Erasmus University Rotterdam

MSc in Maritime Economics and Logistics

2009/2010

Ship Spare Parts Network Optimization: the case of Golden Cargo

By

Nikolaos Filopoulos

Acknowledgements

First I would like to thank Mr. Chris Bravos, operation manager of Golden Cargo, who was showed a great interest in my idea and was every day available this summer to answer my questions and provide me with the necessary data even at his. Also I have to thank the rest of the personnel of Golden Cargo for dedicating a lot of their valuable time to explain me all the details I needed to complete this thesis and for making me feel like a part of the company.

I would also like to give many thanks to my supervisor, Professor Rommert Dekker, who assisted me with his remarks and added quality to my thesis. Without his help this summer would have been more difficult.

I want to thank the MEL staff and my classmates for the remarkable year we spent together. The experience was unforgettable and I am very happy I joined the MEL program.

Also I want to thank my dear friend, Kostanstinos Tsakalozos, whose knowledge in JAVA helped me deal with numerous difficulties and I wish him every success in finalizing his PhD studies soon.

Last but not least I would like to thank my family for their support the past year as well as for all the previous years that I was studying. It was as stressful for them as it was for me and I was grateful to have them next to me each and every moment.

Abstract

Every company in the logistics sector targets on having an efficient network, which guarantees on time deliveries with a reasonable cost. As the companies expanded and now offer worldwide coverage their networks became extensive and rather complex. The old manual methods of network optimization must now be changed with modern ones, which fully utilize the vast computational capabilities of the modern computers. In the big networks we have to deal we cannot use exhaustive algorithms to find a solution as it would take a massive amount of time to produce results. The effectiveness of a network is becoming more crucial for a company day by day.

As a test case we used Golden Cargo Logistics, one of the leading companies in ships spare parts supply. The network of the company has been researched in order for its specific attributes to be identified and to give us an insight of how we should deal with them. The network has worldwide coverage and is responsible for the distribution of a considerable amount of spare parts. This thesis tries to provide a solution by using a different than the usual methods, the method of simulated annealing. The history of the method is presented together with some problems it has been used for. The implementation of the method presents different difficulties in every problem. Then we used the method to optimize a network according to the company's operational schedule. It is a smaller than the original network will be but this fact allowed us to further research different solutions in a limited amount of time. The program that was created produces a complete plan of how the packages of the company should be routed. Finally, the results of the implementation of the method are presented.

Table of contents

1. Introduction

The spare part network is a very important part of the supply chain. The shipping industry has shown tremendous growth and has become more complex, making the availability of the spare parts even more critical than in the past. There are many parties involved and there various many difficulties in making it work smoothly and cost effectively. But what defines the efficiency of a spare part network? First of all it needs to supply the necessary demand, ideally in every situation. Lack of timely support or an incomplete support will probably cause unexpected delays, which will then lead to losses. The losses can rarely be compensated and the result is another added cost to the procedure/production. If this was the only concern then spare parts supply networks would be easy to design.

The other great problem is the cost of operating such a network. Many shipping companies have grown to large multinational corporations which offer worldwide coverage. There are more than 53000 ships operating in the seas around the globe and all of them have specific needs. The operating environment is not gentle at with all the mechanical parts and the conditions often cause failures. In all this we have to add the scheduled maintenance. It is obvious that there is a crucial need for an efficient network in order to supply the ships. The modern container and bulk carrier terminals have made this need even more important. They help the ships to minimize their port time, as the idleness of the ship costs money without generating any. This fact provides the logistic companies with even less time to supply the necessary equipment to the ships. No one wants to be responsible for delaying a 15.000 TEU container vessel or a 300.000 dwt tanker.

We will study this problem from the viewpoint of Golden Cargo, a leading company in the logistics and forwarding business. The company specializes among others in the ship spare parts delivery network. They have a client base of more than 2.000 ships, which makes them one of the biggest companies of the kind in the world, if not the biggest. The network and the orders of the company will be studied, alongside with their operational procedures. The company's objective is to satisfy its customers alongside with maintaining a low operational cost. Every day it receives dozens, maybe even hundreds, of orders from around the globe. Then by activating its service network it has to deliver the parts at their destinations inside the requested time frame.

This task would not be a problem in a smaller sized company. The large amount of deliveries combined with the worldwide network greatly increases the complexity. For the time being the orders are handled manually. This requires a number of experienced and dedicated personnel and a lot of working hours. Also, taking into account the human factor means that the proposed solutions are not always optimal. In fact the company has realized that maintaining this type of operating procedures would result in the long term in serious problems.

In this thesis we will attempt to present an automated solution for the forwarding process. The goal is to provide a plan of how the shipments should be routed, generated by a computer program. The program, using the computational power of modern computers will be able to provide a more efficient solution, quicker than the company's personnel.

We will try to answer the following research question:

-How can the ship spare part supply network of the company be optimized?

The supply network of the company deals mainly with ship spare parts. The spare parts are considered as packages.

This main research question can be broken down into different branches:

-How should each specific package be routed?

A package can be send directly from its origin to its destination. Alternatively it can wait to the origin until another package for the same destination appears. Another option is to ship it to another airport first, which is serving as a hub and then to its destination

-Should the company consolidate certain packages into one shipment?

Cargo consolidation means that more than one package will be shipped together. This is done to benefit from economies of scale and achieve a better freight rate.

-Which airports should be used as hubs and for which package?

We need to decide which packages should be send directly and which will use hubs. For those using hubs we must find the one that serves our purposes better.

In our effort to answer these questions we must keep in mind certain other elements. The solution we will propose must satisfy the following basic requirements:

-Every package needs to arrive on time

-The total cost must be smaller than without using any hubs

Also it is important to mention that the company does not influence the demand at all. It is the shipping companies that define it and the company has to follow it.

There are numerous factors that need to be taken into account in modeling the problem. In order to keep the case smaller and easier to test a few assumptions have been made which will be described in the relevant chapter.

A big part of this thesis is the optimization program we have built. We needed this program to make the decisions for the routing of the packages. First we studied the company's network in order to find its distinctive attributes. Then the optimization program will be built. In order to build it we need to use a decision making process. We chose to use the simulated annealing method. The method is fully described in the relevant chapter.

1.1. Contents

In chapter 2, a description of the operations and the provided services of Golden Cargo, is made. Also the forwarding process and an example of the consolidation service of the company is given.

In chapter 3 the network of the company is described. The different parts we used to model it are mentioned. Also the objective of the thesis is described in more detail and with an example. Chapter 4 contains a review of the method we will use to optimize the network. The reasons we chose it together with the advantages, disadvantages and a list of other problems using the same method are provided.

Further on, in chapter 5 we can see a description of the program we used. The input and the output of the program are mentioned alongside with the main functions of it. In chapter 6 we discuss the results of the optimization program.

Finally, in chapter 7 the conclusion of the research and possible paths for further research are described.

2. Golden Cargo

In this chapter the company for which this study was made will be presented. This will provide a better understanding of the complexities of the forwarding – logistics processes. It will also help the readers identify the problems and the solutions that will be recommended in the following chapters. The information on this chapter comes from the following sources: interviews with the company's personnel, operating manuals, the company's website, as well as from on site research during this summer.

2.1. Company description

Golden Union Shipping is a company based on Piraeus, Greece, founded in 1977. In its early days the company mostly carried bulk cement to facilitate the great demand of the Middle East and West Africa. The company converted several bulk carriers to cement carriers and was one of the pioneers in the pneumatic conveying of the cement. The company used its expertise and its continuous growth and entered different market segments. Traditional cargo ships were bought and the existing fleet was renewed with larger units. Presently the company has achieved a dominant position in dry cargo operators and is oriented in building panama and larger cape size dry bulk carriers.

Throughout the years the company developed a logistics department to facilitate the growing needs of its own fleet. Together with the cooperation of the technical department, the logistics experts were responsible for the warehouses and the inventory of the company. As the fleet was expanding it was becoming more difficult to supply the ships, with the necessary resources, both fast and with the minimum cost. During the 90's the company was very happy with the way the logistics department was operating and decided to expand it. By expanding it they used the already available expertise, network and infrastructure and offered the same service to other companies.

This was the beginning of the company Golden Cargo S.A. In the beginning it served only Golden Union's fleet and then it entered the free market of the logistics area. Golden Cargo S.A. attracted new customers quickly and since its creation has kept growing. It has developed a fast and secure airfreight ship spare parts delivery system with worldwide coverage. In order to avoid problems and potential delays, they move each shipment directly from the supplier facility to the destination place without any 3rd part interference. Through the worldwide network they can offer practically unlimited weight and no size restrictions. In addition the client is continuously informed about the present status of his shipment through an online portal. Today Golden Cargo S.A. has a client base of more than 2000 vessels.

2.2. Provided services

The company is able to handle any type of cargo the clients may need. The services include ocean, air and road freights, S.O.S logistics and warehousing. In more details:

2.2.1. Ocean Freights

The worldwide network of offices and networks assures a reliable door to door service both for LCL (less than container load) traffic and full container units. The company is ready to provide:

- Inland Freight
- Booking Arrangements
- Warehouse and storage
- Pier inspection of cargo
- Total documentation including banking and consular
- Cargo insurance
- Cargo expediting
- Export crating and packing
- Arranging part and full charters
- Custom clearance and delivery to the consignee

2.2.2. Air freights

The company also offers air freights services. In many cases this service can be combined with the sea freights

The company's appropriate department works with the customer to develop and implement systems in their effort to provide:

- SEA / AIR services on more than 20 routes
- Door-to-door express services
- Regular schedule service to major world markets
- Committed space allocation with the major international airlines for reliable services
- Assembly and distribution programs
- Expertise in foreign banking, consular work and document preparation
- IATA agency service for shipments to destination where special circumstances apply
- Experience in the proper handling of dangerous goods shipments
- Arranging part or full charters

2.2.3. Road freight

The network of the company covers all Europe and maintains groupage and full truck services to all main cities / ports. The services include:

- Special truck services
- Groupage truck services
- Warehousing and storage
- Total documentation service
- Custom Clearance
- Delivery to final destination

2.2.4. S.O.S. Logistics

The company has given this name to the department that handles the urgent deliveries. This kind of cargo is the most difficult to handle as a potential delay would cost a lot of money to the customer. The procedure that is followed is different than the one followed for the normal cargo. The main objective is the fast delivery and not the minimum cost. Ships spares logistics terminals are in operation at the following countries:

- Tokyo – Japan
- Rotterdam – Netherlands
- Houston – USA
- Singapore
- Seoul/Pusan - S. Korea
- Piraeus – Greece

The company provides the following services:

- Follow up of spare parts orders via the owned Golden's Logistic Portal (www.gco.gr)
- Weekly inventory reports
- Assembly and distribution programs
- Regularly scheduled service to major shipping ports
- Export crating and packing
- Customs clearance and delivery on board the vessel
- Sea / Air service on more than 20 routes
- Experience in foreign banking , V.A.T exception documents preparation (T-document, ex-el)
- Door-to-door express service

2.3. Aim of the company

The aim of the company is to maintain low and consistent costs for the provided services

It is organized to provide a competitive and comprehensive service, and to ensure that goods and spare parts will be transported across the globe in time and intact. The company's aim is to maintain low and consistent costs for the services provided. With their modern operating procedures they do not to charge according to the predicament of the client or by competing with the expensive alternative of a delayed ship. Experienced personnel works under modern organization and operating principles with excellent results. At the main hubs the company maintains trained and experienced personnel that clears and forwards arriving or departing shipments and deals with any problems that might arise. Exclusive customs clearance agents are also employed to avoid any inconvenience. Also very important are the information mechanisms developed in cooperation with the customers, which are above industry standards. Golden Cargo is a responsible and responsive forwarder who satisfies the needs of the demanding clientele. Nearly all the well-known shipping groups are included in the list of clients.

The company also tries to comply with the highest quality standards. The quality monitoring system consists of a Global Quality Manager cooperating with Country Quality Managers, who constantly monitor service levels at every branch. The service network has achieved ISO9002 certification in most operating countries. The objective is to achieve the specific certification around the world to the whole network. In addition the company is committed to environmental preservation and has attained the ISO4001 certification in Greece.

The quality monitoring procedure can be seen in the following graph:

Golden Cargo picks up and/or receives goods and expedites their transportation and delivery by arranging for warehousing and shipping. It also offers door-to-door delivery services (in wholes or partially). Golden Cargo's expertise in ship spares logistics - handling services on a worldwide basis provides tailor made global services to shipping managers - operators.

2.4. The Company Culture

The company uses the motto Thinking Ahead - Moving Forward. In a few words this means evolution of the logistics operations beyond industry standards in order to provide fully customized, Just in Time deliveries on spot. Exceeding the expectations of the customers in terms of speed and security of delivery is considered by the management as the standard way things should be done. This mentality is passed from the top management to the entire company's personnel. The employees, motivated by the management's commitment, have a high respect of the company culture and focus on the excellence of their performance beyond standards.

2.5. The forwarding process

The whole procedure starts when the captain of a ship realizes there is a need for a spare part. Then he assesses the situation and informs his company about the seriousness of it and for the necessary spare parts. He is the one who makes the decision if the ship is able to sail or not. The shipping company then contacts the appropriate supplier and orders the part. According to the captains reports the company informs the supplier for the urgency of the demand. So far Golden Cargo is not even aware of the situation. It is only after it has been informed by the shipping company that it contacts the supplier of the part to arrange the pickup details. At this point the shipping company also informs Golden Cargo of the deadline it has to deliver the spare parts.

The forwarding process is described in more detail in the next page and the picture that follows:

1. Customer's Purchasing Department proceeds to the Order of a spare part.

   The supplier is instructed to release/deliver the spare upon readiness, to the company's warehouse at the country of origin and send a copy of the Purchasing Order Note (P.O.). A corresponding partner of the network in the country of origin will act as instructed.

2. Upon Receipt of the Purchasing Order Note

   The company's partner immediately contacts the supplier to assess the status and the date of readiness of the spare and provides with a feedback accordingly

3. The spare/s is being delivered/collected

   The partner immediately proceeds to entering the delivery in the IT system, in which the client has access any time. There he is able to see all the details of the order description, weight, dimensions etc. The order is now ready for dispatch

4. The vessel is calling the port

   Instructions are expected from the client in order to forward the spares in concern to the vessel. Specifically the information needed is:

   a. Exact port of calling and nearest airport (if known)
   b. Vessel's expected ETA
   c. Full style address of consignee or/ and vessel's agent

5. Track and Trace

   The company's partner subsequently activates the tracking mechanisms within the Global Network Warehouses to spot the spares for the specific vessel. Then he arranges to forward the shipment in the most economic and safest manner within time limits.

6. Upon release of the shipment

   All shipping details and documents are sent in the vessel's agent at the destination in concern if requested, and a document's copy is also provided to the ship owner.

7. Company monitors every shipment

   Throughout the whole operating procedure from Door to Airport or On Board delivery the company monitors the status of the shipments. If any deviation or change happens it is immediately reported to all parties involved

The forwarding process
From the customer to the vessel

Issues the P.O Note

Calls to order the spare

Supplier of the spare

Calls to arrange pick up details

Sends copy of the P.O Note & Invoice

Agent of country

Warehouse

Trucking Freight

The ship opt/customer

E-mails pick up order
Copy of P.O.

Feedback

Feedback

Delivery on Board (D.O.B.)

Calls for f/w process
Sends copy of P.O., M/V Name, Supplier's name, Order No., size, weight etc.

G.C.O- SOS WWL Administrator

2.6. Consolidation service

In the past the shipping companies used to purchase spare parts from various suppliers and perform individual forwardings per purchase. For each shipment there was a separate Air Way Bill (AWB). Each AWB had to be cleared separately by the customs. This fact cased various problems in the forwarding procedure. Often it led to excessive delivery charges. Thus the company could not offer competitive prices. This led to losing customers and making it more difficult to attract new ones. Another problem was the delays on the arrivals of the parts. The customs offices often do not share the customers need for urgency and delay to clear the shipment. Despite the fact that this is not the company's fault still it is responsible for the delivery of the shipment and it may pay compensation to the customer. Because customs offices are a civil service it is virtually impossible to change the procedures they follow and to make them more efficient.

With the modernization and automatization of the port terminals the calling time for each vessel at a port has decreased significantly. As a result it is more crucial than in the past to for the spare parts to be delivered on time. Port time is idle for the ships and the companies try to minimize it as possible. Therefore it is very important for a logistics company to be able to deliver the necessary spare parts inside a tight time window. The wisest thing the company can do is to minimize the interaction it has with the customs office. The solution to this is to have as few shipments as possible. In a few words to consolidate the shipments into larger packages which need to pass through the customs only once. This procedure contains also some risks. In case a consolidated shipment is delayed for any reason, then the delayed packages would be more than one and the cost of the delay will rise substantially. On the other hand since it is only one shipment there are fewer probabilities that the shipment will be delayed.

Golden Cargo decided to offer a cargo consolidation service to minimize the costs and the risks of delays. This is even easier in Europe, as inside the E.U. countries no custom clearance is necessary. Consolidating several orders under only on airway bill reduces the risk and minimizes the cost of the delivery. Thus the company can offer a better service at even more competitive prices,

The decisions on which of the orders should be consolidated and where it should be done are being taken in the company's headquarters in Piraeus. Currently this process is made manually and its success depends entirely on the experience and the wits of the personnel. The company's continuous growth makes these decisions harder and harder every day. The number of the orders keeps increasing and accordingly increases the number of the suppliers. As the company has a worldwide service network it is easy to understand how it has become really complex to make the consolidating decisions.

A simple example of the consolidating procedure will be described in order to help the understanding of the process and the complexities it has. This example is taken from the operation manual of Golden Cargo We need to point out that the prices are not real and this is used only as an example of the consolidation service.

We are making an assumption that the following items have been purchased by the company's customers

- 25 kg ex Norway to Shanghai
- 15 kg ex Germany to Shanghai
- 35 kg ex Denmark to Shanghai

- 20 kg ex Netherlands to Shanghai

We will then ship the packages using two different scenarios. In the 1st all the items will be shipped individually and in the 2nd the consolidation service will be used. The scenarios are very simple but their purpose it to help the reader have a better understanding of what the designed program wants to do.

In the 1st scenario we will ship each item individually. The costs will be as following:

1st Scenario

1. 25 kg Norway – China	EUR 230 all in + pick up
2. 15 kg Germany – China	EUR 185 all in + pick up
3. 35 kg Denmark – China	EUR 235 all in + pick up
4. 20 kg Netherlands – China	EUR 210 all in + pick up
Total	EUR 860 up to Shanghai
	Airport + pick ups

In the total cost we should add up the customs clearance. This is estimated per Air Way Bill at USD 300. So we have in total 4x300 = USD 1200 ≈ EUR1000.

The grand total has risen to 1000 + 860 = EUR 1860.

In the 2nd scenario we will use the shipment consolidation. As a hub in the specific example we will use the Netherlands. The Netherlands due to their positioning in the heart of Europe and the dominant position of the port of Rotterdam is likely to be used as a hub in many occasions. Another reason for this are the fast custom procedures which further reassure the company that it is unlikely to face any delays, at least from the airports side.

2nd scenario

1. 25 kg Norway – Netherlands	EUR 115 all in + 75 handling +pick up
2. 15 kg Germany – Netherlands	EUR 115 all in + 55 handling + pick up
3. 35 kg Denmark – Netherlands	EUR 160 all in + 65 handling +pick up
4. 20 kg Netherlands – Netherlands	
Total	Handling only + pick up if any
	EUR 585 up to Rotterdam

The 4 items have a total weight of 95 kg. If we send them under only one airway bill the items will only be cleared once in China. Also we can put the shipment on the +100 Kg scale rate in order to enjoy most competitive prices.

So, we will pay:

- 100 Kg x 2.7 EUR/kg = EUR 270
- Handling fee = EUR 55
- Clearance fee = EUR 45
- Airline handling = EUR 20
- Clearance fee (China) = USD440 ≈ EUR330

The grand total is EUR 1305.

It is obvious that in the 2nd scenario we have to pay EUR 555 less. The charges in this example are of course exemplary as they continuously change. The conclusion still stands and if we checked with today's real prices we would still achieve great cost reduction. It is obvious now how important this service is for the company and why the consolidation decisions need to be correct.

2.7. IT program

The operations of the company are supported by a custom made portal (www.gco.gr). The portal is used by the employees of the company as well as by the customers. Through the portal the customers can acquire the necessary information about the status of their shipments. The users of the portal need to log in with a unique User ID and a password.

The portal offers the following options:

- Inquiry
- Instruction
- Order
- Offer
- Dispatch details
- Charges
- Reports
- Event

The client or the partner can see all the inquiries, instructions, orders etc. that are for/from the company. The company's users can see all the data in the system. In this tab the user can search and list all the records with all the available criteria. Also there is the option to add/delete a new record. If you want to add a record a number is automatically given and the user has to supply the rest of the data following the instructions.

The program that the company uses was designed when the company was created. According to the company, it is obsolete and it needs to be upgraded. This is not an easy procedure and it has a substantial cost, both for designing a new one as well as

training the employees to use it. So far they have not taken the decision to change it. The main problem is that it does not provide enough statistical data. Such information could help their planning, but unfortunately it was not taken into account during the initial design. So if we ask the company how many packages have been shipped from Rotterdam to Piraeus in 2009 the program cannot provide this information. It only counts the shipments between the 3 zones, in which the company has divided its network. This is the main reason we could not use statistical data in our program.

3. The network of the company

The company offers worldwide service, even in the most remote areas, through its affiliated partners. The main volume of the shipments is moved between Europe, the Far East and North America. We can divide the globe into 3 zones. The first zone is Europe. In Europe the main hubs of the company are Rotterdam in the Netherlands, Copenhagen in Denmark and Frankfurt in Germany. Of course there are more airports which are used for sending spare parts, but their usage is less than the ones mentioned before. The main ship engines manufacturers are still located in Europe on the contrary with the ship yards. This means that most of the spare parts have a European country as their origin.

In the second zone China, Korea, Japan and the countries of the Indian ocean are included. Most of the new ships are being built in these three countries. Also China has shown tremendous growth rates in the past decade and is expected to keep up like this. The huge size of the country and its way to modernization makes the need for trade bigger and bigger every day. This area is the most important in the ship trade business and it is exactly the same for the company's logistics procedures. The majority of the shipments have these countries as a destination. The third zone consists of North and South America

The company is responsible for delivering tens of thousands of shipments every year. If we combine the shipments to the worldwide destinations it offers, we end up with a very complex network. This network contains hundreds of hubs and thousands of different paths. In this chapter the problem this thesis is trying to solve will be defined.

During the 2nd half of the 20th century the planet witnessed several technological breakthroughs and remarkable growth. The shipping business was and still is one of the main drivers of this growth. Traditionally shipping was one of the most globalised industries. It was used for trade between distant places for millennia. Particularly nowadays, when we face globalization in every aspect of our life the role of shipping is even more important. The invention of the container together with the technological advancements in ship and engines design significantly lowered the sea transportation costs. This allowed the countries to supply their increasing demands of raw materials (oil, coal, iron ore) and commodities with lower prices than in the recent past.

Meanwhile global changes are taking place. Relocation has been used, by large firms in the beginning in order to meet internal sourcing goals. As relocation proved successful the medium sized companies followed. This led to an even bigger demand for transport. The big difference is that despite the larger volumes, the demand is fragmented and it involves more final customers, greater diversity in terms of origins and destinations and a greater variety of goods. Thus the role of suppliers of logistics services is becoming greater every day.

From the logistics provider's point of view the following issues have been raised (OECD 2005):

- A need to known under what conditions goods will be transported from the point of shipment to that of reception, regardless of the routes and modes of transport used; this expectation is particularly strong when outsourced products are delivered to the point of shipment(INCOTERM E or F), and is less pronounced when the supplier is able to deal with price conditions and delivery schedules himself (INCOTERMS D or C) involving a greater variety of goods, more final customers, and greater diversity in terms

- The desire to made shipments secure by giving priority to financial, customs or fiscal aspects and, in terms of transport, compliance with pick-up and delivery schedules, quality (conformity, no breakages), security (no losses), and therefore traceability (tracking/tracing).

- Less stringent requirements than major shipping agents in terms of freight rates, and the almost systematic use of external suppliers for the organization of transport and logistics

- A positive percent of price levels for maritime transport compared with those for inland transport, particularly when movements include a lengthy road segment, although there are negative reactions to fluctuations in freight rates due to surcharges

The whole logistics process has changed and the following trends have been identified:

- Markets are more fragmented and the orders appear to be less uniform, due to product differentiation, opposed to the mass production

- Purchase orders apply to a number products, each one of a different type and packaging, that are shipped together

- Smaller orders with a shorter delivery time. These orders are more constraining when it comes to punctuality and adjustment to fluctuations in the demand of the products

- The logistical support requirements are more demanding and the solutions need to be more sophisticated

In this challenging and changing environment Golden Cargo has to face some serious challenges. The target base of the company is virtually any ship that sails in the seas at the time being. According to the Lloyd's List Fairplay (July 2009), this number is approximately 53000 ships. Despite the big crisis of the last couple of years the trend is that the world fleet will continue to grow steadily in the next years. In the following map

the world's major shipping routes can be identified. We can see the major trading routes are between Asia, Europe and North America and are followed by Africa, Oceania and South America.

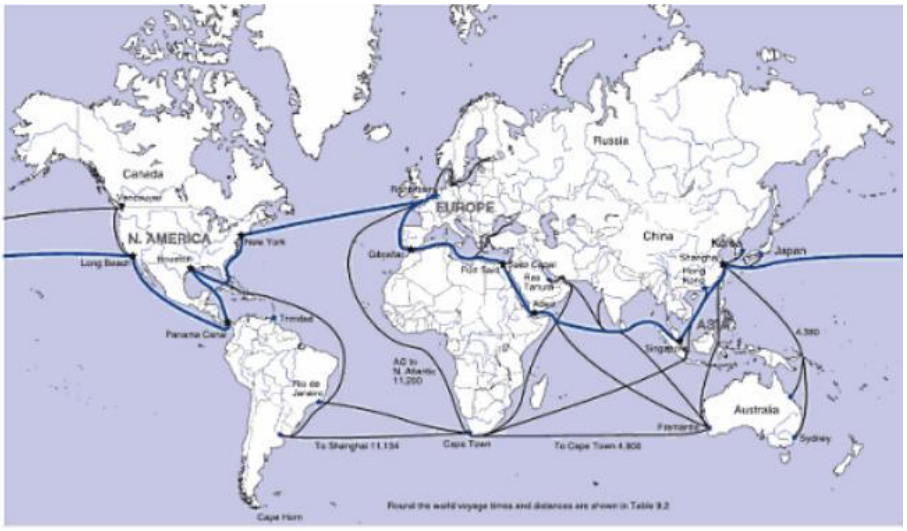
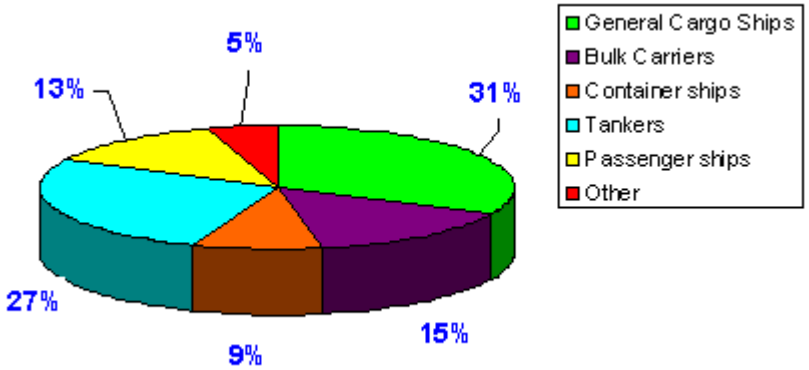


Figure 9.1
The world's major shipping routes, 2007
Source: Martin Stopford 2007

So the facts are the following: there are more than 50000 ships circling the globe, operating in an unfriendly environment under very harsh conditions. Moreover, with the advancements in the technology and the modernization of the terminals, the port time of the ships has significantly decreased, compared with the past. As a result the time window a ship has to perform the necessary maintenance or fix any damages that may occur is now limited. Port time is unproductive for ships and so everyone is trying to keep it limited.



Source: Lloyd's Register Fairplay July 2009

General Cargo ships	17,104
Bulk Carriers	7,787
Container ships	4,678
Tankers	14,095
Passenger ships	6,839
Other	2,502
TOTAL	53.305

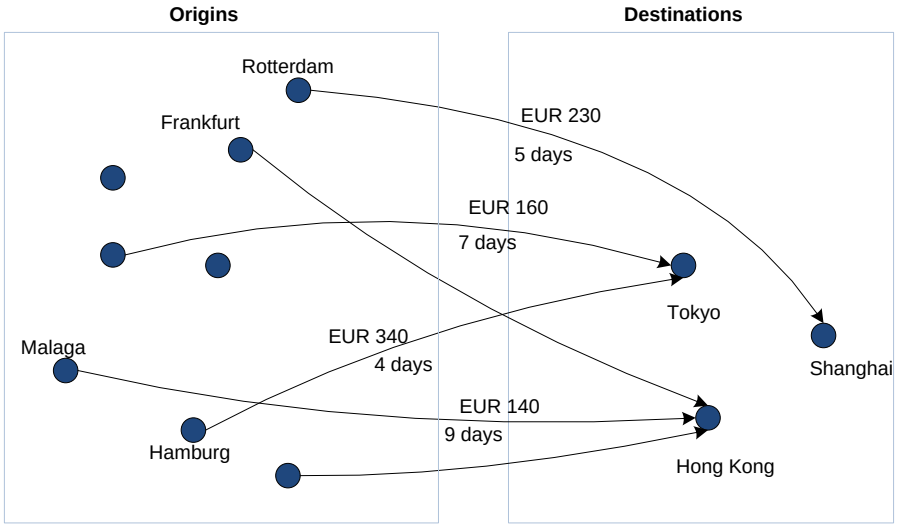
The need for a reliable network that will supply the ships with the necessary equipment is significant. The amount of the ships in the world fleet is what makes the challenge Golden Cargo has to face even bigger.

3.1. The network

Golden Cargo is a company that offers third-party logistics (3PL) services. It is important for the success of the company to design an efficient and effective logistics network. A logistics network consists of suppliers, manufacturing centers, warehouses, distribution centers, and retail outlets as well as channels for the flow of raw materials, work-in-process inventory, and finished products between the facilities (Simchi-Levi et al., 2000). An optimal network must be able to deliver the products to the customers at the minimum cost, while achieving the set service level requirements. There are various models one can use to optimize the network. The objective of each method is, as stated before the increase in the effectiveness of the network design and the utilization of the facilities, alongside with minimizing the relevant costs.

3.2. Definition of a network

The network can be described and defined with graphs. A graph G consists of a finite set V and an irreflexive binary relation on V. V is called the set of vertices. The binary relation on V can be represented as a collection E of ordered pairs or as a function from V to its power set. The ordered pair (v, w) € E is called an edge (Golumbic 1992). This is the strict mathematical definition of a graph. When dealing with network problems usually the term node instead of vertex and arc instead of edge is used. In our case study the graphs are directed. This means that there is a specific way you can travel from each node to another. The nodes are divided into two categories, the origins and the destinations. Certain nodes belong to both categories, they can be both origins and destinations. The nodes are connected between them with the arcs. Each arc is given a value. This value represents the “distance” we need to travel from the one node to the other. The “distance” is not confined to the actual distance but can also be time, cost etc. In the case study we used a smaller version of the company’s network. Despite the smaller size, this network has the same attributes and can be used as a test case. An example of how the network looks like is shown below:



The fact that each arc does not have a specific value is what makes the problem more complex and more difficult to deal with.

3.3. The package

As package we define each item the company has to move between two nodes. The package has the following attributes:

- Origin

  This is the place the package begins its journey. The company is responsible for receiving it there and does not know how the package will arrive to the destination

- Destination

  Destination is the place where the package needs to be delivered. We only consider the airport in which the package will arrive. In case it needs to be transported somewhere else, we consider it a different procedure and we do not take it into account here.

- Weight

  Each package has a specific weight. Te weight is used to calculate the freight rate of each destination. There are different scale rates according to the different weights.

- Deadline

  Every package has to reach its destination before a specific date. This is defined as the deadline of the package. It is needed in order to know when we change the route of the packages if it will still be at the destination on time.

3.4. The airport

We define as airports the nodes of the network. Each airport has the following attributes:

- Clearance

  As clearance we define the clearance fees that are charged for every package that arrives in each airport.

- Port handling

  The (air) port handling fees are charged to each package that is handled in the specific airport.

4.5. The connection

With the connection each arc of the network is described. Each connection has the following attributes:

- From

  This describes the airport of origin of each package

- To

  Here we state the destination of each package.

- Time

  This describes the time that a package needs to travel from the origin to the destination

- Air Handling

  This is the charge of the airline company for the handling of each package. The air handling fees are become higher as the weight of the package increases.

3.5. Objective

The objective is to find which the optimal connection for each package is. This means that, since we know the origin and the destination for each package we must find out if it would be better to add another airport, as a hub to the connection. We will add another airport in order to consolidate the packages. Thus we will have fewer shipments to deal with.

There are two main reasons to do this:

1. Better freight rates

   When we consolidate two or more packages then they are considered as one shipment. If the combined weight of the packages is enough, we can send the shipment with a lower freight rate. The airline companies offer better prices per kilo when the weight is higher

2. Less clearance fees

   Every shipment has to be cleared by the customs when it arrives at its destination. This costs both money and time, especially outside of the E.U. where customs agreements have not yet been implemented. When we consolidate the packages we wait for fewer shipments to be cleared. It is easier for the company to maintain its service level and also it saves a lot of money, as clearance in many countries is rather expensive.

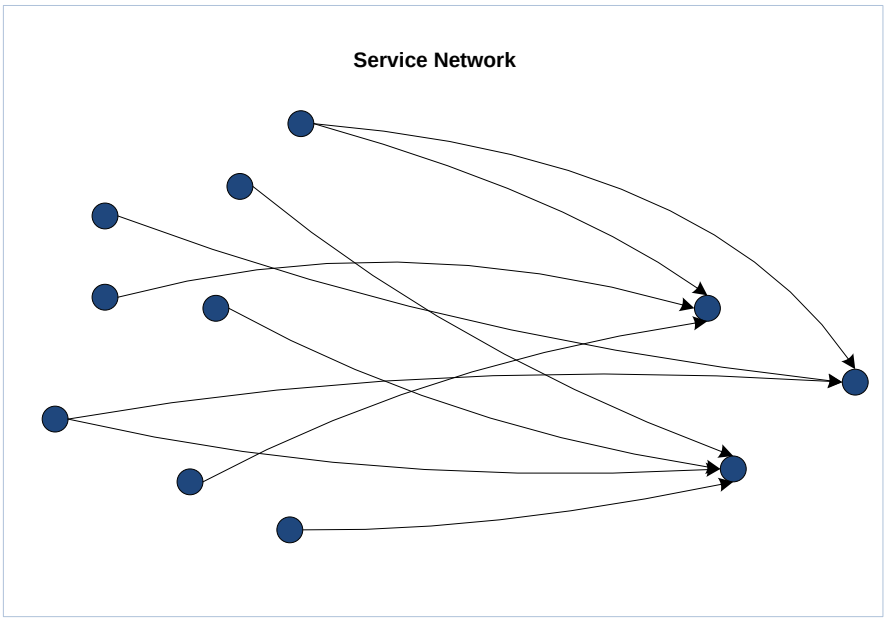
In order to make these decisions we must first see what data we have and what decisions we need to make. Our data come from the spare parts orders the company gets. We have the following information about every package: the origins, the destination, the weight and the deadline.

Then for each package we need to take a decision of how we will send each package. There are several options we can follow. The first is to send the package directly from the origin to the destination. The second one is to send the package to a hub airport and then to its destination. Our goal for doing this is to consolidate this package with other packages. In this case if there are packages that have the hub airport as an origin, they would be delayed so they can also be consolidated.

The hubs will be chosen by the program. This procedure is explained further in chapter 5. Here we need to say that the hubs are not fixed and any airport is a potential hub. All airports have equal probabilities of being chosen as a hub. A careful examination of past years data would allow us to define better the selection of hubs as we could see which airports were used as hubs in most of the cases. Unfortunately this data is not provided by the IT program of the company and it was impossible to collect them manually.

We must be very cautious with the constraints of each package. In case there is a new airport added, it must be checked if the package still arrives on time. Thus we assure that the service level is maintained to satisfactory levels. In the following pictures we show more clearly what the algorithm is trying to achieve. The pictures contain an

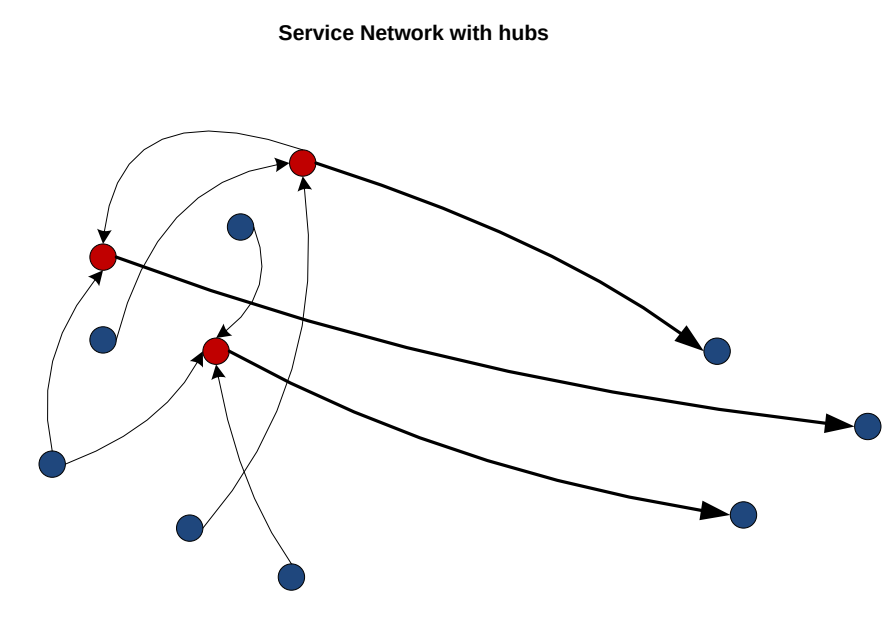
example of the service network of the company. In the first one the delivery of the packages is done directly and in the second one we use hubs and shipment consolidation.



In the case above the total cost is the following:

Cost: $\sum_1^n$ [weight* (EUR/kg) + handling fee + airline handling + clearance at destination]

This is the objective function of the problem. Of course this is without using any consolidation, even for the packages that have same origins and destinations. Our goal is to transform the objective function so it can describe a network that uses some airports as hubs. Moreover if we calculate the cost objective function it should be lower than the one without hubs. We want the transform the network as in the following picture:



Here you can clearly see what we are trying to achieve. The packages are consolidated into the origins airports and in the end only on shipment leaves for any destination. The cost in this case would be:

Cost up to the hub = $\sum$ [weight*(EUR/kg) + handling fee + airline handling + clearance at hub]

From hub to final destination = $\sum$ [($\sum$ weight)*(EUR/kg) + handling fee +airline handling + clearance at destination]

The real network would indeed be much more complex but the figure gives an insight of what the designed program is trying to achieve.

4. Simulated annealing

The method we are going to use for optimizing the company's network is called simulated annealing. In this chapter the history of the method will be described and the way it works will be explained. Also examples of its use will be used. Finally, we will focus on the advantages and the disadvantages of the method. This will help us understand our decision to use this method and why it fits to the current problem. Also it points at any problems that might occur with its use and makes a review of several problems that have used the same or a similar method.

4.1. History

Annealing is a process that has been used for treating metals for more than 5000 years. It is a heat treatment in which the metal is heated for an extended time (usually until glowing) and then it is allowed to slowly cool down. Annealing heat treatments are largely characterized by induced micro structural changes which are then responsible for altering the material's mechanical properties. The goal of the process is to reduce the hardness of the metal and improve its ductility. Then the metal can be further worked out (Ameritherm).

The "cooling" schedule is the way in which the metal will be allowed to lower its temperature. At the high temperature the atoms of the metals have very high energy values. This gives the atoms freedom to restructure themselves. By controlling the cooling schedule we try to direct the way to the atoms will be reformed as they move to lower energy states. The goal is to make them reform in crystal structure which will be both consistent and stable. This will give the metal the desired properties. The cooling may happen slowly until it reaches room temperature or it may be fastened by dipping it into water (quenching). The different cooling schedules result in different properties for the treated metal. A wrong cooling schedule could result in deteriorating the metal's properties instead of bringing the desired results. The cooling schedule is also very important in the simulated annealing procedure. Several tries have been made to find improved cooling schedules (Romeijn and Smith 1993, Dekker and Aarts 1991, Rosen 2005, Elmohamed and Coddington 1998)

In 1953 Metropolis introduced an algorithm, which was named after him as the Metropolis Monte Carlo integration algorithm. He used it to calculate large-dimensional path integrals found in statistical physics problems (Metropolis et.al.1953). The Monte Caro methods were named after the famous resort in the French Riviera. The name states the resemblance of the methods to the way the gamblers played in the casino as it is based into "random" sampling. In 1983 Kirkpatrick showed "how the Metropolis algorithm for approximate numerical simulation of the behavior of a many body system at a finite temperature provides a natural tool for bringing the techniques of statistical mechanics to bear on optimization" (Kirkpatrick et.al. 1983).

We can also formulate the simulated annealing algorithm as an inhomogeneous Markov chain (Seneta 1981, Isaacson 1976). There we have to decrease the value of the control parameter, in our case the cost between subsequent trials. In this case, it has been proved that the algorithm will asymptotically converge. Practically this means that the method can provide the optimal solution in every problem, as long as we allow it to

have an infinite number of iterations. Of course this is not possible because it would also require an infinite amount of time, which is not a viable solution.

Therefore techniques have to be invented to help the algorithm provide a very good solution within a reasonable amount of time. These “finite time” implementations of the algorithm do find good solutions but they cannot guarantee an optimal solution. Nevertheless, it has been found that recommended solutions are of high quality and therefore acceptable (Aarts, Korst 1990).

Combinatorial optimization is defined as the process of finding efficient techniques of estimating maximum and minimum values of a function with many independent variables (Aho 1974, Lowlor 1976). The function is named “objective function”. The objective function can serve as a measurement of how complex a system is. We use such an objective function in our problem. It is the cost function and it is the one we are trying to minimize.

In order to properly implement a simulated annealing method we have to include several parameters (Davidson, Harel 1996):

- The set of configurations, or states, of the system, including an initial configuration (which can be chosen randomly or by using other algorithms).
- A rule to generate the new configurations. This is obtained by defining a neighborhood of each configuration and choosing the next configuration randomly from the neighborhood of the current one. In case there are available statistical data we can choose the next configuration with “smarter” ways and not randomly.
- The objective, function, which needs to be minimized over the configuration space. This is the analogue of the energy in the physical annealing, which is constantly lowered as the algorithm runs.
- The cooling schedule of the control parameter, including initial values and rules for when and how to change it. This is the analogue of the temperature which decreases with time as the annealing process is realized.
- A condition to terminate the algorithm. This can be based on the time and the values of the cost function and/or the control parameter

The success of simulated annealing can be characterized by the following elements (Aarts 1990):

- Performance, i.e. running time, solution quality
- Ease of implementation
- Applicability and flexibility

4.2. Reasons for using simulated annealing

Simulated annealing is not the only method we could have used in order to optimize the company's network. Before examining the advantages and disadvantages of the method we will show why we decided to use it instead of the other methods.

One popular way of solving this kind of problems is by using heuristics algorithms. Most of the heuristics are designed for a specific problem. This makes the implementation of them rather difficult. There is no guarantee that an efficient algorithm for one problem will be as efficient in a similar one. In the problem we are examining the topology of the network is very important. We cannot know if one algorithm is efficient unless we test it. This trial and error procedure is considered inefficient and unproductive and it did not fit in our objectives. The main reason for using simulated annealing is the generality of the method. It has been used in many problems and produced satisfactory results. The main difference when using this method for different problems is the cooling schedule. We tried several values for dropping the temperature, between 80%-99% of the previous one. Finally from our testing we found that to drop the temperature after each iteration to 99% of the previous one was the best value. This is the most commonly used value in the literature. Also a list of problems which have been solved with simulated annealing is provided further on this chapter.

It is possible that companies with similar network with ours have researched for a way to optimize it. Unfortunately such solutions are considered business secrets and are not published. Moreover most of these systems are described as "legacy systems". It is very difficult to maintain or modify them as they are not well documented. Usually it is only one or two persons who have the necessary knowledge and the ability to do this and the company is dependent on them. With the solution we propose everything is well understood and easy to modify. It is not a secret that only the programmer knows but all the personnel that works in the appropriate department has sufficient knowledge over the program.

Another thing is that we can decide the running time of the algorithm. It needs a finite time to run which depends on the settings of the program and it will produce results in any deadline we set. Of course when we give it more time the results will be better.

Moreover it does not exclude the use of custom made heuristics. The annealing criterion is used to decide if we will accept the proposed plan. During the construction of the new plan we can used other algorithms. In fact we have tries several modifications before we ended up to the one we used and is described later on.

Finally we have to mention that simulated annealing is a scaled method. When we use it for more time and with more computational power we get better results. As mentioned before the success of the method is guaranteed when the running time is infinite. Of course the running time is not infinite, but we can safely suggest that it can be a really big number and that the solution will be very close to the optimal. It is easy to understand that if we use a big company's PC network we will be able to produce very good results in minimal time.

4.3. Advantages

Simulated annealing is conceptually very simple and easy to implement. Experience has showed that implementation for new problems often takes only a few days and a typical simulated annealing algorithm has needs only a few hundred lines of code. In the past 20 years it has been used widely and one could argue that it is one of the most applicable and flexible algorithms that exist. The complexity lies to applying the general algorithm to the specifications of each problem

Among the advantages of this method are the following: (Verdonk 2006):

- can deal with arbitrary systems and cost functions
- statistically guarantees finding an optimal solution
- is relatively easy to code, even for complex problems
- generally gives a “good” solution

In case statistical knowledge of the response surface exists, the method gives even better results. Instead of searching the entire feasible region for the global optimal solution, we can add the probabilities each solution has to occur. Thus we rule out many of the “bad “solutions and the algorithm then chooses among the better ones. The variation of the method is called discrete variable simulation optimization method (Rosen 2005). The method focuses in higher quality local optimal solutions. The implementation of the method provided with 1% - 7% better results, with the higher percentage corresponding to multi decision variables. Also this formulation helped the algorithm to develop a solution earlier as it required 15% - 17% fewer simulation runs (Rosen 2005). This makes S.A. an even more attractive option for optimization problems where problem specific heuristic methods are not available.

Another great advantage of this method is that it can be parallelized. This mean that it can be transformed in a way that it makes parallel runs into different computer systems. This practically means that we can significantly lower the running time of the algorithms. Especially in the network of a big company, which can contains dozens of computers, we can very easily use all this computational power for our benefit. Thus we can make the method produce a huge amount of solutions in a very limited amount of time. This will greatly improve the chances of finding a near optimal solution.

4.4. Disadvantages

The method is not always applicable and in certain occasions the problem needs to be reformulated or transformed into an equivalent. Another major problem is the way the neighboring solutions will be defined. Also the definition of a neighboring solution and the rules the algorithm follows to find them present various difficulties.

Other negative features of simulated annealing are the following (Ingber 1993):

- it can be quite time-consuming to find an optimal fit, especially when using the "standard" Boltzmann technique
- most of the times it is difficult to fine tune to specific problems, relative to some other fitting techniques;
- it usually suffers from "over-hype" and faddish misuse, leading to misinterpretation of results
- It can lose its ergodic property by misuse, for example by transforming the simulated annealing into a method of "simulated quenching". The problem is that this method has not been statistically proved to provide an optimal solution.

If the objective cost function is complex to compute, the cooling schedule of the method can prove to be very slow. Also the method cannot identify when a problem has a smooth energy landscape or when there are only a few local minima. Then the simulated annealing method proves to be very complex and it is outperformed by faster and simpler methods (gradient descent). The truth is the number of times the energy landscape is known is minimal.

Other heuristics algorithms are problem specific. In some occasions they take advantage of extra information that are available about the system. Often such methods, due to the fact that they have been customized can produce better results than the general simulated annealing algorithm.

In our opinion there is another major disadvantage about the simulated annealing algorithms. Even if we create an efficient algorithm and use it to find results, we have no way of checking the results. We cannot judge how good the solution is, if it is trapped to a local minimum or if it is very close to the optimal one.

4.5. Problems solved with simulated annealing

The method has been used firstly to calculate large-dimensional path integrals found in statistical physics problems (metropolis 1953). Later it was identified as "a natural tool for bringing the techniques of statistical mechanics to bear on optimization" (Kirkpatrick 1983). It has also been used in multi layer thin – film systems (Niu 1996), the travel salesman problem (Kirkpatrick, Gellat 1983), optimal deployment of missile interceptors (Bohachevsky, Johnson (1988), optimization of statistical functions (Goffe, Ferrier 1988), analysis of the cost function for CT image reconstruction (Haneishi et. al. 1990), to minimize mean weighted tardiness in a flow shop (Parsasharathy 1997), VLSI placement problem (Sechen, Sangivanni-Vincentelli 1985). Furthermore it has been used to determine the sequence of observations for an automated astronomical telescope, arrange connections on chips and switching devices in telephone networks and for random move games determined by the simulated annealing algorithm. This is just a selection of the publications that have used simulated annealing to solve their problems. The complete list is quite impressive and we can support with certainty that

simulated annealing is a powerful tool which can solve many problems, as long as the researcher can adapt it to every problem.

Apart from the above we have witnessed the usage of the simulated annealing algorithms in hub location problems. These are the problems which mainly interest us more as they are more similar to the one we are examining which are closer to the network optimization we are currently trying to do. It was used to solve a capacitated hub location problem (Rodriguez - Martin, Salazar – Gonzalez, 2008). Simulated annealing has also been used, with the Metropolis criterion (the same that we are using in our algorithm) to create a hybrid heuristic for the uncapacitated single hub allocation problem (Chen 2007). The same hub allocation problem has come into attention of more researchers. The genetic algorithm and tabu search (GATS) was introduced as a hybrid heuristic solution to the same problem (Abdinnour – Helm).

5. Optimization program

In this chapter the actual program we used in order to do the optimization will be described. It has been broken down into the most important functions. Several functions were further broken down due to their large size. Every decision the program takes is here described with detail. The program has been given the name “Simulated Annealing Planner”.

5.1. Input

The program in order to calculate and propose a plan for the packages the company has to deliver, needs to be given the appropriate input. First we create a table which lists all the airports in the network and associates each airport with a specific number. The program will use this number to identify the airports.

In the test case the table with the airports is the following:

	Airports
1	Athens
2	Frankfurt
3	Genoa
4	Hamburg
5	Hong Kong
6	Copenhagen
7	Malaga
8	Rotterdam
9	Shanghai
10	Tokyo

Then we create another table with the packages we need to deliver. The table describes all the orders we have received at the specific point in time. This is the actual table that the program will read and make its calculations. It contains an ID for each package, the origin and the destination, the deadline and the weight of the package

ID	Deadline	Destination	Origin	Weight
1	2	5	1	65
2	2	9	4	21
3	2	5	8	59
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
118	12	10	6	27
119	12	5	8	42
120	12	9	4	78

The entire table with all the 120 packages is included in the appendix.

Each package is given an ID. We use this unique ID to identify each package. Also, as we can see in the table the origin and the destination of each package are defined. The given numbers correspond to the table of the airports. The weight of each package is also given in kilos. It is used to calculate the cost of each shipment. Finally we see the deadline of each package. This represents the latest date the package needs to be delivered. The deadline in the table represents a point in time. The first package needs to be delivered in point 2 and the last in point 12. The points can be mapped to specific dates and each point has the duration of a day. So, the last package needs to be delivered 12 days after the day we will run the program, which is defined as day 0.

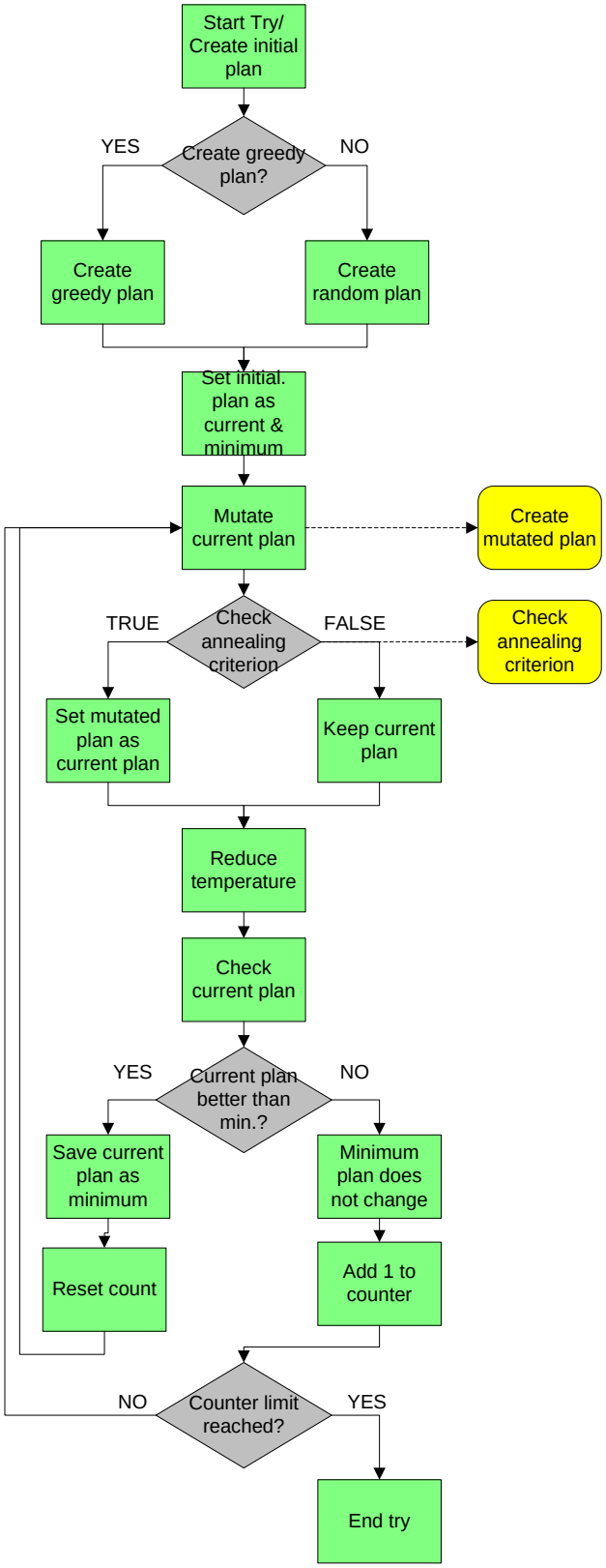
5.2. Description of the program

Here we will give a general description of the main functions of the program. We have divided the program into three main parts in order to make it easier to understand. Each part is accompanied with a diagram that explains how the algorithm takes the decisions.

5.2.1. Main function

Firstly the program reads the input table. Then it creates an initial plan. It will use this initial plan to make the calculations and try to generate a better one. There are two options for creating the initial plan: a random algorithm and a greedy algorithm. After we make this choice the initial plan is created. This is the plan we will attempt to optimize. The optimization decisions will be taken according to the costs. In this case the optimum plan is the one with the minimum cost. So, the program starts “mutating” the initial plan. The procedure of mutating will be described further on. After we have the mutated plan we check it with the annealing criterion. The exact procedure will be described later. According to the annealing criterion we either accept the mutated plan and we set it as current plan or we do not accept it and the current plan remains as it is. Then we check if the current plan is better than the minimum plan. If it is we set a new minimum plan and we reset the counter. If it is not we keep the minimum plan and we add 1 to the counter. Then we check the counter. If it has reached the pre-set number the procedure stops. We have the option to change the counter number every time we open the program.

The above mentioned procedure describes one run of the algorithm. We can specify the number of tries the program will make. The general rule is that when the number is bigger the result will be better.


Start Try/ Create initial plan
Create greedy plan?
YES
NO
Create greedy plan
Create random plan
Set initial. plan as current & minimum
Mutate current plan
Create mutated plan
Check annealing criterion
TRUE
FALSE
Check annealing criterion
Set mutated plan as current plan
Keep current plan
Reduce temperature
Check current plan
Current plan better than min.?
YES
NO
Save current plan as minimum
Minimum plan does not change
Reset count
Add 1 to counter
Counter limit reached?
NO
YES
End try

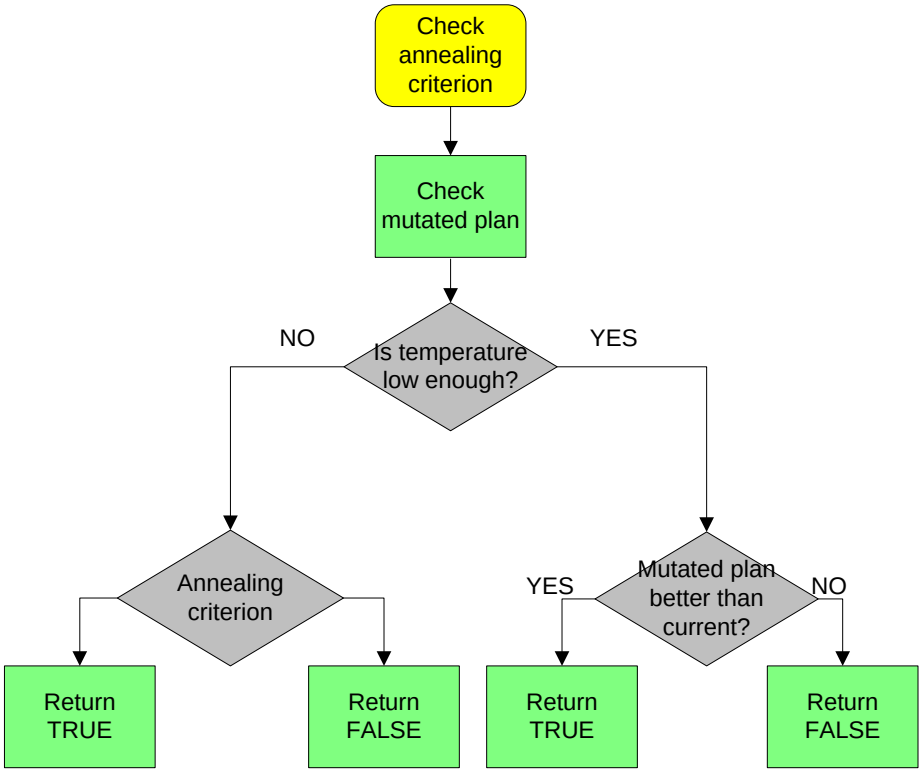
5.2.2. The annealing procedure

Here we will describe how the algorithm uses the simulated annealing method in order to make decisions. Initially we are checking the current temperature. As mentioned the temperature is lowered with every iteration the program makes. The value we chose is that the temperature will be at the 99% of the previous one. This value is the most common used in the simulated annealing algorithms. If the temperature has lowered enough we check it the mutated plan is better than the current one. If it is better we accept it and the function returns the value TRUE, if not we do not accept it and the value FALSE is returned. In case the temperature has not lowered enough the program decides with the annealing criterion. The annealing criterion is the following:

$random < \exp\left(\frac{diff}{temp}\right)$, where

- random = a random number
- diff = the difference between the minimum plan and the mutated plan
- temp = the current temperature

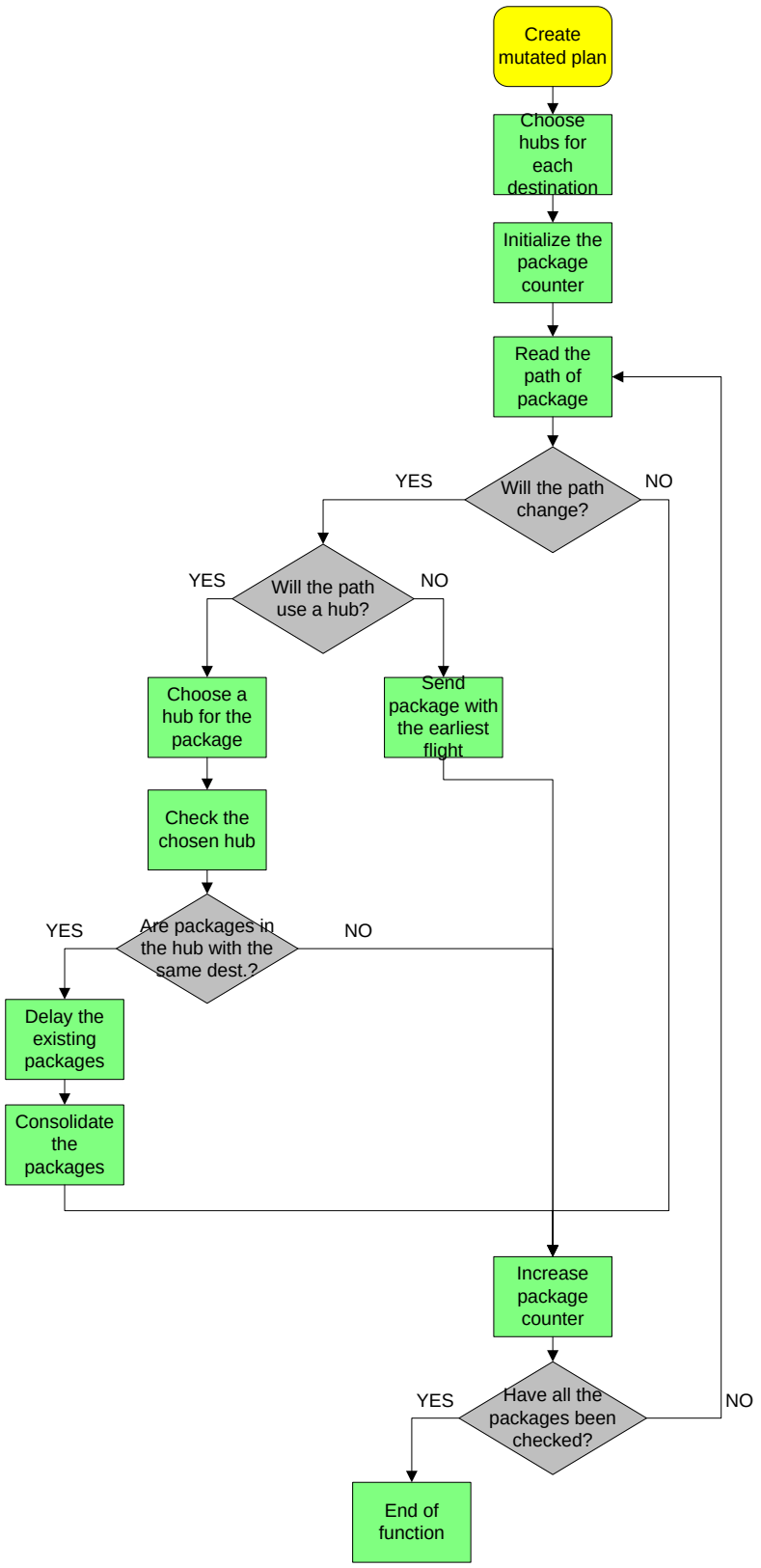
Then according to the criterion the function returns a TRUE or FALSE value



5.2.3. The mutation plan procedure

This procedure takes the current plan and mutates it in order to produce a better one. The changes that will take place concern the routing of the packages. Instead of direct shipping each package the procedure will change the path each package will follow. It will try to send some packages through hubs and consolidate them with other packages. First we need to pick the airports that will be used as hubs. The program chooses randomly one hub for each destination airport. We must mention that it is also possible that two destinations will use the same airport as a hub. After it has chosen the hubs, it will start altering the paths of each package. We have defined the probabilities of altering the path of each package. In the test case we have assigned a 10% probability. Then when the program has decided to change a specific path it has to choose if it will use a hub. We have defined the probabilities of using a hub at 50%. The same is done for every package we have. If a package has been chosen to go through a hub then there is an additional check in that hub. If there are packages that have the same destination and their origin is the hub airport, then they are delayed. They wait for the rest of the packages to arrive at the hub and then they are consolidated into one shipment.

Once this procedure has been done for all the packages the mutated plan has been created. Then the main function continues until the mutation of the plan is called again.


Create mutated plan
Choose hubs for each destination
Initialize the package counter
Read the path of package
Will the path change?
YES
NO
Will the path use a hub?
YES
NO
Choose a hub for the package
Send package with the earliest flight
Check the chosen hub
Are packages in the hub with the same dest.?
YES
NO
Delay the existing packages
Consolidate the packages
Increase package counter
Have all the packages been checked?
YES
NO
End of function

5.3. Output

Here we will describe the output that the program produces. The objective is to find a plan of how we will route the packages.

First of all the program gives us the route the package has to follow and which point in time it has to be shipped. We can separate the results in two big categories, the packages that are shipped directly and the ones that use a hub.

The directly shipped packages look like this:

```
Routing 34 from 6 to 9 no later than 5
Air Connections: (6, 9):1
Routing 35 from 1 to 10 no later than 5
Air Connections: (1, 10):1
```

Here we are given the following information:

- Package 34 has to be shipped from airport 6 to airport 9 and it has to arrive there before day 5. It is scheduled to leave at day 1.
- Package 35 has to be shipped from airport 1 to airport 10 and it has to arrive there before day 5. It is scheduled to leave at day 2.

The other category is the packages that use a hub:

```
Routing 15 from 1 to 9 no later than 3
Air Connections: (1, 8):1 (8, 9):3
Routing 16 from 7 to 10 no later than 3
Air Connections: (7, 2):1 (2, 10):2
```

This output provides us with the following information:

- Package 15 has to go from airport 1 to 9 before day 3. It will travel from 1 to 8 at day 1 and then at day 3 it will go from 8 to 9, which is its final destination.
- Package 16 has to go from airport 7 to 10 before day 3. It will go first from 7 to 2 at day 1 and then from 2 to 10 at day 2.

The same kind of information is given for every package we have given in the input table.

Furthermore the program calculates the total cost in Euros of the proposed plan.

```
Cost: 47301.2
```

Finally some extra information is given. This information can be studied further and give the company an insight of which airport is used more, the average weight of the shipments and the total cost of each proposed route.

```
Connections flight 3, 10 total cost: 1628.0
Connections flight total weight: 166
```

```
Connections flight 4, 5 total cost: 2921.6
Connections flight total weight: 452
```

Here we are informed the following facts:

- From airport 3 to airport 10 we send packages of total weight 1628 kg with a total cost of 1628 €.
- From airport 4 to airport 5 we send packages of total weight 452 kg with a total cost of 2921.6 €.

The same kind of information is given for every connection in the network we are examining.

The complete output table of the program is provided in the appendix.

5.4. The connection flight payment function

These are the payment functions between all the airports in the system. In the data set that was used there were a total of 63 connection flights between 10 airports. The freight rate for a package to be moved from an origin to a destination airport is a scalar function. There are 4 different scale rates. Each rate has a higher weight limit than its previous one whereas the cost per kilo is lower. In case the freight rates change, they have to be also changed in every payment function. In the freight rate cost we add the clearance cost in the destination airport. The reason for this decision is the consolidation we are trying to make. Since the packages are being consolidated in the origin, then they are all shipped together to the destination. So the clearance fee is divided between all the packages in the same flight. For the program to identify this we had to add this cost to the connection flights payment function. This cost is now distributed among all the packages that are in the same flight, as all of them are cleared together only one time and not every package separately.

5.5. Airport function

This function is used to define the different airports, which are the nods to the network. Each airport has as attributes an ID and a name. Also they have a clearance function and a port handling function. The clearance function defines the charges for clearing a package in every airport and the port handling function defines the charges of the airport for the handling of every package. Both functions will be further explained later.

5.6. Connection function

The connection function is used to define the attributes of all the connections, the arcs in the network. Each connection is given a different set of values. Firstly it is given an ID. Then the “from” and “to” values have to be defined. These, state from where the connection starts and where it ends. Also the time a package need to follow each connection and an air handling fee are being defined.

5.7. Package function

Here we give the appropriate values to each of the packages. Each package has an ID number and a specific weight. Thus the program knows in which scale rate to put it. Also the origin and the destination of each package are defined. Lastly a deadline has to be given. This is the latest time a package can arrive to its destination. There is no restriction for a package on arriving earlier, as long as it is available at the given time.

5.8. Path function

This is an auxiliary function which is used for the paths the packages follow. Firstly it clones the selected path for each package. This cloned path is the one that will be changed in the next session of the algorithm. Then it checks that the selected flight for each package is before the package's deadline. If it is not it does not allow the specific path for this package and the algorithm will give it a new path in the next iteration. Also this function is used to print the results. In the current configuration the algorithm returns the following information for each package: the time that it will travel, from where it will start and where it will arrive.

5.9. Plan function

The following function is the plan function. It defines the different plan according to which the packages will be forwarded. Each plan consists of the paths that all the packages will follow.

Firstly we define what each plan contains. Here it is given the variables: path, airport and connection. As explained before each different plan consists of the different paths each package will follow to reach its destination. Then it lists the available airports and connections. So far there are not any paths defined in the current plan. Next it clones the current plan, clears the paths and adds the new paths to the cloned plan. We use the cloned plan to make all the changes the algorithm proposes. So when the program makes a full run if the cloned (changed) plan is better than the current one, we change it and define the cloned as the current. If it is not better, we proceed with the current plan, which has remained unchanged, since we have only altered the clone.

In the following part the cost of each new plan is computed. First of all it reads and stores the weights per origin, destination and flight. Then it runs several loops which checks for all the paths which are contained in the plan and for all the connection flights that exist in a plan. For all the connection flights in the current plan, checks if there are weights in the airport ID. If there is not it adds a new weight list, so it can store the weights in that list. This is done for all the paths that are contained in a plan. In the next two loops the same procedure is being followed in order to count and store the weight of all the packages to their destinations and to count and store the weight per connection. Then it adds the weights from every origin and puts them into the variable total weight. At this point the program adds the weights and calculates the cost to each destination. In order to do so it takes into account all the relevant cost functions. Then it adds the weights and calculates the costs per connection flight. Again all the relevant

cost functions are being called in order for those calculations to be performed.

5.10. SAPlanner function

The SAPlanner function stands for Simulated Annealing Function. It is one of the core functions of the program and explains the rules which are used by this algorithm to help it take decisions. The simulated annealing plan will be described. This is also the part that can be further improved in the program. There have been many versions of this part until its current form was decided. It is the most complex function of the program but on the same time the most crucial. This function will also be broken down into parts. Each part will be explained separately in order to make it most understandable.

First we establish a random number generator. This will be used in various parts of the function. In general in all the simulated annealing algorithms similar generators are used, since the method is based on them. Then we initialize the plans that this function uses in order to clear it from previous values. In these the current plan, the minimum plan and a temporary plan are included. The program first uses the current plan. It sets it as minimum plan and it starts running. The various calculations are being made in a temporary plan. If this is better than the minimum plan it is set as minimum..Next certain values are assigned which are significant for the algorithm. The number of plans it will generate without optimizing the minimum plan. Then the times the algorithm will run with a different initial plan. Afterwards we assign the number of airports (nodes) each package can use (including the origin) in order to reach its destination. In the current configuration this number is 2. This practically means that only 1 stop is allowed per package, meaning that only 1 hub will be used. Then the numbers of attempts that the program will make to mutate the current plan are defined. Here this number is 10. This can be either increased or decreased. The general rule is that as bigger this number is as better. In practice a very large number does help the algorithm produces slightly better results but there is a big payoff in the calculation time.

In the next part the actual planning starts taking place. First we define the variables for the function which are the three plans mentioned above, the airports, the connections, the packages, the destination airports and the hubs. Also the variables temperature and greedy are defined. The temperature is used in the annealing procedure. It acts as a counter of the iterations, which is then checked and acts as a constraint in the algorithm. The greedy variable is used to define if the initial plan will be calculated by using a greedy algorithm. In the current configuration this is the case as it seems that it provides better results. This may not be the case though in all circumstances.

Then we call the “MarkHubs” sub function. For each destination airport it finds the connection flights that end up in this specific airport. Firstly it loops between the connections for each destination airport. Then if the connections have the same destination it puts them as hub candidates. Then for every connection it chooses randomly a hub from the candidate list. If the hub is used for the first time it adds it to a list of hubs. The algorithm now finds the hubs, and then it calls the function that produces the best plan for this try and saves it. This is done for all the number of tries

that we have set. Then the algorithm checks the best plan from each try. After that it compares it with the best plans from every previous try and keeps only the best one.

In the next part the simulated annealing is running. As it has been mentioned before, this algorithm is based on a starting temperature. This temperature is lowered in each iteration of the algorithm by a set value. At the end of the running time the temperature is checked and if it is low enough the mutated plan is kept. If the mutated plan has not been kept, a new one is estimated according to the standard Boltzmann simulated annealing algorithm, which is shown in the source code. Instead of the standard simulated annealing algorithm there are several other algorithms that can be used. Many times a new algorithm is constructed custom for a specific problem. The disadvantage of using such an algorithm is that is usually works very good for the specific problem that it has been designed but it has not guaranteed results for the rest of the problems. For this reason the most common algorithm has been used, which is also according to the literature most generally applicable.

Next the condition that terminates the annealing procedure is set. It is the variable "samecount". It specifies the maximum number of mutations the algorithm will make without changing the current plan. Then when the annealing constraint is satisfied it keeps the mutated plan. After that it checks the cost of the mutated plan. If it is better than the current plan, then it puts the mutated in the position of the current. The rest part of the code is about the cooling schedule of the annealing. It drops the temperature at each iteration to 99% of the previous one. This percentage is the most common used in the literature and there was no obvious reason to change it. If the mutation of the plan fails 100 times then it proceeds to the next cycle.

The following is the part that mutates the current plan. First of all the number of tries it will make to mutate the current plan are defined. Here this value is set to 10. If this value is bigger, in general the results are expected to be better. In the current problem the specific value worked out well. Then it gives a 10% chance to each package to be rerouted. This is done with the help of the random number generator mentioned earlier. More values than this 10% were tested and it seems that this number offers the best results without compromising the programs running time. Afterwards it checks if the mutated plan is valid.

Next the algorithm takes the plan and an index ID on the path that has to be rerouted. Each path is given a 50% chance to use a hub to reach its destination. Then it checks the hubs that send packages to the same destinations. In case a package has origin which has been chosen to become a hub, then the packages wait in their origin for the rest of the packages to arrive. If the origin is not a hub it tries to find the quickest flight from the origin to a hub. In case a hub is not used, it routes the package is routed directly with the "find quickest flight" sub function. This sub function sends the package directly to its destination with the first available flight.

Next we have the two functions that calculate the initial plan. We have two options, a random method and a greedy method. Here is the greedy method for the creation of the initial plan only. After trying the algorithm with different configurations it was obvious that a good initial plan was helping the algorithm to locate a very good solution. In this problem the greedy algorithm in almost all of the tested cases provided better results than the other option, the completely random algorithm, which is presented later on. The method calls the “find greedy path” function in order to find a path for every package. The combination of all the greedy paths creates the greedy plan. Next we see the random method which is used for the calculation of the initial plan only. Like the greedy method, it calls the “find random path” function. This one finds a random path for every package and then it combines them to create the initial random plan

The following part sends every package directly from the origin to the destination. This is done by putting adding a specific value to the length of the path. To force the package travel directly without using a hub we use the value 1 as the length of the path. Next the “find random path” is called, which is the function that finds a random path for each package. The path is not totally random. The algorithm tries to make it as quick as possible, meaning that it has the minimum length. Among those with the minimum length it chooses randomly.

5.11. Test me function

This is the function which runs the entire program. Firstly we define all the connection flights payment functions. Most of them have been omitted since they are all the same. Inside the functions we define all the cost a package has to pay when it travels through the specific connection flight. There is another variable, the airline handling fee. This is not considered important and it would not make any change to the way the algorithm works. We preferred to leave this field blank, so the airline handling fee is not calculated in the costs. In case we want to calculate it we must similar same functions to the connection flights payment functions. Then we have the clearance functions for all the airports. Each airport charges a clearance fee to every package.

Then we initialize all the airports and we attach to them the payment functions of each one. Also then we add them in the memory of the program. We can see 10 airports in total. The airports have been divided, as can be seen in the connection flights, to two different types: origins and destinations. The reasons for this separation are the actual facts of the problem. In reality some airports act only as destinations. In some rare occasions there are certain packages which use them as origins. This case, as it is rare has been omitted from the program.

Now we add the connection flights for each day. In the specific occasion we assumed we have a total of 5 days to send all the shipments. This number will change according to the tested data. Also the connection flights are added to the program. As mentioned before, there are in total 63 connection flights. Only a few of them are shown here as the procedure is the same for all of them

The next lines define the place from which the program will acquire its data. Currently the data set is in the desktop screen of my laptop. Anyone who will attempt to run this program will have to change this path according to where is the dataset saved. As we can see the program reads the data from a single ".txt" file. It is very easy to extract such a file from an "MS Excel" spreadsheet or from any similar program. The "ReadPackets" function is the exact one that is reading the data.

The next part calls the "SAPlanner" function, which has been describer earlier. Here we also define certain important parameters of the program. Firstly we have to define how many iterations the function will make. This how many times the algorithm will run and try to find a better solution, in case the current solution has not changed. Currently it is programmed to run 1000 times in case it does not find a better solution. Then it stops, stores the current best solution and proceeds to the next try. A larger number in general may provide slightly better results. A very large number though does not guarantee a better solution. When the algorithm has already done many iterations, due to the design of the simulated annealing procedure only small changes are allowed. So it is most probable that the best solution the algorithm can calculate ate the specific try has already been found. The payoff for increasing this number is the amount of time required to produce the results, which becomes significantly larger. Then we define the number of tries that the program will make in total. The optimum would be if this number was infinite. Then the algorithm would produce the optimal solution with 100% confidence. This is of course impossible so we have to settle with a reasonable number. Also it has to be mentioned that the amount of time the program needs to find a final solution increases linearly with this number in comparison with the iteration times where it increases exponentially. This means that we prefer to increase this number a lot. From the actual testing of the algorithm it was found that a large number does produce better results without compromising the functionality of the program. Finally a logic variable is given a value here. In case it is "true", the greedy algorithm runs to find an initial solution. If the value is "false", a random initial plan is generated.

The following part is the function that actually reads the input ".txt" file. The process is standardized and can be found in the internet so there is not an actual need of explaining it here. What should be said though is that here we must map to the program the definitions of the data it is reading. The data should be set up in columns in the following order: package ID, deadline, destination, origin and weight.

6. Optimization Results

In this chapter we will discuss about the results of the program we have used in order to optimize the company's network.

First of all we will check the cost of sending all the packages separately to their destinations. This has been calculated to be **72417 €.** Then we will compare it with the minimum cost the program has produced. This cost was calculated to be **45804.7 €.** In this plan the program has proposed the use of two hubs, Rotterdam and Frankfurt.

In total of 120 packages, 8 have been rerouted to travel through Rotterdam and 7 packages have been rerouted to travel through Frankfurt. The packages that already are at the hubs wait for the rest of the packages to arrive there and are not shipped earlier. Also consolidation is made even if we do not use hubs. If two packages have to leave from the same origin and have the same destination, they are shipped together. The algorithm delayed 14 packages in the hubs until the rest of the packages arrived. We have tested the algorithm extensively in order to find the settings that provide the best results. When we tuned the algorithm we researched the quality of the results it gives. We ran it 20 times with the following settings:

- 3000 iterations/try
- 1000 tries

The results we have taken from each run are the following

46535,4	**45868,3**	**46412,3**	**46452,5**	**46395,7**
46666,4	**46852,0**	**45859,0**	**46048,4**	**46476,8**
46441,4	**46502,9**	**45804,7**	**46313,1**	**46301,5**
46972,2	**46271,8**	**46328,8**	**45833,7**	**46843,6**

From this table we can derive the following conclusions:

- The optimum plan costs **45804,7 €**
- The average cost is **46359 €**
- The standard deviation is **341,7 €**

As we can see, between the initial plan of sending each package separately and the plan proposed by the program, there is a big difference of 26621€. We cannot suggest that this is the amount the company will benefit from using the program. The employees would have made a manual optimization and the cost would have been lower. Unfortunately we cannot know the exact difference between the plan that the program proposes and what the company would actually do.

The running time of each try is approximately 2 hours and 40 minutes. The test tries where made in a laptop computer with the following specifications: an Intel Core2Duo

T7300 processor, at 2GHZ and 2GB RAM. To give a better insight about these numbers, this system is more than 3 years old and it is considered relatively outdated. It is obvious that for the 20 tries we needed more than 2 days of pure computational time. Considering that we would record the results each time and then run the algorithm again, the total time was estimated at 4 days. This is not a vial option for the company since it needs to take these decisions in a few hours at most. Comparing the benchmarks from the system we used and some of the newest laptop models we see that they are remarkably faster. Although we did not try, as such a system was not available, it is certain that the computational time would be greatly reduced. Also if we used multiple systems to run the program we could produce very good results in limited time, even with older computers. In case the company decides to use this program, it could very easily build a grid with some of its computers and run it there.

We decided to expose the results to the company and ask their opinion about the proposed plan. Their experience allows them to realize potential mistakes in the plan which we could not see. Although based only on the experience of the personnel we believe that it is a test that would give us very useful results. Their response was very positive and they agreed that the plan looked correct and that they would have constructed a similar one. The comment we had is that they would have preferred to route certain packages in a different way. This happens because they prefer to route some packages from specific airports. They do it in case a package needs special handling and they prefer to send to the airport with the best handling service. Also the company has owned warehouses in some airports and they would prefer to use them even if the transportation part cost a bit more.

7. Conclusion

The simulated annealing algorithm is believed to be working very good in the specific problem. It quickly minimizes the original solution and returns a much lower result. As mentioned before we cannot check if the provided solution is the optimal. What we do know is that it produces one very good solution. We have to be careful with the variables in the algorithm. We must remind that there are a lot of settings we can very easily change in the algorithm and make it run differently. Therefore it should not surprise us that it produces each time different results. This is also the case if we use the same settings and a very limited amount of tries and iterations

In the test runs we did with it, the algorithm proved reliable and effective. With a large amount of tries it produces nearly the same results every time. After carefully checking the output we recognized that only one or two packages were routed differently in each run. We consider this variation acceptable and we have to remind that the amount of tries was kept relatively limited due to time restrictions as we had to perform a large amount of tests. In case it would be actually used by the company to take the decisions the number of tries and the number of iterations in each try would be significantly bigger. A stronger computing system would be able to produce the results relatively quickly. We also have to mention that the data set was relatively limited as was the size of the network. This was done because we did not see an advantage of using the entire network. The algorithm can be easily expanded to involve all the real hubs and more packages than the ones in the test case.

Our belief is that the method we uses proved to be reliable and we can trust it to find an optimal solution. Though we cannot evaluate the cost for the implementation of the method and for attaching to the existing IT program, we strongly believe that in the long run it the company would greatly benefit for it.

7.1. Further research

In the present study we have tried to present an optimization program. The program is based uses a simulated annealing algorithm in order to produce results. Also it certain rules which we have set. It is a fact that there is randomness in the choice of the plans. We rely heavily on the quantity of the tries to produce an acceptable result. This should be improved and we would like to have better quality results with fewer runs. This can be done if we study the past orders of the company. Then we could assign specific probabilities of each airport being used as a hub and the path each package will use, based on the past years data. In our case this was not an option as this data was not available electronically. It was virtually impossible to gather this data for thousand of orders manually one by one. However in case the IT program of the company is modified, this data would be available very easily. Thus the program could be redesigned and probably it would produce better results. Furthermore there are some aspects that could be improved. There are cases that a package is not heavy but it has a big volume. In such a case it is charged by volume. In our case we only take into account the weight of the package and we do not consider the volume at all.

There are other subjects that maybe escape the field we are studying but should be considered. In our approach we considered all the airports equally important. This is not entirely true as there are differences between them. Even inside the E.U. the customs procedures vary and it is possible that we prefer to use one airport instead of another even if it is slightly more expensive. Also in certain airports the company might has owned facilities. Then it is obvious that it will prefer to use its own facilities. Then it could benefit both not only from sending the package but also from storing it.

Bibliography

Aarts E, Korst J (1990). *Simulated annealing and the Boltzmann machine*, Wiley, New York, NY, USA

Abdinnour-Helm S. (1998), *A hybrid heuristic for the uncapacitated hub location problem,* European Journal of Operational Research, issue 106, pp 489 – 499

Abdinnour-Helm S. (2001). Using simulated annealing to solve the p-hub median problem, *International Journal of Physical Distribution & Logistics Management* **31** (3), pp 203–220

Ahmed M., Alkhamis T. (1992). Simulation-based optimization using simulated annealing with ranking and selection. *Computers and Operations Research,* 29, pp 387–402

Ahmed M., Alkhamis T, Hasan M. (1997). Optimizing discrete stochastic systems using simulated annealing and simulation, *Computers and Industrial Engineering*, vol.32 No.4, pp 823-836

Alrefaei, H. M, Andradottir S.(1999). A Simulated Annealing Algorithm with Constant Temperature for Discrete Stochastic Optimization, *Management Science*, Volume 45 , Issue 5 , Pages: 748 – 764

Alumur S., Kara Y. B. (2008). Network hub location problems: The state of the art, *European Journal of Operational Research*, vol. 190, Issue 1, pp 1-21

Anagnostopoulos A., Michel L., van Hentenryck P., Vergados Y. (2006). A simulated annealing approach to the traveling tournament problem, *Journal of Scheduling*, Vol. 9, Issue 2, pp 177 – 193

Aykin T., (1995a). The hub location and routing problem, *European Journal of Operational Research* 83, pp 200–219

AykinT.,(1995b). Networking policies for hub-and-spoke systems with application to the air transportation system, *Transportation Science* 29, pp 201–221

Azencott R. (1987-1988). *Simulated annealing*, Séminaire Bourbaki, 30, Exp. No. 697, pp 15

Bertsimas D., Tsitsiklis J. (1993) Simulated Annealing, *Statistical Science* Vol. 8, No. 1, Report from the Committee on Applied and Theoretical Statistics of the National Research Council on Probability and Algorithms (Feb., 1993), pp. 10-15

Bohachevsky I.O., Johnson M.E., Stein M.L. (1988). Optimal deployment of missile interceptors, *American Journal of Mathematical and Management Sciences,* vol. 8 (3 & 4), pp 361-387

Bravos C., (2010). Interview by author, Operation Manager, Golden Cargo, Piraeus

Campbell J.F., (1994). *Integer programming formulations of discrete hub location problems*, European Journal of Operational Research, issue 72, pp 387-405

Campbell J.F., Ernst A.T., Krishnamoorthy M. (2002). *Hub location problems*, Z. Drezner, H.W. Hamacher (Eds.), Facility Location. Applications and Theory, Springer, Heidelberg, 2002, pp 373–408.

Chen F. (2007). A hybrid heuristic for the uncapacitated single allocation hub location problem, *Omega* **35**, pp 211–220

Connelly D. T. General purpose simulated annealing, *Journal of Operations Research*, 43

Czyzak, P. and Jaszkiewicz, A., Pareto simulated annealing - a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, vol. 7, pp 34-47

Doug Lea, "Concurrent programming in Java: Design Principles and Patterns", Addison-Wesley, 1996

Eglese, R. W. (1990). Simulated annealing: A tool for operational research, *European Journal of Operational Research*, Elsevier, vol. 46(3), pp 271-281

Elmohamed S., Coddington P. D., Fox G. (1997). *Lecture Notes In Computer Science*; Vol.1408 archive Selected papers from the Second International Conference on Practice and Theory of Automated Timetabling II, pp 92 – 114, Springer-Verlag London, UK

Ernst A.T., Krishnamoorthy M. (1998). Exact and heuristic algorithms for the uncapacitated multiple allocation p-hub median problem, *European Journal of Operational Research* **104**, pp 100–112

Gao L., Robinson E.P. Jr. (192). A dual-based optimization procedure for the two-echelon uncapacitated facility location problem, *Naval Research Logistics* **39**, pp 191–212

Gelfand S.B. (1987). *Analysis of simulated annealing type algorithms,* Ph.D. Thesis, MIT, Cambridge,MA.

Gelfand, Saul B., Mitter, Sanjoy K. (2007). *Metropolis-type Annealing Algorithms for Global Optimization in IRd*

Gidas B. (1985) Non stationary Markov chains and convergence of annealing algorithms, *Journal of Statistical Physics, vol.* 39, pp 73-131

Golden Cargo Operations Manual (2009), Golden Cargo, Piraeus

Haddock J., Mittenthal J. (1992). Simulation optimization using simulated annealing, *Computers and Industrial Engineering*, vol. 22 n.4, pp 387-395

Heaton J., (2008) *Introduction to Neural Networks for Java*, 2nd Edition, Heaton Research, 2008

Jadbabaie A., Ozdaglar A., Wei E., Zargham M., A distributed Newton Method for network optimization

Kirkpatrick S, Gelatt CD, Vecchi MP (1983). Optimization by simulated annealing, *Science* 220, pp 671-680

Klincewicz J.G. (1991). Heuristics for the *p*-hub location problem, *European Journal of Operational Research* **53** (1991), pp 25–37

Klincewicz J. G. (1996). *A* dual algorithm for the uncapacitated hub location problem, *Location Science*, vol. 4, Issue 3, pp 173-184

Laarhoven van P.J.M., Aarts E.H.L. (1987). *Simulated Annealing: Theory and Applications*, D. Reidel, Dordrecht, The Netherlands

Lester I. (1993) Simulated Annealing: Practice versus Theory, *Mathematical and Computer Modeling*, vol. 18, No. 11, pp 29-57.

Lester I., (1995). *Adaptive simulated annealing (ASA)*, ftp://alumni.caltech.edu/pub/ingber/

Lester I. (2008) *Adaptive Simulated Annealing (ASA):Global Optimization and Sampling C Code*, Version 4. Knol,

Marin A., Canovas L., Landete M. (2006), *European Journal of Operational Research,* vol. 172, pp 274 292

Mayer G., Wagner B. (2002). HubLocater: An exact solution method for the multiple allocation hub location problem, *Computers & OR* **29**, pp 715–739

Metropolis N. (1987). The Beginning of the Monte Carlo Method, *Los Alamos Science,* No. 15, pp 125-130

Metropolis N., Ulam S. (1949). The Monte Carlo Method, *Journal of the American Statistical Association.* vol. 44, No. 247, pp 335-34

Metropolis N., Rosenbluth A.W., Rosenbluth M.N., Teller A.H., Teller E.(1953). Equation of state calculations by fast computing machines, *Journal of Chemical Physics, vol.* **21** (6), 1087-1092

Organization for Economic Co-operation and Development – OECD (2005). *The role of changing transport costs and technology in industry relocation,* Meeting of the Maritime Transport Committee, 26-27 May 2005

Romeijn H.E., Smith R.L (1994). Simulated annealing for constrained global optimization, *Journal of Global Optimization*, vol. 5(2), pp 101–126

Rosen L Scott., Harmonosky M. C.. (2005) An improved simulated annealing simulation optimization method for discrete parameter stochastic systems, *Computers and Operations Research*, vol.32 No.2, pp 343-358

Siarry P., Berthiau G., Durbin F., Haussy J. (1997) Enhanced simulated annealing for globally minimizing functions of many continuous variables , *ACM Trans. Mathematical Software*, vol. 23, no. 2, pp 209-228

Szu H., Hartley R. L. (1987) Noneconomic optimization by fast simulated annealing, *Proceedings of the IEEE*, vol. 75, pp 1538-1540, Nov. 1987

Szu H., Hartley R. (1987) Fast simulated annealing, *Physics Letters A*, Vol. 122, Issues 3-4, pp 157-162

Suman, B. and Kumar, P., A survey of simulated annealing as a tool for single and multiobjective optimization. *Journal of the Operational Research Society*, vol. 57 issue 10, pp 1143-1160

Tsallis C,,Stariolo D. A.(1996). Generalized simulated annealing, *Physica A*, vol. 233, No.1-2, pp 395-406

Wang, L.,Zhang, L., Stochastic optimization using simulated annealing with hypothesis test. *Applied Mathematics and Computation*, vol. 174, pp 1329-1342

Wagner B. (2004). The latest arrival hub location problem, *Management Science,* vol. 50, No. 12, pp 1751-1755

Xinhui Niu (1996). *Basic Adaptive Simulated Annealing in a Java Environment*, (http://embedded.eecs.berkeley.edu/Alumni/zhengyu/ref2.htm)

Yaman, H., Kara, B.Y., Tansel, B.C., (2005). *The latest arrival hub location problem for cargo delivery systems with stopovers*, Technical report, Bilkent University Industrial Engineering Department, 06800 Bilkent, Ankara, Turkey

APPENDICES

Appendix 1: Source code of the optimization program

1. The connection flight payment function

```
public class RotterdamHongKongPayment implements PaymentFunction {

	@Override
	public Float ComputPayment(Integer weight) {
		if (weight < 45)
			return (float) (10.7*weight +300);
		if (weight < 100)
			return (float) (8.6*weight+300);
		if (weight < 200)
			return (float) (6.8*weight+300);
		return (float) (5.5*weight +300);
	}

}
```

2. Airport function

```
public class Airport {

	String name;
	Integer ID;

	// Package arrives at airport
	PaymentFunction clearance;

	//Package leaving from airport
	PaymentFunction portHandling;

	Airport(String name, Integer ID,
			PaymentFunction clearance, PaymentFunction
portHandling ){
		this.name = name;
		this.ID = ID;
		this.clearance = clearance;
		this.portHandling = portHandling;
```

3. Connection function

```
public class Connection {

	Integer ID;
	Airport from;
	Airport to;
	Integer time;

	PaymentFunction airHandling;

	String getID(){
		return ID+"_"+time;
	}

	Connection(Integer ID, Airport from, Airport to, Integer time,
			PaymentFunction airHandling){
		this.ID = ID;
		this.from = from;
		this.to = to;
		this.time = time;
		this.airHandling = airHandling;
	}
}
```

4. Package function

```
public class Package {
	String ID;
	Integer deadline;
	Integer destination;
	Integer origin;
	Integer weight;


	public Package(String ID,Integer deadline, Integer destination,
Integer origin,
			Integer weight) {
		this.ID = ID;
		this.deadline = deadline;
		this.destination = destination;
		this.origin = origin;
		this.weight = weight;
	}
}
```

5. Path function

```
public class Path {
	Package packet;
	List<Connection> flight = new ArrayList<Connection>();
	public void ClonePath(Path p){
		packet = p.packet;
		for(int i = 0 ; i < p.flight.size(); i++){
			flight.add(p.flight.get(i));
		}
	}
	public boolean Valid(){
		for(int i = 0 ; i < flight.size(); i++){
			if (packet.deadline < flight.get(i).time)
				return false;
		}
		return true;
	}

	public String links_toString() {
		String string = "";
		for(int i = 0 ; i < flight.size(); i++){
			string += "("+flight.get(i).from.ID+",
"+flight.get(i).to.ID+"):"+
			flight.get(i).time+" ";
		}
		return string;
	}

}
```

6. Plan function

```
public class Plan {
	List<Path> paths = new ArrayList<Path>();
	List<Airport> airports;
	List<Connection> connections;
	Float cost = new Float(-1.0);

	Plan(){
	}

	Plan(List<Airport> airports, List<Connection> connections){
		this.airports = airports;
		this.connections = connections;
	}
	public void ClonePlan(Plan p){
		this.airports = p.airports;
		this.connections = p.connections;
		paths.clear();
		for (int i = 0; i < p.paths.size();i++){
```

```
                  Path path = new Path();
                  path.ClonePath(p.paths.get(i));
                  paths.add(path);
            }
public Float ComputeCost(){

            cost = (float)0.0;
            Map<Integer, List<Integer> > weights_per_origin = new
HashMap<Integer, List<Integer>>();
            Map<Integer, List<Integer> > weights_per_destination = new
HashMap<Integer, List<Integer>>();
            Map<String, List<Integer> > weights_per_flight = new
HashMap<String, List<Integer>>();

            Iterator<Path> it = paths.iterator();
            while (it.hasNext()) {
                  Path p = (Path) it.next();
                  Integer w = p.packet.weight;

                  Iterator<Connection> it_flights =
p.flight.iterator();
                  while (it_flights.hasNext()) {
                        Connection connection = (Connection)
it_flights.next();
                        Integer origin = connection.from.ID;

                        List<Integer> orig_weight;
                        if (weights_per_origin.containsKey(origin)){
                              orig_weight =
weights_per_origin.get(origin);
                        }else{
                              orig_weight = new ArrayList<Integer>();
                              weights_per_origin.put(origin,
orig_weight);
                        }
                        orig_weight.add(new Integer(w));
                  Integer destination = connection.to.ID;
                        List<Integer> dest_weight;
                        if
(weights_per_destination.containsKey(destination)){
                              dest_weight =
weights_per_destination.get(destination);
                        }else{
                              dest_weight = new ArrayList<Integer>();
                              weights_per_destination.put(destination,
dest_weight);
                        }
                        dest_weight.add(new Integer(w));

                        String connectionID = connection.getID();
                        List<Integer> flights_weight;
                        if
(weights_per_flight.containsKey(connectionID)){
```

```
                              flights_weight =
weights_per_flight.get(connectionID);
                        }else{
                              flights_weight = new
ArrayList<Integer>();
                              weights_per_flight.put(connectionID,
flights_weight);
                        }
                        flights_weight.add(new Integer(w));

                  }

            }
      Iterator<Entry<Integer, List<Integer>>> it_origins =
weights_per_origin.entrySet().iterator();
            while (it_origins.hasNext()) {
                  Map.Entry<Integer, List<Integer>> entry =
(Map.Entry<Integer, List<Integer>>) it_origins
                  .next();
                  Airport orig_airport =
getAirportFromID(entry.getKey());
                  List<Integer> weights = entry.getValue();
                  Integer total_weight = new Integer(0);
                  Iterator<Integer> wts = weights.iterator();
                  while (wts.hasNext()) {
                        Integer integer = (Integer) wts.next();
                        total_weight += integer;
                  }
                  cost +=
orig_airport.portHandling.ComputPayment(total_weight);
            }
            Iterator<Entry<Integer, List<Integer>>> it_dests =
weights_per_destination.entrySet().iterator();
            while (it_dests.hasNext()) {
                  Map.Entry<Integer, List<Integer>> entry =
(Map.Entry<Integer, List<Integer>>) it_dests
                  .next();
                  Airport dest_airport =
getAirportFromID(entry.getKey());
                  List<Integer> weights = entry.getValue();
                  Integer total_weight = new Integer(0);
                  Iterator<Integer> wts = weights.iterator();
                  while (wts.hasNext()) {
                        Integer integer = (Integer) wts.next();
                        total_weight += integer;
                  }
                  cost +=
dest_airport.clearance.ComputPayment(total_weight);

Iterator<Entry<String, List<Integer>>> it_flight =
weights_per_flight.entrySet().iterator();
            while (it_flight.hasNext()) {
```

```
                  Map.Entry<String, List<Integer>> entry =
(Map.Entry<String, List<Integer>>) it_flight
                  .next();

                  Connection conn_flight =
getConnectionFromID(entry.getKey());
                  List<Integer> weights = entry.getValue();
                  Integer total_weight = new Integer(0);
                  Iterator<Integer> wts = weights.iterator();
                  int count = 0;
                  while (wts.hasNext()) {
                        Integer integer = (Integer) wts.next();
```

```
                  count++;
                  total_weight += integer;
            }

            cost +=
conn_flight.airHandling.ComputPayment(total_weight);        }
            return cost;
      }
```

7. SAPlanner function

```
* A network planner that uses simulated annealing to do the planning
public class SAPlanner {

      Random rand = new Random();

      Plan current_plan = null;
      Plan min_plan = null;
      Plan tmp_plan = null;

      List<Airport> airports;
      List<Connection> connections;
      List<Connection> hubs;
      List<Package> packets;
      List<Airport> destinationairports;

      double temperature;
      int sameCount;
      int tries;
      int max_level = 2 ;
      // number of attempts to mutate plan
      int max_reroute = 10;
```

```
      boolean greedy;
public SAPlanner(List<Airport> airports, List<Connection> connections,
List<Package> packets,
                  List<Airport> destinationairports, double temper, int
same, int retries, boolean greedy) {
            current_plan = new Plan();
            tmp_plan = new Plan();
            min_plan = new Plan();
            sameCount = same;
            tries = retries;
            temperature = temper;
            this.airports = airports;
            this.connections = connections;
            this.packets = packets;
            this.greedy = greedy;
            this.destinationairports = destinationairports;
            this.hubs = new ArrayList<Connection>();
      }
private void MarkHubs() {
            hubs.clear();
            Iterator<Airport> ports =destinationairports.iterator();
            List<Connection> candidates = new ArrayList<Connection>();
            while (ports.hasNext()) {
                  Airport airport = (Airport) ports.next();
                  Iterator<Connection> conn = connections.iterator();
                  while (conn.hasNext()) {
                        Connection connection = (Connection)
conn.next();
                        if (connection.to.ID.equals(airport.ID) &&
                                    connection.time > 1){
                              candidates.add(connection);
                        }
                  }
                  int candidateIndex = Math.abs(rand.nextInt() %
candidates.size());
                  Connection con = candidates.get(candidateIndex);
                  System.out.println("Connection hub: "+con.getID()+"
from "+con.from.ID+" to "+con.to.ID);
                  hubs.add(con);
                  candidates.clear();
            }
      }
public Plan PlanDeployment() throws Exception {
            //if (!ready) {
            if (!Schedule())
                  return null;
            //}
            return min_plan;
      }
public boolean Schedule() throws Exception {

            List<Plan> pop = new ArrayList<Plan>();
            for (int i = 0; i < tries; i++) {
```

```
                  System.out.println("Try "+i);
                  MarkHubs();
                  ProducePlan();
                  Plan p = new Plan();
                  p.ClonePlan(min_plan);
                  pop.add(p);
//annealing plan - standard simulated annealing algorithm
      private boolean anneal(double d, double temp) {
            //          System.out.println("d = "+d+" temp = "+temp);

            //if temp low enough we keep the mutated plan (following
function: mutate plan)
            if (temp < 0.01) {
                  if (d > 0.0)
                        return true;
                  else
                        return false;
            }
            double diff = min_plan.ComputeCost() - d;
            //if we have not kept the mutated plan
            //we estimate a new one according to the following function
            if (Math.random() < Math.exp(diff / temp))
                  return true;
            else
                  return false;
      }
for (int i = 0; i < pop.size(); i++)
                  if (pop.get(i).ComputeCost() <
min_plan.ComputeCost())
                        min_plan.ClonePlan(pop.get(i));

            return true;
      }
private void ProducePlan() throws Exception {
            int same = 0;
            int cycle = 0;
            if (greedy){
                  current_plan = StartingGreedyPlan();
            }else{
                  current_plan = StartingRandomPlan();

            }

            min_plan.ClonePlan(current_plan);
            double temp = temperature;
while (same < sameCount) {

                  try {
                        Plan p = Mutate(current_plan);
                        if (anneal(p.ComputeCost(), temp)) {
                              current_plan.ClonePlan(p);
                        }
```

```
                        if (current_plan.ComputeCost() <
min_plan.ComputeCost()) {
                              min_plan.ClonePlan(current_plan);
                              System.out.println("Minimum update:
"+min_plan.ComputeCost());
                              same = 0;
                        } else
                              same++;

                        temp = 0.99 * temp;
                        cycle = 0;
                  } catch (Exception x) {
                        if (cycle > 100)
                              same++;

                  }
                  cycle++;
            }
      }
```

```
private boolean RerouteHubPacket(Plan tmpPlan, int pathindex) {
            boolean res = false;
            Path path = tmpPlan.paths.get(pathindex);
            Package p = path.packet;

            boolean usehub = rand.nextBoolean();
            if (usehub){
                  Connection hub = null;
                  Iterator<Connection> hubsit = hubs.iterator();
                  while (hubsit.hasNext()) {
                        Connection connection = (Connection)
hubsit.next();
                        if (connection.to.ID.equals(p.destination))
                              hub = connection;
                  }
                  if (hub.from.ID.equals(p.origin)){
                        path.flight.clear();
                        path.flight.add(hub);
                        res = true;
                  }else{
                        Connection hop = findQuickestFlight(p.origin,
hub.from.ID, hub.time);
                        if (hop != null){
                              path.flight.clear();
                              path.flight.add(hop);
                              path.flight.add(hub);
                              res = true;
                        }
                  }
      }else{
```

```
                  Connection direct = findQuickestFlight(p.origin,
p.destination, p.deadline);
                  if (direct != null){
                        path.flight.clear();
                        path.flight.add(direct);
                        res = true;
                  }
            }

            return res;
      }
private Plan StartingGreedyPlan() throws Exception {
            Plan p = new Plan(airports, connections);
            for (int pack = 0 ; pack < packets.size(); pack++){
                  Path path = findGridyPath(packets.get(pack));
                  path.packet = packets.get(pack);
                  p.paths.add(path);
            }
            return p;
      }
      private Plan StartingRandomPlan() throws Exception {
            Plan p = new Plan(airports, connections);
            for (int pack = 0 ; pack < packets.size(); pack++){
                  Path path = findRandomPath(packets.get(pack));
                  path.packet = packets.get(pack);
                  p.paths.add(path);
            }
            return p;
}
```

```
Path findGridyPath(Package packet) throws Exception{
            for(int len = 1; len <= max_level; len++ ){
                  Path p = findAnyRandomPathOfLength(len, packet, new
Path());
                  if (p != null)
                        return p;
            }
            throw new Exception("Cannot find path for packet
"+packet.ID);
      }
Path findRandomPath(Package packet) throws Exception{
            For (int len = max_level; len > 0; len-- ){




                  Path p = findAnyRandomQuickPathOfLength(len, packet,
new Path());
                  if (p != null)
                        return p;
            }
            throw new Exception("Cannot find path for packet
"+packet.ID);
```

8. Test me Function

```
public class TestMe {

	/**
	 * @param args
	 */
	public static void main(String[] args) {
		List<Airport> airports = new ArrayList<Airport>();
		List<Airport> destinationairports = new
ArrayList<Airport>();
		List<Connection> connections = new ArrayList<Connection>();
		List<Package> packets = new ArrayList<Package>();

		NoPayment paymentFunction = new NoPayment();

		AthensFrankfurtPayment athfranpaymentFunction = new
AthensFrankfurtPayment ();
		HamburgAthensPayment hamathpaymentfunction = new
HamburgAthensPayment();
		HamburgTokyoPayment hamtokpaymentfunction = new
HamburgTokyoPayment();
		KopenhagenHongKongPayment kophonpaymentfunction = new
KopenhagenHongKongPayment();
		KopenhagenMalagaPayment kopmalpaymentfunction = new
		MalagaAthensPayment malathpaymentfunction = new
MalagaAthensPayment();
		RotterdamAthensPayment rottathpaymentfunction = new
RotterdamAthensPayment();
		RotterdamFrankfurtPayment rotfranpaymentfunction = new
RotterdamFrankfurtPayment();
		RotterdamClearancePayment rotclear = new
RotterdamClearancePayment();
		ShanghaiClearancePayment shanclear = new
ShanghaiClearancePayment();
		TokyoClearancePayment tokclear = new
TokyoClearancePayment();
```

```
Airport Athens = new Airport("Athens", 1, paymentFunction,
paymentFunction);
		Airport Frankfurt = new
Airport("Frankfurt",2,paymentFunction, paymentFunction);
		Airport Genova = new Airport("Genova",3,paymentFunction,
paymentFunction);
		Airport Hamburg = new Airport("Hamburg",4,paymentFunction,
paymentFunction);
		Airport HongKong = new
Airport("HongKong",5,paymentFunction, paymentFunction);
		Airport Kopenhagen = new
Airport("Kopenhagen",6,paymentFunction, paymentFunction);
```

```
            Airport Malaga = new Airport("Malaga",7,paymentFunction,
paymentFunction);
            Airport Rotterdam = new
Airport("Rotterdam",8,paymentFunction, paymentFunction);
            Airport Shanghai = new
Airport("Shanghai",9,paymentFunction, paymentFunction);
            Airport Tokyo = new Airport("Tokyo",10,paymentFunction,
paymentFunction);
// initial plans two alternatives:
            // true -> greedy
            // false -> random
            SAPlanner planner = new SAPlanner(airports, connections,
packets, destinationairports, 1,
                        1000, 50, true);

            try {
                  Plan goodplan = planner.PlanDeployment();
                  goodplan.Print();
            } catch (Exception x) {
                  x.printStackTrace();
            }
      }
private static void ReadPackets(List<Package> packets, String fName,
List<Airport> airports, List<Airport> destinations) {
            String thisLine; // string variable which take each record
at a time
            try {
                  FileInputStream fis = new FileInputStream(fName);
                  DataInputStream myInput = new DataInputStream(fis);
                  List<Integer> airIDs = new ArrayList<Integer>();


                  while ((thisLine = myInput.readLine()) != null) { //
beginning of outer
                        // while loop
                        StringTokenizer st = new
StringTokenizer(thisLine, "\t");
                        String ID = "";
                        Integer deadline = 0;
                        Integer destination = 0;
                        Integer origin = 0;
                        Integer weight = 0;
                        int count = 0;
                        while (st.hasMoreElements()) {
                              String field = st.nextToken();
                              System.out.print(field + ", ");
                              if (count == 0)
                                    ID = field;
                              if (count == 1)
                                    deadline = Integer.parseInt(field);
                              if (count == 3)
                                    origin = Integer.parseInt(field);
                              if (count == 2){
```

```
                                    destination =
Integer.parseInt(field);
                                    Integer destID = new
Integer(destination);
                                    if (!airIDs.contains(destID))
                                          airIDs.add(destID);

                              }
                              if (count == 4){
                                    weight = Integer.parseInt(field);
                                    Package p = new Package(ID,
deadline, destination, origin, weight);
                                    packets.add(p);
                              }
                              count++;
                        }
                        System.out.println();
                  }
                  Iterator<Integer> airIt = airIDs.iterator();
                  //loops over the IDs of the destination airports
                  while (airIt.hasNext()) {
                        Integer id = (Integer) airIt.next();
                  //we find the airport using each ID
                        Airport airport = getAirportFromID(airports,
id);
                  //adds the airprot to the destinations, in case it is
not already there
                        if (!destinations.contains(airport)){
                              destinations.add(airport);
                        }
                  }
                  System.out.println("Total destinations: "+
destinations.size());

            } catch (IOException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            } // ending of outer while loop
      }
```

```
private static Airport getAirportFromID(List<Airport> airports, Integer
ID){
            for(int i = 0; i < airports.size();i++){
                  if (airports.get(i).ID.equals(ID))
                        return airports.get(i);
            }
            return null;
      }
}
```

Appendix 2 The optimal plan

Cost: 45804.703

Routing 1 from 1 to 5 no later than 2	Air Connections: (4, 5):1
Air Connections: (1, 5):1	Routing 12 from 7 to 5 no later than 3
Routing 2 from 4 to 9 no later than 2	Air Connections: (7, 5):1
Air Connections: (4, 9):1	Routing 13 from 8 to 5 no later than 3
Routing 3 from 8 to 5 no later than 2	Air Connections: (8, 5):1
Air Connections: (8, 5):1	Routing 14 from 7 to 9 no later than 3
Routing 4 from 6 to 5 no later than 2	Air Connections: (7, 2):1 (2, 9):2
Air Connections: (6, 5):1	Routing 15 from 1 to 9 no later than 3
Routing 5 from 4 to 10 no later than 2	Air Connections: (1, 9):1
Air Connections: (4, 10):1	Routing 16 from 7 to 10 no later than 3
Routing 6 from 8 to 9 no later than 2	Air Connections: (7, 10):1
Air Connections: (8, 9):1	Routing 17 from 6 to 10 no later than 3
Routing 7 from 1 to 9 no later than 2	Air Connections: (6, 10):1
Air Connections: (1, 9):1	Routing 18 from 3 to 9 no later than 3
Routing 8 from 7 to 5 no later than 2	Air Connections: (3, 9):1
Air Connections: (7, 5):1	Routing 19 from 7 to 5 no later than 3
Routing 9 from 2 to 10 no later than 2	Air Connections: (7, 5):1
Air Connections: (2, 10):1	Routing 20 from 4 to 10 no later than 3
Routing 10 from 6 to 9 no later than 3	Air Connections: (4, 10):1
Air Connections: (6, 9):1	Routing 21 from 2 to 5 no later than 3
Routing 11 from 4 to 5 no later than 3	Air Connections: (2, 5):1

Routing 22 from 8 to 10 no later than 3

Air Connections: (8, 10):1

Routing 23 from 2 to 9 no later than 3

Air Connections: (2, 9):2

Routing 24 from 7 to 10 no later than 4

Air Connections: (7, 10):1

Routing 25 from 3 to 9 no later than 4

Air Connections: (3, 9):1

Routing 26 from 7 to 10 no later than 4

Air Connections: (7, 10):1

Routing 27 from 4 to 9 no later than 4

Air Connections: (4, 2):1 (2, 9):2

Routing 28 from 6 to 5 no later than 4

Air Connections: (6, 5):1

Routing 29 from 1 to 10 no later than 4

Air Connections: (1, 10):1

Routing 30 from 7 to 5 no later than 5

Air Connections: (7, 5):1

Routing 31 from 4 to 5 no later than 5

Air Connections: (4, 5):1

Routing 32 from 2 to 10 no later than 5

Air Connections: (2, 8):1 (8, 10):4

Routing 33 from 8 to 10 no later than 5

Air Connections: (8, 10):4

Routing 34 from 6 to 9 no later than 5

Air Connections: (6, 9):1

Routing 35 from 1 to 10 no later than 5

Air Connections: (1, 8):1 (8, 10):4

Routing 36 from 8 to 5 no later than 5

Air Connections: (8, 5):1

Routing 37 from 3 to 10 no later than 5

Air Connections: (3, 8):1 (8, 10):4

Routing 38 from 7 to 9 no later than 5

Air Connections: (7, 2):1 (2, 9):2

Routing 39 from 1 to 5 no later than 6

Air Connections: (1, 5):1

Routing 40 from 8 to 5 no later than 6

Air Connections: (8, 5):3

Routing 41 from 7 to 5 no later than 6

Air Connections: (7, 5):1

Routing 42 from 8 to 10 no later than 6

Air Connections: (8, 10):4

Routing 43 from 7 to 9 no later than 6

Air Connections: (7, 2):1 (2, 9):2

Routing 44 from 1 to 10 no later than 6

Air Connections: (1, 10):1

Routing 45 from 3 to 9 no later than 6

Air Connections: (3, 9):1

Routing 46 from 2 to 5 no later than 6

Air Connections: (2, 5):1

Routing 47 from 8 to 10 no later than 6

Air Connections: (8, 10):1

Routing 48 from 6 to 5 no later than 6

Air Connections: (6, 5):1

Routing 49 from 2 to 10 no later than 7

Air Connections: (2, 10):1

Routing 50 from 8 to 5 no later than 7

Air Connections: (8, 5):1

Routing 51 from 7 to 5 no later than 7

Air Connections: (7, 5):1

Routing 52 from 4 to 10 no later than 7

Air Connections: (4, 10):1

Routing 53 from 6 to 9 no later than 7

Air Connections: (6, 9):1

Routing 54 from 1 to 10 no later than 7

Air Connections: (1, 10):1

Routing 55 from 2 to 5 no later than 7

Air Connections: (2, 5):1

Routing 56 from 4 to 10 no later than 7

Air Connections: (4, 10):1

Routing 57 from 8 to 9 no later than 7

Air Connections: (8, 9):1

Routing 58 from 1 to 9 no later than 7

Air Connections: (1, 9):1

Routing 59 from 7 to 9 no later than 7

Air Connections: (7, 2):1 (2, 9):2

Routing 60 from 4 to 9 no later than 7

Air Connections: (4, 2):1 (2, 9):2

Routing 61 from 7 to 10 no later than 7

Air Connections: (7, 10):1

Routing 62 from 1 to 5 no later than 7

Air Connections: (1, 5):1

Routing 63 from 2 to 9 no later than 7

Air Connections: (2, 9):2

Routing 64 from 8 to 5 no later than 8

Air Connections: (8, 5):1

Routing 65 from 3 to 5 no later than 8

Air Connections: (3, 8):1 (8, 5):3

Routing 66 from 1 to 10 no later than 8

Air Connections: (1, 10):1

Routing 67 from 4 to 9 no later than 8

Air Connections: (4, 2):1 (2, 9):2

Routing 68 from 8 to 10 no later than 8

Air Connections: (8, 10):4

Routing 69 from 6 to 5 no later than 8

Air Connections: (6, 5):1

Routing 70 from 1 to 5 no later than 8

Air Connections: (1, 5):1

Routing 71 from 3 to 10 no later than 8

Air Connections: (3, 8):1 (8, 10):4

Routing 72 from 4 to 9 no later than 8

Air Connections: (4, 9):1

Routing 73 from 6 to 10 no later than 8

Air Connections: (6, 10):1

Routing 74 from 2 to 9 no later than 8

Air Connections: (2, 9):2

Routing 75 from 8 to 5 no later than 8

Air Connections: (8, 5):3

Routing 76 from 2 to 10 no later than 8

Air Connections: (2, 10):1

Routing 77 from 3 to 5 no later than 8

Air Connections: (3, 8):1 (8, 5):3

Routing 78 from 1 to 9 no later than 9

Air Connections: (1, 9):1

Routing 79 from 8 to 9 no later than 9

Air Connections: (8, 9):1

Routing 80 from 2 to 10 no later than 9

Air Connections: (2, 10):1

Routing 81 from 4 to 9 no later than 9

Air Connections: (4, 9):1

Routing 82 from 4 to 10 no later than 9

Air Connections: (4, 10):1

Routing 83 from 1 to 10 no later than 9

Air Connections: (1, 10):1

Routing 84 from 3 to 9 no later than 9

Air Connections: (3, 9):1

Routing 85 from 4 to 5 no later than 9

Air Connections: (4, 5):1

Routing 86 from 8 to 10 no later than 9

Air Connections: (8, 10):1

Routing 87 from 2 to 9 no later than 10

Air Connections: (2, 9):2

Routing 88 from 3 to 9 no later than 10

Air Connections: (3, 9):1

Routing 89 from 8 to 10 no later than 10

Air Connections: (8, 10):4

Routing 90 from 6 to 9 no later than 10

Air Connections: (6, 9):1

Routing 91 from 4 to 5 no later than 10

Air Connections: (4, 5):1

Routing 92 from 1 to 10 no later than 10

Air Connections: (1, 8):1 (8, 10):4

Routing 93 from 2 to 9 no later than 10

Air Connections: (2, 9):2

Routing 94 from 8 to 10 no later than 10

Air Connections: (8, 10):1

Routing 95 from 6 to 5 no later than 10

Air Connections: (6, 5):1

Routing 96 from 4 to 10 no later than 10

Air Connections: (4, 10):1

Routing 97 from 7 to 10 no later than 10

Air Connections: (7, 10):1

Routing 98 from 3 to 5 no later than 10

Air Connections: (3, 8):1 (8, 5):3

Routing 99 from 8 to 10 no later than 11

Air Connections: (8, 10):4

Routing 100 from 4 to 9 no later than 11

Air Connections: (4, 9):1

Routing 101 from 4 to 10 no later than 11

Air Connections: (4, 10):1

Routing 102 from 1 to 5 no later than 11

Air Connections: (1, 5):1

Routing 103 from 8 to 10 no later than 11

Air Connections: (8, 10):4

Routing 104 from 2 to 5 no later than 11

Air Connections: (2, 5):1

Routing 105 from 3 to 9 no later than 11

Air Connections: (3, 9):1

Routing 106 from 8 to 10 no later than 11

Air Connections: (8, 10):1

Routing 107 from 6 to 9 no later than 11

Air Connections: (6, 9):1

Routing 108 from 4 to 5 no later than 11

Air Connections: (4, 5):1

Routing 109 from 2 to 5 no later than 11

Air Connections: (2, 5):1

Routing 110 from 1 to 9 no later than 12

Air Connections: (1, 9):1

Routing 111 from 6 to 10 no later than 12

Air Connections: (6, 10):1

Routing 112 from 4 to 5 no later than 12

Air Connections: (4, 5):1

Routing 113 from 8 to 9 no later than 12

Air Connections: (8, 9):1

Routing 114 from 6 to 9 no later than 12

Air Connections: (6, 9):1

Routing 115 from 4 to 5 no later than 12

Air Connections: (4, 5):1

Routing 116 from 6 to 10 no later than 12

Air Connections: (6, 10):1

Routing 117 from 1 to 9 no later than 12

Air Connections: (1, 2):1 (2, 9):2

Routing 118 from 6 to 10 no later than 12

Air Connections: (6, 10):1

Routing 119 from 8 to 5 no later than 12

Air Connections: (8, 5):3

Routing 120 from 4 to 9 no later than 12

Air Connections: (4, 9):1

Connections flight 2, 9 total cost: 4130.4

Connections flight total weight: 684

Connections flight 4, 9 total cost: 2247.0

Connections flight total weight: 330

Connections flight 6, 10 total cost: 1610.8

Connections flight total weight: 226

Connections flight 8, 9 total cost: 2395.5

Connections flight total weight: 381

Connections flight 4, 10 total cost: 2239.8

Connections flight total weight: 318

Connections flight 6, 9 total cost: 1554.4

Connections flight total weight: 224

Connections flight 1, 5 total cost: 2304.8

Connections flight total weight: 358

Connections flight 3, 9 total cost: 1751.8

Connections flight total weight: 238

Connections flight 2, 5 total cost: 1688.8

Connections flight total weight: 248

Connections flight 2, 8 total cost: 81.0

Connections flight total weight: 18

Connections flight 2, 10 total cost: 1497.2

Connections flight total weight: 164

Connections flight 1, 10 total cost: 2483.8

Connections flight total weight: 358

Connections flight 4, 5 total cost: 2921.6

Connections flight total weight: 452

Connections flight 7, 2 total cost: 623.5

Connections flight total weight: 215

Connections flight 7, 10 total cost: 1654.5

Connections flight total weight: 215

Connections flight 8, 10 total cost: 1810.5

Connections flight total weight: 265

Connections flight 6, 5 total cost: 1862.4

Connections flight total weight: 279

Connections flight 7, 5 total cost: 1733.5

Connections flight total weight: 235

Connections flight 1, 8 total cost: 385.0

Connections flight total weight: 110

Connections flight 8, 5 total cost: 1939.0

Connections flight total weight: 298

Connections flight 3, 8 total cost: 1000.50006

Connections flight total weight: 345

Connections flight 8, 10 total cost: 3623.1

Connections flight total weight: 583

Connections flight 1, 9 total cost: 2288.3

Connections flight total weight: 337

Connections flight 1, 2 total cost: 55.0

Connections flight total weight: 10

Connections flight 8, 5 total cost: 1416.5

Connections flight total weight: 203

Connections flight 4, 2 total cost: 506.0

Connections flight total weight: 220