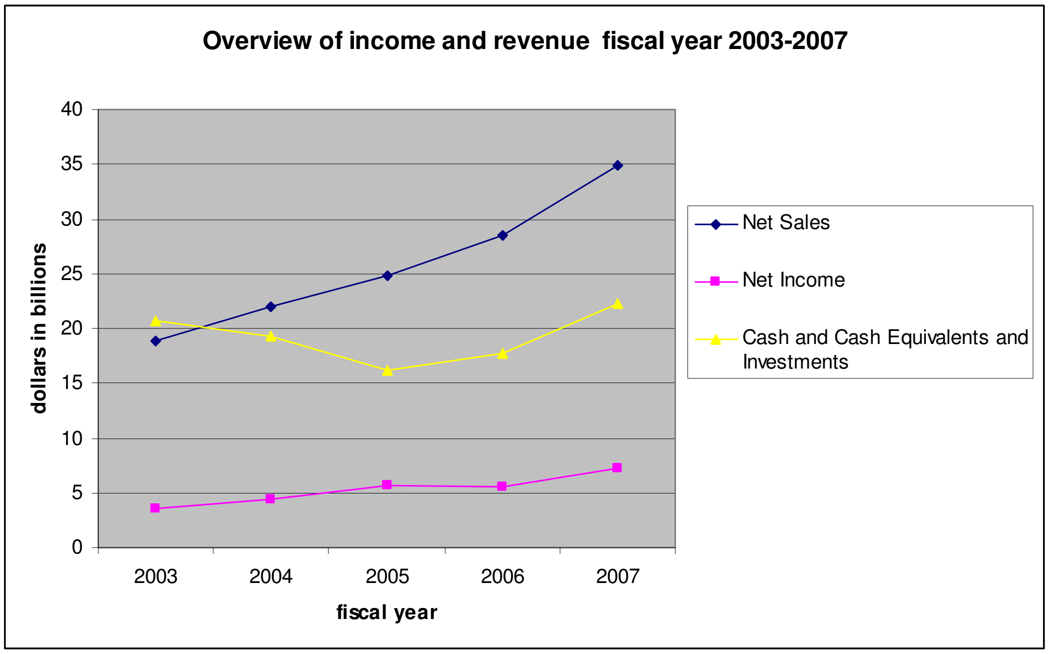


Figure 1.1 Overview of Income and Revenue Fiscal 2003-2007

1.2 Products and Core Technologies

Cisco innovation continues with industry-leading products in the core areas of routing and switching, as well as advanced technologies in areas such as home networking, IP Communications, optical solutions, network security, storage networking, and wireless LAN technology. Cisco remains committed to creating secure networks that are smarter, faster, and more durable, with a generational approach to an evolutionary infrastructure.

1.3 Market

Cisco divides its global market into six theaters:

■ West Emerging Markets, including Latin America and the Caribbean

■ East Emerging Markets, including the Middle East and Africa, Russia and other members of the Commonwealth of Independent States (CIS), and eastern European nations.

■ European Markets, including more than 20 countries, which are United Kingdom & Ireland, Germany, Italy, France, Nordics, Benelux, Alpine, Mediterranean.

■ United States and Canada Markets

■ Asia Pacific Markets, all Asia Pacific countries except Japan

■ Japan theatre

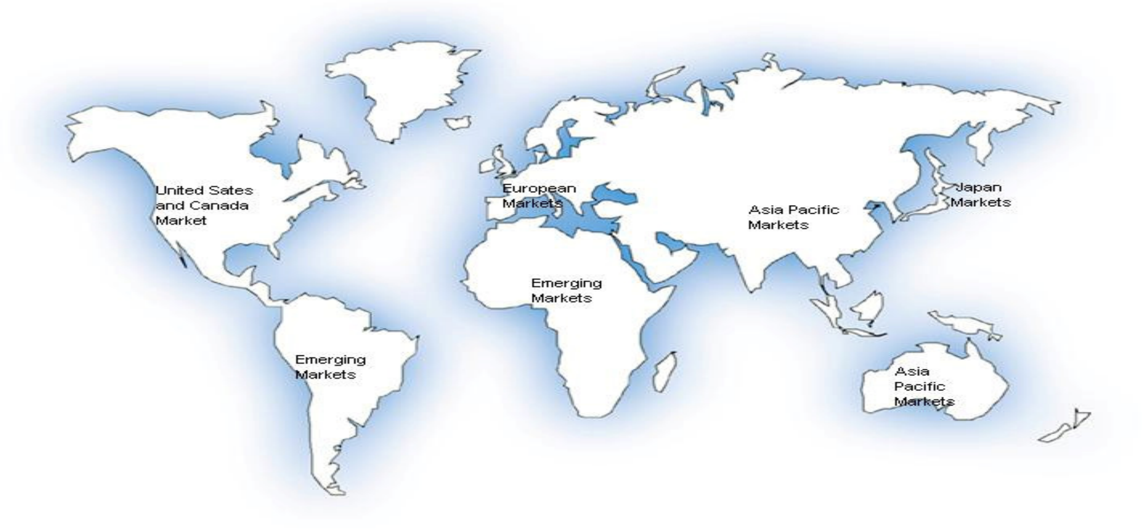


Figure 1.2 Cisco System Worldwide Markets

1.4 Service and Supply Chain Management

Cisco Customer Advocacy mission is to accelerate customer success with Cisco through innovative services and world-class people, partners, process, and tools. From its mission, it can be seen that service is very important for Cisco. Cisco Technical Services have four components:

1. Software support: Innovative System and application software and IT-related support that help maximize customer’s technology investment

2. Cisco.com: Increases customer’s self-sufficiency and productivity with registered access to online tools and resources

3. Advance Hardware Replacement: Flexible and responsive hardware replacement support that helps maximize customer’s operational reliability

4. Technical Assistance Center (TAC): Supplements customer’s in-house staff with access to highly-trained network and application software engineers and R&D engineers

2. Research Assignment and Methodology

2.1 Problem Definition

Cisco SSCD (Service Supply Chain Delivery) is responsible for the service of Cisco products in European market. Advance Hardware Replacement is one of most important components of technical services. In order to supply high-quality service to European customers, Cisco locates one central warehouse (NL1) in Helmond (The Netherlands) and 160 local depots (RFD) around European region. The map of European depots is show in figure 2.1

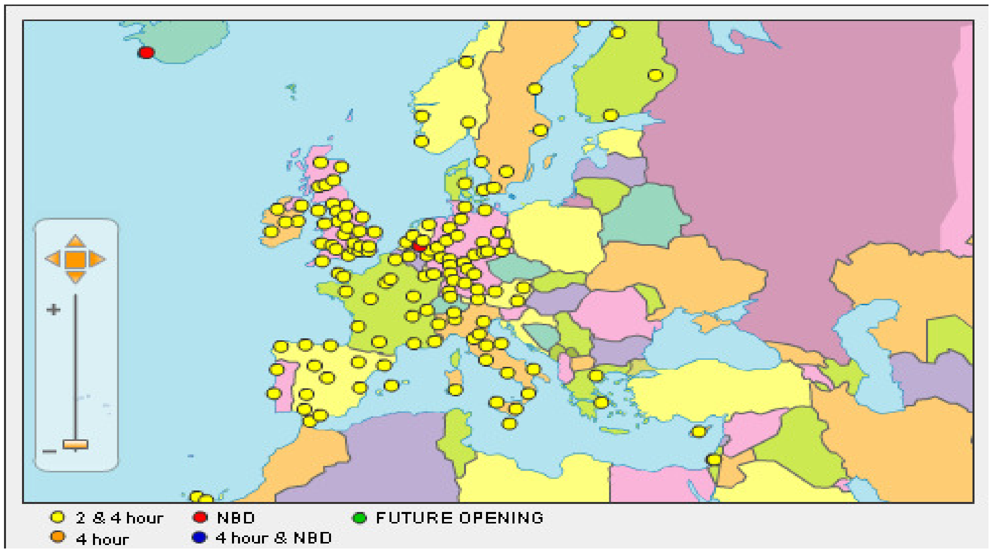


Figure 2.1 Cisco System Local Depots in European Region

Cisco offers four Advance Hardware Replacement (AHR) service for customers, which are:

1. Same Day Delivery
2. Next Business Day Delivery (NBD)
3. Two-hour Delivery
4. Four-hour Delivery

When Cisco products failed, the fastest way of Advance Hardware Replacement service is Two-hour Delivery, which means the new spare part will be delivered to customers within two hours. The other three deliveries are within four hours, same day, and next business day respectively.

The first two services are operated by central warehouse in Helmond, which means Cisco ships the spare parts in central warehouse to customers directly. The last two contracts are operated by local depots which are close to the local customers, because the delivery time is very short. If customers want to make changes on their product contracts, for example, one customer wants to change his service contract from Four-hour Delivery to Same Day Ship, which results excess spare parts in that local depot, because the service for the customer is switch from local depot to central warehouse. Cisco needs to pull back the excess spare parts to central warehouse to fulfill its service demand. In the mean time, if the customer updates his contract from Same Day Ship to Four-hour Delivery who is near another depot, Cisco needs to ship more spare parts from central warehouse to that depot in order to keep high service level. Sometimes, customers want to update their Cisco products, and then old versions of products are not used anymore, which causes previous spare parts obsolete in this nearby local warehouse. Cisco needs to pull back these obsolete spare parts to central warehouse in Helmond as well, while central warehouse could use these obsolete spare parts to supply its customers directly. The customer service contracts are changing frequently and the new service will be started immediately after customers' changing request. Currently, all local depots in European region face the problem of excess stock. Cisco faces difficulty to deliver high service level with low cost, therefore Cisco can not utilize excess stocks, and has to deliver more spare parts from central warehouse to the local depots when they have shortage problem.

In order to solve above problem, currently Cisco pulls these excess stocks back to its central warehouse (NL1) periodically. The purpose of excess pull back is to redistribute the excess stock among all local depots including central warehouse. Cisco believes current excess pull back procedure should be reviewed because it has possibility to be improved on the basis of total cost. That is the point where this project initiates.

2.2 Research questions and project boundaries

Because the inventory policies in central warehouse (NL1) and local depots are given by Cisco planning team in UK, this project is mainly focused on optimizing transportation policy during excess pull back procedure, analyze current excess pullback procedure in Cisco and develop models for improvement of transportation policy. Specifically:

> *Develop model of optimal transportation policy on the basis of total cost consideration in excess pullback procedure, which states when shipments must take place in which depot for each shipment, in such a way that the sum of transportation cost, handling cost and management cost are minimized.*

In this project, the scope is only within European region (geographic regions defined by Cisco System: UK & Ireland, Germany, Italy, France, Nordics, Benelux, Alpine, Mediterranean). Third-part logistics provider Kuehne+Nagel are operating excess pullback procedure. Because Cisco planning team in United Kingdom has decision on inventory policy, optimization of inventory policy is not considered in this project. All the parameters related to inventory policy is treated as given.

2.3 Research design (methodology)

The research approach of master project in Industrial Engineering and Management is proposed by Van Aken, Van der Bij and Berends (2006). This methodology suggests that there are two phases during a master project: diagnostic phase and redesign phase.

The first phase of project should be diagnostic phase. The purpose of this phase is to diagnose the current situation by literature theory and interview, identify the problem to be talked. The structure of diagnostic phase is depicted as follows:

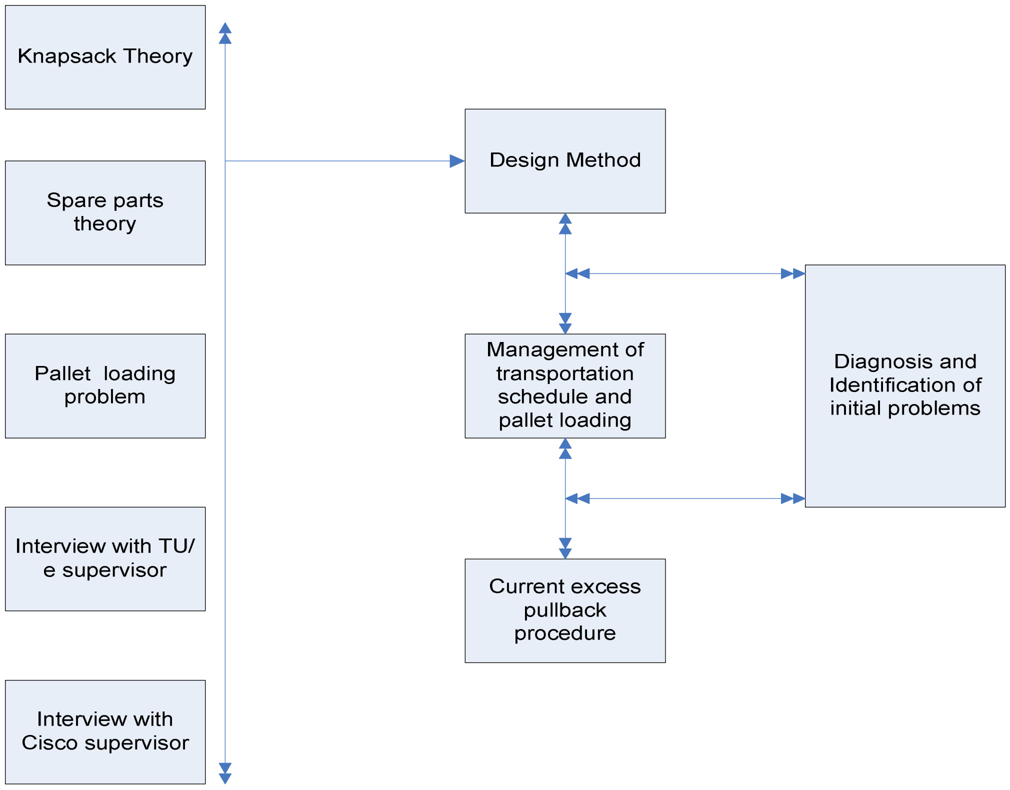

Knapsack Theory
Spare parts theory
Pallet loading problem
Interview with TU/e supervisor
Interview with Cisco supervisor
Design Method
Management of transportation schedule and pallet loading
Current excess pullback procedure
Diagnosis and Identification of initial problems

Figure 2.2 Research model for diagnostic research

The second phase of project is redesign phase. After identifying initial problems, the model of problem solution is proposed in this phase. In order to propose redesign model and implementation plan, design guidelines and constraints need to be obtained which are acquired by literature study and interview with experts. The research model for this phase is depicted as below.

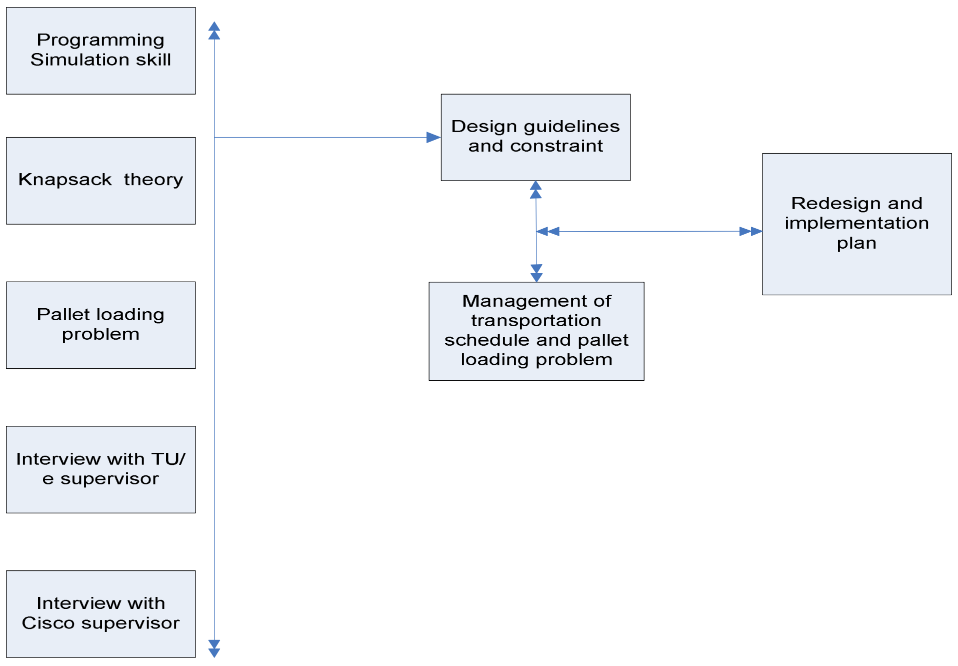

Programming Simulation skill
Knapsack theory
Pallet loading problem
Interview with TU/e supervisor
Interview with Cisco supervisor
Design guidelines and constraint
Management of transportation schedule and pallet loading problem
Redesign and implementation plan

Figure 2.3 Model for design oriented research

2.4 Structure of the remainder of this report

The remainder of this report is structured as follows:

Section 3 is the analysis of current excess pullback procedure. In section 4, the proposed excess pullback model is given and several algorithms are proposed for solving problems. Section 5 shows computational results of proposed model, and the comparison between this model and original plan. Section 6 and 7 are about implementation of proposed plan and future improvement for excess pullback procedure respectively.

3. Current practice for excess pullback

The below analyses are made on the basis of interviews with employees from Cisco and Kuehne+Nagel.

As stated in section 2.1, Cisco system has around 160 local depots in European region, and Cisco tries to pull back excess stock from these depots weekly by minimum cost. The overall structure of current practice of excess pullback at Cisco System is depicted in figure 3.1(arrows depict goods flows):

.

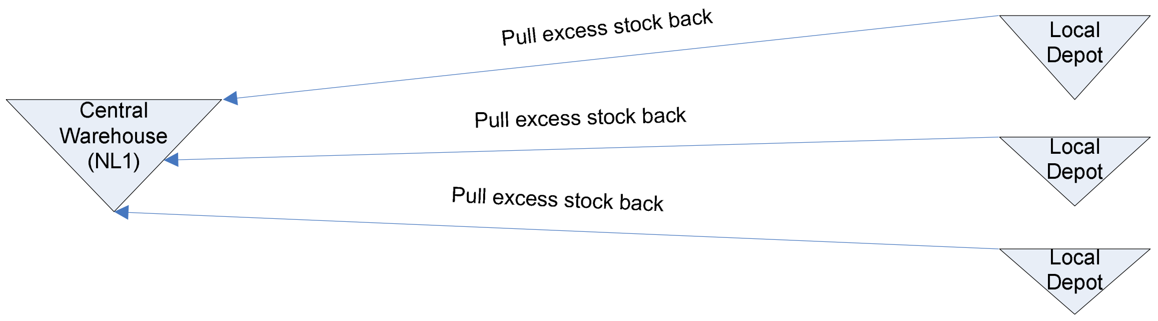


Figure 3.1 Overview procedure of excess pullback

From the above figure, it can be seen that all excess spare parts is pulled back from local depots to central warehouse in Helmond. The direction of goods flow is one way.

3.1 Excess spare parts

The target of this project is dealing with excess inventory. Cisco planning team has its definition of excess inventory in each local depot. Cisco planning team sets T4 inventory level in each local depot. T4 is an inventory level threshold, when inventory level in one local depot is above T4 level, then this local depot has excess inventory, otherwise no excess inventory in this local depot. Cisco only has available excess inventory data from week 49 of 2007 to week 13 of 2008. Cisco started to execute excess pullback operation from week 49 of 2007, which is the reason why they only have detail data from that time. Excess inventory level in European countries can be found in figure 3.2

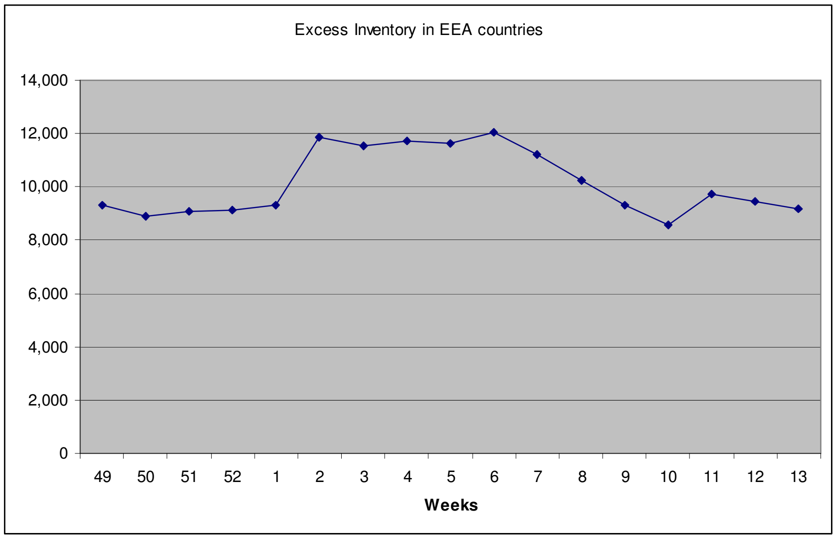


Figure 3.2 Excess inventory in EEA countries

In week 49 of 2007, the total sum of excess inventory parts (SKU) in EEA countries is 9316. The Cisco planning team stated that the total amount of inventory in EEA countries is around 55000 parts. The excess inventory in EEA countries accounts for around 17% of total inventory, while Cisco's target is at most 10% of total inventory. From fiscal year of 2009, Cisco's target level of excess inventory becomes 5% of total inventory. From week 49 of 2007, Cisco started to execute excess pullback procedure. After several weeks excess pullback operation, the total amount of excess inventory in week 13 of 2008 is 9199, which means the Cisco's pullback plan did not work very well. And there is still big gap between target level (10%) and current excess inventory level.

3.2 The procedure of excess pullback

From the stock redistribution perspective, Cisco evaluates the benefits of excess pullback is much more than costs involving excess pullback process which include handling costs in both pullback depots and central warehouse (NL1), transportation costs from pullback depots to central warehouse, and management costs involves in excess pullback procedure. The more excess spare parts are pulled back from local depots, the more spare parts can be used for future usage. The benefits are very significant from Cisco perspective. That's the reason why Cisco wants to pull back excess parts as many as possible.

As stated in section 3.1, Cisco started current excess pullback procedure from week 49 of 2007. Currently, the whole duration of excess pullback procedure is three weeks. When Cisco decides the excess parts of which depots need to be pulled back, these depots start picking up parts, loading parts, and all management work in first week. It takes another week to transport excess parts from all pullback local depots to central warehouse (NL1). The third week is for handling time in central warehouse (NL1) including duration of unloading, packing, and management work. On the basis of this process, the minimal pullback interval can be drawn – one week. The process is depicted in figure 3.3:

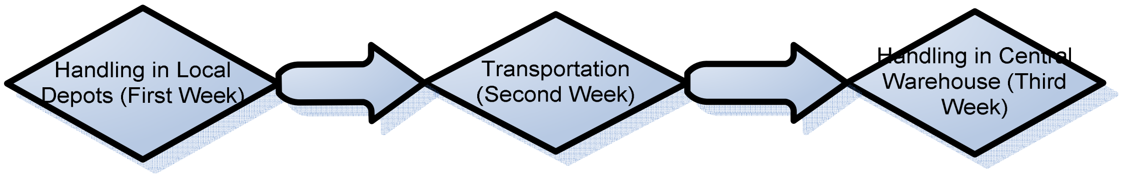

Handling in Local Depots (First Week)
Transportation (Second Week)
Handling in Central Warehouse (Third Week)

Figure 3.3 Excess pullback duration

Currently, every week Cisco pulls back excess stock from several local depots to central warehouse (NL1). Cisco sorts all local depots in European region on the basis of their quantity of excess spare parts. Cisco selects several depots whose the amount of excess stock are most. The basic rule is the number of total involved pullback local depots is no more than 15; total number of involved pullback countries is no more 2, and all involved pullback local depots should be operated by same depot vendor (either 3PL2 or 3PL1). The more countries involved in excess pullback process, the more management cost is spent, and more complex the process is.

After excess parts have been delivered to central warehouse, these parts need to be packing in order to be used for future usage, because there are many old labels on boxes (transportation label, etc). The maximal pullback amount of excess parts within one week is 1000(±5%) parts due to handling capacity in central warehouse (NL1) in one week.

3.3 The cost components of excess pullback

In section 3.2, it is stated that the total cost involves in excess pullback consists of three components which are management cost during excess pullback, transportation cost from local depots to central warehouse (NL1), handling cost in both pullback depots and NL1. Figure 3.4 depicts this three cost components.

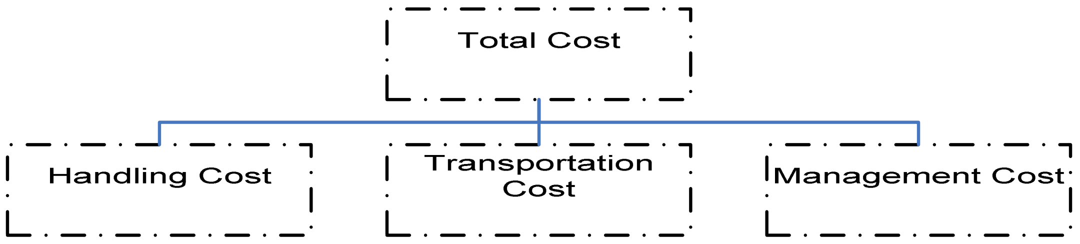


Figure 3.4 The cost components of excess pullback

The management cost includes all labor costs related to transportation coordination, depot management and other costs during excess pullback process. The transportation of excess pullback is operated by Kuehne+Nagel. Kuehne+Nagel charges Cisco for transportation cost based on the amount of delivered pallets. The charging price of delivering one pallet is different from location to location within European region. Future discount can be obtained by delivering more pallets from one location. Because of confidential reason, pallet charging costs of each depot are not listed. Handling cost includes both handling costs in pullback depots and central warehouse (NL1). Handling costs in local depots are charged differently by vendors, and handling cost is charged on the amount of delivered parts. Currently, two depot vendors (3PL1 and 3PL2) are operating the local depots in European region. The average handling cost for picking a part at 3PL2 is 2.3 euro per part, for 3PL1 it is 2.1 euro.

3.4 Performance of current plan

Cisco started its excess pullback procedure from week 49 of 2007. Due to the lack of historical data, only 6 sets of excess pullback data are available, which are week 51 of 2007; week 4, 8, 9, 10, 11 of 2008. The performance (based on total cost) of current excess pullback plan is listed as below:

Week 51 of 2007
5 Depots in United Kingdom were involved (3PL1)

Depot Name	Pullback Parts	Delivered Pallets	Transportation Price	Handling fee (Parts*2.1)	Cost/Part	Total cost(Euros)
GLC	87	4	350			
LDS	138	9	567			
LHR	143	8	470			
LTN	103	5	391			
MAN	253	10	600			
	724		2377	1520.4	5.38	3897.4

Table 3.1 Current plan performance of week 51 of 2007

It can be seen that, in week 51 of 2007, 5 depots in UK from 3PL1 are involved in excess pullback procedure. The total pullback amount is 724. And total cost is 3897.4 euros. The value of cost/part is 5.38.

Week 4 of 2008
6 Depots in United Kingdom were involved (3PL1)

Depot Name	Pullback Parts	Delivered Pallets	Transportation Price	Handling fee (Parts*2.1)	Cost/Part	Total cost(Euros)
SWS	53	3	369			
WRX	69	2	332			
XVH	114	3	312			
BHX	140	6	472			
CWL	65	2	264			
EMA	96	5	419			
	537		2168	1127.7	6.14	3295.7

Table 3.2 Current plan performance of week 4 of 2008

Week 8 of 2008
9 Depots in Germany were involved (3PL2)

Depot Name	Pullback Parts	Delivered Pallets	Transportation Price	Handling fee (Parts*2.3)	Cost/Part	Total cost(Euros)
AAC	129	7	232			
BER	63	9	412			
BFE	111	7	239			
BRE	76	5	211			
CGN	134	5	192			
D1S	183	10	244			
DTM	63	4	191			
F1A	83	8	302			
GIE	51	5	212			
	900		2235	2070	4.78	4305

Table 3.3 Current plan performance of week 8 of 2008

Week 9 of 2008
10 Depots in Germany were involved (3PL2)

Depot Name	Pullback Parts	Delivered Pallets	Transportation Price	Handling fee (Parts*2.3)	Cost/Part	Total cost(Euros)
HAJ	48	2	176			
HAM	139	6	271			
LEJ	56	3	247			
MUC	203	7	417			
NUE	94	5	282			
QFB	55	3	247			
QKA	99	4	231			
QOL	63	7	356			
RLG	43	1	216			
STR	146	7	324			
	946		2767	2175.8	5.23	4942.8

Table 3.4 Current plan performance of week 9 of 2008

Week 10 of 2008
10 Depots in France were involved (3PL2)

Depot Name	Pullback Parts	Delivered Pallets	Transportation Price	Handling fee	Cost/Part	Total cost(Euros)

				(Parts*2.3)		
BOD	63	6	420			
CDG	140	5	227			
ETZ	212	7	274			
LIL	54	2	125			
LYS	98	4	305			
MRS	66	4	357			
NCE	77	8	560			
ORE	190	6	277			
RNS	93	6	355			
TLS	56	5	359			
	1049		3259	2412.7	5.41	5671.7

Table 3.5 Current plan performance of week 10 of 2008

Week 11 of 2008
7 Depots in Italy were involved (3PL2)

Depot Name	Pullback Parts	Delivered Pallets	Transportation Price	Handling fee (Parts*2.3)	Cost/Part	Total cost(Euros)
AOI	70	3	531			
BLQ	130	5	591			
FCO	223	7	1032			
FLR	62	4	615			
MXP	257	8	815			
T1N	154	5	577			
VCE	115	7	686			
	1003		4847	2306.9	7.13	7153.9

Table 3.6 Current plan performance of week 11 of 2008

From above figures, it can be seen that

1. The total quantities of pullback excess parts in each week vary from 547 to 1049. It seems that there is no fix target quantity for weekly excess pullback operation, which makes current operation disorderly.

2. All weekly excess pullback operations are executed within same country. If new plan can be executed within several countries, it should be more flexible than current one.

3. There is no management cost is considered in above calculation. In section 4.1, the forth assumption will explain the management cost consideration in this project.

4. In each table, there are values for total cost and cost per excess part of excess pullback operation. It is impossible to know whether current plan is good or bad without

comparison. In section 5, the proposed plan will draw its result based on same sets of data, and give the comparison.

4. Excess pullback model

4.1 Assumptions:

For ease of reference all assumptions for this model are stated here:

1. According to Cisco planning team, it is given that either pulling back all excess parts in one depot or keeping these excess parts. Every week, Cisco pulls back as many parts from depots as possible under the restriction of handling capacity (1000±5%) in central warehouse (NL1).

2. The values of all excess parts are treated as same in this project, because it is impossible to evaluate the real value of parts which including spare parts' price and future value of excess parts. This assumption makes no pullback priority for different excess inventory categories. The benefits of pulling back each excess part are same; the only consideration in making plan is the cost of pulling back each excess part.

3. Within three cost components, handling cost in central warehouse (NL1) is not considered in this report. Because during each excess pullback procedure, the total amount of pullback parts is always within some range (1000±5%). The difference of handling cost in central warehouse (NL1) can be negligible, which means the only considered handling cost is the handling cost of all involved local depots.

4. As stated in section 3.2, management cost of excess pullback process is increasing with increase amount of involved local depots, countries and depot vendors. Compared to management cost which involves different depots and different countries, the management cost involves different depot vendors is extremely high. Because management cost of different depot vendors is dominant in all management cost components, only this cost component is considered in rest of report. Within each excess pullback process, the management costs of depot vendors are same (3PL1 = 3PL2).

5. Several depots are not considered in proposed excess pullback plan, because there is no fix transportation tariff for these depots now. Cisco will have tariff information for these depots soon. Once Cisco gets the information, it can be added to the proposed model easily. The list of these depots can be found in Appendix I. After removing these depots form the list, the costs of excess pullback operations for the rest local depots can be calculated accurately, and it is easier to compare with current excess pullback plan.

6. Cisco's target for excess inventory is at most 10% of total inventory at this moment. In proposed solution, amount of total inventory is not considered. Because the data of total inventory level of European region is obtained from Cisco planning team in UK, and it is hard to be got simultaneously and accurately. After the implementation of this proposed plan, Cisco checks if the excess inventory is more than 10% of total inventory before using proposed solution. If it is, starting to use the proposed model to get an optimal excess pullback plan, otherwise, no excess pullback plan is executed.

7. The total amount of weekly pullback parts in each candidate solutions should be within amount range between 950 and 1050. If amount of pullback parts of two candidate solutions are both within this range, the pullback effect of these two candidate solutions are treated as same. The only comparing criterion is the cost. For example, if total amount of pullback parts in one solution is 980, while the other one is 990, from amount of pullback parts perspective, no one is superior to the other. The only consideration is cost, the less cost, the better solution. The reason of setting an amount range not a specific value is because of handling capacity in central warehouse. As stated in section 4.1, the weekly handling capacity in central warehouse is 1000±5% parts. If one excess pullback solution is an optimal solution, the amount of pullback parts should be consistent with handling capacity. For instance, the handling capacity of this week is 980, and then the solution of pulling back 990 parts is worse than solution of 980; while when weekly handling capacity is 1000, the 990 solution is better than 980. That's why amount range is set not a specific value.

8. According to Kuehne+Nagel's regulation, five Italian island depots should be treated as one depot, which means either pulling back all excess parts in these five depots simultaneously or keeping all excess inventory in five island depots . These codes of five depots are BRI, NAP, PMO, CTA, and C2G.

4.2 Literature review

In this project, managing excessive stocks is the core problem. Holding excessive stock is very harmful for inventory management of an organization. Wee and Chung (2005) stated the disadvantage of holding excessive stocks, which reduce working capital, decrease return on investment, and increase costly storage space. Managing excessive stocks is therefore necessary in efficient inventory management of an organization.

Tersine and Toelle (1984) stated the reasons for occurrence of excessive stocks, which are

1. A redesign of a product
2. a change in methods of production
3. a reduction in demand for a product
4. new technological innovations
5. forecast errors
6. over-zealous purchasing practices
7. record keeping errors;
8. an introduction of new products

Several ways can be done to manage excessive stocks, Tersine and Toelle (1984) stated that reduction of inventory can be accomplished in three modes: (1) increase the outflow (demand) of items, (2) limit the inflow (supply) of items, (3) reduce the level of surplus items, in this project Cisco System chooses to use the third option, which is reducing the level of surplus items.

The research question of this project is to develop model of optimal transportation policy on the basis of total cost consideration in excess pullback procedure, which stating when shipments must take place in which depot for each shipment, in such a way that the sum of transportation cost, handling cost and management are minimized. In this project, the core problem in excess pullback model is how to select pullback depots from all local depots of European region in one excess pullback procedure. Every week, excess inventory in 160 European local depots have to be pulled back to central warehouse (NL) with a weekly capacity of 1000±5%, each local depot j needs cost c_j to pull back the excess parts a_j , and the problem is to select a subset of local depots whose total pullback parts are within capacity of 1000±5%, while the total cost involved is minimum.

On the basis of above analysis, the research problem can be modeled as follows:

$$Min(z) = \sum_{j=1}^{n} c_j x_j$$

(eq. 4.1)

$$950 \le \sum_{j=1}^{n} a_j x_j \le 1050, x_j \in \{0,1\}, j = 1,......,160$$

Z is total cost of each excess pullback process
x_j is a binary variable, indicating either to pull excess parts back from this depot or not
j is the index of European region depots which is from 1 till 129.
a_j is the amount of excess parts in local depot j
n is number of local depots in European region
c_j is total cost of local depot j, if excess parts in depot j will be pulled back
$c_j = c_{mj} + c_{hj} + c_{tj}$
c_{mj} is management cost of pulling back the amount of a_j parts in depot j.
c_{hj} is handling cost of pulling back the amount of a_j parts in depot j
c_{tj} is transportation cost pulling back the amount of a_j parts in depot j.

Kellerer (2004) stated the definition of Min-knapsack problem, which is given n pairs of positive integers (c_j, a_j) and a positive integer M, find $x_1, x_2, \ldots, x_n$ so as to

$$Min(z) = \sum_{j=1}^{n} c_j x_j$$

(eq. 4.2)

$$\sum_{j=1}^{n} a_j x_j \ge M, x_j \in \{0,1\}, j = 1,......n$$

It can be seen that eq.4.1 and eq.4.2 are quite similar. The only difference is in eq. 4.1, the knapsack volume is a range, while in eq.4.2 only lower boundary is needed. The above analysis shows that the research problem in this project can be modeled as Min-Knapsack problem.

Despite its importance in practical applications only few methods for solving min-knapsack problem have been proposed in the English literature. Most of results and algorithms are translated from Russian. Because of the scarcity of literature for min-knapsack problem, it is also helpful to study the literatures of max-knapsack problem. The definition of max-knapsack problem is stated as below:

$$Max(z) = \sum_{j=1}^{n} p_j x_j$$

$$\sum_{j=1}^{n} w_j x_j \le C, x_j \in \{0,1\}, j = 1,......n \qquad \text{(eq. 4.3)}$$

Where a subset of n items has to be packed into a knapsack of capacity C, each item j has a profit P_j and a weight W_j, and the problem is to maximize the profit sum of the chosen items without exceeding the capacity C.

The 0-1 max-knapsack problem is NP-hard problem and can be solved in pseudo-polynomial time. There are two main methods for solving knapsack problem. It can be exactly solved by branch and bound algorithm (Martello and Toth, 1977) and dynamic programming algorithm (Toth, 1980). The computational performance of branch and bound algorithm depends largely on the type of data sets considered. If the values of P_j and W_j are independent, the performance is much better than those two values have strong correlation. The storage requirements of dynamic programming algorithm procedures grow deeply with the size of C, so only problem with moderate values of C can be solved in reasonable amount of time.

The number of variables of the problem can be decreased by utilizing a reduction procedure developed by Ingargiola and Korsh (1973). Within this reduction procedure, the value of C can be decreased. When the values of P_j and W_j have very strong correlation, Martello et al (1999) proposed an algorithm which combines dynamic programming with tight bounds called combo.

There are some literatures have proposed solutions on the single-sink fixed-charge transportation problem (SSFCTP) which is the extension problem of min-knapsack problem. The single-sink fixed-charge transportation problem (SSFCTP) is to decide on the amounts x_j>=0 of shipments to be made from a given set of suppliers j=1…n to a single sink in such a way that the suppliers' capacities b_j are respected and the sink's demand D is satisfied at minimum shipment cost. The cost of shipping x_j>0 units from a supplier j to the sink involve a fixed-charge f_j as well as costs $c_j x_j$ that are proportional to the quantity shipped:

$$Min(z) = \sum_{j=1}^{n} (c_j x_j + f_j y_j)$$

$$\sum_{j=1}^{n} x_j = D$$

$$y_j \in \{0,1\}, j = 1,......n$$

$$0 \le x_j \le b_j y_j, j = 1,......n$$

Haberl (1991) proposes enumeration algorithm for SSFCTP problem. Basic enumeration procedure is to sort suppliers according to non-increasing relative cost $e_j=c_j+f_j/b_j$; then enumerate all feasible solutions y_j in lexicographically decreasing manner. Herer (1996) et al. improve previous enumeration algorithm by pegging binary variables y_j to zero and one respectively (computing lower bounds).

Alidaee and Kochenberger (2005) introduced a dynamic programming method to solve SSFCTP problem. The dynamic programming algorithm are described as below

$$G_i(S) = \min \sum_{j=1}^{n} (c_j x_j + f_j y_j)$$

$$\sum_{j=i}^{n} x_j = D - S$$

$$y_j \in \{0,1\}, j = 1,\ldots\ldots n$$

$$0 \le x_j \le b_j y_j, j = 1,\ldots\ldots n$$

$$G_1(0) = Z; x_i^{\max} = \min\{b_i, D - S\}$$

$$G_i(S) = \min\{G_{i+1}(S), \min\{f_i + c_i x_i + G_{i+1}(S + x_i) : x_i = 1,\ldots., x_i^{\max}\}\}$$

The above algorithm can be solved within O(nDmax$_j$b$_j$).

4.3 Excess pullback model

4.3.1 General excess pullback procedure plan

As Stated before, Cisco started its excess pullback plan from week 49 of 2007, and the excess stock data is available from this week too. So starting time of proposed plan is the same as Cisco's old plan.

In section 3.1, it is shown that, at the beginning of week 49 of 2007, the total sum of excess inventory parts (SKU) in EEA countries is 9316, around 17% of total inventory. The Cisco's target is at most 10%, because of the large gap between target excess stock level and current excess stock level, it is better to pull back as many parts (1000±5%) as possible per week at the beginning (handling capacity in central warehouse).

Within each excess pullback procedure, drawing out an optimal solution by minimizing involved total cost which includes handling cost in local depots, transportation cost, and management cost. All three cost components are related to involved local depots within each excess pullback. The handling cost is related to depot vendor (3PL2 or 3PL1). The transportation fee is charged on number of delivered pallets. The less number of delivered pallets, the less transportation cost is charged. Minimizing pallet cost is to load as many parts on one pallet as possible. Currently, block pallet is used for excess pullback which is 120cm*100cm*220cm. The safety volume is based on the analysis of historical data which is from week 49 of 2007 until now. The last component of total cost is management cost. The forth assumption in section 4.1 stated how the management cost is calculated.

After several weeks of excess pullback procedure, if the total amount of excess parts reaches the target level of 10% of total inventory, then the excess pullback procedure is stopped temporally. The total inventory amount in European region is updated every fortnight. The amount of excess inventory can be reviewed by each week, whether the total excess parts are above 10% level of total inventory amount, if it is, starting new excess pullback procedure again. Otherwise, no pullback procedure will be executed.

4.3.2 Min-Knapsack Model

As stated in section 4.2, the research problem of this project can be modeled as min-knapsack problem, which is:

$$Min(z) = \sum_{j=1}^{n} c_j x_j$$

$$950 \leq \sum_{j=1}^{n} a_j x_j \leq 1050, x_j \in \{0,1\}, j = 1, \ldots\ldots n \qquad \text{(eq. 4.1)}$$

Z is total cost of each excess pullback process
x_j is a binary variable, indicating either to pull excess parts back from this depot or not
j is the index of European region depots which is from 1 till 129.
a_j is the amount of excess parts in local depot j
n is number of local depots in European region
c_j is total cost of local depot j, if excess parts in depot j will be pulled back
$c_j = c_{mj} + c_{hj} + c_{tj}$
c_{mj} is management cost of pulling back the amount of a_j parts in depot j.
c_{hj} is handling cost of pulling back the amount of a_j parts in depot j
c_{tj} is transportation cost pulling back the amount of a_j parts in depot j.

As stated before, C_{mj} is management cost of depot vendor. The total cost is calculated separately by depot vendor (3PL1 and 3PL2). Because the management cost of 3PL1 is same as 3PL2 in each excess pullback process, C_{mj} is not considered in the model. C_{hj} is different from vendor to vendor. The average handling cost for picking a part at 3PL2 is 2.3 euro, for 3PL1 it is 2.1 euro. Transportation price C_{tj} is charged based on number of pallets delivered.

On the basis of historical data (week51 of 2007, week 4, 8, 9, 10, 11 of 2008), the average loading capacity for each pallet is 24 parts. This value is the average loading capacity of above 6 weeks in all involved local depots. After getting the amount of excess parts in depot j (a_j), the number of delivered pallets of depot j can be obtained by $a_j/24$, and round up to the nearest integer. Then getting the prices of transportation, which according to the number of delivered pallets of depot j by tariff information from Cisco. For each depot j, C_j and a_j can be known at first.

4.4 Algorithms

Csirik, etc (1991) proposed some very useful algorithms, which is used for solving min-knapsack problem. Three algorithms refer to algorithm GR, GR+, and IGR. Algorithm GR is basic algorithm, while Algorithm GR+ and Algorithm IGR are developed on the basis of Algorithm GR. Three Algorithms are described in detail in following sub sections (GR stands for Greedy).

4.4.1 Algorithm GR

The basic algorithm is called Algorithm GR.

1. Calculating the c_j/a_j value of each depot j, a_j is the amount of excess parts in local depot j, c_j is total cost of local depot j, so c_j/a_j is relative cost of pulling back each part.

2. Sorting and indexing c_j/a_j with nondecreasing order, which means $c_1/a_1 <= c_2/a_2 ...... <= c_j/a_j ...... <= c_n/a_n$.

3. Summing up a_j, let k_1 be the first index for which $\sum_{i=1}^{k_1} a_i < 950 \le \sum_{i=1}^{k_1+1} a_i$, then define the set $S_1 = \{a_1, a_2, ...a_{k_1}\}$, so first candidate solution for min-knapsack problem can be written as $S_1 \cup \{a_{k_1+1}\}$.

4. If $950 \le \sum_{i=1}^{k_1} a_i + a_{k_1+2}$ holds, $S_1 \cup \{a_{k_1+2}\}$ is another candidate solution for min-Knapsack problem. If it is possible, Let K_1+3, $K_1+4 \ldots K_2-1$, be a series of indices so that all of the corresponding items the following holds: $950 \le \sum_{i=1}^{k_1} a_i + a_j \quad a_j \in B_1 = \{a_{k_1+3}, a_{k_1+4} ...., a_{k_2-1}\}$ which means $S_1 \cup \{a_{k_1+3}\}$, $S_1 \cup \{a_{k_1+4}\} \ldots S_1 \cup \{a_{k_2-1}\}$ are all candidate solutions for min-knapsack problem.

5. Let K_2 be the first next index for which $\sum_{i=1}^{k_1} a_i + a_{k_2} \le 950$, and $\sum_{i=1}^{k_1} a_i + a_{k_2+1} \le 950$, until $k_3 >= k_2$ be the first index for which $\sum_{i=1}^{k_1} a_i + \sum_{i=k_2}^{k_3} a_i \le 950 \le \sum_{i=1}^{k_1} a_i + \sum_{i=k_2}^{k_3+1} a_i$, Let's get one set which is $S_2 = \{a_{k_2}, a_{k_2+1} ...., a_{k_3}\}$, So, $S_1 \cup S_2 \cup \{a_{k_3+1}\}$ is also one of candidate solutions. Repeat above indexing procedure until the end of list. So the whole list is divided into several sets S and B.

$$\overbrace{a_1, a_2 ...., a_{k_1}}^{S_1} \overbrace{a_{k_1+1}, a_{k_1+2} ...., a_{k_2-1}}^{B_1} \overbrace{a_{k_2}, a_{k_2+1} ...., a_{k_3}}^{S_2} \overbrace{a_{k_3+1}, a_{k_3+2} ...., a_{k_4-1}}^{B_2} ....,$$

$$\overbrace{a_m, a_{m+1} ...., a_n}^{S_m}$$

The last set is possibly empty. The items in S sets are called small items, because sum of the items in each S set is smaller than 950, while in B sets are called big items. From above analysis, the candidate solutions are $S_1 \cup a_j, a_j \in B_1$; $S_1 \cup S_2 \cup a_j, a_j \in B_2$; $S_1 \cup S_2 \cup S_3 \cup a_j, a_j \in B_3$ and so on.

4.4.2 Algorithm GR+

Algorithm GR+ is an improved version of Algorithm GR, which is described as follows:

1. If a_1, a_2……, a_k, is a candidate solution for GR (a_1, a_2…… a_{k-1} are small items which belong to one of S sets and a_k is a big item which belong to one of B sets). We start to delete the last small item in S set: a_{k-1}.

2. After first deletion step, if $\sum_{i=1}^{k-2} a_i + a_k \geq 950$, we can continue to delete a_{k-2}, if $\sum_{i=1}^{k-3} a_i + a_k \geq 950$, we keep on deleting a_{k-3}. Hence we could continue deleting items until a_l, when $\sum_{i=1}^{l-1} a_i + a_k \leq 950$, after finishing above procedures, the candidate solution {a_1, a_2……, a_k} becomes {a_1, a_2… a_l, a_k}.

3. All candidate solutions can be improved by same deletion procedure as stated above.

4. After all improvements, all candidate solutions should be checked if their sum is more than 1050, because in this project, the total amount of pullback parts could not be more than 1050. For example, after above improvement, we get candidate solution {a_1, a_2… a_l, a_k}, it should be checked if $\sum_{i=1}^{l} a_i + a_k \geq 1050$, if it is, abandon this solution, otherwise, leave it to candidate solution sets.

The candidate solutions which are given by GR+ Algorithm are the minimum cost candidates. After getting all candidate solutions, calculating all solution on the basis of total cost, and gets the optimal one which has lowest total cost.

4.4.3 Algorithm IGR

There is another way of improvement on the basis of algorithm GR, which is called algorithm IGR. The procedure of this algorithm is stated as follows:

1. Let L = {a_1, a_2…… a_n}, a_n is the amount of excess parts in local depot n.

2. After sorting and indexing, one knapsack problem arises. It is assumed that the number of big items in L set is i. Set L_i=L/{a_i}, which means remove one big item from L set, and

becoming new set L_i, a_i is one of these big items. There become several new sets and the amount of new sets is i.

3. M_i=M-a_i, M is capacity of knapsack, in this project, M=950. This equation means the capacity of each new knapsack problem is equal to the original knapsack capacity minus the volume of each removed big item.

4. After above procedures, the original one knapsack problem becomes to i new knapsack problems. All these i knapsack problems can be solved by above GR+ algorithm.

5. The total cost for each candidate solution is defined as IGR_i. IGR_i is equal to total cost of L_i plus C_i. Total cost of L_i (new knapsack problem) is calculated by GR+ algorithm and C_i is cost of removed big item a_i. After getting all cost of IGR_i, the minimal cost of IGR algorithm can be easily drawn as Min (IGR_i).

5. Results

5.1 Selection of Programming Language

In order to compute the algorithms of section 4, a programming language is needed. C++ is chosen as programming language because:

1. C++ is an object oriented and very powerful programming language. Our three algorithms can be realized by using class objects in C++ and it can save large amount of codes.

2. C++ supports the point operations, which make our algorithm more efficient.

3. C++ supports structure operations. In our project, each depot can be encapsulated as one structure, which makes the structure of data clearer.

4. I have experience on C++ programming language

All *.xls files should be converted to *. csv files before using as input file of program, the reasons are as below:

1. The structure of *.csv files is very simple, which is almost same as text file's.

2. It is very easy to convert *.xls file to *.csv file, and file size is less than *.xls.

3. Because small size of *.csv file, it is easy to store and transfer via internet.

The three algorithms in section 4 are coded in C++ programming language. For the use and further development of proposed plan, Cisco needs to buy Microsoft Visual Studio software, which costs approximately €950.-. The detail source code can be found in Appendix II. The source code is only needed when the model needs to be changed critically. For example, the strategy of plan has changed. The people who use this program don not need to understand what these codes means. They only need to know how to use it. The program has its friendly interface and it is easy to use.

5.2 Proposed program

The proposed program is discussed in this section. First of all, the interface of program is showed in figure 5.1 and figure 5.2. From figure 5.1, it can be seen that there are two menu buttons in the program, which are "File" and "Process". Under "File" menu, there are five sub-menus, which are "Read Pallet Price File", "ReadFile", "ReadIslandof 3PL2File", "SaveFile", and "Exit" separately. First three sub-menus are set for reading input files. The last two sub-menus are for saving file and exiting the program. There are two sub-menus under "Process" menu button. First button is "Result", which is giving the best solution of three algorithms. Second button is “amount range”, which is setting amount range of weekly excess pullback.

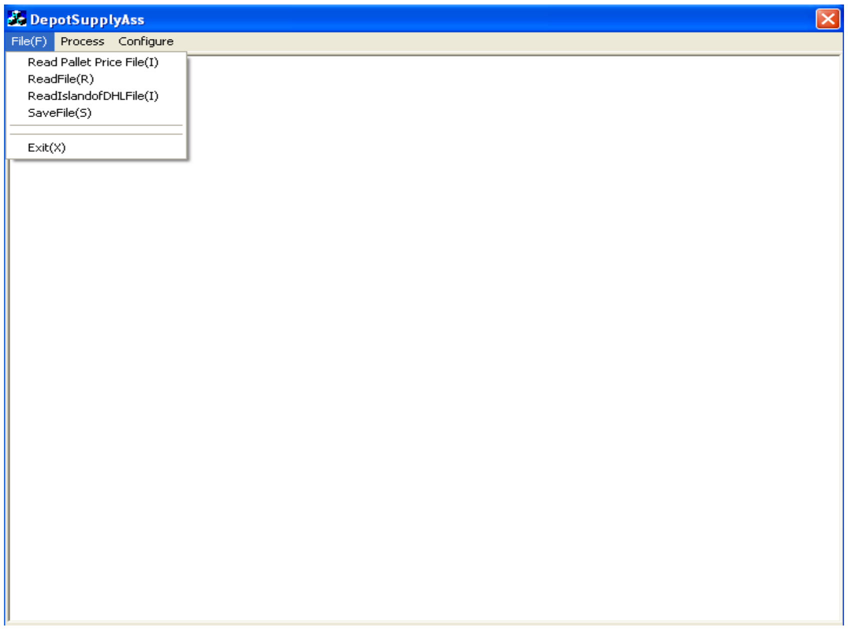


Figure 5.1 The interface of proposed program

After getting the interface of program, the procedure of using this program is described as follows:

1. Before operating the proposed program, all related input files should be changed from excels files to *. Csv files.
2. As show in the appearance of program, the file of pallet price for each depot should be loaded at first.
3. Second, the file which contains information of the number of excess parts in each depot should be loaded. If first file is from 3PL2 depots, the file contains information of five Italian depots also should be loaded next. If first file is from 3PL1 depots, no island depot file needed.
4. Third, set amount range of weekly excess pullback. Normally, it is 950-1050.
5. Forth, get the optimal solution by clicking the “Calculate” sub-button.
6. Fifth, saving result of each algorithm by clicking “SaveFile” button.

The detail user guidance of proposed program is described in one handout file, in which you can find:

How to run the proposed program
How to read the message of proposed program
How to interpret the outcome of proposed program

5.3 Results

In this section, the historical data of excess pullback procedure is test. And the results of proposed plan are listed. The basic procedure is as follows:

1. Loading the file which contains the information of delivering pallet price for each depot in European Region.

2. Loading the file which contains the information of quantities of excess stock in each depot in European Region (3PL2 depots or 3PL1 depots separately). If the depots are from 3PL2, the input files should also include one file, which contains the excess stock information of 5 Italian island depots.

3. Click "calculate", getting the best result and save it.

5.3.1 Result of proposed plan

Due to lack of historical data, only six sets of weekly data have been tested by proposed excess pullback plan. During each week test, the target amount of excess pullback stock is set within certain range. The minimal quantity of pullback stock is same as actual amount of weekly pullback stock. For example, in week 51 of 2007, the amount of pullback excess stock is 724. Then the minimal target amount of pullback excess stock in proposed plan is 724, the maximal amount is 800 (randomly but close to minimal amount).

The results of week 51 of 2007, week 4, 8, 9, 10, and 11 of 2008 are shown in tables 5.1-5.12. All detail procedures of running proposed program are listed in Appendix III.

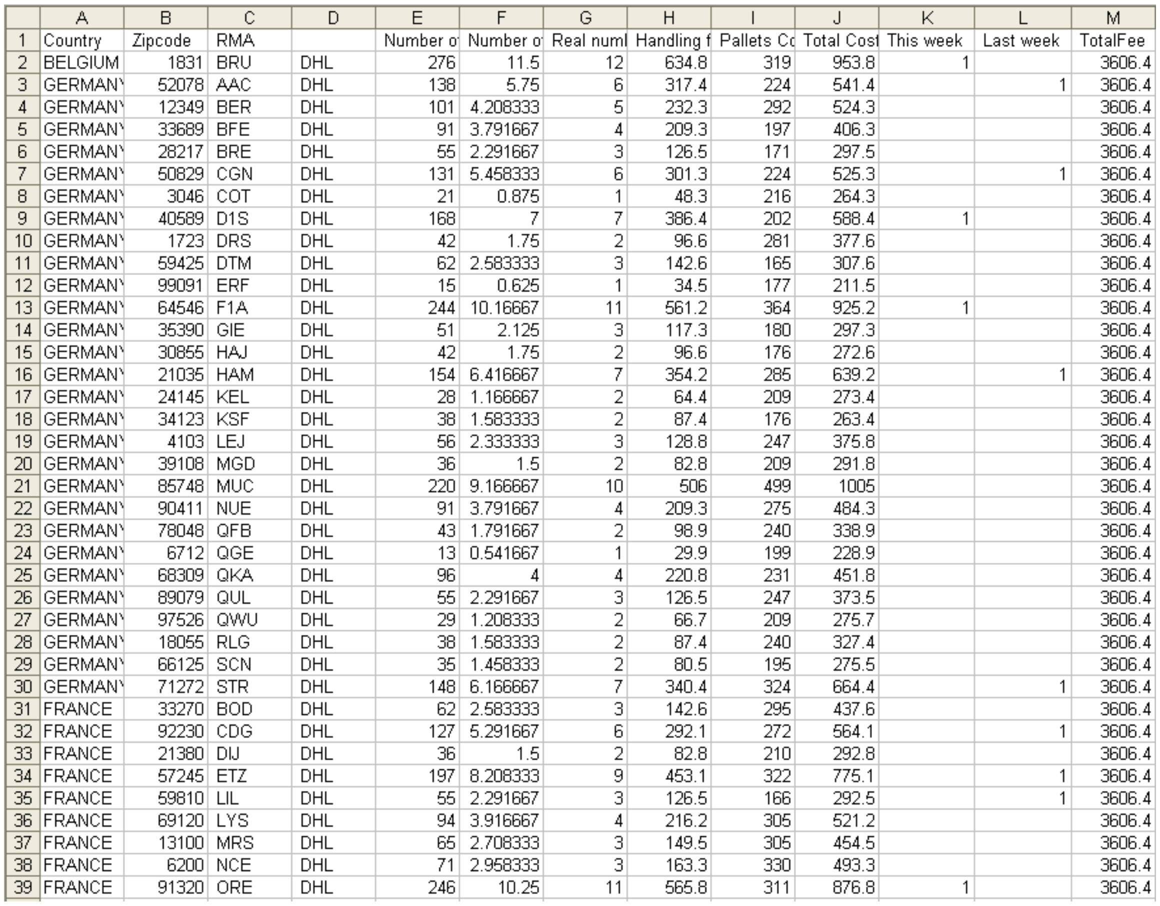
Time	Amount of pullback parts	Total Cost (€)	Involved depots	Depot vendor
Week 51	748	2772.4	BRU,CGN,F1A,ORE	3PL2
Week 4	545	2020.5	BRU, AAC, CCN	3PL2
Week 8	915	3313.5	BRU, D1S, F1A, ORE	3PL2
Week 9	969	3743.7	BRU, HAM, ORE, CDG, ETZ	3PL2
Week 10	1053	4406.9	BRU, DRS, CDG, ETZ, LIL, ORE, RNS	3PL2
Week 11	1003	5020	LGW, LHR, MAN, MHZ, AMS EIN, GRQ, QYP, RTM, GOT	3PL1

Table 5.1 Performance of proposed plan

From the above table, in row of week 51, it can be seen that, the proposed plan suggests to pull quantity of 748 of excess stock from 4 3PL2 depots, which are BRU from Belgium, CGN from Germany, F1A from Germany, ORE from France, the total cost of excess pullback procedure is 2772.4 Euros. The results of other rows can be interpreted by same way.

5.3.2 Using program in consecutive week

In section 5.4.1, the proposed program shows that it can get its result based on the excess stock information of singe week. Actually, this program can also be used in plan for consecutive weeks, where the quantity of excess parts in each depot is constant during these weeks. The first step of running program is same as which has been shown in section 5.4.1. After getting program result of first week, continue to click three algorithm sub-buttons again, and the result of this week can be drawn. The result excel file can be shown as below:



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Country	Zipcode	RMA		Number o	Number o	Real numl	Handling f	Pallets Cc	Total Cost	This week	Last week	TotalFee
2	BELGIUM	1831	BRU	DHL	276	11.5	12	634.8	319	953.8	1		3606.4
3	GERMAN	52078	AAC	DHL	138	5.75	6	317.4	224	541.4		1	3606.4
4	GERMAN	12349	BER	DHL	101	4.208333	5	232.3	292	524.3			3606.4
5	GERMAN	33689	BFE	DHL	91	3.791667	4	209.3	197	406.3			3606.4
6	GERMAN	28217	BRE	DHL	55	2.291667	3	126.5	171	297.5			3606.4
7	GERMAN	50829	CGN	DHL	131	5.458333	6	301.3	224	525.3		1	3606.4
8	GERMAN	3046	COT	DHL	21	0.875	1	48.3	216	264.3			3606.4
9	GERMAN	40589	D1S	DHL	168	7	7	386.4	202	588.4	1		3606.4
10	GERMAN	1723	DRS	DHL	42	1.75	2	96.6	281	377.6			3606.4
11	GERMAN	59425	DTM	DHL	62	2.583333	3	142.6	165	307.6			3606.4
12	GERMAN	99091	ERF	DHL	15	0.625	1	34.5	177	211.5			3606.4
13	GERMAN	64546	F1A	DHL	244	10.16667	11	561.2	364	925.2	1		3606.4
14	GERMAN	35390	GIE	DHL	51	2.125	3	117.3	180	297.3			3606.4
15	GERMAN	30855	HAJ	DHL	42	1.75	2	96.6	176	272.6			3606.4
16	GERMAN	21035	HAM	DHL	154	6.416667	7	354.2	285	639.2		1	3606.4
17	GERMAN	24145	KEL	DHL	28	1.166667	2	64.4	209	273.4			3606.4
18	GERMAN	34123	KSF	DHL	38	1.583333	2	87.4	176	263.4			3606.4
19	GERMAN	4103	LEJ	DHL	56	2.333333	3	128.8	247	375.8			3606.4
20	GERMAN	39108	MGD	DHL	36	1.5	2	82.8	209	291.8			3606.4
21	GERMAN	85748	MUC	DHL	220	9.166667	10	506	499	1005			3606.4
22	GERMAN	90411	NUE	DHL	91	3.791667	4	209.3	275	484.3			3606.4
23	GERMAN	78048	QFB	DHL	43	1.791667	2	98.9	240	338.9			3606.4
24	GERMAN	6712	QGE	DHL	13	0.541667	1	29.9	199	228.9			3606.4
25	GERMAN	68309	QKA	DHL	96	4	4	220.8	231	451.8			3606.4
26	GERMAN	89079	QUL	DHL	55	2.291667	3	126.5	247	373.5			3606.4
27	GERMAN	97526	QWU	DHL	29	1.208333	2	66.7	209	275.7			3606.4
28	GERMAN	18055	RLG	DHL	38	1.583333	2	87.4	240	327.4			3606.4
29	GERMAN	66125	SCN	DHL	35	1.458333	2	80.5	195	275.5			3606.4
30	GERMAN	71272	STR	DHL	148	6.166667	7	340.4	324	664.4		1	3606.4
31	FRANCE	33270	BOD	DHL	62	2.583333	3	142.6	295	437.6			3606.4
32	FRANCE	92230	CDG	DHL	127	5.291667	6	292.1	272	564.1		1	3606.4
33	FRANCE	21380	DIJ	DHL	36	1.5	2	82.8	210	292.8			3606.4
34	FRANCE	57245	ETZ	DHL	197	8.208333	9	453.1	322	775.1		1	3606.4
35	FRANCE	59810	LIL	DHL	55	2.291667	3	126.5	166	292.5		1	3606.4
36	FRANCE	69120	LYS	DHL	94	3.916667	4	216.2	305	521.2			3606.4
37	FRANCE	13100	MRS	DHL	65	2.708333	3	149.5	305	454.5			3606.4
38	FRANCE	6200	NCE	DHL	71	2.958333	3	163.3	330	493.3			3606.4
39	FRANCE	91320	ORE	DHL	246	10.25	11	565.8	311	876.8	1		3606.4

Figure 5.3 The results of consecutive weeks

From above figure, it can be seen that, column K, and L are shown the result of this week and last week separately. The figure shows that, last week the excess pullback plan is executed in depots: AAC, CGN, HAM, STR, CDG, ETZ, and LIL. The excess stock in depots BRU, D1S, F1A, and ORE are pulled back to central warehouse (NL1), and the total cost of this week is 3606.4 euros.

5.4 Comparison

In section 3.4, the performance of current excess pullback procedure is shown, while the performance of proposed excess pullback plan is listed in section 5.4. Both results are drawn on the basis of same sets of historical data, which makes the comparison results much more straightforward.

In this section, two ways of comparison are listed below. First, two plans (original plan and proposed plan) pull back approximate same amount of excess parts (proposed plan pulls no less excess stock than original plan), then comparing total cost of two plans. This has been done in two sections. Second, setting the quantity of pullback stock of proposed plan same to Cisco's target quantity (950-1050), and getting the value of cost per part; then comparing this value to original plan.

■ *Week 51 of 2007*

If optimal plan pulls back approximate same amount of excess parts as original plan. The comparison is as follows:

	Amount of pullback parts	Involved local depots	Total Cost (€)	Amount Difference	Cost Difference(€)	Cost Saving Percentage
Original Plan	724	GLC, LDS, LHR, LTN, MAN	3897.4	+24	-1125	28.8%
Proposed Plan	748	BRU, CGN, F1A, ORE	2772.4			

Table 5.2 Comparison result for week 51 of 2007

If optimal solution pulls back with target amount of excess parts (results in Appendix IV), the comparison is as follows:

	Amount of pullback parts	Total Cost (€)	Cost/Part	Cost Saving Percentage
Original Plan	724	3897.4	5.38	30.5%
Proposed Plan	960	3587	3.74	

Table 5.3 Comparison result for week 51 of 2007

■ *Week 4 of 2008*

	Amount of pullback parts	Involved local depots	Total Cost (€)	Amount Difference	Cost Difference(€)	Cost Saving Percentage
Original Plan	537	SWS, WRX, XVH, BHX, CWL, EMA	3295.7	+8	-1275.2	38.7%
Proposed Plan	545	BRU, CGN, AAC	2020.5			

Table 5.4 Comparison result for week 4 of 2008

If optimal solution pulls back with target amount of excess parts (results in Appendix V), the comparison is as follows:

	Amount of pullback parts	Total Cost (€)	Cost/Part	Cost Saving Percentage
Original Plan	537	3295.7	6.14	38.8%
Proposed Plan	958	3606	3.76	

Table 5.5 Comparison result for week 4 of 2008

■ ***Week 8 of 2008***

	Amount of pullback parts	Involved local depots	Total Cost (€)	Amount Difference	Cost Difference (€)	Cost Saving Percentage
Original Plan	900	AAC, BER, BFE, BRE, CGN, D1S, DTM, F1A, GIE	4305	+15	-991.5	23%
Proposed Plan	915	BRE, D1S, F1A, ORE	3313.5			

Table 5.6 Comparison result for week 8 of 2008

If optimal solution pulls back with target amount of excess parts (results in Appendix V), the comparison is as follows:

	Amount of pullback parts	Total Cost (€)	Cost/Part	Cost Saving Percentage
Original Plan	900	4305	4.78	21.8%
Proposed Plan	953	3566	3.74	

Table 5.7 Comparison result for week 8 of 2008

■ ***Week 9 of 2008***

	Amount of pullback parts	Involved local depots	Total Cost (€)	Amount Difference	Cost Difference(€)	Cost Saving Percentage
Original Plan	946	HAJ, HAM, LEJ, MUC, NUE, QFB, QKA, QOL, RLG, STR	4942.8	+23	-1199.1	24.3%
Proposed Plan	969	BRU, HAM, ORE, CDG, ETZ	3743.7			

Table 5.8 Comparison result for week 9 of 2008

If optimal solution pulls back with target amount of excess parts (results in Appendix V), the comparison is as follows:

	Amount of pullback parts	Total Cost (€)	Cost/Part	Cost Saving Percentage
Original Plan	946	4942.8	5.23	26.2%
Proposed Plan	969	3743	3.86	

Table 5.9 Comparison result for week 9 of 2008

■ *Week 10 of 2008*

	Amount of pullback parts	Involved local depots	Total Cost (€)	Amount Difference	Cost Difference(€)	Cost Saving Percentage
Original Plan	1049	BOD, CDG, ETZ, LIL, LYS, MRS, NCE, ORE, RNS, TLS	5671.7	+4	-1261.2	22.2%
Proposed Plan	1053	BRU, DRS, CDG, ETZ, LIL, ORE, RNS	4406.9			

Table 5.10 Comparison result for week 10 of 2008

If optimal solution pulls back with target amount of excess parts (results in Appendix V), the comparison is as follows:

	Amount of	Total Cost (€)	Cost/Part	Cost Saving Percentage

	pullback parts			
Original Plan	1049	5671.7	5.41	25.3%
Proposed Plan	954	3855.2	4.04	

Table 5.11 Comparison result for week 10 of 2008

■ ***Week 11 of 2008***

	Amount of pullback parts	Involved local depots	Total Cost (€)	Amount Difference	Cost Difference(€)	Cost Saving Percentage
Original Plan	1003	AOI, BLQ, FCO, FLR, MXP, T1N, VCE	7153.9	+0	-2133	29.8%
Proposed Plan	1003	LGW, LHR, MAN, MHZ, AMS, EIN, GRQ, QUYP, RTM, GOT	5020.9			

Table 5.12 Comparison result for week 11 of 2008

If optimal solution pulls back with target amount of excess parts (results in Appendix V), the comparison is as follows:

	Amount of pullback parts	Total Cost (€)	Cost/Part	Cost Saving Percentage
Original Plan	1003	7153.9	7.13	29.5%
Proposed Plan	950	4774	5.03	

Table 5.13 Comparison result for week 11 of 2008

The simulation results of proposed plan on the basis of historical data have been shown. Meanwhile, the comparison between two plans (proposed plan and original plan) has been drawn. From the comparison result, it seems that the proposed plan has improved significantly, both on total involved cost and cost per pullback part.

5.5 Verification and Validation

The excess pullback model and three algorithms have been incorporated in a program in the programming language C++. The model has been verified and validated, in order to make sure the proposed solution process is correct.

Verification deals with determining whether the model assumptions have been correctly translated into computer program, the process of checking if a program does what it is intended to do. Kelton et al. (2002) present a number of techniques that can be used for verifying a model. The following methods have been applied in this project: First, the error messages that appeared when running the programs were tackled, i.e. the programs were debugged generally. Second, the code of programming language has been checked carefully and several times, no bad descriptions were found.

Validation has been done after verification; it is a much boarder concept than verification. Validation is defined as “substantiation that a computerized model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model” (Sargent 2005). It consists of conceptual model validation, data validity and operation validation.

Data validity is defined as ensuring that data necessary for building, evaluation and testing of the model are adequate and correct. This is done by obtaining all parameters directly from transportation management of K&N, and it has been checked by Cisco system, which ensures the validity of used data.

Conceptual model validity is defined as determining that the theories and assumptions underlying the conceptual model are correct and that the model representation of the problem entity is reasonable for the intended purpose. Therefore all assumptions have been listed and discussed with Cisco system management before, during and after development of the model. Conceptual validity is obtained by sensitivity analysis with respect to certain input parameters, this shows how the model behaves for various parameter settings and continuity, consistency, and degeneracy of model.

Last not least validation is operational validation, which is defined as determining that the models output behavior has sufficient accuracy for the models intended purpose over the domain and intended applicability. In all model tests, subjective tests are usually the most convincing. The operational validation is obtained by comparing the output of the proposed model plan to the output of the original plan.

5.5.1 Sensitivity Analysis

In this section, the sensitivity analysis of several input parameters on the total involved cost of proposed plan is provided. Two most interesting factors are considered in this section: amount of excess parts on per delivered pallet and the amount range of weekly excess pullback excess parts.

In proposed plan, the value of amount of parts on per pallet is 24. In figure 5.4, the overview of total cost based on different values is provided. The values 20, 24, 28, 32, and 36 are chosen and week 51 of 2007 is chosen as sample week. It can be concluded that the impact of amount of parts per pallet is significant. The total cost is decreasing significantly with increase of amount of parts per pallet. This is not a surprise, because more parts be loaded on one pallet, less pallets are delivered, and then less transportation cost is charged.

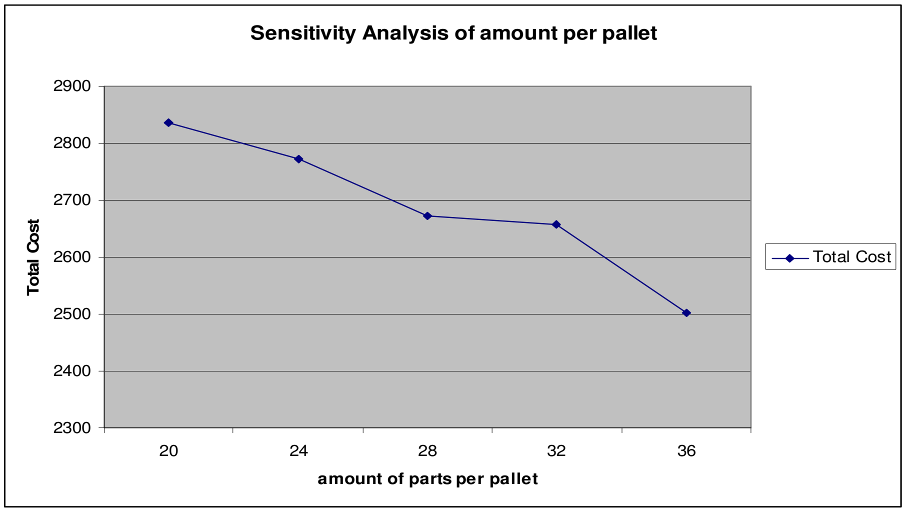


Figure 5.4 Sensitivity analysis of excess parts amount per pallet

The amount range of weekly excess pullback parts is another interesting factor. The same as first analysis, week 51 of 2007 is selected as sample week. The values of amount range are 650-750, 750-850, 850-950, 950-1050, and 1050-1150. Table 5.14 shows all results. In figure 5.5, it can be seen that the amount range of weekly excess pullback parts is also a very significant factor on total cost.

Amount Range	Actual Pullback Amount	Total Cost (euro)
650-750	673	2456.9
750-850	751	2889.3
850-950	851	3150.3
950-1050	969	3611.7
1050-1150	1091	4036.3

Table 5.14 Amount range, actual amount, and total cost

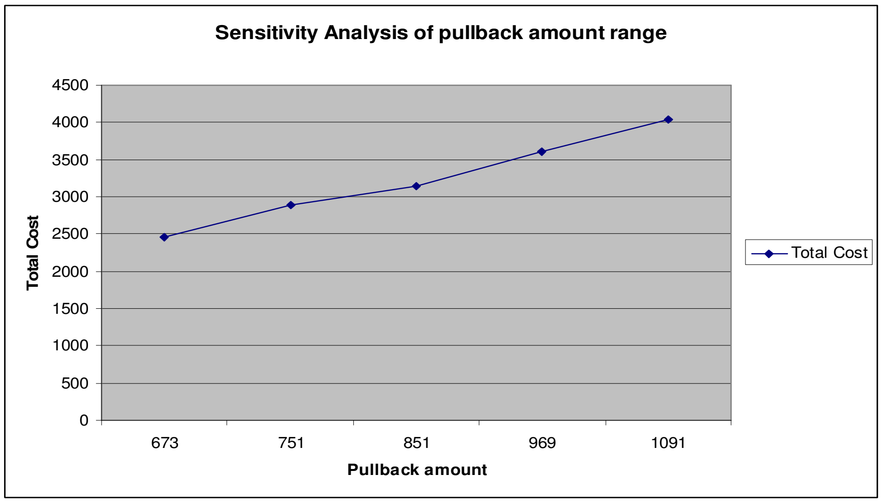


Figure 5.4 Sensitivity analysis of excess pullback amount range

The above analysis shows the significance of two factors in the proposed model: amount of excess parts on per delivered pallet and the amount range of weekly excess pullback excess parts. Both factors are very important in deciding total cost of excess pullback procedure. The first factor (amount of parts on per pallet) is the one should be paid attention. If Cisco can improve the loading capacity of each pallet, the total cost can be decreased significantly. There is another obvious factor for reducing total cost, which is transportation tariff of each delivered pallet.

5.5.2 Model solution compared to output of original plan

In order to obtain operational validation, the outputs of two plans are compared. In table 5.1, the proposed plan shows the weekly selected excess pullback depots and total cost. If the original plan chooses the same depots, and gets similar total cost results, the model structure is operationally validated. Because all model assumptions have been validated, if total cost results are similar, then the model is operationally validated. The results are shown in table 5.15. It can be seen that the output of original plan is quite similar to proposed model solution.

	Original Plan Cost (euro)	Proposed Plan Cost (euro)	Similarity
Week 51	2716.9	2772.4	97.9%
Week 4	2000.3	2020.5	99%
Week 8	3446.5	3313.5	96.2%
Week 9	3874.8	3743.7	96.7%
Week 10	4265.3	4406.9	97%
Week 11	4723.9	5020	94.1%

Table 5.15 The similarity between outputs of two plans

6. Implementation plan

The final stage for this project is to implement proposed program to current excess pullback procedure. The most important for implementing the proposed model in current procedure is commitment and willingness to change. The commitment and willingness are prerequisites for success.

In order to reach above situation, it is important to show Cisco System how the new plan can improve the present situation and what the positive impacts are for Cisco System with respect to better service level and lower total cost involved in excess pullback procedure. From the results of section 5, it can be seen that, the average percentage of improvement on total cost is 27.8%, which results better service level. It is also important that Cisco System check all the results again and test the proposed model for certain period.

Though this model is quite different from what was used before, not much will change for excess pullback planners, basic operations are the same. In current plan, Cisco planner only needs to know the amount of excess inventory in each local depot before starting the excess pullback operation. In proposed plan, having information of transportation price is another prerequisite to excess pullback operation. Besides that, the planner needs to pay special attention to five Italian island depots, in proposed plan these five depots are treated as one. Planner needs to pick up information of these five depots, and make it as one excel file. Planner can start to operate proposed software after getting all required information. The proposed tool has very friendly interface and it is easy to operate even if the planner has no knowledge of programming language.

The procedures of implementation plan are described as follows:

1. After verifying all input data (it has been done in validation section), compare the results of proposed model to current excess pullback procedure. It is also important for Cisco to become familiar with the outcome of this program, how to interpret it, and how to change the parameters of program to fulfill the specific needs and advanced requirements.

3. The second phase is to test the proposed model in certain period (three months). It is advised to use same parameters as previous months, and understand the difference between these two plans and how proposed plan improves current excess pullback procedure.

4. After the successful test period, the excess pullback procedure in European region should be operated under the guidance of proposed model.

Not only technical implementation is needed, but also organizational issues, such as setting responsibilities, have to be considered. On the basis of results comparison in section 5, it is believed that the success of implementation will be realized. But people are always the keys for all work. Thereby it is important to note that whether implementation is successful or not, highly depends to a high extent on the support for the new excess pullback procedure in Cisco System. Therefore one person should be formally responsible for successful implementation. He is mainly involved in above four steps of implementation plan and responsible for gathering all required information for proposed tool. Cisco system outsources its logistics operation to Kuehne+Nagel, which includes excess pullback operation. This planner should have good connection with them, because all relevant data are got from Kuehne+Nagel. The planner should be familiar with excess pullback operations and Cisco European local depots.

In order to use proposed tool, it is required to check the excess inventory level in European region weekly. If the excess inventory is more than 10% of total inventory level, the planner needs to run the proposed tool to draw an optimal excess pullback plan. Before that, the planner needs to gather all information which is required by proposed tool. It can be seen that, getting correct and accurate information is most important during proposed plan.

For operation of proposed plan, Cisco needs to buy Microsoft Visual Studio software, which costs approximately €950.-. Before developing this tool further Cisco should carefully think why it needs to develop this tool further and set clear goals for development. The planner also needs to understand the tool and its parameters, but does not need to understand the programming language C++. In order to develop advance requirement in the future, Cisco needs an expert who is specialized in programming language C++. Other cost in proposed plan is mostly management cost. In proposed plan, involved management cost is more than current plan, because of more coordination between Cisco and Kuehne+Nagel; between Cisco and depot vendors (3PL2 and 3PL1); between local depots. Cisco needs to evaluate the benefits of proposed plan and involved costs.

All the inputs and assumptions have certain possibilities to change during the implementation plan. In order to show the robustness of proposed plan in mid-term, some solutions for these changes are given below.

1. It has high possibility that both transportation tariff of local depots and the amount of local depots in European region will change in the near future. The model can be easily adjusted based above two changes. The tariff information and amount of local depots can be adjusted in the file of “pallet price for each depot”. The new depots can be added to the file by same regulation and structure as existed depots. The tariff information can be adjusted by changing the values in the cells.

2. It is also possible that the handling capacity of central warehouse increases sooner or later. If Cisco improves the loading capacity of each delivered pallet, the model needs to be adjusted too. These two changes can be made by clicking sub-button “totalsupply” of button “configure”. The only thing needs to do is change the values in the dialogue box.

7. Conclusion and Future Research

7.1 Conclusion

In this report, this following research question has been analyzed

> Develop model of optimal transportation policy on the basis of total cost consideration in excess pullback procedure, which stating when shipments must take place in which depot for each shipment, in such a way that the sum of transportation cost, handling cost and management are minimized.

After analyzing original transportation plan, several improvements are recommended in section 3. In section 4, based on suggested transportation model, some algorithms are proposed to solve the model and try to get optimal solution. In section 5, the results of proposed model are drawn by programming three suggested algorithms. It has been shown that the total cost of excess pullback procedure is improved significantly by proposed plan. The average percentage of improvement on total cost is 27.8%.

Three conclusions can be drawn as follows:

1. The original plan has been improved significantly by proposed plan. The original plan was mainly drawn based on amount of excess parts in each depot; the depots which have most excess spare parts are selected as pullback depots. This plan is very rough and imprecise. The original plan has a lot of space to be improved. The proposed plan is modeled on basis of min-knapsack problem, and the transportation problem is solved by algorithm, which makes transportation plan more efficient and less cost.

2. The result of comparison between original plan and proposed plan shows the significant improvement of proposed plan, both on total involved cost and cost per pullback part. The saving percentages of two criteria are 27.8% and 28.7% respectively.

3. The outcome of proposed plan is not only reducing cost and but also improving service quality. In proposed plan, the value of cost per part has significantly decreased as total cost, which also means Cisco System can pull back much more excess parts with same amount of total cost as before. Under this circumstance, Cisco System can fulfill more demands from different customers, which means the improvement on service quality.

7.2 Further Research

Recommendations for further research are stated as below:

1. Transshipment in excess pullback procedure

Tagaras (1999) pointed that lateral transshipment is an effective means of improving customer service and reducing total system costs. Lateral transshipment in our project means distributing excess parts among all local depots. If transshipment is involved in this project, not all excess parts need to be pulled back to central warehouse, and then sent to another depot; some excess parts can be distributed among depots. Excess parts are delivered directly from one depot to another depot which needs these excess parts. This integrated plan is more efficient and more complicated, which can be one of research areas in the future.

2. Maximize pallet usage

In the proposed plan, calculation for pallet loading volume is based on Cisco historical data, which is imprecise. In the literatures, there exists a research area which is about pallet loading problem. The pallet loading problem is generally tackled by attempting to maximize the number of rectangles that can be fitted orthogonally within a larger containing rectangle. The main issue is about maximizing the pallet space to load as many parts as possible. Several methods have been studied in recent years. But these methods may produce solutions which do not satisfy such real-life problems as load stability and transportability. Carpenter (1985) indicates that the criteria to cater for above problems are not only depending on the type of layout proposed and the way in which the pallet stack is built up. The physical characteristics of the boxes being loaded and the requirements of individual packaging staff also need to be considered. Besides that, in real life, no one would like to spend much time to load products with proposed pallet loading solution, which may only save a little cost. All above constraints show the difficulties of making ideal pallet loading solution. But from other point, because of its difficulty, pallet loading problem with practical consideration would be interesting to investigate in future research.

3. Different values of excess parts

In this project, Cisco treats all excess parts as same value, which means no priority of parts when they are pulled back. But in some situations, excess parts do have priority. So putting excess inventory pullback priority into consideration would be future research area.

Reference:

Cisco Annual Report 2003-2007;
http://www.cisco.com/web/about/ac49/ac20/about_cisco_annual_reports.html;

Carpenter H, Dowsland W B, “Practical considerations of the pallet-loading problem”. *The Journal of the Operational Research Society* 1985; Vol. 36; No. 6; p. 489-497

Csirik J, Frenk J B G, Labbe M, Zhang S, “Heuristics for the 0-1 min-knapsack problem”. *Acta Cybernetica* 1991; 10 (1-2); p. 15-20

Kellerer H, Pferschy U, Pisinger D, *Knapsack Problem*. Springer, 2004

Kelton W D, Sadowski R P, Sadowski D A, *Simulation with Arena*, 2nd Edition, McGraw – Hill, 2002

Sargent R G, “Verification and Validation of simulation models”, *Winter Simulation Conference*, Proceedings of 37th conference on winter simulation, 2005, p. 130-143

Tagaras G, “Pooling in multi-location periodic inventory distribution systems”. *Omega* 1999; 27; 1; p. 39-59

Tersine R J, Toelle R A, “Optimum stock levels for excess inventory items”. *Journal of Operations Management* 1984; 4; 3; p. 245-258

Wee H M, Chung S L, “An optimal production model for disposing excessive deteriorating stocks”. *Journal of the Chinese Institute of Industrial Engineers* 2005; 22; 5; p. 369-378

Appendix I. European depots with unknown tariff

Number	Depot Name	Depot vendor
1	C2G	3PL2
2	LPA	3PL2
3	PMI	3PL2
4	TFS	3PL2
5	TSL	3PL2
6	NIC	3PL2
7	M1A	3PL2
8	BFS	3PL2
9	LJV	3PL2
10	G1I	3PL1
11	IOM	3PL1
12	S1F	3PL1
13	VAR	3PL1
14	TLL	3PL1
15	G1B	3PL1
16	RIX	3PL1
17	VNO	3PL1
18	BGO	3PL1
19	FBV	3PL1
20	SVG	3PL1
21	TOS	3PL1
22	TRD	3PL1
23	BTS	3PL1
24	KSC	3PL1
25	REK	3PL3
26	RIX	3PL3

Appendix II. Main codes of program

CommonDefine.h

```
typedef struct T_DEPOT {
        int iID;
        CString strNameofDepots;
        int iNumofParts;
        double dTotalCost;
        double dRelativeCost;
        BOOL b;//true Îª¿ÉÐÐ½âÄÚ flaseÎª¿ÉÐÐ½âÍâ
}T_DEPOT;

typedef struct T_PALLETPRICE{
        CString strNameofDepots;
        int iCompany;                        //1--3PL2,2--3PL1
        int iP1;               //850kg
        int iP2;               //1700kg
        int iP3;               //2550kg
        int iP4;               //3400
        int iP5;               //4250
        int iP6;               //5100
        int iP7;               //5950
        int iP8;               //6800
        int iP9;               //7650
        int iP10;                      //8500
        int iP11;                      //9350
}T_PALLETPRICE;

typedef struct T_SmallSection{
        int iBegin;
        int iEnd;
        int iTotal;
        BOOL b;
}T_SMALLSECTION;


typedef struct T_SolutionSet{
        int iSmallSectionID;
        int iBigSectionID;
        double dTotalFee;
}T_SOLUTIONSET;


typedef struct T_IGR{
        int iBigID;
```

```
	double dTotalFee;
}T_IGR;
#define MIN_PARTS 950
#define MAX_PARTS		1050
#endif
```

Depotsupplyassdlg.h

```
class CDepotSupplyAssDlg : public CDialog
{
// Construction
public:
	CDepotSupplyAssDlg(CWnd* pParent = NULL);	// standard constructor
	int iIndex;
	void Display(CString str);
	void Empty();
	double CalPalletFee(CString                                    strName,int
iPalletNum,CList<T_PALLETPRICE*,T_PALLETPRICE*> *pTPrice);
	void ProcessIsland();
	int m_iIsLandParts;
	double m_dIsLandFee;
	//PalletPrice
	CString m_strPathPrice;
	//PalletPrice
	CInputCsv	m_pPalletPriceCsv; //Pallet Price for each depot file
	//PalletPrice
	CList<T_PALLETPRICE*,T_PALLETPRICE*>	m_pTPalletPriceList;
	//PalletPrice
	int m_iTPalletPriceNum;
	CString m_strPath;
	CInputCsv   m_pCsv;
	CInputCsv m_pIslandCsv;
	CString m_strIslandPath;
	CMethod1 m_p;
	CMethod2 m_p2;
private:
	BOOL m_bPrice;
	BOOL m_bRead;
// Dialog Data
	//{{AFX_DATA(CDepotSupplyAssDlg)
	enum { IDD = IDD_DEPOTSUPPLYASS_DIALOG };
	CListBox	m_ListBox;
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CDepotSupplyAssDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);	// DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	HICON m_hIcon;
	// Generated message map functions
	//{{AFX_MSG(CDepotSupplyAssDlg)
	virtual BOOL OnInitDialog();
	afx_msg void OnPaint();
	afx_msg HCURSOR OnQueryDragIcon();
	virtual void OnOK();
	virtual void OnCancel();
	afx_msg void OnRead();
	afx_msg void OnSave();
	afx_msg void OnReadprice();
	afx_msg void OnExit();
	afx_msg void OnUpdateReadprice(CCmdUI* pCmdUI);
	afx_msg void OnUpdateRead(CCmdUI* pCmdUI);
	afx_msg void OnProcess1();
	afx_msg void OnProcess2();
	afx_msg void OnProcess3();
	afx_msg void OnReadisland();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif
// !defined(AFX_DEPOTSUPPLYASSDLG_H__93570915_173F_4CAE_9A68_A0E16
F0C9D7F__INCLUDED_)
```

Method1.h

```
class CMethod1
{
public:
	CMethod1();
	virtual ~CMethod1();
	void InitDLg(CDepotSupplyAssDlg *pDlg);
	CDepotSupplyAssDlg *m_pDlg;
	//method1
	void EmptyListofMethod1(void);

	BOOL                                        ReadFileofMethod1(CInputCsv
*m_pInputCsv,CList<T_PALLETPRICE*,T_PALLETPRICE*> *p);

	double                                      CalPalletFee(CStringstrName,int
iPalletNum,CList<T_PALLETPRICE*,T_PALLETPRICE*> *p);

	BOOL SortListofMethod1(void);

	BOOL ClassifyListofMethod1(void);

	BOOL FindOptimalofMethod1(void);

	int FindMinofMethod1(double* p,int Num);

	BOOL MoreOptimalMethod(void);

	BOOL WriteResultofMethod1(CInputCsv *m_pInputCsv,CString strFilePath);

	//method1
	CList<T_DEPOT*,T_DEPOT*>	m_pTDepotList;
	int m_iTDepot;
	CList<T_SMALLSECTION*,T_SMALLSECTION*>	m_pTSmSectList;
	int m_iSmSect;
	CList<T_SOLUTIONSET*,T_SOLUTIONSET*> m_pTSolutionList;
	int m_iSolutionNum;
	int m_iMinInex;
	double m_dTotalFee;
	double pTotalFee[10000];
};
#endif
// !defined(AFX_METHOD1_H__FA37C831_9A97_4C5B_9122_C973BBA828E2__IN
CLUDED_)
```

Onprocess1[]

```
#include "stdafx.h"
#include "DepotSupplyAss.h"
#include "DepotSupplyAssDlg.h"
#include "math.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
```

```
#endif

/////////////////////////////////////////////////////////////////////////////
// CDepotSupplyAssDlg dialog
CDepotSupplyAssDlg::CDepotSupplyAssDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CDepotSupplyAssDlg::IDD, pParent)
{//{{AFX_DATA_INIT(CDepotSupplyAssDlg)
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
	m_bPrice = TRUE;
	m_bRead = FALSE;
	m_iIsLandParts = 0;
	m_dIsLandFee = 0;
}
void CDepotSupplyAssDlg::DoDataExchange(CDataExchange* pDX)
{	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CDepotSupplyAssDlg)
	DDX_Control(pDX, IDC_LIST1, m_ListBox);
	//}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CDepotSupplyAssDlg, CDialog)
	//{{AFX_MSG_MAP(CDepotSupplyAssDlg)
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_COMMAND(ID_READ, OnRead)
	ON_COMMAND(ID_SAVE, OnSave)
	ON_COMMAND(ID_READPRICE, OnReadprice)
	ON_COMMAND(ID_EXIT, OnExit)
	ON_UPDATE_COMMAND_UI(ID_READPRICE, OnUpdateReadprice)
	ON_UPDATE_COMMAND_UI(ID_READ, OnUpdateRead)
	ON_COMMAND(ID_PROCESS1, OnProcess1)
	ON_COMMAND(ID_PROCESS2, OnProcess2)
	ON_COMMAND(ID_PROCESS3, OnProcess3)
	ON_COMMAND(ID_READISLAND, OnReadisland)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CDepotSupplyAssDlg message handlers
BOOL CDepotSupplyAssDlg::OnInitDialog()
{CDialog::OnInitDialog();
	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon
	// TODO: Add extra initialization here
	m_p.InitDLg(this);
	m_p2.InitDLg(this);
	m_ListBox.ResetContent();
	iIndex = 0;
	return TRUE;  // return TRUE  unless you set the focus to a control}
// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CDepotSupplyAssDlg::OnPaint()
{if (IsIconic())
	{		CPaintDC dc(this); // device context for painting
		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(),
0);
// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}	else
{CDialog::OnPaint();}}
// The System calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CDepotSupplyAssDlg::OnQueryDragIcon()
{	return (HCURSOR) m_hIcon;
}
void CDepotSupplyAssDlg::OnOK()
{
	// TODO: Add extra validation here
	//EmptyListofMethod1();
	Empty();
	CDialog::OnOK();}
void CDepotSupplyAssDlg::OnCancel()
{// TODO: Add extra cleanup here
	Empty();
	CDialog::OnCancel();}
void CDepotSupplyAssDlg::Display(CString str)
{m_ListBox.InsertString(iIndex++,str);
	}
void CDepotSupplyAssDlg::OnRead()
```

```
{         // TODO: Add your command handler code here
          static char szFilter[] = "CSV Files(*.csv)|*.csv|";
          CFileDialog FileDlg( TRUE, NULL, NULL,OFN_HIDEREADONLY, szFilter );
          CString strFilePath;
          if( FileDlg.DoModal() == IDOK )
          {m_strPath = FileDlg.GetPathName();}else
          {return;}
          m_pCsv.LoadFile(m_strPath);
          CString str = "Has read:" + m_strPath;
          Display(str);}

void CDepotSupplyAssDlg::OnReadisland()
{         // TODO: Add your command handler code here
          static char szFilter[] = "CSV Files(*.csv)|*.csv|";
          CFileDialog FileDlg( TRUE, NULL, NULL,OFN_HIDEREADONLY, szFilter );
          CString strFilePath;
          if( FileDlg.DoModal() == IDOK )
          {m_strIslandPath = FileDlg.GetPathName();
}else
          {return;}
          m_pIslandCsv.LoadFile(m_strIslandPath);
          CString str = "Has read:" + m_strIslandPath;
          Display(str);
}
void CDepotSupplyAssDlg::OnSave()
{CString str1 = "CSV Files(*.csv)|*.csv|";
          str1 = m_strPath + str1;
          CFileDialog FileDlg( FALSE, NULL, NULL,OFN_HIDEREADONLY, str1);
          CString strFilePath;
          if( FileDlg.DoModal() == IDOK )
          {strFilePath = FileDlg.GetPathName();
                    m_p.WriteResultofMethod1(&m_pCsv,strFilePath);
                    }else
          {return;
          }         }

void CDepotSupplyAssDlg::OnReadprice()
{
          static char szFilter[] = "CSV Files(*.csv)|*.csv|";
          CFileDialog FileDlg( TRUE, NULL, NULL,OFN_HIDEREADONLY, szFilter );
          CString strFilePath;
          if( FileDlg.DoModal() == IDOK )
          {         m_strPathPrice = FileDlg.GetPathName();
          }else
          {return;
          }         // TODO: Add your command handler code here
```

```
        m_pPalletPriceCsv.LoadFile(m_strPathPrice);
        int iItemCount = m_pPalletPriceCsv.GetItemCount();
        for (int i = 0; i < iItemCount; i++)
        {
                CString strName;
                CString strCompany;
                CString str1,str2,str3,str4,str5,str6,str7,str8,str9,str10,str11;
                m_pPalletPriceCsv.GetItem(2,i,&strName);
                m_pPalletPriceCsv.GetItem(3,i,&strCompany);
                m_pPalletPriceCsv.GetItem(4,i,&str1);
                m_pPalletPriceCsv.GetItem(5,i,&str2);
                m_pPalletPriceCsv.GetItem(6,i,&str3);
                m_pPalletPriceCsv.GetItem(7,i,&str4);
                m_pPalletPriceCsv.GetItem(8,i,&str5);
                m_pPalletPriceCsv.GetItem(9,i,&str6);
                m_pPalletPriceCsv.GetItem(10,i,&str7);
                m_pPalletPriceCsv.GetItem(11,i,&str8);
                m_pPalletPriceCsv.GetItem(12,i,&str9);
                m_pPalletPriceCsv.GetItem(13,i,&str10);
                m_pPalletPriceCsv.GetItem(14,i,&str11);
                if (strName != ""/* && strCompany == "3PL1"*/)
                {T_PALLETPRICE * p = new T_PALLETPRICE;
                        p->strNameofDepots = strName;
                        p->iCompany = 2;
                        p->iP1 = atoi(str1);
                        p->iP2 = atoi(str2);
                        p->iP3 = atoi(str3);
                        p->iP4 = atoi(str4);
                        p->iP5 = atoi(str5);
                        p->iP6 = atoi(str6);
                        p->iP7 = atoi(str7);
                        p->iP8 = atoi(str8);
                        p->iP9 = atoi(str9);
                        p->iP10 = atoi(str10);
                        p->iP11 = atoi(str11);
                        m_pTPalletPriceList.AddTail(p);
                }       }
        m_iTPalletPriceNum = m_pTPalletPriceList.GetCount();
        CString str = "Has read:" + m_strPathPrice;
        Display(str);
        m_bPrice = FALSE;
}
void CDepotSupplyAssDlg::OnExit()
{
        // TODO: Add your command handler code here
        OnOK();}
```

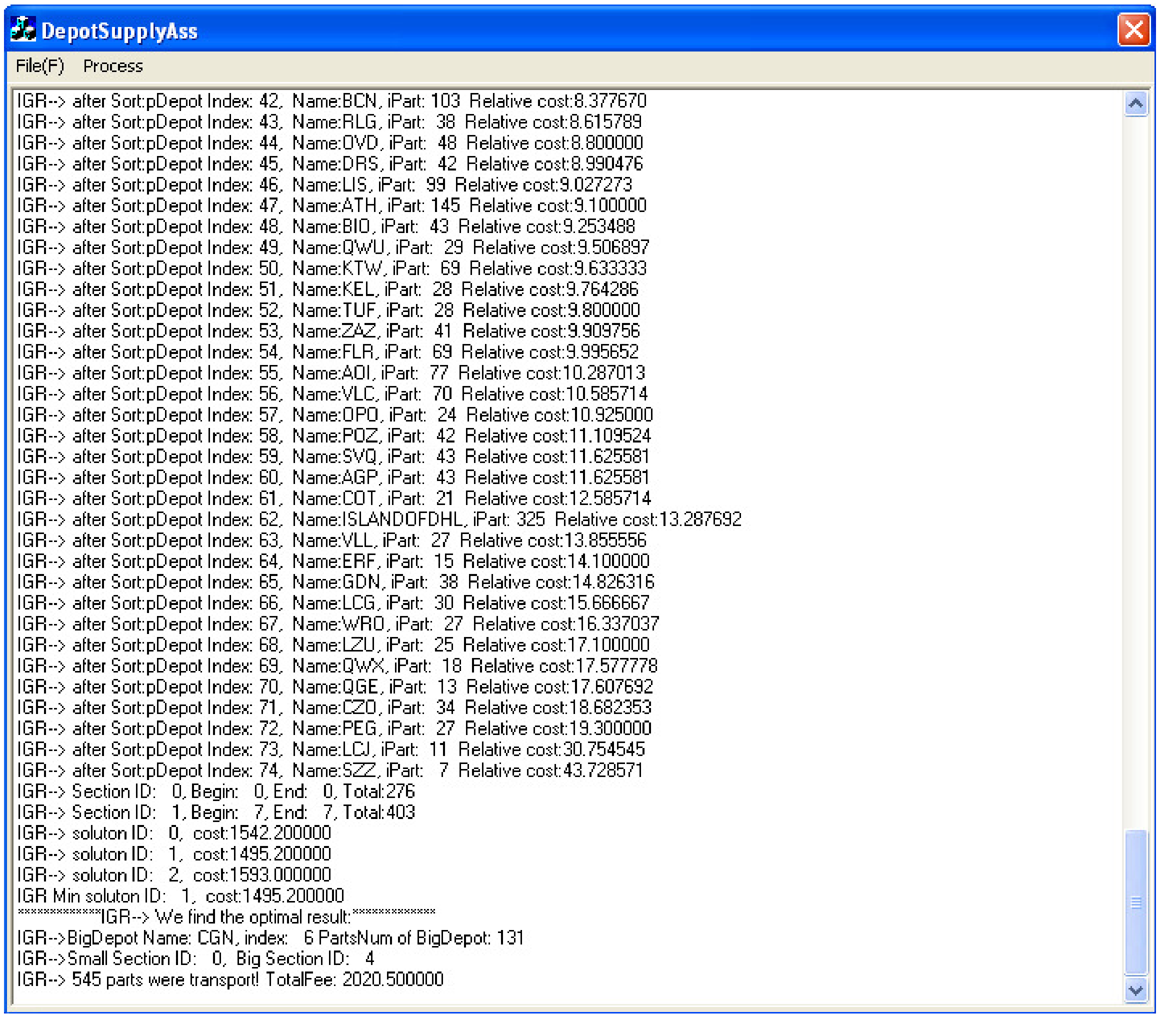
```
void CDepotSupplyAssDlg::Empty()
{
	for (int i = 0; i < m_iTPalletPriceNum; i++)
	{T_PALLETPRICE*                         p                        =
m_pTPalletPriceList.GetAt(m_pTPalletPriceList.FindIndex(i));
		delete p;
		p = NULL;	}
	m_pTPalletPriceList.RemoveAll();
}
void CDepotSupplyAssDlg::OnUpdateReadprice(CCmdUI* pCmdUI)
{// TODO: Add your command update UI handler code here
	pCmdUI->Enable(m_bPrice);
}
void CDepotSupplyAssDlg::OnUpdateRead(CCmdUI* pCmdUI)
{// TODO: Add your command update UI handler code here
	//pCmdUI->Enable(m_bRead);
}
void CDepotSupplyAssDlg::OnProcess1()
{// TODO: Add your command handler code here
	m_p.EmptyListofMethod1();
	ProcessIsland();
	m_p.ReadFileofMethod1(&m_pCsv,&m_pTPalletPriceList);
	m_p.SortListofMethod1();
	m_p.ClassifyListofMethod1();
	m_p.FindOptimalofMethod1();
	MessageBox("Complete!\n,Please Save the Result!");
	OnSave();
}
void CDepotSupplyAssDlg::OnProcess2()
{// TODO: Add your command handler code here
	ProcessIsland();
	m_p.EmptyListofMethod1();
	m_p.ReadFileofMethod1(&m_pCsv,&m_pTPalletPriceList);
	m_p.SortListofMethod1();
	m_p.ClassifyListofMethod1();
	m_p.FindOptimalofMethod1();
	m_p.MoreOptimalMethod();
	MessageBox("Complete!\n,Please Save the Result!");
	OnSave();
}

void CDepotSupplyAssDlg::OnProcess3()
{// TODO: Add your command handler code here
	ProcessIsland();
	m_p2.EmptyList();
```

```
        m_p2.ReadFile(&m_pCsv,&m_pTPalletPriceList);
        m_p2.SortList();
        m_p2.ClassifyList();
        m_p2.FindOptimal();
        int iResult = m_p2.IGR();
        MessageBox("Complete!\n,Please Save the Result!");
        CString str1 = "CSV Files(*.csv)|*.csv|";
        str1 = m_strPath + str1;
        CFileDialog FileDlg( FALSE, NULL, NULL,OFN_HIDEREADONLY, str1);
        CString strFilePath;
        if( FileDlg.DoModal() == IDOK )
        {strFilePath = FileDlg.GetPathName()
                if (iResult == 1)//GR
                {m_p2.WriteResult(&m_pCsv,strFilePath);}
                else//IGR
                {m_p2.WriteResultofIGR(&m_pCsv,strFilePath);}
        }}
void CDepotSupplyAssDlg::ProcessIsland()
{CString str;
        m_pCsv.GetItem(3,0,&str);
        if (str != "3PL2")
        {       return;
        }
        int iIslandDepotNum = m_pIslandCsv.GetItemCount();
        for (int i = 0; i < iIslandDepotNum; i++)
        {
                CString strName;
                CString strParts;

                m_pIslandCsv.GetItem(2,i,&strName);
                m_pIslandCsv.GetItem(4,i,&strParts);

                int iParts = atoi(strParts);
                double dPalletNum = iParts/24.0;
                int iPalletNum = (int)ceil(dPalletNum);
                double                          dPalletcost                             =
CalPalletFee(strName,iPalletNum,&m_pTPalletPriceList);
                double dHandingFee = 2.3*iParts;
                double dtotalCost = dPalletcost + dHandingFee;

                CString strNumofPallet;
                CString strRNumofPallet;
                CString strHandingFee;
                CString strPalletCost;
                CString strTotalCost;
                strNumofPallet.Format("%f",dPalletNum);
```

```
		strRNumofPallet.Format("%d",iPalletNum);
		strHandingFee.Format("%f",dHandingFee);
		strPalletCost.Format("%f",dPalletcost);
		strTotalCost.Format("%f",dtotalCost);

		m_pIslandCsv.UpdateItem(5,i,strNumofPallet);
		m_pIslandCsv.UpdateItem(6,i,strRNumofPallet);
		m_pIslandCsv.UpdateItem(7,i,strHandingFee);
		m_pIslandCsv.UpdateItem(8,i,strPalletCost);
		m_pIslandCsv.UpdateItem(9,i,strTotalCost);
		m_iIsLandParts += iParts;
		m_dIsLandFee += dtotalCost;
	}

	m_pIslandCsv.SaveFile(m_strIslandPath);
	CString strParts;
	CString strTotalCost;
	strParts.Format("%d",m_iIsLandParts);
	strTotalCost.Format("%f",m_dIsLandFee);
	m_pCsv.UpdateSpeicial("ISLANDOF3PL2",strParts,strTotalCost);
	m_pCsv.SaveFile(m_strPath);
}
double			CDepotSupplyAssDlg::CalPalletFee(CString			strName,int
iPalletNum,CList<T_PALLETPRICE*,T_PALLETPRICE*> *pTPrice)
{
	int iPriceNum = pTPrice->GetCount();

	for (int i = 0; i < iPriceNum; i++)
	{T_PALLETPRICE *p = pTPrice->GetAt(pTPrice->FindIndex(i));
		if (p->strNameofDepots == strName)
		{ switch(iPalletNum)
			{case 1: return p->iP1;
				case 2: return p->iP2;
				case 3: return p->iP3;
				case 4: return p->iP4;
				case 5: return p->iP5;
				case 6: return p->iP6;
				case 7: return p->iP7;
				case 8: return p->iP8;
				case 9: return p->iP9;
				case 10: return p->iP10;
				case 11: return p->iP11;
				default:
					break;}break;}}
return 0;}
```

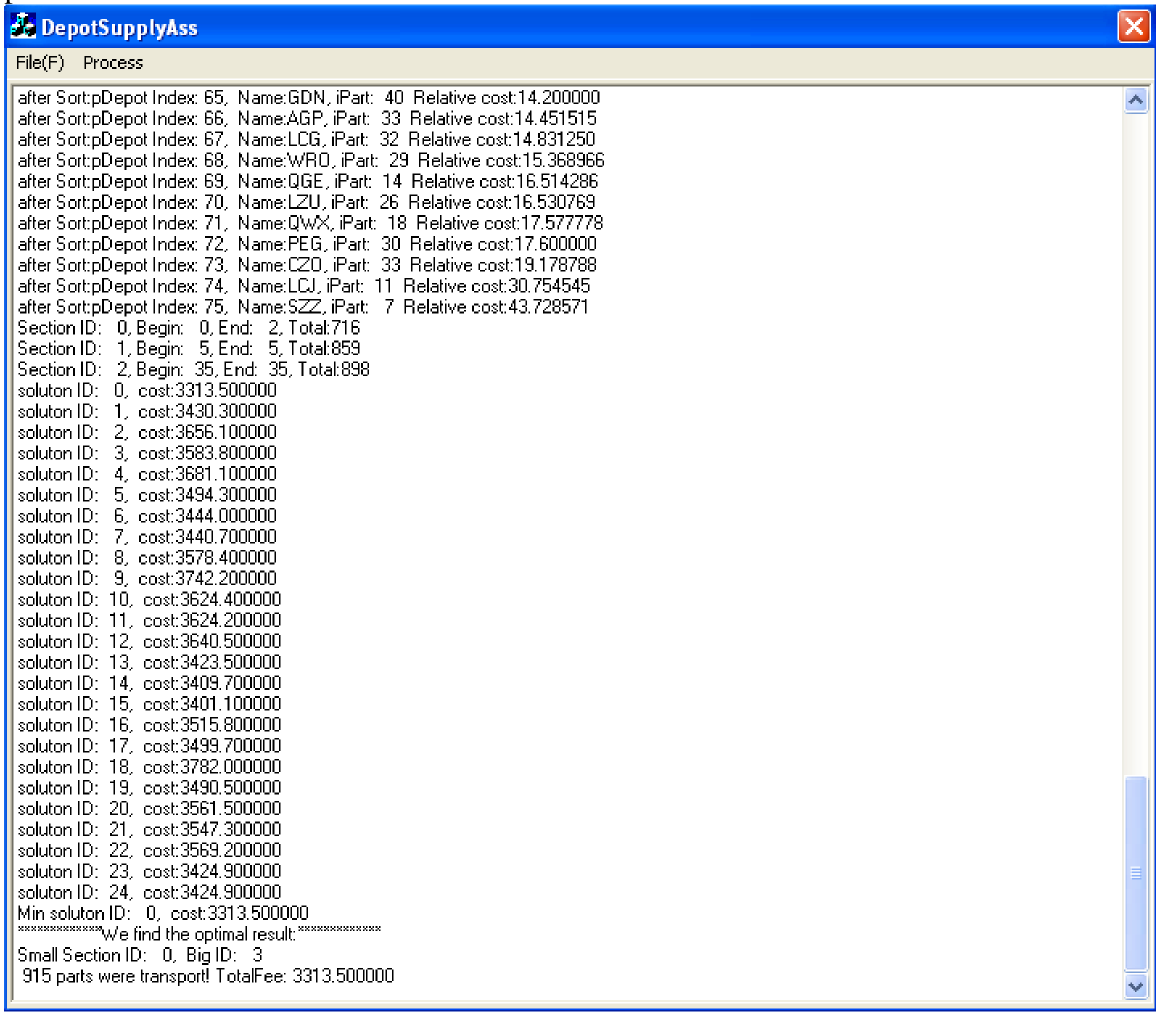
Appendix III. Results of proposed plan

■ Week 4 of 2008, the range is changed to 537-600; the input data file is the excess parts information of week 2/2008



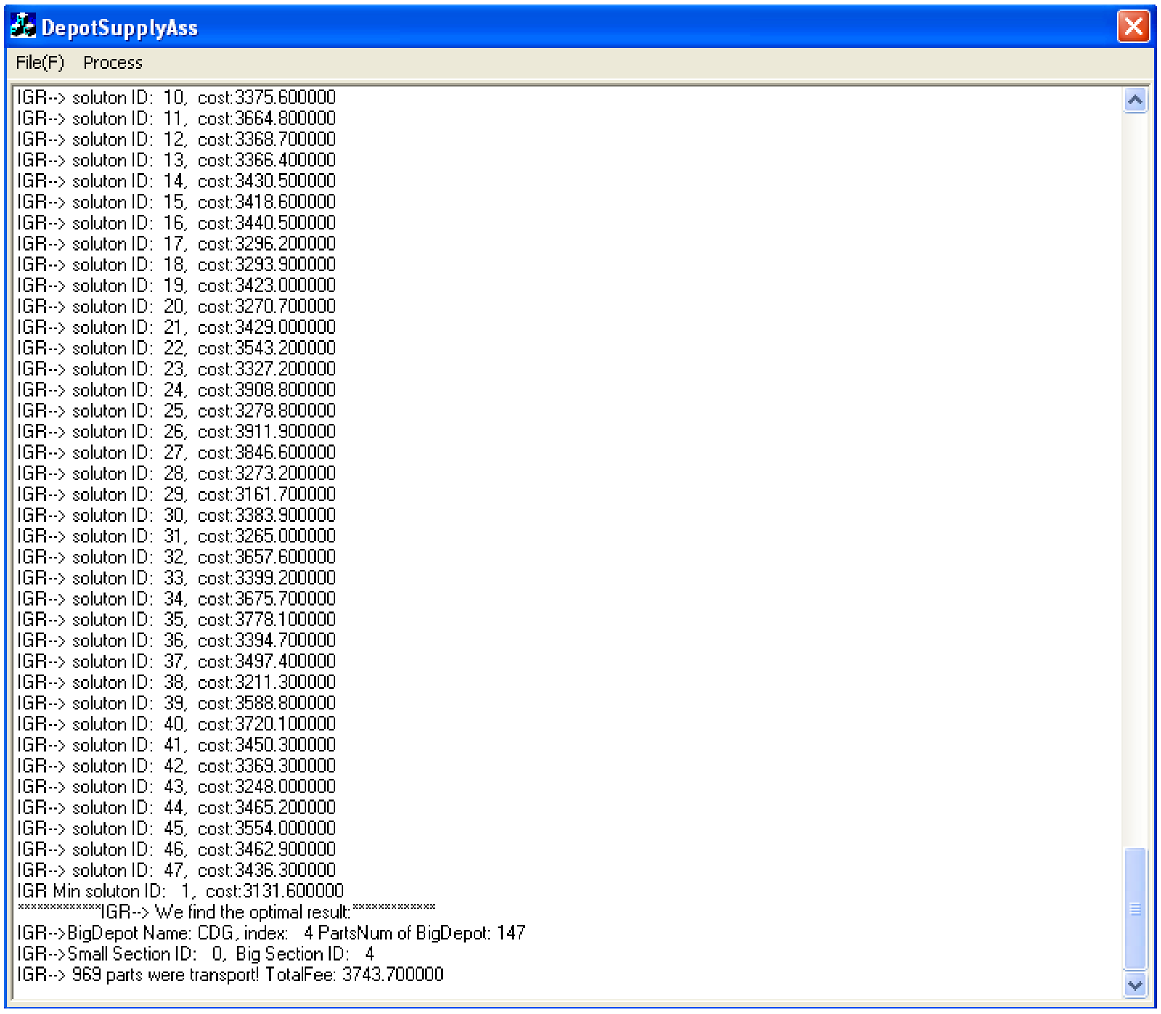
```
IGR--> after Sort:pDepot Index: 42,  Name:BCN, iPart: 103  Relative cost:8.377670
IGR--> after Sort:pDepot Index: 43,  Name:RLG, iPart:  38  Relative cost:8.615789
IGR--> after Sort:pDepot Index: 44,  Name:OVD, iPart:  48  Relative cost:8.800000
IGR--> after Sort:pDepot Index: 45,  Name:DRS, iPart:  42  Relative cost:8.990476
IGR--> after Sort:pDepot Index: 46,  Name:LIS, iPart:  99  Relative cost:9.027273
IGR--> after Sort:pDepot Index: 47,  Name:ATH, iPart: 145  Relative cost:9.100000
IGR--> after Sort:pDepot Index: 48,  Name:BIO, iPart:  43  Relative cost:9.253488
IGR--> after Sort:pDepot Index: 49,  Name:QWU, iPart:  29  Relative cost:9.506897
IGR--> after Sort:pDepot Index: 50,  Name:KTW, iPart:  69  Relative cost:9.633333
IGR--> after Sort:pDepot Index: 51,  Name:KEL, iPart:  28  Relative cost:9.764286
IGR--> after Sort:pDepot Index: 52,  Name:TUF, iPart:  28  Relative cost:9.800000
IGR--> after Sort:pDepot Index: 53,  Name:ZAZ, iPart:  41  Relative cost:9.909756
IGR--> after Sort:pDepot Index: 54,  Name:FLR, iPart:  69  Relative cost:9.995652
IGR--> after Sort:pDepot Index: 55,  Name:AOI, iPart:  77  Relative cost:10.287013
IGR--> after Sort:pDepot Index: 56,  Name:VLC, iPart:  70  Relative cost:10.585714
IGR--> after Sort:pDepot Index: 57,  Name:OPO, iPart:  24  Relative cost:10.925000
IGR--> after Sort:pDepot Index: 58,  Name:POZ, iPart:  42  Relative cost:11.109524
IGR--> after Sort:pDepot Index: 59,  Name:SVQ, iPart:  43  Relative cost:11.625581
IGR--> after Sort:pDepot Index: 60,  Name:AGP, iPart:  43  Relative cost:11.625581
IGR--> after Sort:pDepot Index: 61,  Name:COT, iPart:  21  Relative cost:12.585714
IGR--> after Sort:pDepot Index: 62,  Name:ISLANDOFDHL, iPart: 325  Relative cost:13.287692
IGR--> after Sort:pDepot Index: 63,  Name:VLL, iPart:  27  Relative cost:13.855556
IGR--> after Sort:pDepot Index: 64,  Name:ERF, iPart:  15  Relative cost:14.100000
IGR--> after Sort:pDepot Index: 65,  Name:GDN, iPart:  38  Relative cost:14.826316
IGR--> after Sort:pDepot Index: 66,  Name:LCG, iPart:  30  Relative cost:15.666667
IGR--> after Sort:pDepot Index: 67,  Name:WRO, iPart:  27  Relative cost:16.337037
IGR--> after Sort:pDepot Index: 68,  Name:LZU, iPart:  25  Relative cost:17.100000
IGR--> after Sort:pDepot Index: 69,  Name:QWX, iPart:  18  Relative cost:17.577778
IGR--> after Sort:pDepot Index: 70,  Name:QGE, iPart:  13  Relative cost:17.607692
IGR--> after Sort:pDepot Index: 71,  Name:CZO, iPart:  34  Relative cost:18.682353
IGR--> after Sort:pDepot Index: 72,  Name:PEG, iPart:  27  Relative cost:19.300000
IGR--> after Sort:pDepot Index: 73,  Name:LCJ, iPart:  11  Relative cost:30.754545
IGR--> after Sort:pDepot Index: 74,  Name:SZZ, iPart:   7  Relative cost:43.728571
IGR--> Section ID:   0, Begin:   0, End:   0, Total:276
IGR--> Section ID:   1, Begin:   7, End:   7, Total:403
IGR--> soluton ID:   0,  cost:1542.200000
IGR--> soluton ID:   1,  cost:1495.200000
IGR--> soluton ID:   2,  cost:1593.000000
IGR Min soluton ID:   1,  cost:1495.200000
**************IGR--> We find the optimal result:**************
IGR-->BigDepot Name: CGN, index:   6 PartsNum of BigDepot: 131
IGR-->Small Section ID:   0,  Big Section ID:   4
IGR--> 545 parts were transport! TotalFee: 2020.500000
```

■ Week 8 of 2008, the range is changed to 900-1000; the input data file is the excess parts information of week 6/2008



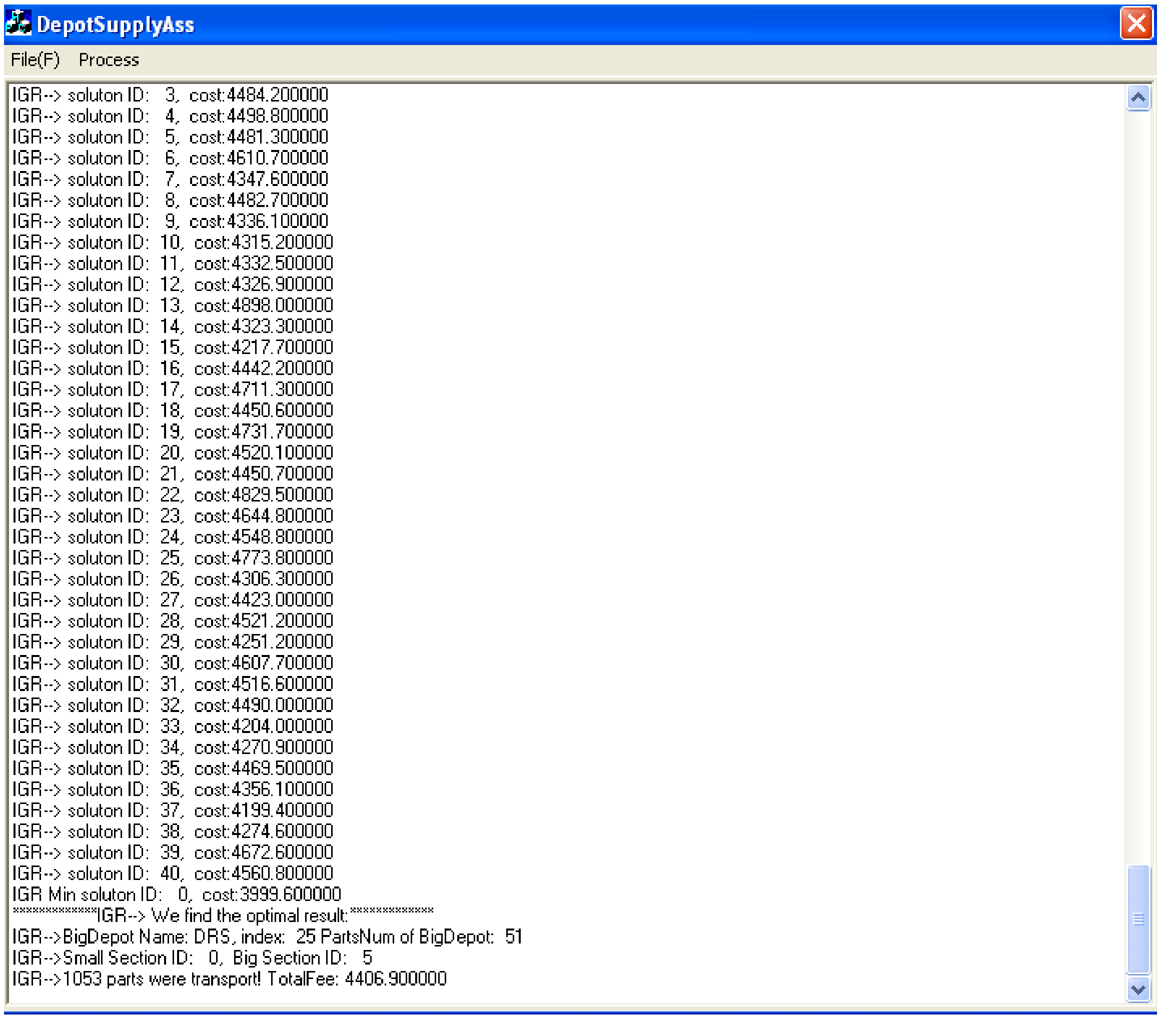
```
after Sort:pDepot Index: 65,  Name:GDN, iPart:  40  Relative cost:14.200000
after Sort:pDepot Index: 66,  Name:AGP, iPart:  33  Relative cost:14.451515
after Sort:pDepot Index: 67,  Name:LCG, iPart:  32  Relative cost:14.831250
after Sort:pDepot Index: 68,  Name:WRO, iPart:  29  Relative cost:15.368966
after Sort:pDepot Index: 69,  Name:QGE, iPart:  14  Relative cost:16.514286
after Sort:pDepot Index: 70,  Name:LZU, iPart:  26  Relative cost:16.530769
after Sort:pDepot Index: 71,  Name:QWX, iPart:  18  Relative cost:17.577778
after Sort:pDepot Index: 72,  Name:PEG, iPart:  30  Relative cost:17.600000
after Sort:pDepot Index: 73,  Name:CZO, iPart:  33  Relative cost:19.178788
after Sort:pDepot Index: 74,  Name:LCJ, iPart:  11  Relative cost:30.754545
after Sort:pDepot Index: 75,  Name:SZZ, iPart:   7  Relative cost:43.728571
Section ID:   0, Begin:   0, End:   2, Total:716
Section ID:   1, Begin:   5, End:   5, Total:859
Section ID:   2, Begin:  35, End:  35, Total:898
soluton ID:   0,  cost:3313.500000
soluton ID:   1,  cost:3430.300000
soluton ID:   2,  cost:3656.100000
soluton ID:   3,  cost:3583.800000
soluton ID:   4,  cost:3681.100000
soluton ID:   5,  cost:3494.300000
soluton ID:   6,  cost:3444.000000
soluton ID:   7,  cost:3440.700000
soluton ID:   8,  cost:3578.400000
soluton ID:   9,  cost:3742.200000
soluton ID:  10,  cost:3624.400000
soluton ID:  11,  cost:3624.200000
soluton ID:  12,  cost:3640.500000
soluton ID:  13,  cost:3423.500000
soluton ID:  14,  cost:3409.700000
soluton ID:  15,  cost:3401.100000
soluton ID:  16,  cost:3515.800000
soluton ID:  17,  cost:3499.700000
soluton ID:  18,  cost:3782.000000
soluton ID:  19,  cost:3490.500000
soluton ID:  20,  cost:3561.500000
soluton ID:  21,  cost:3547.300000
soluton ID:  22,  cost:3569.200000
soluton ID:  23,  cost:3424.900000
soluton ID:  24,  cost:3424.900000
Min soluton ID:   0,  cost:3313.500000
*************We find the optimal result:*************
Small Section ID:   0,  Big ID:   3
 915 parts were transport! TotalFee: 3313.500000
```

■ Week 9 of 2008, the range is changed to 946-1046; the input data file is the excess parts information of week 7/2008



```
IGR--> soluton ID:  10,  cost:3375.600000
IGR--> soluton ID:  11,  cost:3664.800000
IGR--> soluton ID:  12,  cost:3368.700000
IGR--> soluton ID:  13,  cost:3366.400000
IGR--> soluton ID:  14,  cost:3430.500000
IGR--> soluton ID:  15,  cost:3418.600000
IGR--> soluton ID:  16,  cost:3440.500000
IGR--> soluton ID:  17,  cost:3296.200000
IGR--> soluton ID:  18,  cost:3293.900000
IGR--> soluton ID:  19,  cost:3423.000000
IGR--> soluton ID:  20,  cost:3270.700000
IGR--> soluton ID:  21,  cost:3429.000000
IGR--> soluton ID:  22,  cost:3543.200000
IGR--> soluton ID:  23,  cost:3327.200000
IGR--> soluton ID:  24,  cost:3908.800000
IGR--> soluton ID:  25,  cost:3278.800000
IGR--> soluton ID:  26,  cost:3911.900000
IGR--> soluton ID:  27,  cost:3846.600000
IGR--> soluton ID:  28,  cost:3273.200000
IGR--> soluton ID:  29,  cost:3161.700000
IGR--> soluton ID:  30,  cost:3383.900000
IGR--> soluton ID:  31,  cost:3265.000000
IGR--> soluton ID:  32,  cost:3657.600000
IGR--> soluton ID:  33,  cost:3399.200000
IGR--> soluton ID:  34,  cost:3675.700000
IGR--> soluton ID:  35,  cost:3778.100000
IGR--> soluton ID:  36,  cost:3394.700000
IGR--> soluton ID:  37,  cost:3497.400000
IGR--> soluton ID:  38,  cost:3211.300000
IGR--> soluton ID:  39,  cost:3588.800000
IGR--> soluton ID:  40,  cost:3720.100000
IGR--> soluton ID:  41,  cost:3450.300000
IGR--> soluton ID:  42,  cost:3369.300000
IGR--> soluton ID:  43,  cost:3248.000000
IGR--> soluton ID:  44,  cost:3465.200000
IGR--> soluton ID:  45,  cost:3554.000000
IGR--> soluton ID:  46,  cost:3462.900000
IGR--> soluton ID:  47,  cost:3436.300000
IGR Min soluton ID:   1,  cost:3131.600000
*************IGR--> We find the optimal result:*************
IGR-->BigDepot Name: CDG, index:   4 PartsNum of BigDepot: 147
IGR-->Small Section ID:   0, Big Section ID:   4
IGR--> 969 parts were transport! TotalFee: 3743.700000
```

■ Week 10 of 2008, the range is changed to 1049-1149; the input data file is the excess parts information of week 8/2008

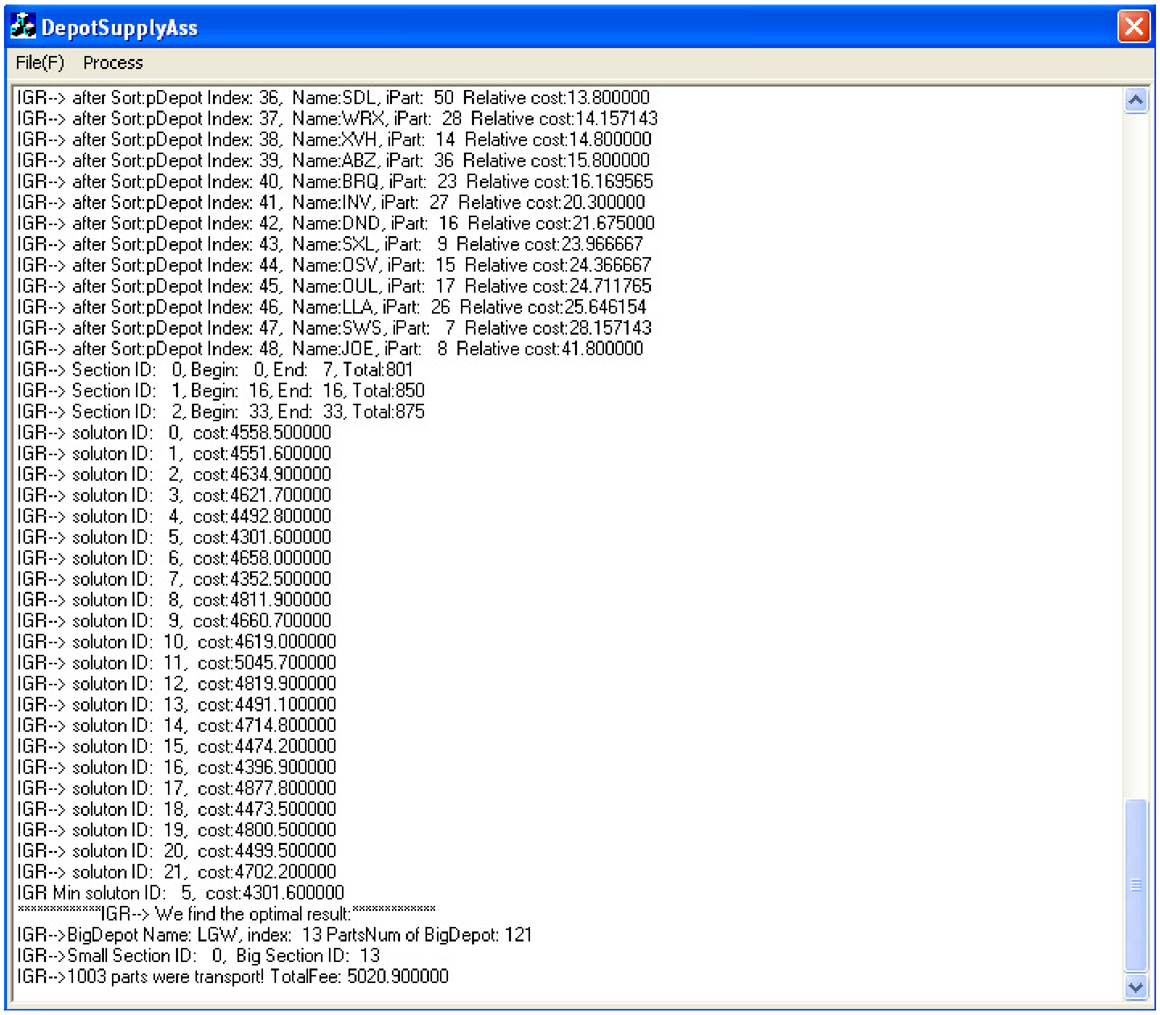


DepotSupplyAss

File(F)   Process

```
IGR--> soluton ID:   3,  cost:4484.200000
IGR--> soluton ID:   4,  cost:4498.800000
IGR--> soluton ID:   5,  cost:4481.300000
IGR--> soluton ID:   6,  cost:4610.700000
IGR--> soluton ID:   7,  cost:4347.600000
IGR--> soluton ID:   8,  cost:4482.700000
IGR--> soluton ID:   9,  cost:4336.100000
IGR--> soluton ID:  10,  cost:4315.200000
IGR--> soluton ID:  11,  cost:4332.500000
IGR--> soluton ID:  12,  cost:4326.900000
IGR--> soluton ID:  13,  cost:4898.000000
IGR--> soluton ID:  14,  cost:4323.300000
IGR--> soluton ID:  15,  cost:4217.700000
IGR--> soluton ID:  16,  cost:4442.200000
IGR--> soluton ID:  17,  cost:4711.300000
IGR--> soluton ID:  18,  cost:4450.600000
IGR--> soluton ID:  19,  cost:4731.700000
IGR--> soluton ID:  20,  cost:4520.100000
IGR--> soluton ID:  21,  cost:4450.700000
IGR--> soluton ID:  22,  cost:4829.500000
IGR--> soluton ID:  23,  cost:4644.800000
IGR--> soluton ID:  24,  cost:4548.800000
IGR--> soluton ID:  25,  cost:4773.800000
IGR--> soluton ID:  26,  cost:4306.300000
IGR--> soluton ID:  27,  cost:4423.000000
IGR--> soluton ID:  28,  cost:4521.200000
IGR--> soluton ID:  29,  cost:4251.200000
IGR--> soluton ID:  30,  cost:4607.700000
IGR--> soluton ID:  31,  cost:4516.600000
IGR--> soluton ID:  32,  cost:4490.000000
IGR--> soluton ID:  33,  cost:4204.000000
IGR--> soluton ID:  34,  cost:4270.900000
IGR--> soluton ID:  35,  cost:4469.500000
IGR--> soluton ID:  36,  cost:4356.100000
IGR--> soluton ID:  37,  cost:4199.400000
IGR--> soluton ID:  38,  cost:4274.600000
IGR--> soluton ID:  39,  cost:4672.600000
IGR--> soluton ID:  40,  cost:4560.800000
IGR Min soluton ID:   0,  cost:3999.600000
*************IGR--> We find the optimal result:*************
IGR-->BigDepot Name: DRS, index:  25 PartsNum of BigDepot:  51
IGR-->Small Section ID:   0,  Big Section ID:   5
IGR-->1053 parts were transport! TotalFee: 4406.900000
```

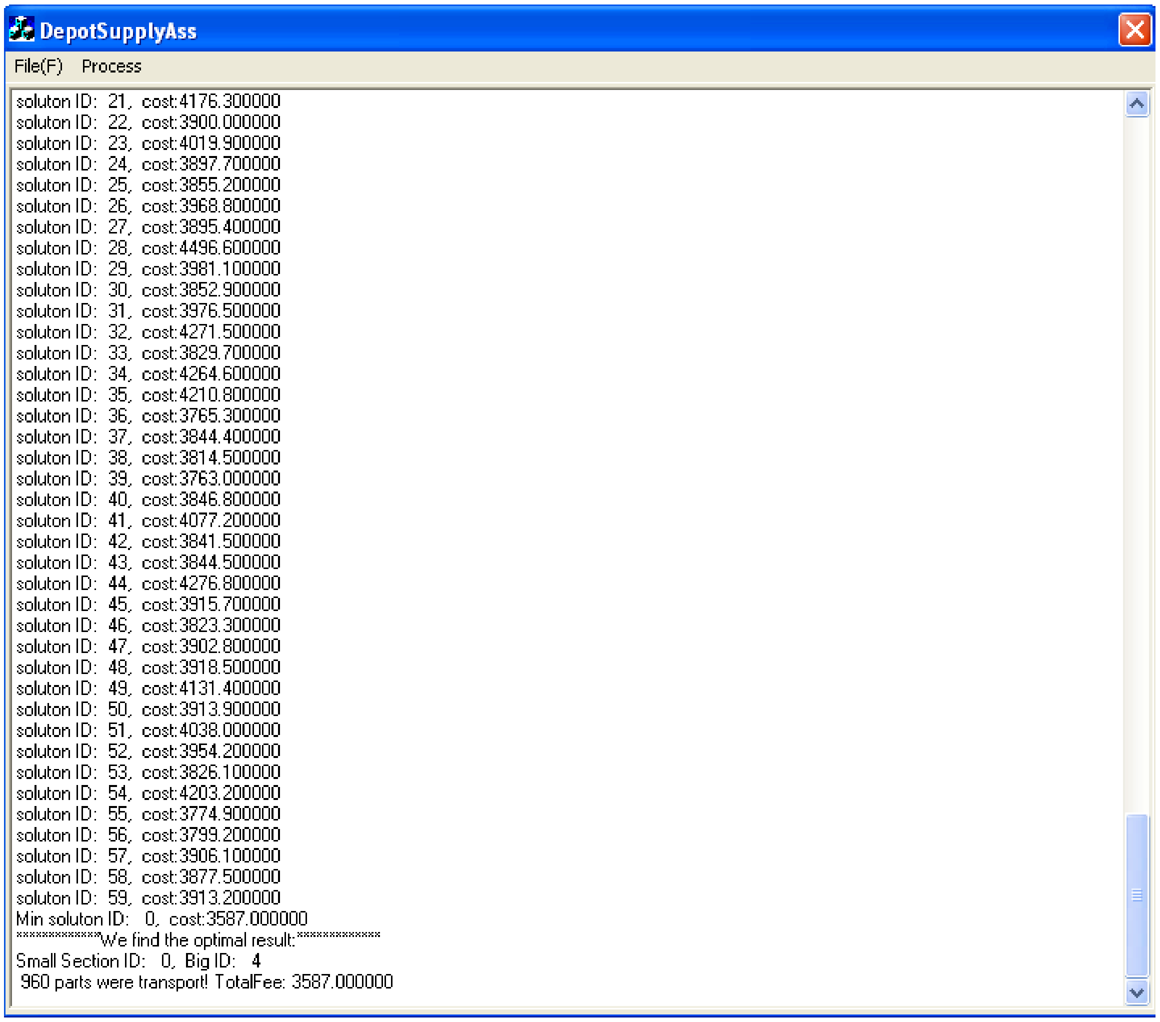
■ Week 11 of 2008, the range is changed to 1003-1103; the input data file is the excess parts information of week 9/2008



```
IGR--> after Sort:pDepot Index: 36, Name:SDL, iPart:  50  Relative cost:13.800000
IGR--> after Sort:pDepot Index: 37, Name:WRX, iPart:  28  Relative cost:14.157143
IGR--> after Sort:pDepot Index: 38, Name:XVH, iPart:  14  Relative cost:14.800000
IGR--> after Sort:pDepot Index: 39, Name:ABZ, iPart:  36  Relative cost:15.800000
IGR--> after Sort:pDepot Index: 40, Name:BRQ, iPart:  23  Relative cost:16.169565
IGR--> after Sort:pDepot Index: 41, Name:INV, iPart:  27  Relative cost:20.300000
IGR--> after Sort:pDepot Index: 42, Name:DND, iPart:  16  Relative cost:21.675000
IGR--> after Sort:pDepot Index: 43, Name:SXL, iPart:   9  Relative cost:23.966667
IGR--> after Sort:pDepot Index: 44, Name:OSV, iPart:  15  Relative cost:24.366667
IGR--> after Sort:pDepot Index: 45, Name:OUL, iPart:  17  Relative cost:24.711765
IGR--> after Sort:pDepot Index: 46, Name:LLA, iPart:  26  Relative cost:25.646154
IGR--> after Sort:pDepot Index: 47, Name:SWS, iPart:   7  Relative cost:28.157143
IGR--> after Sort:pDepot Index: 48, Name:JOE, iPart:   8  Relative cost:41.800000
IGR--> Section ID:   0, Begin:   0, End:   7, Total:801
IGR--> Section ID:   1, Begin:  16, End:  16, Total:850
IGR--> Section ID:   2, Begin:  33, End:  33, Total:875
IGR--> soluton ID:   0,  cost:4558.500000
IGR--> soluton ID:   1,  cost:4551.600000
IGR--> soluton ID:   2,  cost:4634.900000
IGR--> soluton ID:   3,  cost:4621.700000
IGR--> soluton ID:   4,  cost:4492.800000
IGR--> soluton ID:   5,  cost:4301.600000
IGR--> soluton ID:   6,  cost:4658.000000
IGR--> soluton ID:   7,  cost:4352.500000
IGR--> soluton ID:   8,  cost:4811.900000
IGR--> soluton ID:   9,  cost:4660.700000
IGR--> soluton ID:  10,  cost:4619.000000
IGR--> soluton ID:  11,  cost:5045.700000
IGR--> soluton ID:  12,  cost:4819.900000
IGR--> soluton ID:  13,  cost:4491.100000
IGR--> soluton ID:  14,  cost:4714.800000
IGR--> soluton ID:  15,  cost:4474.200000
IGR--> soluton ID:  16,  cost:4396.900000
IGR--> soluton ID:  17,  cost:4877.800000
IGR--> soluton ID:  18,  cost:4473.500000
IGR--> soluton ID:  19,  cost:4800.500000
IGR--> soluton ID:  20,  cost:4499.500000
IGR--> soluton ID:  21,  cost:4702.200000
IGR Min soluton ID:   5,  cost:4301.600000
**************IGR--> We find the optimal result:**************
IGR-->BigDepot Name: LGW, index:  13 PartsNum of BigDepot: 121
IGR-->Small Section ID:   0,  Big Section ID:  13
IGR-->1003 parts were transport! TotalFee: 5020.900000
```

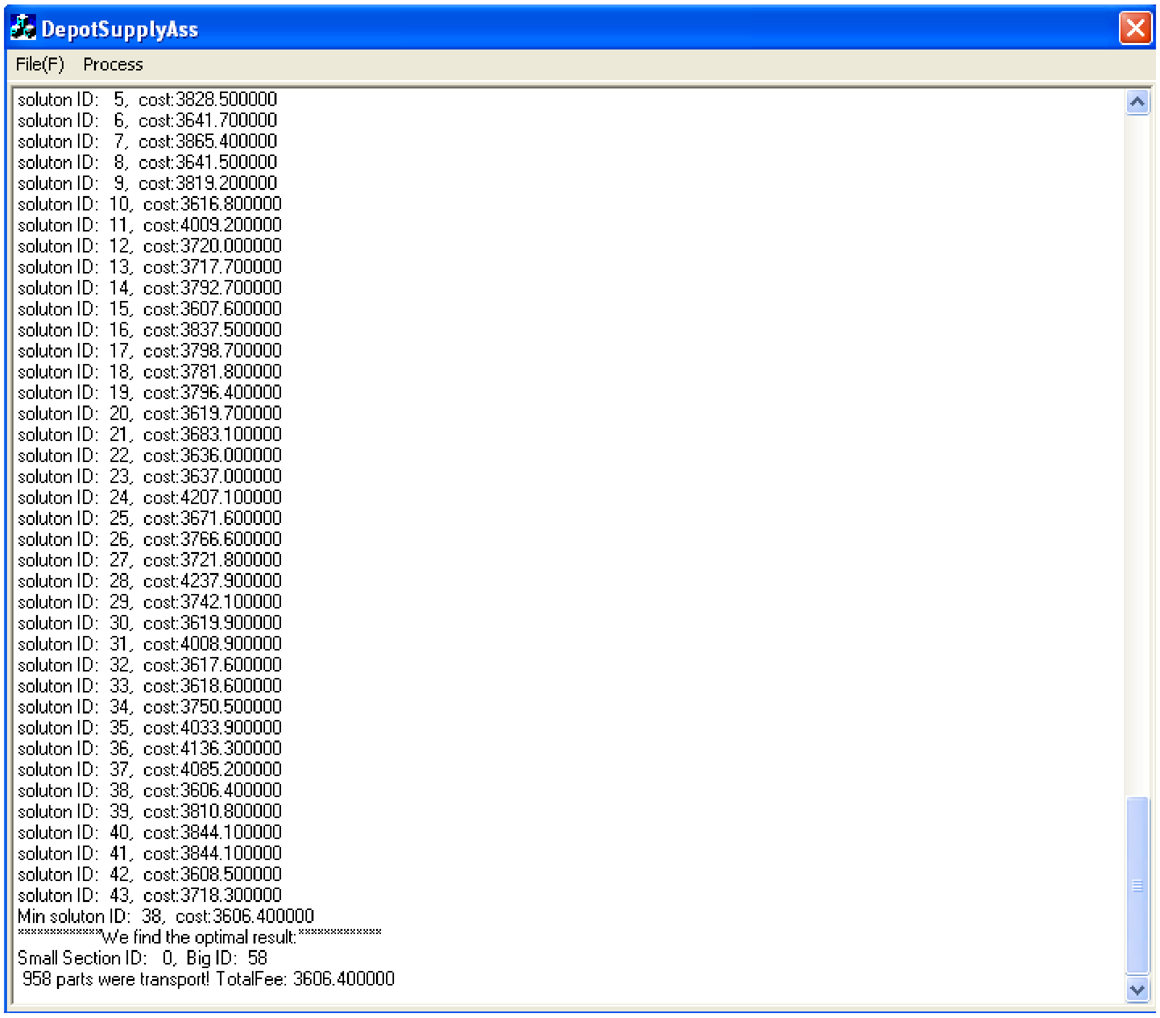
Appendix IV. Results of proposed plan

In week 51 of 2007, the target amount is 950-1050; the input data file is the excess parts information of week 49/2007



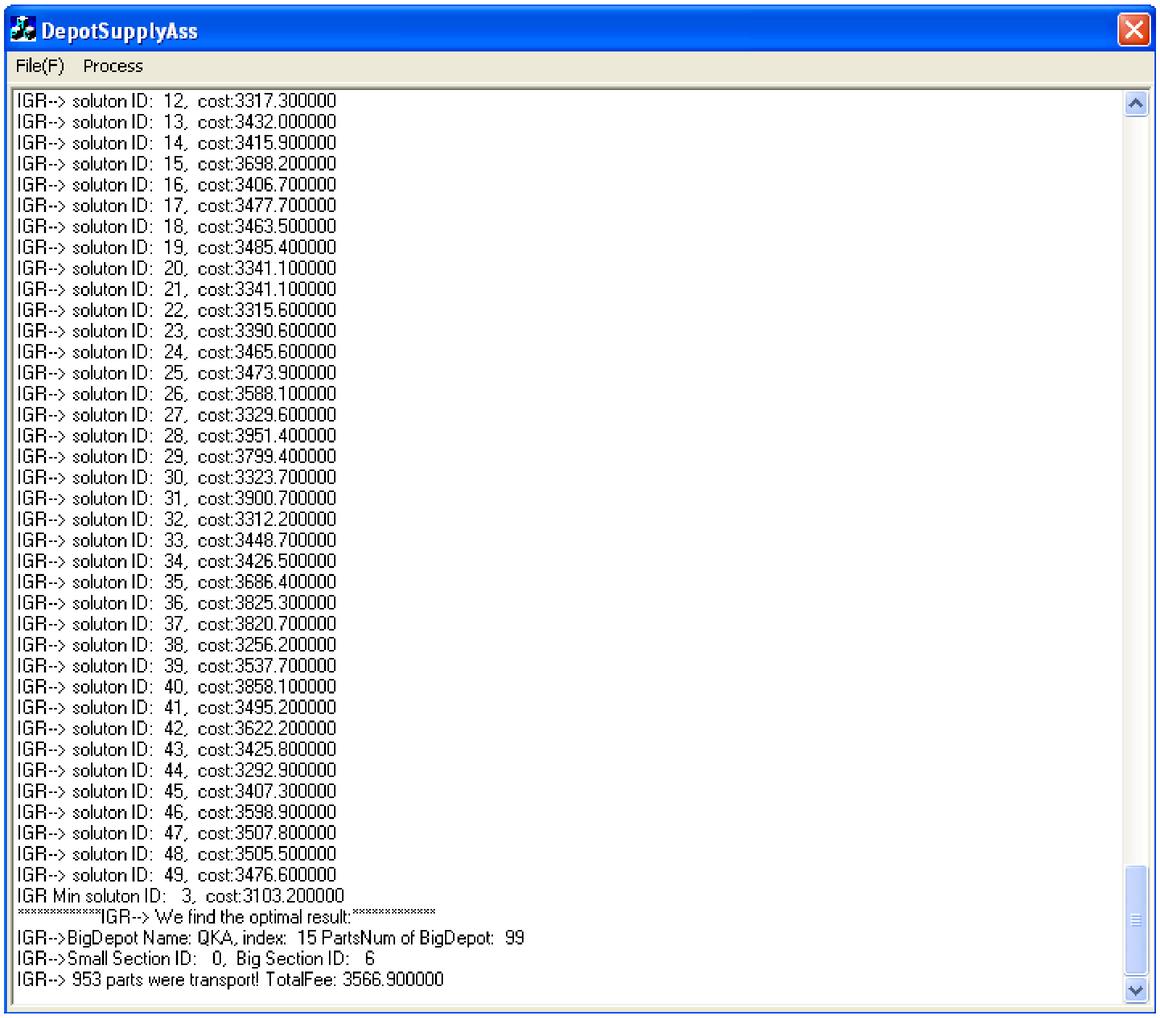
```
soluton ID:  21,  cost:4176.300000
soluton ID:  22,  cost:3900.000000
soluton ID:  23,  cost:4019.900000
soluton ID:  24,  cost:3897.700000
soluton ID:  25,  cost:3855.200000
soluton ID:  26,  cost:3968.800000
soluton ID:  27,  cost:3895.400000
soluton ID:  28,  cost:4496.600000
soluton ID:  29,  cost:3981.100000
soluton ID:  30,  cost:3852.900000
soluton ID:  31,  cost:3976.500000
soluton ID:  32,  cost:4271.500000
soluton ID:  33,  cost:3829.700000
soluton ID:  34,  cost:4264.600000
soluton ID:  35,  cost:4210.800000
soluton ID:  36,  cost:3765.300000
soluton ID:  37,  cost:3844.400000
soluton ID:  38,  cost:3814.500000
soluton ID:  39,  cost:3763.000000
soluton ID:  40,  cost:3846.800000
soluton ID:  41,  cost:4077.200000
soluton ID:  42,  cost:3841.500000
soluton ID:  43,  cost:3844.500000
soluton ID:  44,  cost:4276.800000
soluton ID:  45,  cost:3915.700000
soluton ID:  46,  cost:3823.300000
soluton ID:  47,  cost:3902.800000
soluton ID:  48,  cost:3918.500000
soluton ID:  49,  cost:4131.400000
soluton ID:  50,  cost:3913.900000
soluton ID:  51,  cost:4038.000000
soluton ID:  52,  cost:3954.200000
soluton ID:  53,  cost:3826.100000
soluton ID:  54,  cost:4203.200000
soluton ID:  55,  cost:3774.900000
soluton ID:  56,  cost:3799.200000
soluton ID:  57,  cost:3906.100000
soluton ID:  58,  cost:3877.500000
soluton ID:  59,  cost:3913.200000
Min soluton ID:   0,  cost:3587.000000
**************We find the optimal result:**************
Small Section ID:   0,  Big ID:   4
 960 parts were transport! TotalFee: 3587.000000
```

In week 4 of 2008, the target amount is 950-1050; the input data file is the excess parts information of week 2/2008



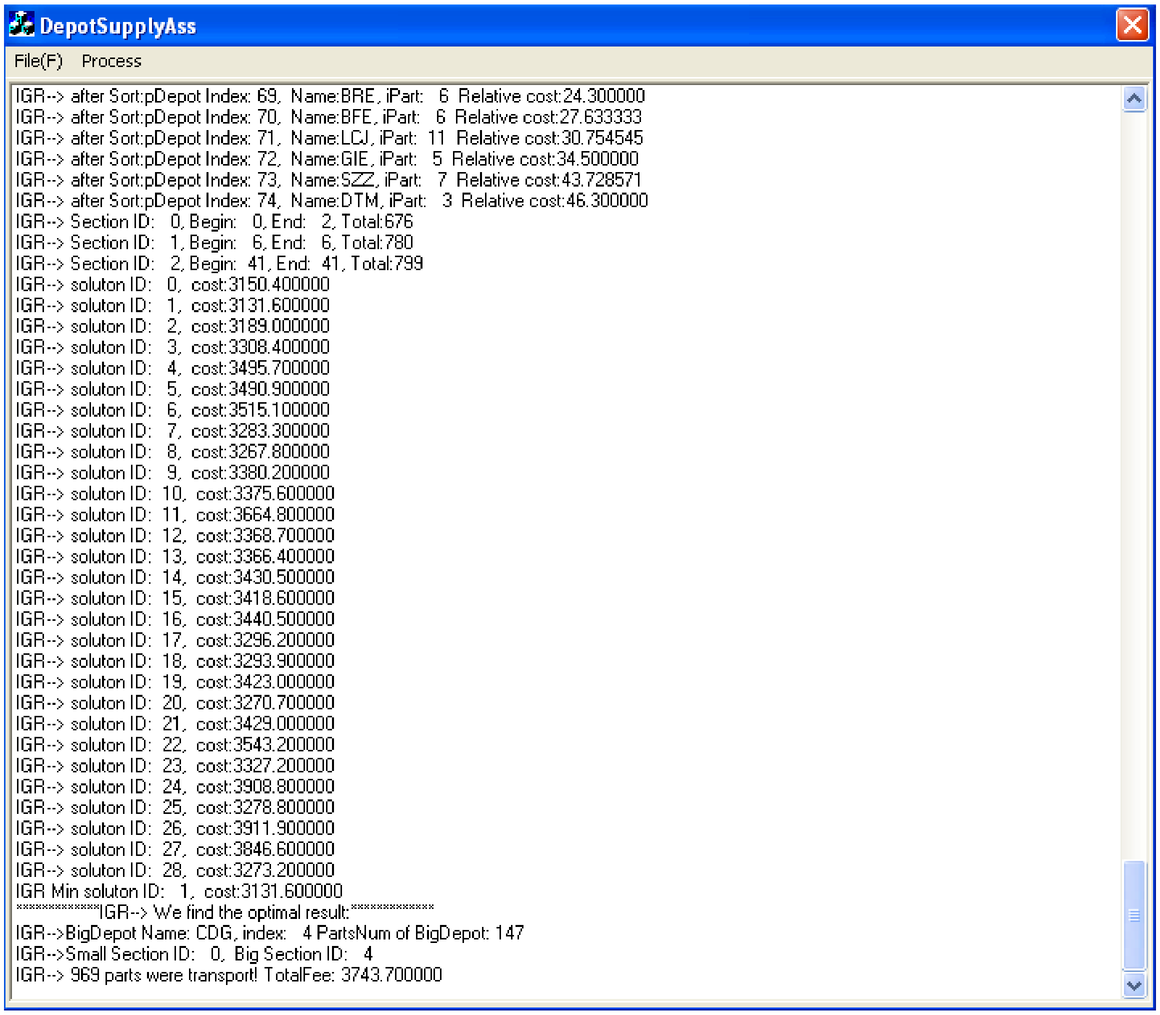
```
soluton ID:   5,  cost:3828.500000
soluton ID:   6,  cost:3641.700000
soluton ID:   7,  cost:3865.400000
soluton ID:   8,  cost:3641.500000
soluton ID:   9,  cost:3819.200000
soluton ID:  10,  cost:3616.800000
soluton ID:  11,  cost:4009.200000
soluton ID:  12,  cost:3720.000000
soluton ID:  13,  cost:3717.700000
soluton ID:  14,  cost:3792.700000
soluton ID:  15,  cost:3607.600000
soluton ID:  16,  cost:3837.500000
soluton ID:  17,  cost:3798.700000
soluton ID:  18,  cost:3781.800000
soluton ID:  19,  cost:3796.400000
soluton ID:  20,  cost:3619.700000
soluton ID:  21,  cost:3683.100000
soluton ID:  22,  cost:3636.000000
soluton ID:  23,  cost:3637.000000
soluton ID:  24,  cost:4207.100000
soluton ID:  25,  cost:3671.600000
soluton ID:  26,  cost:3766.600000
soluton ID:  27,  cost:3721.800000
soluton ID:  28,  cost:4237.900000
soluton ID:  29,  cost:3742.100000
soluton ID:  30,  cost:3619.900000
soluton ID:  31,  cost:4008.900000
soluton ID:  32,  cost:3617.600000
soluton ID:  33,  cost:3618.600000
soluton ID:  34,  cost:3750.500000
soluton ID:  35,  cost:4033.900000
soluton ID:  36,  cost:4136.300000
soluton ID:  37,  cost:4085.200000
soluton ID:  38,  cost:3606.400000
soluton ID:  39,  cost:3810.800000
soluton ID:  40,  cost:3844.100000
soluton ID:  41,  cost:3844.100000
soluton ID:  42,  cost:3608.500000
soluton ID:  43,  cost:3718.300000
Min soluton ID:  38,  cost:3606.400000
*************We find the optimal result:*************
Small Section ID:   0,  Big ID:  58
 958 parts were transport! TotalFee: 3606.400000
```

In week 8 of 2008, the target amount is 950-1050; the input data file is the excess parts information of week 6/2008



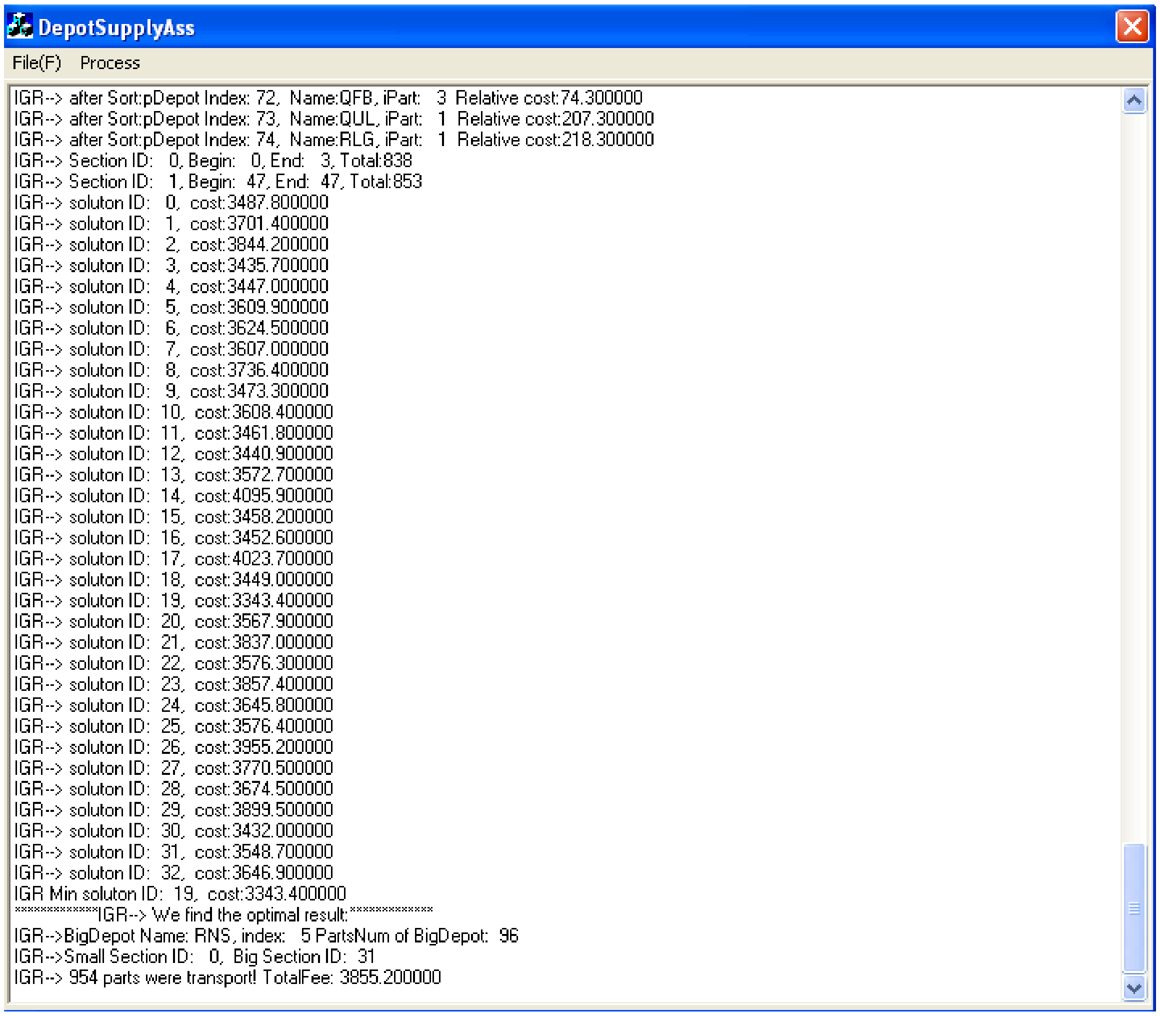
```
IGR--> soluton ID:  12,  cost:3317.300000
IGR--> soluton ID:  13,  cost:3432.000000
IGR--> soluton ID:  14,  cost:3415.900000
IGR--> soluton ID:  15,  cost:3698.200000
IGR--> soluton ID:  16,  cost:3406.700000
IGR--> soluton ID:  17,  cost:3477.700000
IGR--> soluton ID:  18,  cost:3463.500000
IGR--> soluton ID:  19,  cost:3485.400000
IGR--> soluton ID:  20,  cost:3341.100000
IGR--> soluton ID:  21,  cost:3341.100000
IGR--> soluton ID:  22,  cost:3315.600000
IGR--> soluton ID:  23,  cost:3390.600000
IGR--> soluton ID:  24,  cost:3465.600000
IGR--> soluton ID:  25,  cost:3473.900000
IGR--> soluton ID:  26,  cost:3588.100000
IGR--> soluton ID:  27,  cost:3329.600000
IGR--> soluton ID:  28,  cost:3951.400000
IGR--> soluton ID:  29,  cost:3799.400000
IGR--> soluton ID:  30,  cost:3323.700000
IGR--> soluton ID:  31,  cost:3900.700000
IGR--> soluton ID:  32,  cost:3312.200000
IGR--> soluton ID:  33,  cost:3448.700000
IGR--> soluton ID:  34,  cost:3426.500000
IGR--> soluton ID:  35,  cost:3686.400000
IGR--> soluton ID:  36,  cost:3825.300000
IGR--> soluton ID:  37,  cost:3820.700000
IGR--> soluton ID:  38,  cost:3256.200000
IGR--> soluton ID:  39,  cost:3537.700000
IGR--> soluton ID:  40,  cost:3858.100000
IGR--> soluton ID:  41,  cost:3495.200000
IGR--> soluton ID:  42,  cost:3622.200000
IGR--> soluton ID:  43,  cost:3425.800000
IGR--> soluton ID:  44,  cost:3292.900000
IGR--> soluton ID:  45,  cost:3407.300000
IGR--> soluton ID:  46,  cost:3598.900000
IGR--> soluton ID:  47,  cost:3507.800000
IGR--> soluton ID:  48,  cost:3505.500000
IGR--> soluton ID:  49,  cost:3476.600000
IGR Min soluton ID:   3,  cost:3103.200000
*************IGR--> We find the optimal result:*************
IGR-->BigDepot Name: QKA, index:  15 PartsNum of BigDepot:  99
IGR-->Small Section ID:   0,  Big Section ID:   6
IGR--> 953 parts were transport! TotalFee: 3566.900000
```

In week 9 of 2008, the target amount is 950-1050; the input data file is the excess parts information of week 7/2008



```
IGR--> after Sort:pDepot Index: 69, Name:BRE, iPart:  6 Relative cost:24.300000
IGR--> after Sort:pDepot Index: 70, Name:BFE, iPart:  6 Relative cost:27.633333
IGR--> after Sort:pDepot Index: 71, Name:LCJ, iPart: 11 Relative cost:30.754545
IGR--> after Sort:pDepot Index: 72, Name:GIE, iPart:  5 Relative cost:34.500000
IGR--> after Sort:pDepot Index: 73, Name:SZZ, iPart:  7 Relative cost:43.728571
IGR--> after Sort:pDepot Index: 74, Name:DTM, iPart:  3 Relative cost:46.300000
IGR--> Section ID:   0, Begin:   0, End:   2, Total:676
IGR--> Section ID:   1, Begin:   6, End:   6, Total:780
IGR--> Section ID:   2, Begin:  41, End:  41, Total:799
IGR--> soluton ID:   0,  cost:3150.400000
IGR--> soluton ID:   1,  cost:3131.600000
IGR--> soluton ID:   2,  cost:3189.000000
IGR--> soluton ID:   3,  cost:3308.400000
IGR--> soluton ID:   4,  cost:3495.700000
IGR--> soluton ID:   5,  cost:3490.900000
IGR--> soluton ID:   6,  cost:3515.100000
IGR--> soluton ID:   7,  cost:3283.300000
IGR--> soluton ID:   8,  cost:3267.800000
IGR--> soluton ID:   9,  cost:3380.200000
IGR--> soluton ID:  10,  cost:3375.600000
IGR--> soluton ID:  11,  cost:3664.800000
IGR--> soluton ID:  12,  cost:3368.700000
IGR--> soluton ID:  13,  cost:3366.400000
IGR--> soluton ID:  14,  cost:3430.500000
IGR--> soluton ID:  15,  cost:3418.600000
IGR--> soluton ID:  16,  cost:3440.500000
IGR--> soluton ID:  17,  cost:3296.200000
IGR--> soluton ID:  18,  cost:3293.900000
IGR--> soluton ID:  19,  cost:3423.000000
IGR--> soluton ID:  20,  cost:3270.700000
IGR--> soluton ID:  21,  cost:3429.000000
IGR--> soluton ID:  22,  cost:3543.200000
IGR--> soluton ID:  23,  cost:3327.200000
IGR--> soluton ID:  24,  cost:3908.800000
IGR--> soluton ID:  25,  cost:3278.800000
IGR--> soluton ID:  26,  cost:3911.900000
IGR--> soluton ID:  27,  cost:3846.600000
IGR--> soluton ID:  28,  cost:3273.200000
IGR Min soluton ID:   1,  cost:3131.600000
*************IGR--> We find the optimal result:*************
IGR-->BigDepot Name: CDG, index:   4 PartsNum of BigDepot: 147
IGR-->Small Section ID:   0, Big Section ID:   4
IGR--> 969 parts were transport! TotalFee: 3743.700000
```

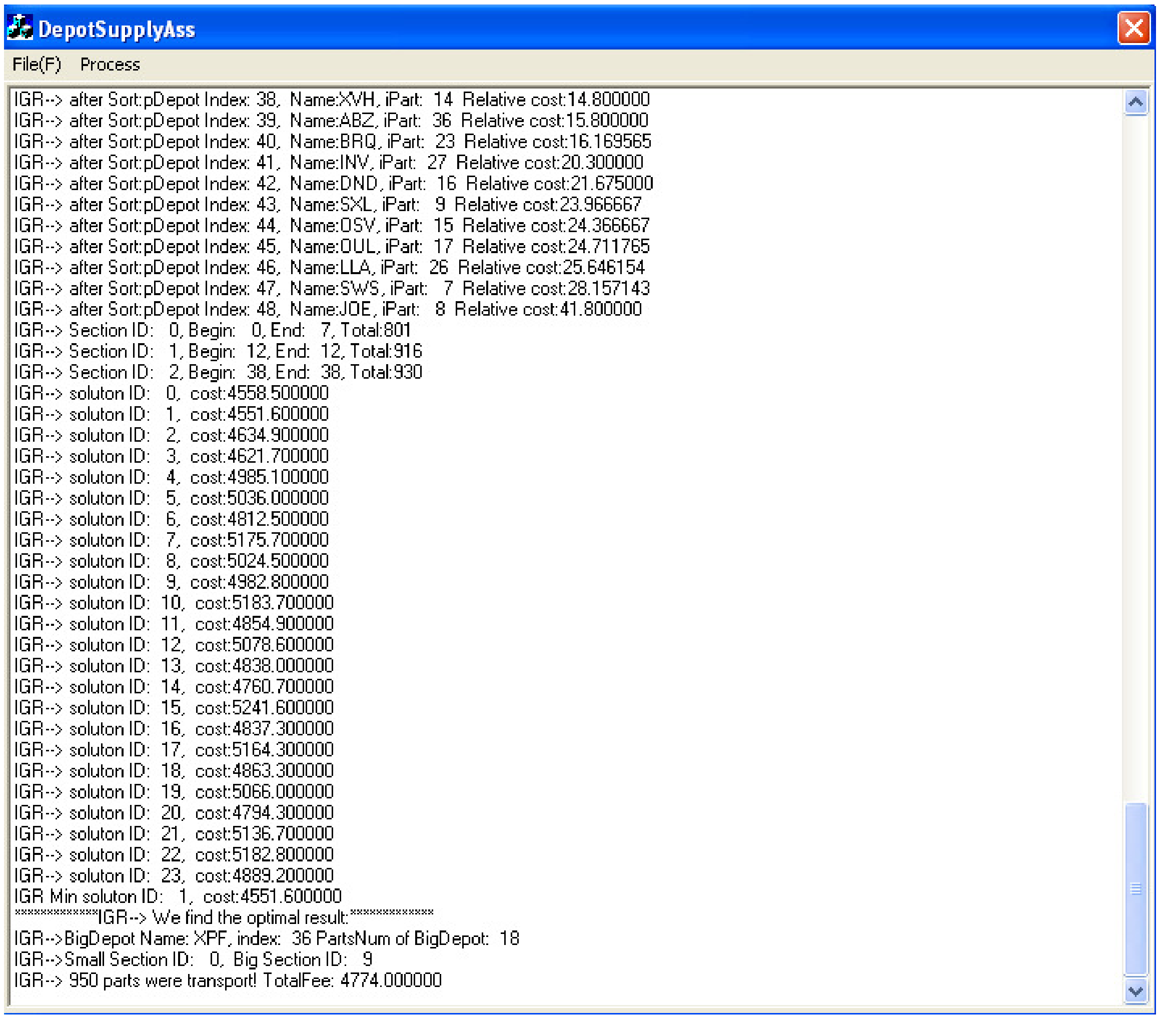
In week 10 of 2008, the target amount is 950-1050; the input data file is the excess parts information of week 8/2008


DepotSupplyAss

File(F) Process

```
IGR--> after Sort:pDepot Index: 72,  Name:QFB, iPart:   3  Relative cost:74.300000
IGR--> after Sort:pDepot Index: 73,  Name:QUL, iPart:   1  Relative cost:207.300000
IGR--> after Sort:pDepot Index: 74,  Name:RLG, iPart:   1  Relative cost:218.300000
IGR--> Section ID:   0, Begin:   0, End:   3, Total:838
IGR--> Section ID:   1, Begin:  47, End:  47, Total:853
IGR--> soluton ID:   0,  cost:3487.800000
IGR--> soluton ID:   1,  cost:3701.400000
IGR--> soluton ID:   2,  cost:3844.200000
IGR--> soluton ID:   3,  cost:3435.700000
IGR--> soluton ID:   4,  cost:3447.000000
IGR--> soluton ID:   5,  cost:3609.900000
IGR--> soluton ID:   6,  cost:3624.500000
IGR--> soluton ID:   7,  cost:3607.000000
IGR--> soluton ID:   8,  cost:3736.400000
IGR--> soluton ID:   9,  cost:3473.300000
IGR--> soluton ID:  10,  cost:3608.400000
IGR--> soluton ID:  11,  cost:3461.800000
IGR--> soluton ID:  12,  cost:3440.900000
IGR--> soluton ID:  13,  cost:3572.700000
IGR--> soluton ID:  14,  cost:4095.900000
IGR--> soluton ID:  15,  cost:3458.200000
IGR--> soluton ID:  16,  cost:3452.600000
IGR--> soluton ID:  17,  cost:4023.700000
IGR--> soluton ID:  18,  cost:3449.000000
IGR--> soluton ID:  19,  cost:3343.400000
IGR--> soluton ID:  20,  cost:3567.900000
IGR--> soluton ID:  21,  cost:3837.000000
IGR--> soluton ID:  22,  cost:3576.300000
IGR--> soluton ID:  23,  cost:3857.400000
IGR--> soluton ID:  24,  cost:3645.800000
IGR--> soluton ID:  25,  cost:3576.400000
IGR--> soluton ID:  26,  cost:3955.200000
IGR--> soluton ID:  27,  cost:3770.500000
IGR--> soluton ID:  28,  cost:3674.500000
IGR--> soluton ID:  29,  cost:3899.500000
IGR--> soluton ID:  30,  cost:3432.000000
IGR--> soluton ID:  31,  cost:3548.700000
IGR--> soluton ID:  32,  cost:3646.900000
IGR Min soluton ID:  19,  cost:3343.400000
**************IGR--> We find the optimal result:**************
IGR-->BigDepot Name: RNS, index:   5 PartsNum of BigDepot:  96
IGR-->Small Section ID:   0,  Big Section ID:  31
IGR--> 954 parts were transport! TotalFee: 3855.200000
```

In week 11 of 2008, the target amount is 950-1050; the input data file is the excess parts information of week 9/2008



```
IGR--> after Sort:pDepot Index: 38,  Name:XVH, iPart:  14  Relative cost:14.800000
IGR--> after Sort:pDepot Index: 39,  Name:ABZ, iPart:  36  Relative cost:15.800000
IGR--> after Sort:pDepot Index: 40,  Name:BRQ, iPart:  23  Relative cost:16.169565
IGR--> after Sort:pDepot Index: 41,  Name:INV, iPart:  27  Relative cost:20.300000
IGR--> after Sort:pDepot Index: 42,  Name:DND, iPart:  16  Relative cost:21.675000
IGR--> after Sort:pDepot Index: 43,  Name:SXL, iPart:   9  Relative cost:23.966667
IGR--> after Sort:pDepot Index: 44,  Name:OSV, iPart:  15  Relative cost:24.366667
IGR--> after Sort:pDepot Index: 45,  Name:OUL, iPart:  17  Relative cost:24.711765
IGR--> after Sort:pDepot Index: 46,  Name:LLA, iPart:  26  Relative cost:25.646154
IGR--> after Sort:pDepot Index: 47,  Name:SWS, iPart:   7  Relative cost:28.157143
IGR--> after Sort:pDepot Index: 48,  Name:JOE, iPart:   8  Relative cost:41.800000
IGR--> Section ID:   0, Begin:   0, End:   7, Total:801
IGR--> Section ID:   1, Begin:  12, End:  12, Total:916
IGR--> Section ID:   2, Begin:  38, End:  38, Total:930
IGR--> soluton ID:   0,  cost:4558.500000
IGR--> soluton ID:   1,  cost:4551.600000
IGR--> soluton ID:   2,  cost:4634.900000
IGR--> soluton ID:   3,  cost:4621.700000
IGR--> soluton ID:   4,  cost:4985.100000
IGR--> soluton ID:   5,  cost:5036.000000
IGR--> soluton ID:   6,  cost:4812.500000
IGR--> soluton ID:   7,  cost:5175.700000
IGR--> soluton ID:   8,  cost:5024.500000
IGR--> soluton ID:   9,  cost:4982.800000
IGR--> soluton ID:  10,  cost:5183.700000
IGR--> soluton ID:  11,  cost:4854.900000
IGR--> soluton ID:  12,  cost:5078.600000
IGR--> soluton ID:  13,  cost:4838.000000
IGR--> soluton ID:  14,  cost:4760.700000
IGR--> soluton ID:  15,  cost:5241.600000
IGR--> soluton ID:  16,  cost:4837.300000
IGR--> soluton ID:  17,  cost:5164.300000
IGR--> soluton ID:  18,  cost:4863.300000
IGR--> soluton ID:  19,  cost:5066.000000
IGR--> soluton ID:  20,  cost:4794.300000
IGR--> soluton ID:  21,  cost:5136.700000
IGR--> soluton ID:  22,  cost:5182.800000
IGR--> soluton ID:  23,  cost:4889.200000
IGR Min soluton ID:   1,  cost:4551.600000
**************IGR--> We find the optimal result:**************
IGR-->BigDepot Name: XPF, index:  36 PartsNum of BigDepot:  18
IGR-->Small Section ID:   0,  Big Section ID:   9
IGR--> 950 parts were transport! TotalFee: 4774.000000
```