Simulation and Optimization for Supply Chain Based on Multi-agent Reinforcement Learning: A Case Study on a Large-scale Refinery

PAN Yanchun
College of Management, Shenzhen University, P. R. China, 518060
pan_yc@163.com

Abstract: A case study on a large-scale refinery is implemented in this paper. Considering the plenty of stochastic factors existing in real life, general-purpose simulation platform ARENA is employed to model the complex supply chain of this refinery and obtain the system’s performance indices. With VBA and Object Oriented Programming technology, a kind of architecture is proposed to integrate simulation with optimization. Then a Multi-agent Reinforcement Learning algorithm is designed to optimize the ordering and distributing policies of the refinery. Research results show that the methodology proposed can effectively solve the optimization problems existing in real-life and complicated supply chain.

Keywords: Supply Chain; Simulation; Optimization; Reinforcement Learning

1 Introduction

Today’s enterprises face intense competition and cooperation in uncertain environment. Complex supplier relationships, increasing customer expectations and changing market demands drive the business community to pay more and more attention to the design and optimization of their supply chain. As a result, SCM (Supply Chain Management) has become a research hotspot[1]. Due to the complexity and the bullwhip effect, many scholars conduct their researches on coordination and benefit distribution among different nodes in the supply chain from the point of view of information theory and game theory. For example, Moyaux and his/her collaborators divided market demands into original demands and adjustable demands. Then they proposed two different rules to reduce the amplification effect of fluctuations in demands with information-sharing technologies[2]. Modern heuristic algorithms, such as Ant Colony Optimization, Genetic Algorithms and Multi-agent technologies play an important role in the process of optimizing inventory, distributing policy, supplier selection and so on in SCM[3][4]. Because of the high computing complexity, simulation is usually employed to model the problems and obtain the systems’ performance indices. It seems to be the most promising method to integrate simulation with heuristic algorithms to optimize supply chain. Vito, Nunzia and Ilaria investigated supply chain cooperation in industrial districts. They used simulation to model the system. MARL (Multi-agent Reinforcement Learning) technology was adopted to optimize its supply chain. They found that moderate competition and cooperation could improve product quality and reduce costs[5]. Tarokh and his/her collaborators introduced many methods for simulating supply chain in their paper[6]. Reference [7] and [8] used simulation-optimization to solve the problems of supply chain management.

However, most of the current studies on SCM stay in the theoretical level. They usually suppose a theoretical model of supply chain and can only take few factors into consideration. It’s still rare to solve the real-life problems in supply chain management and optimization, which is very complicated because it always includes many stages, such as suppliers, manufacturers, distributors, end users and the third party logistics (3PLs). In this paper, a case study on a large-scale refinery is introduced. A kind of MARL algorithm integrated with simulation tool ARENA is employed to model the complex supply chain of the refinery and optimize its ordering and distributing policies. As a result, the total operations cost is reduced. Furthermore, the simulation and optimization for real-life, large-scale and complicated supply chain is discussed.

2 Problem Description
The crude oil resources are very uneven in the world. OPEC members have nearly 75% oil reserves and satisfy 40% oil demands in the world. The world oil market can be affected and the oil prices may be fluctuated when social situation is unrest in the Middle East countries. Most of the crude oil and oil products have to be stored and transported by very large ships, which are usually rented from 3PLs. The transportation time is always long and easily affected by typhoon.

Compared to other industries, the production and operations processes of oil industry are very complex. Oil exploration and mining are the core businesses in oil supply chain. For petrochemical enterprise, crude oil purchase costs account for about 70% to 80%, and then the costs of logistics. Production costs are only a small fraction of the total costs. As a result, in order to reduce the costs of petrochemical enterprise, it is very necessary to optimize its ordering and distributing policies in the supply chain.

Under this background of oil industry, a case study on a large-scale refinery R is implemented in this paper. Figure 1 shows the supply chain structure of R. There are three crude oil suppliers and four oil products distributors. Due to quota limitation, the procurement proportions from E, D and M are about 5%, 25% and 70%. In practice, the procurement quantity from E and D cannot exceed their quotas but can be less than the quotas. M has no limitation. That is, if the quantity purchased from E and D is not enough, R can buy more crude oil from M. Because VMI (Vendor Managed Inventory) is implemented in the supply chain, R can get the inventory information of each distributor every day.

According to the analysis of primary data, shipping time and oil prices generally fit triangle distribution. Arriving interval time of typhoon and order as well as social situation almost fit exponential distribution. Ordering quantity fits mean distribution. Different suppliers have different crude oil prices. Transportation cost is subject to transportation means and distance.

Annual operations cost of refinery R is regarded as performance index for evaluating the solutions.
Annual operations cost is the sum of all kinds of costs about crude oil and oil products. Crude oil costs include purchase cost, ordering cost, loading and unloading costs, transportation cost, inventory cost, shutdown cost of refinery, processing cost, waiting cost of laden ships etc. Oil products costs include inventory costs of all tanks, loading and unloading costs, transportation costs, distribution costs, costs for out of stock etc. The optimization problem is illustrated as follows: When to order crude oil? Which supplier should be chosen? How much crude oil should be ordered? When should the oil products be distributed? Which distributor should be chosen? How much oil products should be distributed? We must determine these answers to minimize the annual operations cost. This optimization problem can hardly be solved by traditional analytical methods due to its complexity.

Simulation is employed to model this supply chain and obtain the system’s performance indices. Reinforcement learning is introduced to design a kind of multi-agent decision making system to optimize the ordering and distributing policies. The agents include: a purchase agent to determine the procurement action according to current state, four distributing agents to determine the distributing actions according to current state, a global agent to coordinate with other agents to make them take good actions. All these agents can perceive the state information of environment and make decisions every day. In the one-year period, a decision sequence will occur according to the whole decision making process. Our goal is to find a satisfied sequence or decision policy to minimize the annual operations cost of the supply chain.

3 Architecture for simulation-optimization

General-purpose tool ARENA developed by Rockwell is taken as simulation platform to model the supply chain. The core modules of ARENA are all encapsulated in its hundreds of class libraries. In order to easily integrate simulation with optimization, Visual Basic 6.0 (VB for short) and Object Oriented Programming (OOP for short) technology are employed to design and develop a general-purpose interface SOIBA (Simulation-Optimization Interface Based on ARENA), which encapsulates the functions in common use for simulation-optimization. SOIBA appears as a Dynamic Link Library (DLL for short) and can be referred to in any development environment to control the running process of ARENA simulation model. As a result, the effective integration between simulation and optimization is realized. Figure 2 explains how to call this interface in VB environment. It’s easy to see that SOIBA is really very simple and practical.

```vbnet
Dim makespan As Double 'declare a variable for saving result
Dim mySOIBA As New SOIBA 'declare an interface object
mySOIBA.OpenARENA 'open ARENA
Call mySOIBA.OpenModel("C:\testModel.doe") 'open the given simulation model
mySOIBA.NumberOfReplications = 10 'set the max number of replications to be ten
mySOIBA.RunModel 'running the model until its end
makespan = mySOIBA.VariableArrayValue(mySOIBA.SymbolNumber("makespan")) 'get the simulation result
mySOIBA.EndRun 'end the simulation running
mySOIBA.QuitARENA 'quit ARENA
Set mySOIBA = Nothing 'release the interface object
```

Figure 2: Calling method of SOIBA in VB environment

Furthermore, VBA (Visual Basic for Application) is embedded in ARENA, which makes it possible for users to integrate other systems into ARENA model. Due to its strong and practical functions, ARENA can be effectively used to model and simulate the complex systems, such as refinery[9]. Figure 3 shows this simulation-optimization system architecture. The MARL algorithm implemented in this paper is directly embedded into ARENA model and becomes a part of the model via VBA. With SOIBA, the interaction between simulation and optimization is realized. The performance indices obtained by simulation are transformed into an input for the optimization algorithm. The optimized solution coming from the optimization algorithm is then transformed into an input parameter of simulation. This process won’t stop until some stop rule is satisfied. Figure 4 illustrates the main simulation logic.
In terms of theory, inventory quantity, ordering quantity, distributing quantity and oil prices are all continuous variables. They are discretized in this paper for the existence of minimal economic batch and for the sake of computing convenience. Inventory quantity and oil prices are divided into several intervals as necessary. Ordering quantity and distributing quantity are expressed by discrete values. Although it is not completely precise, the results can be accepted in practice. After discretized, the state variables and action variables of all agents are shown as follows:

\[ s_0 = (I_{co}, I_{po}, P_{co}, z) \] //state of purchase agent  
\[ a_0 = (v, I_o) \] //action of purchase agent  
\[ s_2 = (I_{po}, I_z) \] //state of distributing agent Z  
\[ s_L = (I_{po}, I_z) \] //state of distributing agent L  
\[ s_J = (I_{po}, I_z) \] //state of distributing agent J  
\[ s_C = (I_{po}, I_z) \] //state of distributing agent C  

\[ I_{co} \] : crude oil inventory interval  
\[ I_{po} \] : oil products inventory interval in the refinery  
\[ P_{co} \] : crude oil price interval  
\[ z \] : season, 0 for normal, 1 for typhoon season  
\[ v \] : supplier, 1 for M, 2 for D, 3 for E  
\[ I_o \] : ordering quantity  
\[ I_z \] : inventory interval of distributor Z  
\[ I_L \] : inventory interval of distributor L  
\[ I_J \] : inventory interval of distributor J  
\[ I_C \] : inventory interval of distributor C

The action of each distributing agent is its distributing quantity. When the oil products inventory in refinery R is less than the sum of all distributing quantities, the global agent will determine which distributor should get the oil products first according to the following rules:

\[ I_z = I_{2loss} - I_z; \quad I_L = I_{1loss} - I_L; \quad I_J = I_{1loss} - I_J; \quad I_C = I_{1loss} - I_C \]

The subscript loss above stands for out of stock. The distributor with larger value above has higher priority. That is, the distributor that has more loss or has less inventory quantity will get oil products first.

Q-Learning algorithm is introduced to the agents to search in the state-action space. Q-Learning, which is one of the typical algorithms for reinforcement learning, uses the Q value of state-action to estimate its cumulated rewards as follows:

\[ Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)] \] \hspace{1cm} (1)
In the above formula, $\alpha$ is study rate $\alpha \in (0,1]$. Because large $\alpha$ indicates that immediate reward and following rewards have great impact on current state-action, the larger $\alpha$ is, the more far-sighted the agent is, but the slower the convergence speed is. $\gamma \in [0,1]$ is discount rate, which represents the time preference that the agent gets the reward. The larger $\gamma$ is, the more far-sighted the agent is, but the slower the convergence speed is. Namely, agent will consider more information that may affect the following decisions when it makes current decision. $s'$ is a new system state when agent takes action $a$ at state $s$. $r$ is the immediate reward that agent obtains when it takes action $a$ at state $s$ and the system state becomes $s'$. The value of $r$ is up to the reward function $Q(s,a)$ is the $Q$ value when agent takes action $a$ at state $s$. According to the theory of reinforcement learning, when the algorithm converges,

$$\pi(s) = a^* = \arg\max_a Q(s,a)$$

is the best action when the state is $s^{[10]}$.

Suppose the total operations cost of the supply chain is $C_{total}^t$ at time $t$. Then the average cost is $C_{avg}^t = C_{total}^t / t$. Define the immediate reward function as follows:

$$r_1(s, a, s') = \begin{cases} \beta_1 \times (C_{avg} - C_{avg}'), & C_{avg} < C_{avg}' \\ 0, & \text{others} \end{cases}$$

$\beta_1 > 0$ is the coefficient of immediate function. All decision nodes $(s_i, a_i, s_i)$ will be record in the process of reinforcement learning. $(s_i, a_i, s_i)$ denotes that the system state becomes $s_i$ when agent takes action $a_i$ at state $s_i$. When a learning cycle that is one year passes, an annual reward will be fed back to the agents as the following formula according to the total operations cost of this year and last year:

$$r_2(s, a, s') = \begin{cases} \beta_2 \times (C_{total} - C_{total}'), & C_{total} < C_{total}' \\ 0, & \text{others} \end{cases}$$

$\beta_2 > 0$ is the coefficient of annual reward function. $C_{total}$ is the total operations cost of current year. $C_{total}'$ is the total operations cost of last year. Then the corresponding state-action information record in all decision nodes will be updated according to this annual reward and formula (1).

The idea for designing the above reward functions is that the actions leading to the decrease in average cost and annual operations cost should be reinforced. An immediate reward function and an annual reward function are proposed at the same time because it is better for improving the learning efficiency and effectiveness of the agents.

A $\varepsilon$-greedy policy is adopted when agent takes action at a given state, that is, the agent will select the completely greedy action $a^*$ shown as formula (2) with a high probability $1 - \varepsilon$, and select action randomly with a low probability $\varepsilon \in (0.1)$ called greedy rate$^{[10]}$. The smaller $\varepsilon$ is, the greedier the algorithm is, and the more suitable for the agent to implement exploitation according to the current learning results. The larger $\varepsilon$ is, the more random the algorithm is, and the more suitable for the agent to explore new and more effective actions.

Due to the high ratio of crude oil purchase cost, in order to reduce the total cost, agents may stop purchasing crude oil in the process of reinforcement learning. So some new instructive rules are proposed as follows when designing the policies for agents to make decisions: (1) The purchase agent must order crude oil when the inventory quantity is no more than zero. (2) The distributing agents must send oil products to the distributor whose inventory quantity is no more than zero and the inventory quantity of refinery is more than the minimal distributing quantity. (3) The agents won’t make decisions until its state changes.

Figure 5 illustrates how agents interact with the environment. The action is chosen by $\varepsilon$-greedy policy according to the agent's current state. A new state occurs and the agent gets an immediate reward after it takes the chosen action. Then the state-action information is updated by
Q-Learning algorithm. When a decision cycle ends, the decision sequence will also be updated by Q-Learning algorithm according to the annual reward. This process goes round and round and the agents become more and more clever to adjust their action policies to optimize the system goal.

4 Results of case study

According to the current research\(^\text{(10)}\), set the recommended value to each parameter in MARL algorithm. That is,

\[ \alpha = 0.1, \quad \gamma = 0.9, \quad \varepsilon = 0.1. \]  

After some experiments, we set \( \beta_1 = 1 \) and \( \beta_2 = 3 \).

Figure 6 shows the simulation animation at some time. Figure 7 illustrates the cumulated costs in different iterations. With the increase in learning iteration, the decisions of agents are more and more sensible. The agents improve their decision policies continually to optimize the system goal. As a result, the total annual operations cost decreases gradually and even in the same year the operations costs go up more and more smoothly. Figure 8 indicates the total annual operations cost, annual income and annual profit when the algorithm arrives at stable state after 1000 iterations. It’s easy to see that the learning process of agents is effective to reduce the annual operations cost of the supply chain. However, due to the fluctuations of the price of oil products and the market demands, the sales income changes. So the profit doesn’t always rise with the decrease in operations cost.

The research results provide good references for managers to make decisions. At the same time, the methodology proposed here is validated too. The optimization problems for large-scale complex supply chain in real life can really be analyzed and solved by integrating simulation with optimization.
5 Conclusions

At present, scholars always suppose a theoretical model to discuss the coordination and benefit distributing problems among the stages when they conduct research on optimizing supply chain. Due to the complexity of supply chain, theoretical model has to ignore some stochastic factors existing in real life, which weakens its practicability. A case study on a large-scale refinery is implemented in this paper. General-purpose simulation tool ARENA is used to model the complicated supply chain and obtain the system’s performance indices. Combined with VBA and OOP, an architecture integrating simulation with optimization is proposed. Then a kind of multi-agent reinforcement learning algorithm is designed to optimize the ordering and distributing policies of the refinery. The research results help the managers make more reasonable decisions. At the same time, it proves that the methodology proposed in this paper can used to solve the problems of large-scale and complex supply chain in real life. The future research may be as follows:

1) Improve current optimization algorithm, such as introducing the adaptability of the parameters into the algorithm and designing better reward functions etc.

2) Consider the impacts on costs when system environment changes, such as adding docks and adjusting the oil storage capacity etc.

References