Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization: Master in Computer Science	Spring semester, 2014 Restricted access
Writer: Wei Liao	…… (Writer's signature)
Faculty supervisor: DR. R. DAVIDRAJUH External supervisor(s): Shengtong Zhong	
Title of thesis: Fleet Scheduling & Optimization	
Credits (ECTS): 30	
Key words: Maritime Inventory Routing Stochastic Optimization	Pages: …38…… +enclosure: …1 CD…… Stavanger 15th June, 2014

Acknowledgements

I would like to express my gratitude to Arild Vervik, who is the manager of Tieto Corporation, for giving me this opportunity to work on this project. Also I would like to thank Shentong Zhong, my external supervisor for his helpful suggestions and guide.

I would like to thank my faculty supervisor DR. Reggie Davidrajuh, professor in Department of Computer Science of UIS, who carefully reviewed my thesis and gave me a lot of advices.

Additionally I would like to thank the University of Stavanger for giving me the opportunity of learning the master study program, which gave me a lot of wonderful experiences and memories.

Contents

Abstract

In this thesis, we introduce a multi-seeds stochastic optimization algorithm for MIRP (Maritime Inventory Routing Problem) in the LNG business. A liquefied natural gas (LNG) producer usually has a liquefaction plan with limited capacity of inventory, a loading port, a heterogeneous fleet of LNG ships, a list of world wild customers and a set of contracts.

The goal of this thesis is a case study for one of the world's largest LNG producer, and tries to create an annual delivery plan (ADP) to arrange the voyages of fleet at minimum cost to fulfill all the contracts.

Multi-seeds stochastic optimization algorithm first randomly generate several ADPs as seeds, and then try to optimize each of them by using stochastic optimization algorithm. At last we choose the best ADP among them.

1.Introduction

1.1 What is LNG

"Liquefied natural gas (LNG) is natural gas (predominantly methane, CH4) that has been converted to liquid form for ease of storage or transport [1]". Natural gas can be condensed into liquid form at a temperature of -256ºF(-161ºC) at atmospheric pressure. Liquefaction reduces the volume of gas by approximately 600 times thus making it more economical to store and transport over long distance.

1.2 Brief history of LNG business

The first LNG plant was built in West Virginia in 1912 and started working in 1917. In early 1959, the Methane Pioneer which is the first LNG ship in the world, carried 7000 barrels LNG cargos from Lake Charles, Louisiana to Canvey Island, United Kingdom. This first voyage demonstrated that large quantities of liquefied natural gas could be transported across oceans.

Back to 1964, Algeria built a natural gas liquefaction plant named Arzew GL4Z, it became the first LNG importer and United Kingdom is the first customer.
Today, global demand for energy is significantly increasing. Impacted by the Tragedy nuclear flight from Japan's Fukushima nuclear plant three year ago, more and more countries consider LNG as an alternative energy source other than nuclear. Since the LNG suppliers and customers spread across mid-east, Asia, North-American and Europe, the traditional transportation through pipelines is not feasible any more(Figure 1.1 shows the change of supply and demand of LNG from 2005 to 2010 [2]). However, the LNG fleet is not a bad solution for long distance, high volume transportation of LNG cargo. As a result, the total LNG fleet has grown from 105 ships in 1998 to 365 ships in 2012, and this number will keep growing rapidly in the next 10 years.

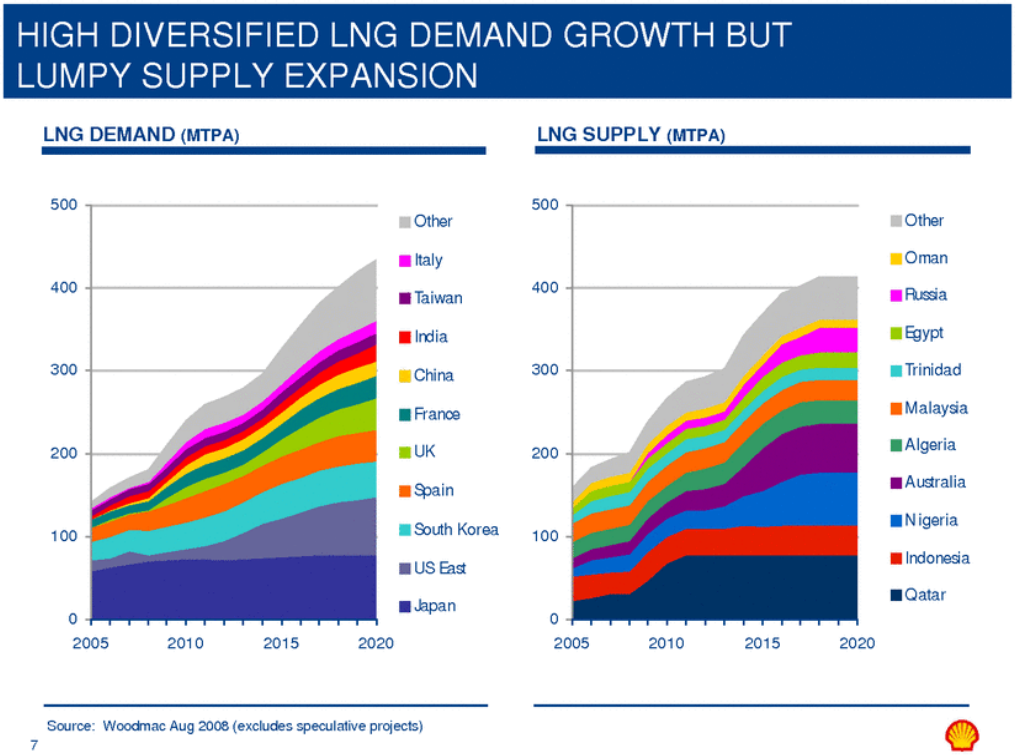

Figure 1.1: supply and demand of LNG from 2005 to 2010 [2]

1.3 LNG supply chain

In order to show a better picture of the LNG business, this section introduces how LNG supply chain works.

The natural gas is extracted from onshore or offshore gas fields and is sent through pipelines to liquefaction plants where the natural gas is liquefied into LNG. The natural gas consists of many impurities such hydrogen sulphide, water, and other heavier hydrocarbons, so the first step of liquefying process is to remove all of them especially which could result in freeze or blockages. After purifying process, the ideal natural gas should be mainly methane with little propane, ethane and nitrogen. In the end, volume of LNG has been reduced as 1 out of 600 its original volume. After all these process, LNG products are stored in special designed storage tanks.

To transport LNG products to customer, we need LNG vessels which are designed only for LNG products. A typical vessel may have four or six tanks located along the center-line of the vessel, and each tank has a double-shell which consists of ballast tanks, cofferdams and voids [4]. Unlike other normal vessels, LNG vessels not

only have high capacity of maximum volume, but also have a low capacity of minimum volume. That is because if the whole tank is empty, it is hard to keep the storage tanks cool. So a LNG vessel can be fully loaded, but cannot be fully discharged, there is always a small amount of LNG products left in its storage tanks. The LNG products stored in the storage tanks on the LNG vessel is cooled down to approximately -163 °C at normal atmospheric pressure, but it is still boiling off during the voyage. According to WGI, the boil-off rate of a typical voyage is estimated 0.1–0.25% of total cargoes each day.[5] Since a typical voyage lasts around 20 to 30 days, 2–7.5% of the total volume of LNG products may be lost. But good news is that this boil-off gas is not totally wasted, they could be used as fuel for the LNG vessel. Furthermore, the latest report says that a new cargo containment system has been developed to reduce the boil-off rate from average 0.15% to 0.1%. [6]

Once the LNG vessel arrive the consumption port, LNG products can either be stored in the storage tanks for later use or directly sent to the regasification plants. At regasification plants, LNG will be vaporized to natural gas and sent to end users through pipelines.

Figure 1.2 illustrates the whole supply chain of LNG business.

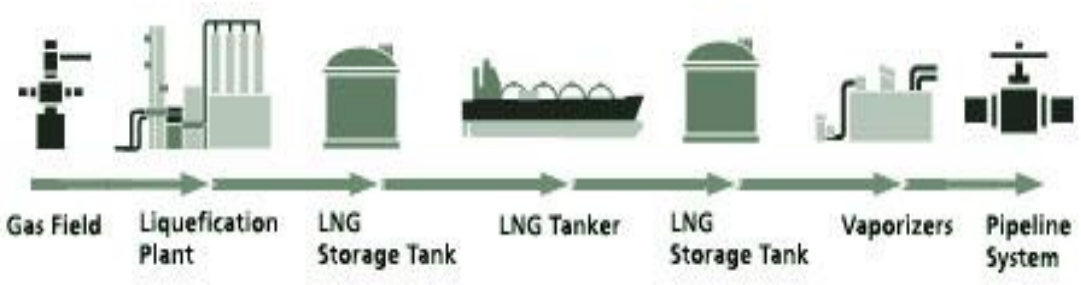

Figure 1.2: LNG supply chain, source: CMS Energy [3]

1.4 Maritime LNG transportation and inventory management

Maritime LNG transportation and inventory management can be seen as a combination of two typical problems, which are Maritime Transportation Problem and Inventory Routing Problem.

As we know, there are three general operation modes in Maritime Transportation Problem: liner, tramp and industrial. In liner mode, ships follow a certain itinerary and schedule like a public bus. A liner ship company tries to maximum the profit from carrying cargos. In tramp mode, ships are more similar as taxi, they follow available cargos. A tramp ship company may have some mandatory contracts and some optional cargoes. So the objective for a tramp ship company is to carry as much

optional cargos during the voyage of carrying the mandatory cargoes. Industrial shipping operators usually are the owners of cargoes and ships. Therefore, their goals are to minimize the shipping cost. In the LNG industry, they operate like combined tramp and industrial mode and usually the LNG producer own the whole fleet. Ships pick up LNG from its liquefaction plant(s), and directly deliver it to long-term customers. If the production of LNG is greater than long-term demands, selling LNG on spot market is also an option.

Inventory Routing Problem (IRP) is defined as a central supplier and a set of customers, where supplier and customers maintain an inventory of one or more types of products. Inventory Routing Problem focuses on two main factors: inventory management and vehicle routing. A balance between the two plays an important role in solving IRP. Maritime Inventory Routing Problem (MIRP) can be seen as a special type of IRPs which has maritime features. Road based IRPs usually have several types of products to delivery, and deliver small amount to several different customers in a single trip. While in MIRPs, one kind of product is carried and the full ship load product is delivered to one customer in one voyage.

Typical MIRP has two main components: ports and vessels. Port can be either a production port or a consumption port. At production port, cargoes are produced and loaded onto vessels. At consumption port, cargoes are unloaded from vessels and consumed. Inventory should be managed in both of production port and consumption port, and there is a high boundary for each port that cargoes cannot exceed. Also, each port may have some other properties: a fixed number of berths that allow limited vessels to load or discharge simultaneously in a time period; production/consumption rate which may differ in different time period. Vessels may have different capacities, cruising speed and travel cost.

1.5 Literature in MIRPs

In this section, we present a review of papers about MIRPs. Since the rapid growth in oil and gas industry last decade, maritime transportation attracted more and more attentions. A lot of researches have been done in MIRPs especially from Norway and Canada.

Geir BrZnmo (2005) [7] present a multi-start local search heuristic for ship scheduling problem, which not only has advantages in computation time, but also in solution quality. However the problem Geir BrZnmo (2005) [7] work on is similar with tramp shipping mode, not involved in inventory management. Another researcher Magnus Stålhane (2009) [8] who presents a similar way called a construction and improvement heuristic to solve Maritime Inventory Routing Problem. It first creates an Annual Delivery Plan (ADP), and then improves this ADP

iteratively by using local search heuristic. Two types of LNG (LLNG and RLNG) are produced by the LNG producer, but only one type of LNG can be carried on one ship. If one ship wants to carry the other type of LNG in the next voyage, the storage tanks on this ship need to perform a fully cleaning process which cost a lot of money. Rakke (2011) [9] who works with Stålhane gives another method called rolling horizon heuristic to create ADP. It divides the whole planning horizon into shorter horizons, and it solves the large scale of planning horizon by solving the subproblems. Grønhaug and Christiansen (2009) (2010) [10][11]bring us an even more complex structure of MIRP. This problem has more than one production ports, inventory management must be applied at both production ports and consumption ports. Ships can pick up cargoes at multiple production ports in one voyage. However, this problem has less ships and limited planning horizon otherwise it will not be feasible for computing.

Papageorgiou (2013) [13] introduces a library for MIRPs, its main goal is "to help maritime inventory routing gain maturity as an important and interesting class of planning problems". [13] It focuses on the core mode described by Christiansen and Fagerholt (2009) [14], and provides MIRPs instances sets and best known results as well. Someone who works on MIRPs can test their solutions with the instances and compare the results with the best known one.

1.6 Problem description

In before, the LNG producers schedule their fleets by man power. However, with rapid increasing of fleets and customers, it makes it impossible to arrange the delivery plan of whole year manually. Thus, a decision support system is needed in order to help people to accomplish this difficult task.

This is a realistic case study, in this thesis we will present a solution of the Maritime Inventory Routing Problem for one of the world's largest LNG producer. This LNG producer has several natural gas wells which produce natural gas and transport it to the liquefaction plants through pipeline. In liquefaction plan, natural gas is liquefied and stored in several storage tanks with limited capacity. Unlike the problem described in Magnus Stålhane (2009) [8], there is only one type of LNG product. The inventory level of those storage tanks should always be with upper and lower bounds. In this specific case, we do not have to manage the inventory level of customers' ports, and we assume they are all well managed by customers themselves. To us, the capacity of customers' ports is infinite, and we can unload as much LNG cargoes as possible.

The production rate of LNG is pre-estimated before planning. It is not fixed, but varies in independent time window. The gas wells, the pipelines, and also the

liquefaction plants have maintenance plan, so the production rate will decrease during that time window.

The LNG producer owns a large number of heterogeneous fleet, and all ships are supposed to stay not far from the production port at the start of planning. But in realistic situation, some ships might be on voyage to customer or just sailing back to production port. So if any ships are not available at some time windows, we need to mark it as unavailable and also set a time when it is available again. The unavailable time window can also be used when ships are in maintenance state.

The voyage is defined as a round trip, starting from the production port, unloading at one of customers' port, and sailing back to the production port. Another constraint is that all ships are fully loaded at production port and fully unloaded at one consumption port. In section 1.3 we mentioned that the storage tanks on LNG ships cannot be completely empty because it is hard to keep low temperature. So the concept of capacity of LNG ship is actually not real capacity, it equals to the real capacity minus the low bound of the storage tanks on the ship. The travel time and cost from production port to one of consumption ports by a given ship are fixed, so we do not need to calculate it by speed, distance, etc. Besides, the loading and unloading time is also included in the voyage time.

The contracts are usually a one year contract with specific requirements in different time intervals. The contract defines destination port, low bound and high bound of LNG product in both long term goal and monthly goal. Under-delivery or over-delivery are not encouraged but allowed in our case, so we will try to avoid them. In reality, penalty should be considered for under-delivery or over-delivery, but for our case we just ignore it. The price per volume of each contract is fixed and different from different customers.

A contract, a ship and a start date, the three elements create a voyage. A Contract has a departure port and a destination port, with a certain ship we know how much time and money this voyage costs. We know that ship is full loaded, so the LNG product on this ship is the max capacity. The voyage end date can be calculated since we know the voyage time and the start date. If we say a ship is on a voyage, which means this ship is not available for any other voyage from the start date to end date. In additional, a voyage's profit can be calculated by price per volume in the contract multiple the max capacity of the ship, then minus the cost of this voyage.

An ADP consists of a set of voyages, and a feasible ADP must be well scheduled so that the inventory level of every day through the whole planning horizon stays between the low bound and the high bound. Our goal is to create a feasible and max profit ADP. To achieve this goal, we will use multi-seeds, stochastic optimization algorithm which will be presented in section 2 and 3.

2. Mathematical Model

In this section, we will present the mathematical mode for our case.

Let $\mathcal{V}$ be the set of all LNG ships.

Let v be one of the LNG ships, $v \in \mathcal{V}$.

Let v^m be the maximum LNG cargo ship v can carry.

Let C be the set of all contracts.

Let c be one of the contracts, $c \in C$.

Let c^p be the price per volume of LNG for contract c.

Let c^- be the low bound of contract c.

Let c^+ be the high bound of for contract c.

Let c^s be the shipped volume of LNG for contract c.

Let $\mathcal{T}$ be the set of time period in the planning horizon.

Let I^t be the inventory level of day t.

Let I^- be the minimum inventory level.

Let I^+ be the maximum inventory level.

Let Cost^{cv} be the cost of ship v executes contract c.

2.1 The Constraints

In this mathematical description, we use a binary variable y^{cvt}. It stands for 1 if ship v serves contract c on day t, otherwise it is 0. Then the set partitioning formulation for our case can be given as follows:

$$Max \quad \sum_{c\in C}\sum_{v\in V}\sum_{t\in T} c^{p} * v^{m} * y^{cvt} - \sum_{c\in C}\sum_{v\in V}\sum_{t\in T} Cost^{cv} * y^{cvt} \tag{1}$$

$$I^{-} \leq \quad I^{t} \quad \leq \quad I^{+} \qquad \forall t \in \mathcal{T}. \tag{2}$$

$$c^{-} \leq \quad c^{s} \leq \quad c^{+} \qquad \forall c \in C. \tag{3}$$

$$y^{cvt} \in \{0,1\} \quad \forall v \in \mathcal{V}, c \in C, t \in \mathcal{T} \tag{4}$$

The objective function (1) maximizes the profit gained by LNG ships serving the contracts minus the total voyage cost. Since ships are fully loaded for every voyage, the profit can be calculated by the price per volume of contract multiple the maximum capacity of ship. In our case, we assume the voyage cost is fixed for a certain ship serving a certain contract, which includes fuel, porting service fuel and human cost etc. Constraints (2) ensure that the daily inventory level of LNG must sit between the high limit and low limit. Constraints (3) ensure that the shipped volume of LNG does not exceed the high limit and not below the low limit of the contract. At last, constraints (4) impose the binary requirement on the variable.

2.2 Feasible Delivery Plan

This section introduces the initial delivery plan construction and how to improve it by stochastic optimization algorithm. A delivery plan consists of several voyages, and each voyage contains one ship, one contract and a certain starting date. The time horizon of a delivery plan can be several weeks, months or a whole year which is called annual delivery plan (ADP). A feasible delivery plan must satisfy the two constraints (2) and (3) we mentioned in previous section. The complexity of creating a delivery plan grows with the time horizon, and it could be very difficult and extremely time-consuming to find a feasible ADP.

Once we have some initial feasible ADPs, we run our optimization algorithm on each ADP. Then we choose the best one among them as our final ADP.

Figure 3.1 shows the main workflow of our solution

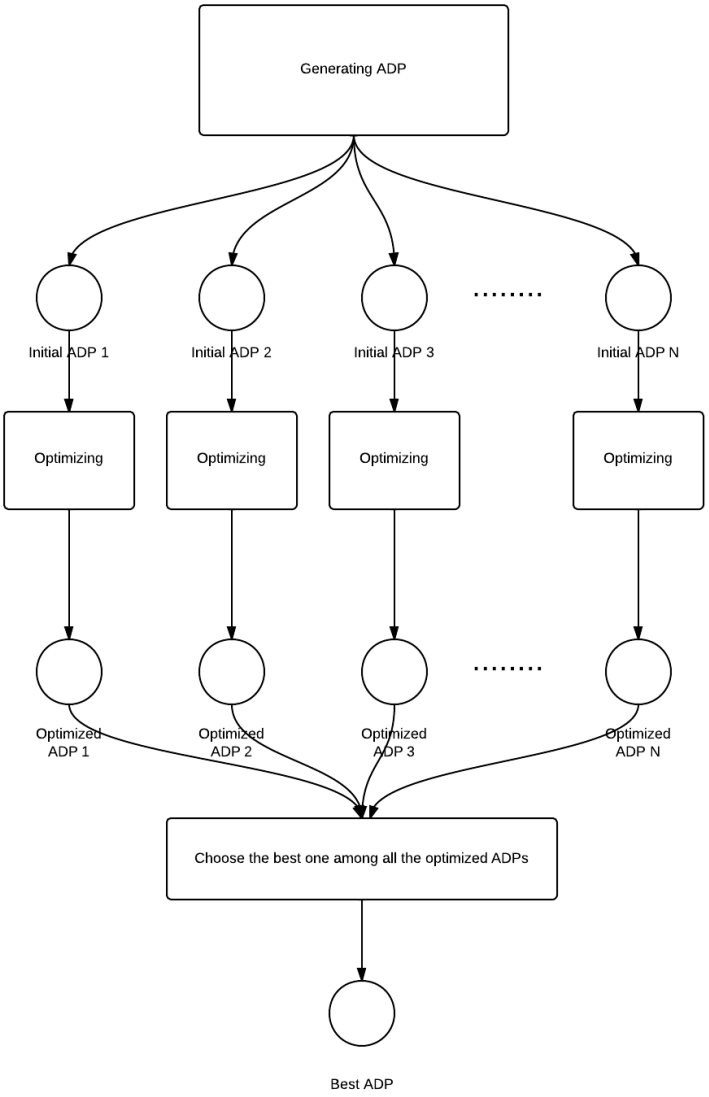

Figure 3.1 Main Workflow

2.3 Generate initial delivery plan

In order to present my solution better, we need introduce more symbols here:

Let $\mathcal{V}^t$ be the set of all available LNG ships on day t. $\mathcal{V}^t \subseteq \mathcal{V}$

Let C^t be the set of all available contracts on day t. $C^t \subseteq C$

2.3.1 Original idea

The original idea of creating a delivery plan is as follows:

1. From start date to end date of planning horizon, for each day t we randomly choose a contract c from C^t and a ship v from V^t to form a voyage until the inventory level of producer's storage is lower than the capability of the smallest available ship .
2. Update the inventory level for next day and repeat step 1.
3. Until the whole planning horizon ends.

Unfortunately this idea has two defects:

A. Since contract c is randomly chosen, it leads to unbalanced result. Some contracts may be over executed, but some contracts' shipped cargo may not reach the low limit.

B. This defect is related to random ship chosen process. Considering a scenario that on day t, there are two ships available and the inventory level only allows one of the ships carrying out a voyage. Further, if ship 1 is chosen, the remained inventory level plus the daily incremental will exceed the high limit of the storage tank. Only ship 2 is chosen can secure the inventory level of the next day stay in between the low and high limit. Since there is no such mechanism in the original idea to ensure the inventory level of the next day, the overflow LNG could be huge.

To solve these two defects, some constraints and terms are introduced into the construction of a delivery plan.

i) Ranking system for contracts:

Adding ranking for each contract is to avoid defect A. We want all contracts to be evenly spread, and the most important goal is that the low limit of each contract should be satisfied. In the meaning while, the price of LNG for each contract is different and it is also an important fact to determine the ranking value. Pseudo code to calculate ranking for each contract is given in algorithm 1.

```
Input : D = current date
Output : ranking of this contract
if D ≥ expire date of this contract then
return 0
end if
rank = price per unit
if shipped cargo of this contract ≤ low limit of this contract then
rank = rank + 10
end if
if expire date of this contract – D ≤ 30 days then
rank = rank + 20
end if
return rank
```

Algorithm 1. Calculate contract's ranking

We sort all available contracts C^t by their ranking, and we get C^{ts} which means sorted available contracts on day t. We do not randomly choose a contract but always choose the highest ranking contract to form a voyage. By doing this, the most urgent contract (will expire in 30 days and the low limit has not been satisfied yet) will be dealt with in high priority. The ranking of each contract will be update every day and expired contract should be removed from C^{ts} as well.

ii) Estimate the inventory level of next day before choosing a ship:

As we mentioned in defect B, randomly ship choosing may lead to inventory overflow for the next day. To avoid that, an estimation of the next day's inventory level must be done before a ship is chosen. More specifically, if one ship is randomly chosen, we calculate the remaining inventory level plus the daily incremental to see if it already exceeds the high limit of storage tank. If it does, that means this ship is not suitable, and we should choose another ship.

However, this method does not 100% guarantee that there will not be any overflow. Restrictions on the number of ships, the range of inventory level, the length of planning horizon, uncertainty of randomly ship chosen, we may encounter that none of available ships can be chosen to avoid inventory overflow. This kind of scenario usually occurs at some point of planning horizon where the number of

available ships is limited and no suitable size of ship can be utilized. Since we will create several delivery plans in total, those which have inventory overflow issue can be abandoned.

2.3.2 Improved idea

The improved idea of creating a delivery plan is as follows:

1. From start date to end date of planning horizon, at each day t we update rankings for each contract.
2. Select the highest ranking contract and a random ship v from $\mathcal{V}^t$ to form a voyage. Estimate the inventory level of the next day, if it is safe, add this voyage.
3. Update the inventory level for next day and repeat step 1.
4. Until the whole planning horizon ends.

Pseudo code of creating delivery plan is given in Algorithm 2.

---

t = start date
while $t \leq$ end date **do**
  Update ranking for each $c \in C^t$, and sort C^t from high ranking to low ranking
  Sort $\mathcal{V}^t$ with random order
  for $c \in C^t$ **do**
    for $v \in \mathcal{V}^t$ **do**
      if $I^- \leq I^{t+1} \leq I^+$ then
      Add voyage $vy(c,v,t)$
      end if
    end for
  end for
    Update inventory level
end while

Algorithm 2. Create delivery plan

2.3.3 Optimize delivery plan

This section presents a method to improve the delivery plans, and tries to maximize the value of the objective function mentioned in Section 2.1. For a given delivery plan, it actually is a set of voyages. We can calculate the profit for each voyage, the sum of all the profit will be the total profit for the delivery plan. Hence the goal of an optimization algorithm is to improve every single profit of voyages. A stochastic optimization algorithm is introduced to improve an initial delivery plan.

We have implemented two operators for stochastic optimization:

1. Changing ship :

   Randomly choose a voyage (c,v,t) and another ship v', then try to replace v with v' to form a new voyage (c,v',t).

2. Swapping ships :

   Randomly choose two voyages (c,v,t) and (c',v',t'), then try to swap v with v' to form two new voyages (c,v',t) and (c',v,t').

In order to determine if a swapping or changing is feasible or not, a couple of restricts must not be violated.

Firstly, no schedule should be conflicted after operation. For changing ship, the duration of new voyage (c,v',t) should not conflict with the exist schedule of v'. For swapping ships, the duration of the new two voyage (c,v',t) and (c',v,t') should not conflict with the exist schedule of both v and v'.

Secondly, the number of inventory level overflow should not be increased. Since different ships have different capacity and ships are full-loaded, the inventory level may be changed after operation. Furthermore, the inventory change will influence the following days' inventory level. For this reason, we need to recalculate all inventory level from the day of ship changing or swapping, and abandon those changes which will increase inventory overflow.

Thirdly, the total profit should be larger than before. Our goal is to maximum the profit of the delivery plan, so the optimization algorithm is expected to increase the profit. It would be totally unnecessary if a ship changing or swapping reduce the profit.

Only if all of the three restricts are satisfied, we can say this operation is feasible. Otherwise the operation should be discarded return.

Pseudo code for ship changing and ship swapping is given in Algorithm 3 and Algorithm 4.

Randomly choose a voyage $vy(c,v,t)$ and a ship v'
if $v == v'$ **then**
    return
end if
if v'.notFeasible(c) **then**
    return
end if
create the new voyage $v'y'(c,v',t)$.
if v'.overlap($v'y'$) **then**
    return
end if
if $v'y'$.profit() <= vy.profit() **then**
    return
end if

Update $I^{t'}$ from start date to end date
if ($I^- > I^{t'} \parallel I^{t'} > I^+$) $\forall t \in \mathcal{T}$. **then**
    return
end if

Remove old voyage from delivery plan, and add new voyage into delivery plan.

Algorithm 3. Ship changing iteration

Randomly choose two voyage $vy(c,v,t)$ and $v'y'(c',v',t')$

if $v == v'$ **then**

return

end if

if (v'.notFeasible(c) || v.notFeasible(c')) **then**

return

end if

Create the new voyage $nvy(c,v',t)$ and $n'v'y'(c',v,t')$.

if (v'.overlap(nvy) || v.overlap($n'v'y'$)) **then**

return

end if

if ($v'y'$.profit() + vy.profit()) >= (nvy.profit() + $n'v'y'$.profit()) **then**

return

end if

Update $I^{t'}$ from start date to end date

if ($I^- > I^{t'}$ || $I^{t'} > I^+$) $\forall t \in T$. **then**

return

end if

Remove the two old voyages from delivery plan, and add the two new voyages into delivery plan.

Algorithm 4. Ship swapping iteration

3. Implementation

This section presents the detail implementation of delivery plan construction and optimization. Java is selected as the programming language for this task, since it is easy to use and has great quantity of third party libraries. Because the information of ports, contracts, fleet and other source from Q company is saved in a XML file, an open source library Dom4j is used to handle XML reading and writing. All coding and debugging are done by using Eclipse.

The whole source code is divided into three packages according to their function, MIRP package, stochasticOptimize package and xmlReadWrite package. In MIRP package, we define all the elements of maritime inventory routing problems including contract, port, vessel, voyages and storage. StochasticOptimize package provides optimize solution to improve delivery plan. In xmlReadWrite package, we use Dom4j to access the XML file. In this architecture, if we want to develop another optimization algorithm, we can build another package (take LocalSearchOptimze package for example) and import it. In this way, we can easily switch between different optimization algorithms.

3.1 Classes definition

Java is an Object-Oriented language, and all objects are instantiated by classes. Well-defined classes could be very important for a good java program. In this section we will present and explain the key classes defined in our program.

3.1.1 Contract class

- **id** : contract id, string
- **name** : contract name, string
- **productType** : cargo type of contract
- **depPort** : departure port
- **desPort** : destination port
- **startDate** : start date of this contract
- **endDate** : end date of this contract
- **minVol** : minimum volume
- **maxVol** : maximum volume
- **shippedVol** : shipped volume
- **pricePerVol** : price per volume
- **rank** : ranking of this contract

- **voyages** : voyages list which carry this contract

One instance of Contract class stands for one real contract with minimum and maximum volume requirements in a certain time interval. **Figure 3.1** shows a sample contract 17US-4.

We can see contract 17US-4 is divided into 12 time intervals, and a minimum volume and a maximum volume for each of them. In our program, we consider this as 12 different contract instances with same id, name and price. However, they are not totally disconnected, the time intervals overlap each other. The first contract's time interval is from 2014-01-01 to 2014-02-01, and the last contract's is from 2014-01-01 to 2015-01-01. So the last one's shipped volume is the sum of all the first 11 contracts' shipped volume and itself.

```
</contract>
<contract id="17US-4" name="Contract 17 (US-4, LLNG)" storage="US-4-LLNG" product="LLNG" type="sale_exship">
  <limit start_time="2014-01-01T00:00:00" end_time="2014-02-01T00:00:00" min="193379.3143690735" max="580137.9431072205"/>
  <limit start_time="2014-01-01T00:00:00" end_time="2014-03-01T00:00:00" min="386758.628738147" max="1160275.886214441"/>
  <limit start_time="2014-01-01T00:00:00" end_time="2014-04-01T00:00:00" min="580137.9431072205" max="1740413.8293216615"/>
  <limit start_time="2014-01-01T00:00:00" end_time="2014-05-01T00:00:00" min="773517.257476294" max="2320551.772428882"/>
  <limit start_time="2014-01-01T00:00:00" end_time="2014-06-01T00:00:00" min="966896.5718453675" max="2900689.7155361025"/>
  <limit start_time="2014-01-01T00:00:00" end_time="2014-07-01T00:00:00" min="1160275.886214441" max="3480827.658643323"/>
  <limit start_time="2014-01-01T00:00:00" end_time="2014-08-01T00:00:00" min="1353655.2005835145" max="4060965.6017505435"/>
  <limit start_time="2014-01-01T00:00:00" end_time="2014-09-01T00:00:00" min="1547034.514952588" max="4641103.544857764"/>
  <limit start_time="2014-01-01T00:00:00" end_time="2014-10-01T00:00:00" min="1740413.8293216615" max="5221241.4879649845"/>
  <limit start_time="2014-01-01T00:00:00" end_time="2014-11-01T00:00:00" min="1933793.143690735" max="5801379.431072205"/>
  <limit start_time="2014-01-01T00:00:00" end_time="2014-12-01T00:00:00" min="2127172.4580598085" max="6381517.3741794255"/>
  <limit start_time="2014-01-01T00:00:00" end_time="2015-01-01T00:00:00" min="2320551.772428882" max="6961655.317286646"/>
  <price_curve>
    <price start_time="2000-01-01T00:00:00" value="12.00"/>
  </price_curve>
</contract>
```

Figure 3.1: Contract Sample

3.1.2 Port class

- **id** : port id, string
- **name** : port name, string
- **location** : port location
- **berth** : berth number
- **portType**: port type
- **storages**: port storage list

3.1.3 Storage class

- **id** : storage id, string
- **name** : storage name, string
- **productType** : cargo type in this storage
- **minVol** : minimum volume
- **maxVol** : maximum volume
- **curVol** : current volume
- **productRateCurve** : the curve of production rate in different time period
- **productRate** : current production rate.
- **dailyInventoryLevel**: stores every day's inventory level

portType clarify a port is a production port or consumption port. A production port may have one or more storages. Shown in **Figure 3.2**, production rate varies in different time intervals, and they are saved in **productRateCurve**. There is a very important element **dailyInventoryLevel** which saves the high and low inventory level of each day. The reason why we keep tracking every day's inventory level is because we need make sure any optimization operation does not violate the inventory limitations.

```
<port id="RLF" name="RLF">
  <location id="l-RLF"/>
  <load_capabilities load_speed="14000" discharge_speed="0"/>
  <storage id="RLF-LLNG" name="Lean LNG (QG)" type="producing" volume_min_level="32500" volume_capacity="375000">
    <inventory_model type="variable_production">
      <production product="LLNG">
        <rate start_time="2014-01-01T00:00:00" rate="8280.9166666667"/>
        <rate start_time="2014-01-13T00:00:00" rate="4161.9166666667"/>
        <rate start_time="2014-01-16T00:00:00" rate="6242.875"/>
        <rate start_time="2014-01-23T00:00:00" rate="8280.9166666667"/>
        <rate start_time="2014-07-15T00:00:00" rate="6242.875"/>
        <rate start_time="2014-07-31T00:00:00" rate="8280.9166666667"/>
      </production>
    </inventory_model>
    <initial_state storage="RLF-LLNG" time="2014-01-01T00:00:00">
      <cargoes>
        <cargo product="LLNG" quantity="109572"/>
      </cargoes>
    </initial_state>
  </storage>
  <closure start_time="2014-05-05T00:00:00" end_time="2014-05-06T00:00:00" type="operations"/>
</port>
```

Figure 3.2: Port and Storage Sample

3.1.4 Vessel class

- **id** : port id, string
- **name** : port name, string
- **maxVol** : capacity of this vessel
- **curVol** : current volume in this vessel
- **productType** : cargo type in this vessel
- **unAvailablePeriod** : a list of time period, which means this vessel is either on voyage or in maintenance
- **voyages** : voyages list which this vessel carried during the whole time horizon

The goal of **unAvailablePeriod** is to distinguish the time window that this ship is not able to carry a voyage. There are two possible reasons for an unavailable period, the first one is carrying a voyage and the second one is in maintenance. Therefore, once a voyage is determined, the voyage time period is added to **unAvailablePeriod** list on this ship. Also we can import the maintenance plan to **unAvailablePeriod** list in advance then we do not schedule a voyage for this ship during the maintenance period.

3.1.5 Voyage class

- **vessel** : vessel carries this voyage
- **contract** : contract
- **depPort** : departure port
- **desPort** : destination port
- **startDate** : date that the voyage starts
- **arrivelDate** : date that the vessel arrives destination port
- **endDate** : date that the vessel finishes the voyage and arrives departure port
- **cargoVol** : cargo volume this voyage carries
- **cost** : the cost for this voyage

The definition of **Voyage** class is quite clear, and it contains all the information for an actual voyage. We can calculate gross profit of this voyage by **cargoVol** and **pricePerVol** of contract. Moreover, we can get net profit by gross profit and **cost**. Once an instance of **Voyage** class is created, it will be added into **MIRP.Voyages** which is the list of all voyages.

3.1.6 RouteKey class

- **vesselName** : vessel name
- **departLoc** : port name, string
- **desportLoc**: port location

3.1.7 RouteValue class

- **cost** : money cost of this route
- **time :** time cost of this route

In our design, we ignore the speed of ships and the distance of two ports and we do not calculate the voyage time from port to port. Instead we leave this computation work to another software, and we get the result as input data. All possible routes are given as route entries in the input file. In other words, if we want a ship **S** execute a voyage from port **P1** to **P2**, there must be a route entry like this <entry vessel=" **S** " from_location=" **P1**" to_location=" **P2**" time="22320" cost="271225"/> in the input file. So if we have a ship name, a departure port and a destination port, we know how long this voyage will be and how much it will cost. In additional, if we do not want a certain ship to execute a certain route, we can just do not provide route entry in the input file. In our code, all possible routes are stored in a linkedHashMap<routeKey, routeMap> **MIRP.routeMaps**. Therefore, if we want to know whether a route is possible or not, we just try to locate a **routekey** (S,P1,P2) in **MIRP.routeMaps**. If this route is not possible, nothing is returned otherwise a **routeMap**(time,cost) is returned. **Figure 3.3** shows a part of route entries for ship LNG_01.

```
<table type="vessel/from_location/to_location">
  <entry vessel="LNG_01" from_location="I-ASIA-1" to_location="I-RLF" time="22320" cost="271225"/>
  <entry vessel="LNG_01" from_location="I-ASIA-2" to_location="I-RLF" time="21600" cost="342977"/>
  <entry vessel="LNG_01" from_location="I-ASIA-3" to_location="I-RLF" time="19440" cost="215605"/>
  <entry vessel="LNG_01" from_location="I-FR-1" to_location="I-RLF" time="19440" cost="424050"/>
  <entry vessel="LNG_01" from_location="I-RLF" to_location="I-ASIA-1" time="22320" cost="435481"/>
  <entry vessel="LNG_01" from_location="I-RLF" to_location="I-ASIA-2" time="21600" cost="301491"/>
  <entry vessel="LNG_01" from_location="I-RLF" to_location="I-ASIA-3" time="19440" cost="387979"/>
  <entry vessel="LNG_01" from_location="I-RLF" to_location="I-FR-1" time="19440" cost="385283"/>
  <entry vessel="LNG_01" from_location="I-RLF" to_location="I-SP-1" time="18720" cost="236587"/>
  <entry vessel="LNG_01" from_location="I-RLF" to_location="I-SP-2" time="15120" cost="249290"/>
  <entry vessel="LNG_01" from_location="I-RLF" to_location="I-SP-3" time="15120" cost="200813"/>
  <entry vessel="LNG_01" from_location="I-RLF" to_location="I-SP-4" time="15120" cost="271020"/>
  <entry vessel="LNG_01" from_location="I-RLF" to_location="I-SP-5" time="15840" cost="408937"/>
  <entry vessel="LNG_01" from_location="I-RLF" to_location="I-UK-1" time="19440" cost="380215"/>
  <entry vessel="LNG_01" from_location="I-RLF" to_location="I-US-3" time="30240" cost="317767"/>
  <entry vessel="LNG_01" from_location="I-RLF" to_location="I-US-4" time="30240" cost="363504"/>
  <entry vessel="LNG_01" from_location="I-SP-1" to_location="I-RLF" time="18720" cost="205483"/>
  <entry vessel="LNG_01" from_location="I-SP-2" to_location="I-RLF" time="15120" cost="315939"/>
  <entry vessel="LNG_01" from_location="I-SP-3" to_location="I-RLF" time="15120" cost="431267"/>
  <entry vessel="LNG_01" from_location="I-SP-4" to_location="I-RLF" time="15120" cost="315113"/>
  <entry vessel="LNG_01" from_location="I-SP-5" to_location="I-RLF" time="15840" cost="308428"/>
  <entry vessel="LNG_01" from_location="I-UK-1" to_location="I-RLF" time="19440" cost="441612"/>
  <entry vessel="LNG_01" from_location="I-US-3" to_location="I-RLF" time="30240" cost="233180"/>
  <entry vessel="LNG_01" from_location="I-US-4" to_location="I-RLF" time="30240" cost="340955"/>
  <entry vessel="LNG_02" from_location="I-ASIA-1" to_location="I-RLF" time="22320" cost="345584"/>
  <entry vessel="LNG_02" from_location="I-ASIA-2" to_location="I-RLF" time="21600" cost="378041"/>
  <entry vessel="LNG_02" from_location="I-ASIA-3" to_location="I-RLF" time="19440" cost="251062"/>
  <entry vessel="LNG_02" from_location="I-FR-1" to_location="I-RLF" time="19440" cost="449548"/>
  <entry vessel="LNG_02" from_location="I-RLF" to_location="I-ASIA-1" time="22320" cost="373361"/>
  <entry vessel="LNG_02" from_location="I-RLF" to_location="I-ASIA-2" time="21600" cost="242903"/>
  <entry vessel="LNG_02" from_location="I-RLF" to_location="I-ASIA-3" time="19440" cost="223081"/>
  <entry vessel="LNG_02" from_location="I-RLF" to_location="I-FR-1" time="19440" cost="200619"/>
```

Figure 3.3: Part of route entries for ship LNG_01.

3.1.8 MaritimeInventoryRoutingProblem class

- **contracts** : all contracts list
- **sortedContracts:** contracts list sorted by ranking order
- **vessels** : all vessel list
- **availableVessels:** vessels list which is available
- **unAvailableVessels**: vessels list which is available
- **sortedContracts:** contracts list sorted by ranking order
- **routeMaps** : all possible routes, saved as a hashMap
- **productionPorts:** production ports list
- **comsumptionPorts:** comsumption ports list
- **voyages** : voyages list
- **minCapVessel** : the minimum capacity of all vessels
- **startDateStr:** string of start date of planning horizon
- **endDateStr:** string of end date of planning horizon

- **totalProfitBeforeOpt:** total profit for the initial delivery plan
- **totalProfitAfterOpt:** total profit for the optimized delivery plan

If we say all the classes above are branch classes, then **MaritimeInventoryRoutingProblem** class is the trunk class. It contains all the information about this problem, and all the operations are running on it. As the name suggests, **availableVessels** stores available ships on current date which are randomly selected during ADP generation process. So at the beginning of a day, we go through another list **unAvailableVessels** to release ships which have returned from last voyage or finished its maintenance to **availableVessels** list. **minCapVessel** is the minimum capacity of all ships. If the inventory level is less than **minCapVessel**, we can just skip today because all ships must be full-loaded. **totalProfitBeforeOpt** is the net profit of this initial delivery plan, and **totalProfitAfterOpt** is the net profit of optimized delivery plan. We save both of them just to know how much it is improved.

3.1.9 Class diagram

Class diagram is given in **Figure 3.4**

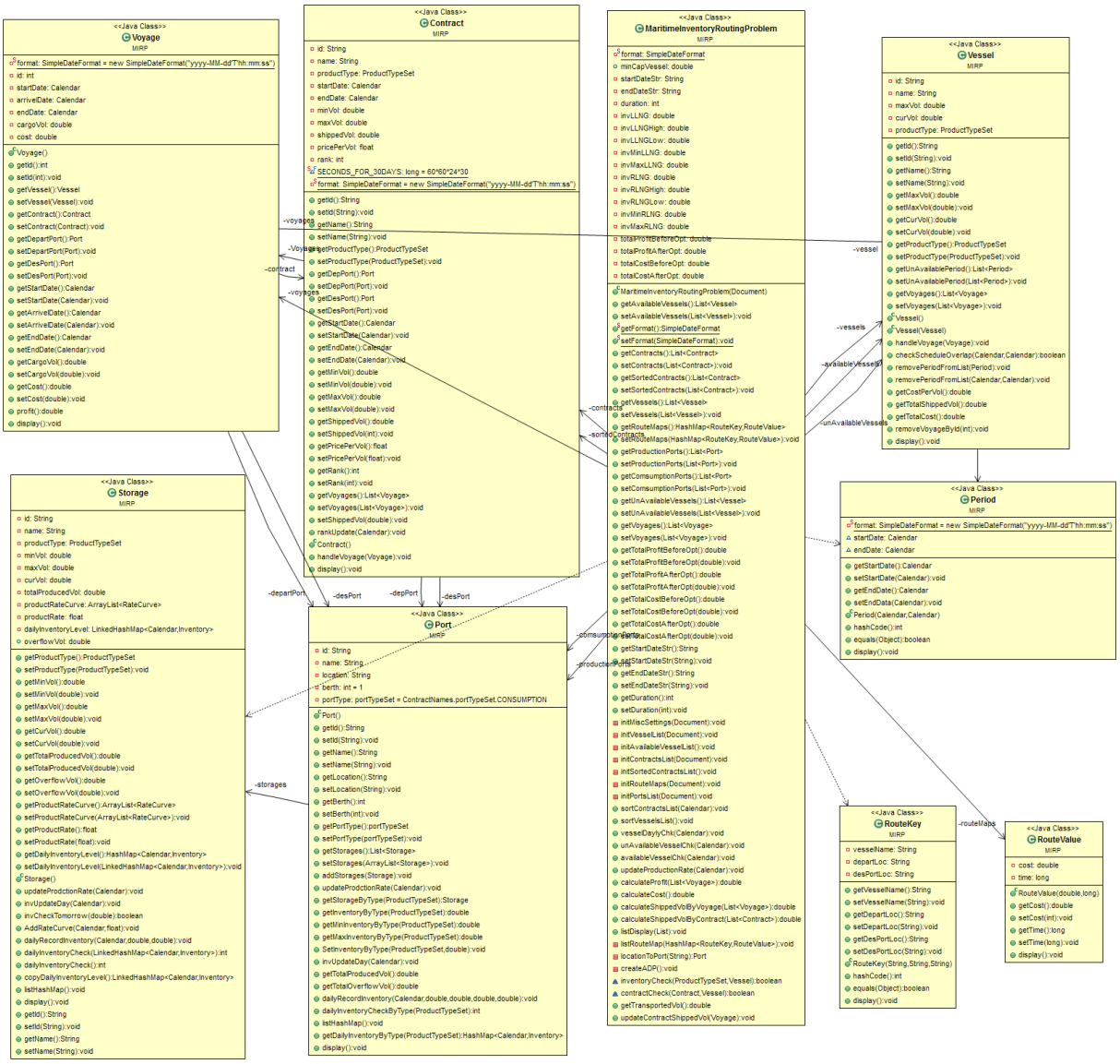

Figure 3.4 class diagram

4. Computational Results

In this section we test our solutions based on the realistic data from Q company, and try to generate the ADP and some shorter delivery plan as well. First a brief introduction of our instance is presented. Then we test this instance with different parameters, like how many initial delivery plans and how many iterates we should run. Afterwards we test our solutions with different time horizons to see which it fits well. At last we analyze the two optimizations algorithms.

All tests have been performed on my laptop with Intel 8-core i7-3610M CPU, 4 GB RAM on Windows 7 operation system. The software development environment is on Eclipe with Java 7.0.

4.1 Instance introduction

Table 1 gives the basic information of this Maritime Inventory Routing Problem, such as number of ships, contracts and planning days.

Basic information	
Number of ships	50
Number of contracts	8
Number of planning days	365
Start date	01-01-2014
End date	30-12-2014
Number of Production port	1
Number of destination port	10

Table 1

Table 2 gives the information of production port, including high and low bound of inventory volume. Since the production rate varies from different time periods, it is given in **Table 3**

Production port information	
Port location	RLF
Production type	LLNG
Min inventory volume	32500
Max inventory volume	375000
Production rate	Table 3

Table 2

Production rate (per hour)	Start date	End date
8280.9166666667	01-01-2014	13-01-2014
4161.9166666667	13-01-2014	16-01-2014
6242.875	16-01-2014	23-01-2014
8280.9166666667	23-01-2014	15-07-2014
6242.875	15-07-2014	31-07-2014
8280.9166666667	31-07-2014	30-12-2014

Table 3

4.2 Parameter test

In our design, there are two parameters seeds number and iteration number. They do not only have influence on the computing time, but also determine the quality of our solutions. Large seeds number brings more possibilities of solution samples, and large iteration number improves the optimization result. Theoretically, infinite seeds number and iteration number can produce the best solution.

In order to balance the results and the time cost, we have done some experiments and tried to find the relative best solution at acceptable time. Three objectives have been taken considered, Profit, Time and Cost.

4.2.1 Seeds Number Test

In this test, we test our solutions with different seeds number and fixed iterate number. Seven modes are created as follows:

Mode A: 10 seeds, 500000 iterations
Mode B: 20 seeds, 500000 iterations
Mode C: 50 seeds, 500000 iterations
Mode D: 100 seeds, 500000 iterations
Mode E: 200 seeds, 500000 iterations
Mode F: 500 seeds, 500000 iterations
Mode G: 1000 seeds, 500000 iterations

Test result is given in **Table 4**.

Mode	Time (s)	Profit Before OPT($)	Profit After OPT($)	Profit Gap($)	Cost Before OPT($)	Cost After OPT($)	Cost Reduced (%)
A	22	879937484	897925711	0	120319516	104690289	14.9
B	44	882473934	899006599	1080888	118243066	104796401	12.8
C	104	880307874	899676923	1751212	121911126	103699077	17.5
D	214	883478777	899678053	1752342	119107223	104257947	14.2
E	421	882480445	900008517	2082806	118782555	103179483	15.1
F	1056	881292600	900549163	2623452	119420400	103273837	15.6
G	2234	881447559	900490050	2564339	120001441	103713949	16.3

Table 4. Seeds Number Test Results

From this table, we can see how the seeds number influence on the results. No strange the least **Profit After OPT**(marked as red) shows up at mode A, and the profit of mode F and G are relevant the highest and very close. So we conclude that once the seeds number reaches a certain amount, the influence might be less sensitive. The column **Profit Gap** means the **Profit After OPT** at this mode minus the least **Profit After OPT** of this table. By Profit Gap, we see that with seeds number increasing, the growth of **Profit After OPT** is determinate. **Cost Redueced** value of all the seven modes are very close, because it is only affected by the iteration number which we will discuss in the next section. One interesting thing is that **Profit Before OPT** of each mode is quite approximate, and it does not effected by the seeds number. Furthermore, a high value of **Profit before OPT** not necessary leads to a high value of **Profit After OPT**. Mode D has the highest value of **Profit Before OPT** but the fourth highest value of **Profit After OPT**. Mode F has the fourth highest value of **Profit Before OPT** but the highest value of **Profit After OPT**.

All in all, 500 seeds are enough for a quick solution, but 1000 seeds can be more secure. More than 1000 seeds may not be necessary.

4.2.2 Iteration Number Test

In this test, we test our solutions with fixed seeds number and different iterate number. Six modes are created as follows:

Mode A: 50 seeds, 100000 iterations
Mode B: 50 seeds, 200000 iterations
Mode C: 50 seeds, 500000 iterations
Mode D: 50 seeds, 1000000 iterations
Mode E: 50 seeds, 2000000 iterations
Mode F: 50 seeds, 5000000 iterations
Mode F: 50 seeds, 10000000 iterations

Test result is given in **Table 5**.

Mode	Time (s)	Profit before OPT($)	Profit After OPT($)	Profit Gap($)	Cost Before OPT($)	Cost After OPT($)	Cost Reduced (%)
A	22	881035062	897434623	0	119675938	106871377	12.8
B	41	880237827	898972993	1538370	118243066	104796401	13.4
C	104	880307874	899676923	2242300	121911126	103699077	17.5
D	195	879137949	899745306	2310683	122931051	104161694	18.0
E	404	881008519	899704884	2270261	120575481	104245116	16.3
F	1017	878671496	898984053	1549430	121378504	104154947	17.2
G	1921	879612355	900378373	2943750	120575645	103289627	16.7

Table 5. Iteration Number Test Results

The results in Table 5 seem not like that type of data we are expecting. First, **Profit After OPT** is not consistently growing with iteration number. It has a big downfall at mode F, and the value of each mode from C to D is tight. Second, we are expecting the **Cost Reduced** value to increase along with iteration number. But in fact, the best value of **Cost Reduced** exists at mode D which is not even the top three of iteration number. It looks like there is a limitation around 16% to 18%, and it is very hard to climb once the iteration number is bigger than 200000. There may have three possible explanations for this phenomenon:

i) The final solution for each mode is selected due to the value of Profit After OPT, which solution is not necessary the best cost reduced solution.

ii) The planning horizon is too long which is one whole year. All the ships are almost fully occupied, it is hard to find any available ship to swap or change. That means it may only find one feasible swapping or changing possibility over 10000 attempts

iii) Initial seed may be critical for optimization process.

The conclusion can be drawn that for a quick computation, 500 seeds number and 500000 iteration number will be enough. The computation time should be less than 1 hour, and the quality of this solution should be very close to the best solution.

4.2.3 Planning Horizon test

In previous section we discuss how the parameters influence on the results, in this section we test our algorithm with different planning horizon.

As we mentioned in section 4.2.2, due to the limited resources like ship quantity, it is very difficult to improve the original seed for a whole year planning. Because all ships have been occupied, there is no available one can be swapped or changed. Then we are wondering if it does better with less planning days. Therefore, we choose 5 different planning horizons, 31 days, 59 days, 90 days, 120 days and 181 days to test our solution.

Mode A: 500 seeds, 500000 iterations, 31 days
Mode B: 500 seeds, 500000 iterations, 59 days
Mode C: 500 seeds, 500000 iterations, 90 days
Mode D: 500 seeds, 500000 iterations, 120 days
Mode E: 500 seeds, 500000 iterations, 181 days
Mode F: 500 seeds, 500000 iterations, 270 days

Test result is given in **Table 6**.

Mode	Time (s)	Profit before OPT($)	Profit After OPT($)	Cost Before OPT($)	Cost After OPT($)	Cost Reduced (%)
A	702	65364336	68746922	8846664	6660078	32.8
B	751	120868159	129652000	19181253	15267344	25.6
C	765	210992208	217842933	28649792	23844067	20.2
D	800	282574369	290623582	38601631	32002418	20.6
E	879	413960803	424328469	58881197	50224531	17.2
F	1485	639356740	654588869	88334260	74813131	18.0

Table 6: Test for different planning horizons

Mode A has the shortest planning days (30) also has the highest Cost Reduced rate (32.8%). The top 2 of longest planning days, mode E and F , have the lowest Cost Reduced rate (17.2% and 18.0%). This result verifies our idea that the ships number is the limitation. In shorter planning horizon, there are more available ships, so there are more swapping or changing possibilities. See Fig 4, it is very clear that how Cost Reduced rate declines.

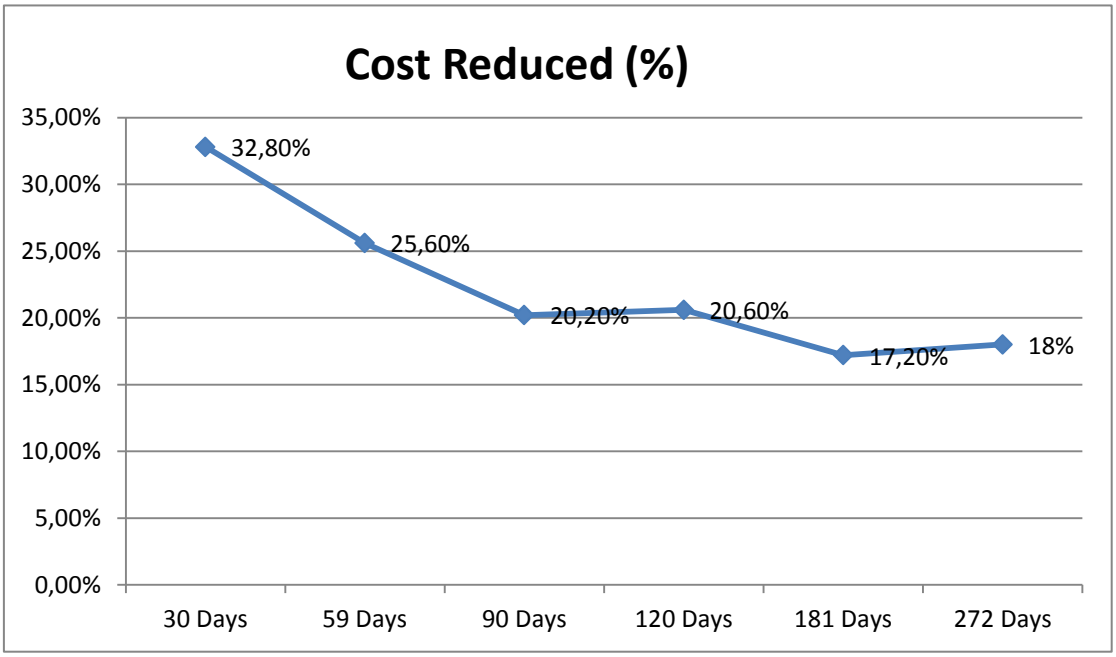

Figure 4.1: Cost Reduced rate falls with increasing planning days

4.2.4 Solution Output

The final solution is a XML file with a XLS transformer, so you can open it by using any web browser. There are three parts in this solution: the first part displays the overall statistic of optimization result (Figure 5.2), the second part displays the contracts status (Figure 5.3) and the final part displays voyage plans for every ship (Figure 5.4). Since the output file is too big, we just paste a short screen shot from each part.

Optimization Result Statistic

Start Date	2014-01-01T00:00:00
End Date	2015-01-01T00:00:00
Duration	365
total_produced	7.1119263E7
total_transported	7.0989E7
total_overflow	0.0
total_profit_before_optimization	8.82513135E8
total_profit_after_optimization	9.01154974E8
total_cost_before_optimization	1.19701865E8
total_cost_after_optimization	1.02977026E8
total_voyages	363

Figure 4.2: Optimization statistic

Contracts Plan

Contract Name	Start Date	End Date	Low Volume Bound	High Volume Bound	Shipped Volume
Contract 16 (UK-1, LLNG)	2014-01-01T12:00:00	2014-02-01T12:00:00	488278.0	1464834.0	619000.0
Contract 16 (UK-1, LLNG)	2014-01-01T12:00:00	2014-03-01T12:00:00	976556.0	2929668.0	1156000.0
Contract 16 (UK-1, LLNG)	2014-01-01T12:00:00	2014-04-01T12:00:00	1464834.0	4394502.0	1915000.0
Contract 16 (UK-1, LLNG)	2014-01-01T12:00:00	2014-05-01T12:00:00	1953112.0	5859336.0	2504000.0
Contract 16 (UK-1, LLNG)	2014-01-01T12:00:00	2014-06-01T12:00:00	2441390.0	7324170.0	2864000.0
Contract 16 (UK-1, LLNG)	2014-01-01T12:00:00	2014-07-01T12:00:00	2929668.0	8789004.0	3433000.0
Contract 16 (UK-1, LLNG)	2014-01-01T12:00:00	2014-08-01T12:00:00	3417946.0	1.0253838E7	3836000.0
Contract 16 (UK-1, LLNG)	2014-01-01T12:00:00	2014-09-01T12:00:00	3906224.0	1.1718672E7	4498000.0
Contract 16 (UK-1, LLNG)	2014-01-01T12:00:00	2014-10-01T12:00:00	4394502.0	1.3183506E7	4901000.0
Contract 16 (UK-1, LLNG)	2014-01-01T12:00:00	2014-11-01T12:00:00	4882780.0	1.464834E7	5304000.0
Contract 16 (UK-1, LLNG)	2014-01-01T12:00:00	2014-12-01T12:00:00	5371058.0	1.6113174E7	5867000.0
Contract 16 (UK-1, LLNG)	2014-01-01T12:00:00	2015-01-01T12:00:00	5859336.0	1.7578008E7	6277000.0
Contract 17 (US-4, LLNG)	2014-01-01T12:00:00	2014-02-01T12:00:00	193379.3143690735	580137.9431072205	367000.0
Contract 17 (US-4, LLNG)	2014-01-01T12:00:00	2014-03-01T12:00:00	386758.628738147	1160275.886214441	596000.0
Contract 17 (US-4, LLNG)	2014-01-01T12:00:00	2014-04-01T12:00:00	580137.9431072205	1740413.8293216615	812000.0
Contract 17 (US-4, LLNG)	2014-01-01T12:00:00	2014-05-01T12:00:00	773517.257476294	2320551.772428882	950000.0
Contract 17 (US-4, LLNG)	2014-01-01T12:00:00	2014-06-01T12:00:00	966896.5718453676	2900689.7155361027	1088000.0
Contract 17 (US-4, LLNG)	2014-01-01T12:00:00	2014-07-01T12:00:00	1160275.886214441	3480827.658643323	1297000.0
Contract 17 (US-4, LLNG)	2014-01-01T12:00:00	2014-08-01T12:00:00	1353655.2005835145	4060965.6017505433	1513000.0

Figure 4.3: Contracts plan

Voyage Plans

Vessel Name	LNG_01
Vessel Id	LNG_01
Max Capacity	138000.0
Total Served Volume	1104000.0
Total Cost	2503331.0
Cost Per Vol	2.267509963768116

Voyage ID	Ship Name	Start Date	Arrival Date	End Date	From Port	To Port	Shipped Volume
15	LNG_01	2014-01-17T12:00:00	2014-01-30T12:00:00	2014-02-13T12:00:00	RLF	UK-1	138000.0
48	LNG_01	2014-02-19T12:00:00	2014-03-12T12:00:00	2014-04-02T01:00:00	RLF	US-3	138000.0
91	LNG_01	2014-04-03T12:00:00	2014-04-16T12:00:00	2014-04-30T12:00:00	RLF	UK-1	138000.0
128	LNG_01	2014-05-10T12:00:00	2014-05-31T12:00:00	2014-06-21T12:00:00	RLF	US-3	138000.0
190	LNG_01	2014-07-11T12:00:00	2014-07-21T12:00:00	2014-08-01T12:00:00	RLF	SP-4	138000.0
226	LNG_01	2014-08-16T12:00:00	2014-08-26T12:00:00	2014-09-06T12:00:00	RLF	SP-3	138000.0
283	LNG_01	2014-10-12T12:00:00	2014-11-01T11:00:00	2014-11-22T11:00:00	RLF	US-3	138000.0
331	LNG_01	2014-11-29T12:00:00	2014-12-20T12:00:00	2015-01-10T12:00:00	RLF	US-3	138000.0

Vessel Name	LNG_02
Vessel Id	LNG_02
Max Capacity	138000.0
Total Served Volume	966000.0
Total Cost	1966071.0
Cost Per Vol	2.0352701863354037

Voyage ID	Ship Name	Start Date	Arrival Date	End Date	From Port	To Port	Shipped Volume
118	LNG_02	2014-04-30T12:00:00	2014-05-21T12:00:00	2014-06-11T12:00:00	RLF	US-3	138000.0
198	LNG_02	2014-07-19T12:00:00	2014-07-30T12:00:00	2014-08-10T12:00:00	RLF	SP-5	138000.0
85	LNG_02	2014-03-28T12:00:00	2014-04-07T01:00:00	2014-04-18T01:00:00	RLF	SP-4	138000.0
60	LNG_02	2014-03-03T12:00:00	2014-03-14T12:00:00	2014-03-25T12:00:00	RLF	SP-5	138000.0

Figure 4.4: Voyage plans

5. Conclusion and future work

In this thesis we have presented a solution for maritime inventory routing problem with a large scale of ships and planning horizon. The goal is to create a feasible annual delivery plan to fulfill customer's long team contracts with minimum cost.

In order to solve this problem, we have not only developed a multi-seeds stochastic optimization algorithm but also built a scalable framework that we can introduce other different algorithms for this problem. The multi-seeds stochastic algorithm may not be the best solution for this kind of problem, but it can produce a reasonable good result within a short time.

We chose Java to implement this software because it is a very popular programming language and very easy to access XML file with third party library. Most important reason is that it will be much easier to integrate the algorithm into the company's ERP system which is written by java.

Until now, this software has not been completed yet. We need to implement some other details which we ignored in this thesis, such as boil-off rate, loading time, discharging time, and so on. Moreover, some other optimization algorithm should be introduced like Generic algorithm and Tabu search algorithm in future.

6. Reference

1) Wikipedia page of LNG http://en.wikipedia.org/wiki/Liquefied_natural_gas.
2) Demand and supply of LNG www.vibrantgujarat.com/sector-presentation/pdf/oil_gas_power/mr_doug_shell_2009_01_09_lng_sd.pdf.
3) Centre for Energy http://www.centreforenergy.com/AboutEnergy/ONG/LiquifiedNaturalGas/Overview.asp?page=1
4) Wikipedia page of LNG carrier, "Cargo handling" http://en.wikipedia.org/wiki/LNG_carrier
5) World Gas Intelligence, 30 July 2008
6) News from DNV about new cargo containment system, 02/04/2013. http://www.dnv.com/industry/maritime/publicationsanddownloads/publications/updates/tanker/2013/1_2012/upgraded_lng_containment_system_forreduced_boiloff.asp
7) Geir BrZnmo, Marielle Christiansenb, Kjetil Fagerholtc, BjZrn Nygreenb (2005). A multi-start local search heuristic for ship scheduling--a computational study. Computers & operations research.
8) Magnus Stålhane, Jørgen Glomvik Rakke, Christian Rørholt Moe, Henrik Andersson, Marielle Christiansen, Kjetil Fagerholt. 2009. A construction and improvement heurstic for a liquefied natural gas. Computers & Industrial Engineering 62 (2012) 245–255
9) Jørgen Glomvik Rakke, Magnus Stslhåne ,Christian Rørholt Moe, Henrik Andersson, Marielle Christiansen, Kjetil Fagerholt.(2011). A rolling horizon heuristic for creating a liquefied natural gas annual delivery program. Transportation Research Part C, 19(5), 896–911
10) Grønhaug Roar, Marielle Christiansen.(2009). Supply chain optimization for the liquefied natural gas business. Innovations in Distribution Logistics, Lecture Notes in Economics and Mathematical Systems, vol. 619. Springer, pp. 195–218.
11) Grønhaug Roar, Marielle Christianse, Guy Desaulniers, Jacques Desrosiers. (2010). A branchand-price method for a liquefied natural gas inventory routing problem.Transportation Science, 44(3), 400–415
12) Henrik Andersson, Marielle Christiansen, Kjetil Fagerholt. 2010. Transportation planning and inventory management in the LNG supply chain. Energy, Natural Resources and Environmental Economics. Springer, pp. 429–441.
13) Papageorgiou, D.J., Nemhauser, G.L., Sokol, J., Cheon, M-S., Keha, A.B., MIRPLib - A Library of Maritime Inventory Routing Problem Instances: Survey, Core Model, and Benchmark Results, European Journal of Operational Research (2013), doi: http://dx.doi.org/10.1016/j.ejor.2013.12.013
14) M. Christiansen and K. Fagerholt. 2009. Maritime inventory routing problems. In C. A. Floudas and P. M. Pardalos, editors, Encyclopedia of Optimization, pages 1947–1955. Springer-Verlag, second edition