CAPSTONE DESIGN

Location Model for Distribution Centers for Fulfilling Electronic Commercial Orders of Fresh Foods with deterministic demand

Advisor: M.Sc. Ngô Thị Thảo Uyên

HoChiMinh City, 1 December 2022

Table of Contents

Acknowledgement

First and foremost, we would want to express our sincere appreciation to my advisor Ngo Thi Thao Uyen, for her guidance on the project and for consulting on our Capstone questions. With her experience in the topic field, the insights we gained were priceless.

She was very dedicated to guiding us and always willing to give feedback to us during the Capstone project. She was also patient in coaching us on the essential skills as well as providing diverse valuable knowledge to aid our analysis and implementation of the project. We also would want to show our gratitude to all the staff of our School of Industrial Engineering and Management for giving us the Capstone guideline and needed support during the semester. We are grateful to our family for always providing the best environment for my studies. Thank all our friends for the encouragement so that we could overcome all the challenges while doing the thesis.

Thank you!

Abstract

A strategic decision dilemma for firms is where to locate distribution hubs to provide fresh food as part of e-commerce. In terms of time and freshness sensitivities, this article establishes a model for locating distribution facilities that takes customer demand for fresh product into account. In this study, the distribution center location problem is resolved using a Genetic Algorithm (GA) optimization algorithm. The model put forth in this study is useful for choosing distribution center locations both theoretically and practically.

Keywords: Locating Distribution Center, Matheuristics, Mathematical Programming, Metaheuristics, Optimization in Operation.

Chapter 1. INTRODUCTION

1. Motivation

Recently, with the rapid development of e-commerce platforms, the demand of customers for convenient shopping food at home and delivered to their door has also increased. Now, there are many big e-commerce platforms like Shopee Mart, Grab Mart, etc. and even big supermarkets like GO!, Aeon, LotteMart, etc. all have their own applications to provide these services to customers' home shopping needs and deliver them to them as quickly as possible. But besides that, there is a challenge that most businesses face that is how to preserve and deliver to customers the freshest and best quality products, because in customer orders most likely have fresh food. Therefore, e-commerce companies are still faced with the challenge of distributing these fresh products in the most efficient and cost-effective approach in order to continually discover the best possible solution. Additionally, it can be challenging for distribution centers to transport all of the merchandise, which results in defective goods being held on the shelf for an extended period of time or not being delivered at all. In order to minimize delivery time and distance from the distributor to the client, a good supply chain and strategically placed distribution centers can help these fresh products be stocked in warehouses and delivered in an appropriate amount of time and distance. As a result, consideration of the location of distribution centers, which can guarantee the freshness of fresh products, plays an important part in the process of transporting fresh goods from the hands of the supplier to the consumer.

2. Problem Statement

Although many e-commerce businesses have several distribution locations in Hanoi and Ho Chi Minh City, covering the need for Vietnam's two most heavily populated cities, the number of distribution centers is still limited in some promising regions with high potential growth rates in the West of Vietnam. Almost all people in the West are now having smart electronic devices, and they also have a lot of needs to make their lives more convenient. Failure to meet these demands may lead to an unsatisfactory customer experience, which in turn leads to further loss of sales down the line. Therefore, there is an increasing need to expand the market to cover more customers' needs and attract potential customers. Considering the high production capacity of suppliers and the variety of retailers in this region such as Bach Hoa Xanh, Winmart+, Co-op mart, etc., the improvement of the supply chain system here will bring a lot of opportunities for e-commerce businesses to increase sales and expand more markets.

Moreover, there are many suppliers here that are already distributed throughout the region, so placing a distribution center here will help in transporting goods from suppliers to distribution centers and retailers will be saving a lot of money. Moreover, it will save holding costs of retailers' inventory by using a distribution center as an inventory that not only plays a role as a warehouse but also a center for delivering products through an online channel in the rapid rise of e-commerce. In addition, factors such as the size and location of the distribution center will affect the freshness of fresh food, shipping times, and distribution costs of an e-commerce business.

Therefore, companies must carefully analyze the demand of customers in the current market to be able to plan for placing a distribution center somewhere in the area. Based on the analysis of customer needs in the market, the design of the distribution center as well as its location will be determined. Not only that, but distribution centers also must solve the problem of storing fresh food, fresh products are not left in the distribution center for too long, so it is a problem of the flow of goods in the distribution center must also be considered here.

3. The Design Project Objectives and Requirements

The targeted problem in this study is to apply the knowledge learned to design an effective supply chain that can replace or complement the current supply chain to assist the expansion of e-commerce businesses' market in western Vietnam by opening new distribution centers in some potential and optimal location, making items more accessible to target customers with a cheaper price and improving manufacturing, transportation, and distribution procedures.

Furthermore, white leg shrimp and cobia supply chain data in western Vietnam, or more specifically in Hoa Binh district of Bac Lieu, will be included in the model to give the reader a better idea of how placing a distribution center will help distribute these fresh products faster and more economical.

4. Scope and Limitations

In our study, it is presumed that e-commerce businesses will establish several distinct distribution centers during a specific time and in a particular location. Products are bought from numerous providers of fresh commodities, and they are then stored in various distribution center locations. Additionally, we presumptively know the quantity and satisfaction needs of the customer of fresh commodities, but we may not be confident of the precise requirements for each order. In other words, many fresh types can be ordered for one order. The scope of this study is to target the supply chain of fresh foods in the Mekong Delta after the Covid-19 pandemic. To be more specific, the supply chain needs clarifying: the distribution centers to be opened, the transportation mode, transportation means to use, the location, size, quantity, and how to deliver the products to customers. This study will take into consideration factors like the shelf life of various perishable goods, their timeliness and freshness, and the design of a distribution center position model that satisfies client demand for quick product delivery. The capacity of anti-interference in the model of finding distribution centers to solve the issue of uncertainty can also be improved by robust optimization. The robust optimization is resolved using an enhanced fruit fly algorithm. When distributing items from the distribution center to the clients, the location model seeks to satisfy customer needs while maintaining the lowest cost with a minimum delivery time. However, there will be factors such as customer segmentation, product sales, population density, etc that will not be mentioned in this study because this model will only focus on placing a distribution center that is most convenient for transportation and optimizes costs and delivery time to keep the freshness of the product.

II/ DESIGN CONCEPTS CONSIDERATION

1/ Introduction

1.1. Design supply chains

The challenge with network design is deciding which facilities to establish in order to perform distribution with the lowest possible cost while maintaining a good degree of customer service. Due to their interaction with the organizational, social, and economic sectors, supply chain design issues are never simple. Therefore, both the industrial and scientific communities are paying increasing attention to the creation of a strong and effective logistic network.

1.2. Distribution center problems

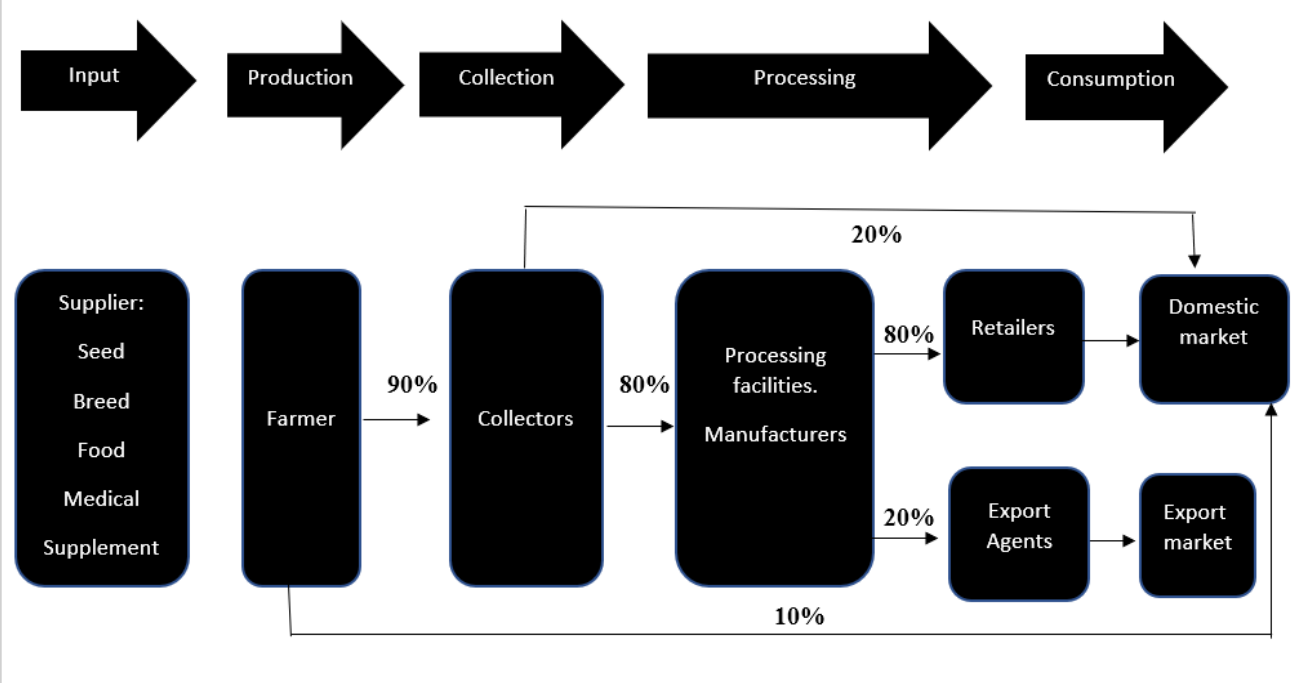
The problem of locating distribution centers for delivering fresh food as a part of electronic commerce is a strategic decision problem for enterprises. The project will find a solution to establish a model for locating distribution centers that can be guaranteed fresh goods in terms of time sensitiveness and freshness.

2/ Literature review

Many scholars have conducted detailed studies on the location of fresh food distribution centers and have proposed various models and algorithms for optimizing location selection.

Hao Zhang and partners established a model of distribution center location and path optimization mainly considering the high quality of perishable food minimizing the total cost of distribution and established a network of distribution centers fast and safe. The model is solved by the method of improved fruit fly optimization algorithm (IFOA). Arta Selimi and Therese Svensson recommend a location for a new distribution center for Svenska Odlarlaget and Samodlarna by using Qualitative and quantitative methods, for more producers to meet Coop‟s and ICA‟s requirements for locally produced fruits and vegetables to address the high demand for domestically produced products. Wang and partners applied discrete (or mixed integer) models, linear mixed integer model and general algorithms, heuristics to find a method to optimize the location and number of research sub-DCs based on carbon policies, carbon tax policies and carbon capitalization trade policies. Rebecca Cleary and partners offers solutions that optimize the strategic design of a robust and reliable supply chain. Addresses facility location problems but also offers new perspectives on building key features to cover facility location issues by applying systematic approaches. The model is solved by linear programming, non-linear programming, multi-objective programming, heuristics, algorithms and metaheuristics, and fuzzy programming. Chong Lin established a model that simultaneously determines optimal location, allocation, inventory, and freshness effort decisions to maximize total profit for a fresh food product using the System dynamics-based methodological approach, continuous approximation method.

3. Current System Investigation



- Supplier:

Main suppliers of raw materials for production: suppliers of seeds and breeds, suppliers of equipment and feedstuff, nutritional and resistance supplements (vaccines, ...).

- Farmer:

The main participants in the livestock production process creating the raw product are the people in the second stage when buying from the suppliers in the first stage (breeds, equipment, foods, etc). In addition, they have a close relationship with traders and lack cooperation with processors (Manufacturers).

- Collectors:

As a bridge in the supply chain between farmers and product processing factories, collectors will find out and help farmers consume their products then provided raw product to processing plant or manufacturing facility.

- Processing facilities:

This is an important link that helps goods to be processed thoroughly as well as helps products meet standards before being sold to the market.

- Retailers:

Who buys goods from the factory and sells them to consumers in the domestic market?

- Export Agents:

Places to handle goods and documents to help goods meet standards when exporting abroad, as well as a place to link and find foreign customers.

- Market (domestic markets and export markets):

In the domestic market, customers prefer to buy fresh products over processed products. One of the top concerns of domestic customers is product safety, as they will be exporting high quality products to export markets.

5. Design Concepts Consideration

- **Exact Method (MILP)**

Exact methods refer to inference procedures that can accurately quantify the uncertainty associated with the statistical model for any finite sample size, in contrast to approximate methods that typically rely on large-sample asymptotic results.

- **GA: Genetic Algorithm**

The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals from the current population to be parents and uses them to produce children for the next generation. Over successive generations, the population "evolves" toward an optimal solution. You can apply the genetic algorithm to solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, nondifferentiable, stochastic, or highly nonlinear. The genetic algorithm can address problems of mixed integer programming, where some components are restricted to being integer-valued.

- **The Fruit fly Optimization Algorithm**

FOA is one of the recently proposed swarm intelligence optimization algorithms used to solve continuous complex optimization problems. FOA has been invented by Pan in 2011 and it is based on

the food search behavior of fruit flies. The FOA has a simple framework, and it is easy to implement for solving an optimization problem with different characteristics. The FOA is also a robust and fast algorithm and some researchers used FOA to solve discrete optimization problems.

- **PSO (Particle swarm metaheuristics)**

Swarm Intelligence algorithms are computational intelligence algorithms inspired by the collective behavior of real swarms such as ant colony, fish schools, bee colony, bat swarms, and other swarms in nature. Swarm Intelligence algorithms are used to obtain the optimal solution for NP-Hard problems that are strongly believed that their optimal solution cannot be found in an optimal bounded time.

Chapter III. SYSTEM DESIGN

1. Approaches Comparison and Selection

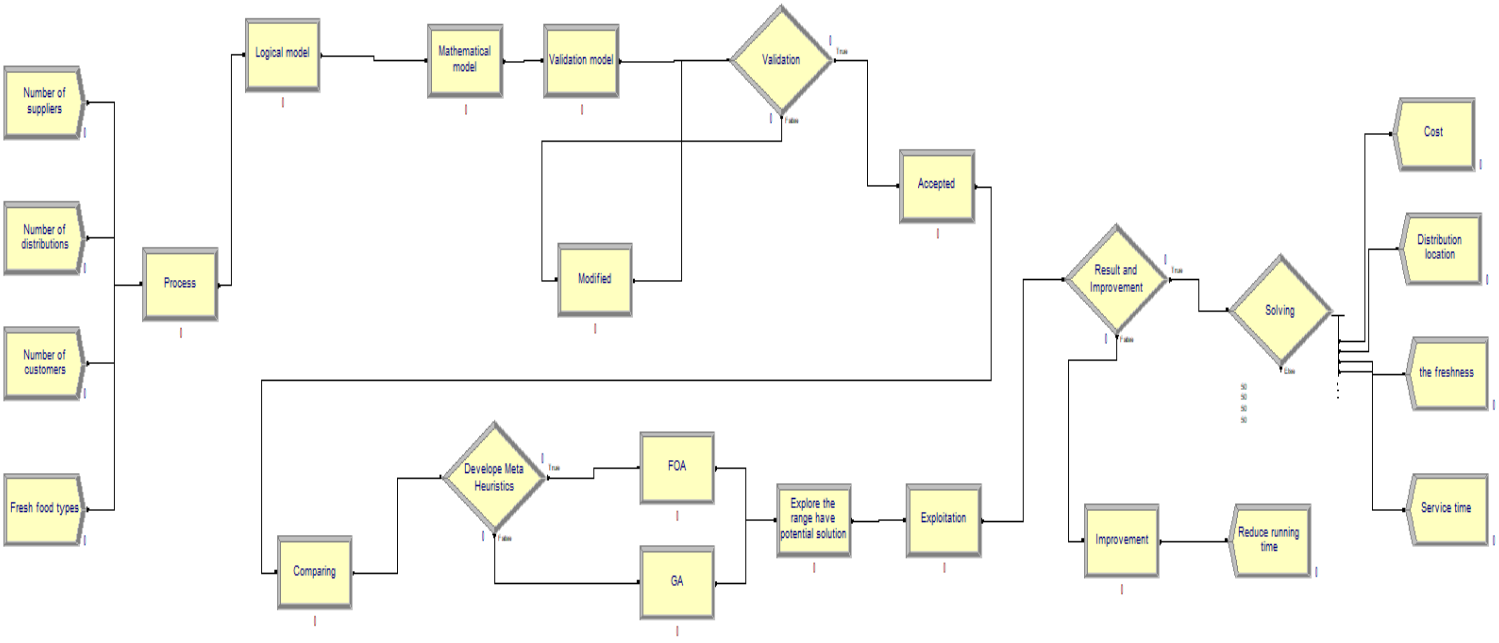
	Advantage	*Disadvantage*	*Optimization*
Genetic Algorithms	• The concept is easy to understand. • GA search from a population of points, not a single point. • GA use payoff (objective function) information, not derivatives. • GA supports multi-objective optimization. • GA use probabilistic transition rules, not deterministic rules. • GA is good for "noisy" environments. • GA is robust w.r.t. to local minima/maxima. • GA is easily parallelised. • GA can operate on various representation.	• GA implementation is still an art. • GA requires less information about the problem but designing an objective function and getting the representation and operators right can be difficult. • GA is computationally expensive i.e. time-consuming.	In computer science, there is a large set of problems, which are NP-Hard. What this essentially means is that, even the most powerful computing systems take a very long time (even years!) to solve that problem. In such a scenario, GAs proves to be an efficient tool to provide usable near-optimal solutions in a short amount of time.

	• GA is stochastic. • GA work well on mixed discrete/continuous problems.		
Particle Swarm Optimization	• Insensitive to scaling of design variables. • Easily parallelized for concurrent processing. • Derivative free. • Very few algorithm parameters. • A very efficient global search algorithm.	• PSO's optimum local searchability is weak. • Need memory to update velocity. • Early convergence	PSO algorithm is superior in finding the optimal solution in terms of accuracy and iteration. In addition, the PSO algorithm is also superior to the simplicity of the techniques used, PSO has a stable condition in finding the optimal solution and PSO is able to get optimal solutions in various conditions. PSO algorithm has ease of implementation and also has high calculation accuracy.
Fruit Fly Optinization Algoritm	• Simple, less parameters, easy to adjust, less calculation, etc., and shortcomings of low convergence precision and easily relapsing into local extremum.	• FOA cannot solve the optimization problems when there exist negative numbers in the domain. • The candidate solution cannot be uniformly generated in the domain with the increase of	Fruit fly algorithm (FOA) is a new-type optimization evolutionary algorithm, which has advantages of simple, easy to adjust, less calculation, efficient searching ability, less parameters to tune, high speed, easy to implement, simple to understand

		the scopes of location range and fruit fly • The probability that the absolute value of x−axis and y−axis becomes large increases, so solution is easy to fall near zero point, and this can explain why FOA can only solve the optimization problems when the extreme point is zero.	

2. System Design Description

- **From the selected approach, develop <u>design structure of the proposed system</u>.**



- **Design Description**

- In the input parameter, collecting the data through the Internet, and some data will collect through the system of our company in last internship. To determine the parameters of the project, using the data in the research paper to be the benchmark for easy to compare and run of the model.me data will be used in the research paper data to reach the exact solution. For the step collecting, we think that

data is just a part of the testing model, for a good model in any situation means in any data, it can be solved and bring out the best solution for that. This is the main objective of our project wanted to reach in the future, but now we just consider the certainty demand, more knowledge, more experience, we will lead to uncertainty demand.

- In the processing, we will consider Logical modeling and Comparing

In Logical Modeling, in this step, think about the new constraints to adapt the new situation based on the traditional problem in the research paper. This step is the most important step because it will affect the mathematical model and the solution. Representing Logical Modeling in words, and detailing. Finishing the Logical Modeling, then changing it to the mathematical model. Of course, in the step mathematical model, it will appear that lots of nonlinear constraints and some constraints get stuck about the conditions. Solving those problems by linearizing and applying more conditions. A validation model is needed to fix the error and also the unreasonable component that makes the model infeasible. After the Validation step, then it will be Modified to complete the model. It will take more than 5 times to validate and modify the model. So, this step will take much time, and it also takes much time because it was the NP-hard problem means the time to run the model change in e^x.

In Comparing, starting to develop a meta-heuristic after getting the final solution. The final solution must represent in the array to put into the metaheuristics to run the solution to compare. It will have 2 algorithms that are compulsory and 1 that is optional if it has time. Taking most of the time to explore and exploit to get the result then comparing the result, to leak out the best result.

The output of the project, get the objective solution to adapt to the situation in the real-time. Because the situation or the market changes frequently, which means the model must update frequently,

and time to run must decrease, it cannot take too much time such as the original model. This step will lead to the future work.

- **Analyze and justify the techniques to be used.**

Deterministic modeling: calculate a future event exactly, without the involvement of randomness. If something is deterministic, you have all the data necessary to predict (determine) the outcome with certainty.

GA (genetics algorithm) is the optional algorithm if we have time. A strategy for using Genetic Algorithms (GAs) to solve NP-complete problems is presented. Since any NP-complete problem can be mapped into any other one in polynomial time, the strategy described here consists of identifying a canonical NP-complete problem on which GAs work well and solving other NP-complete problems indirectly by mapping them onto the canonical problem. So, we can consider this project because it is also an NP-hard problem.

- **Key Advantages of the proposed system design**

  o Easily in changing or editing the model

  o Detailing in formatting through logical modeling

3) Solution Framework Design

3.1) Matheuristics

3.1.1) Introduction

In this study, there are numerous techniques to tackling the optimization problem. The scheduling model can be solved readily using mathematical programming on a small scale (MP). However, in the real world, the model network's scale is far more complex. Because of the NP stiffness, utilizing MP in this scenario may result in unanticipated issue solving time. As a result, various optimization methods are classified as metaheuristics. The advantage of metaheuristics is that it can solve almost any form of optimization issue. However, due to the complexity of the objective in the research model, the fitness function cannot be determined directly in the metaheuristic method.

As a result, to evaluate the exercise function, an MP solver is integrated into the metaheuristic, and this method is known as the matheurisitcs method. Matheuristics are optimization strategies that combine metaheuristic and MP methodologies. Because metaheurisitcs can be applied to a wide range of issues and software can be utilized to solve MP formulae, matheuristics becomes powerful and adaptable. Several approaches for integrating MP with metaheuristics are becoming more common in the matheuristics literature. This combination can be employed in either metaheuristic improvement or generation MP. Metaheuristics are used to enhance existing MP techniques. However, the first of these two techniques have received far more attention.

The matheuristics technique is used with the combination of GA in problem solving and IBM CPLEX providing adaptive function solving in this work. The initial step in the matheuristics procedure

is to begin the population's first generation. The fitness function is then generated directly on MATHLAB and tested using the CPLEX solver on the initial population. Then, until the termination requirements are met, a loop of metaheuristic operations is run. The loop begins with the selection of a parent group for GA processes, followed by hybridization and mutation. A subset will be produced because of these procedures. The MP solver is re-embedded during this phase to evaluate the resulting set.

According to Elitism, the next generation will be generated by absorbing a group of youngsters into the current population. The cycle is now complete, and the metaheuristic stages are repeated. When the termination criteria are met, the procedure is completed, and the best solution is returned.

3.1.2) Solution representation

Solution representation is one of the key factors that need to be established when building metaheuristics because it has a great impact on the productivity and efficiency of the algorithm. In this study, the solution is described in terms of a sequence of distribution centers serving customers in the region. This design gives the algorithm an advantage in two main points.

First, compared to the original representation of the mathematical model, this representation is much simpler and more concise. In the mathematical model, there are 2 variables to represent the number of distribution centers that will be selected to serve the customer, the number of customers suitable for that distribution center.

- Binary 2-dimensional array representing distribution center is selected.
- 2D array representing this customer to be served by the respective distribution center.

Otherwise, represent this solution as just a one-dimensional array consists of binary and integer values describe number of distribution center and number of customers. It streamlines all GA operating activities while saving storage space and resolving the issue with MP software.

Second, by accurately describing the number of distribution centers to be selected and the number of customers to be served, some model constraints can be introduced into the solution representation. So it is possible to give results faster and better than mathematical models.

Distribution center									Customer												
1st DC			2nd DC			3rd DC			C1	C2	C3	C4	s	C6	C7	C8	C9	s	C10	C11	C12
1	0	1	1	1	0	0	1	1	1	2	3	4	0	6	7	8	9	0	10	11	12

Figure 1: The figure illustrates the solution representation of a network of 3 distribution centers and 12 customers.

3.2) Genetics Algorithm

3.2.1) Flowchart

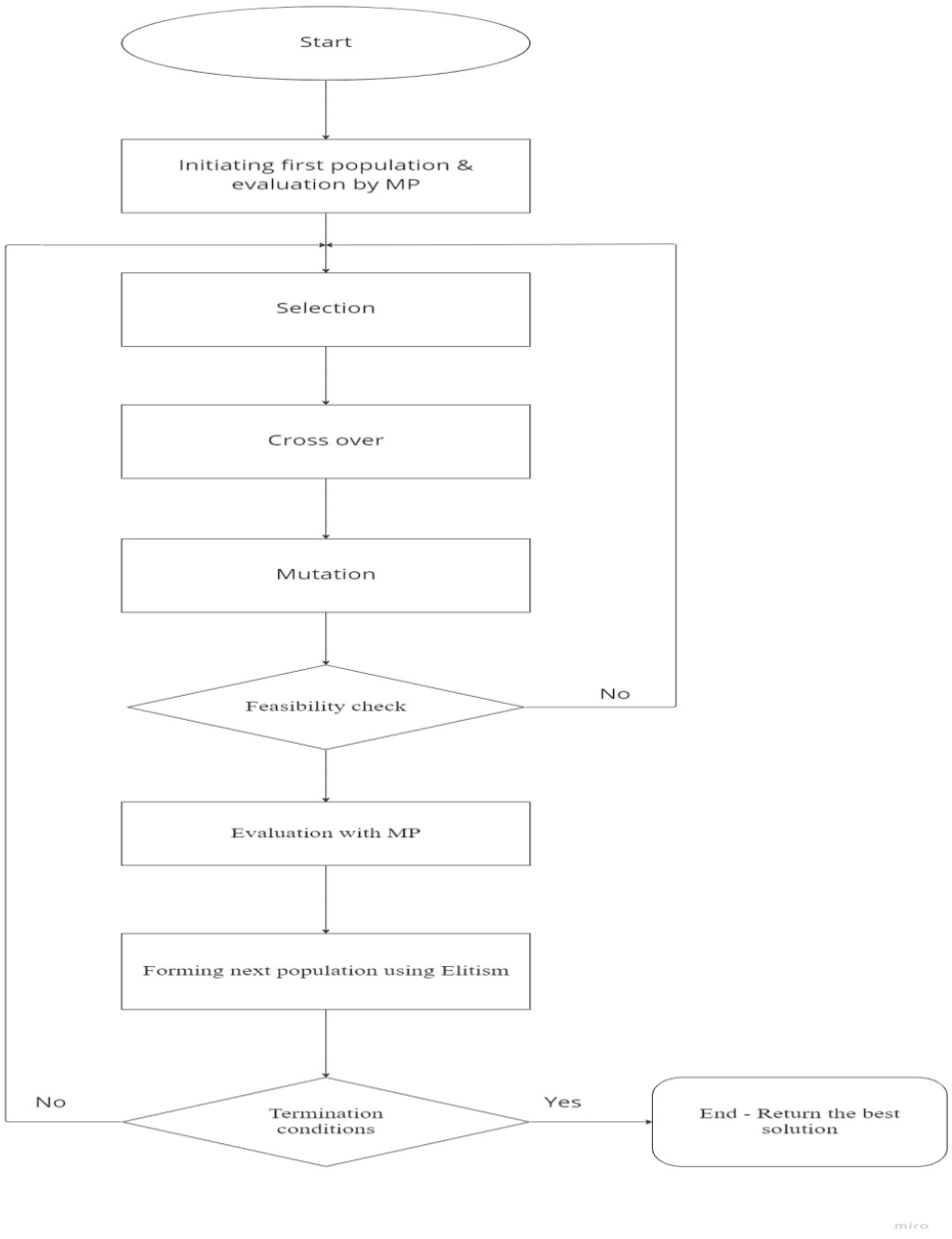


Figure 2: Flowchart of Genetics Algorithm

3.2.2) Initialization and feasibility

The first stage in the GA process is population initialization. The initial population P (0), or first generation, is usually randomly generated. With the structure representing the solution, the first population will be generated by random selection. Each generated chain represents the order in which distribution centers will be served to which customer and must meet conditions such as that the distributions on a chain cannot overlap, and there are no distribution centers with serial numbers. "0", and the customer numbers appear only once. Satisfying the conditions is evaluated for fit, and then added to the original population. Once the procedure collects enough sequences for the population, P (0) is fully generated.

3.2.3) Fitness function

A type of objective function called the fitness function assesses how close a given answer is to the ideal response to a given situation. It evaluates a solution's fit and guides GA operations toward the best design solutions. To evaluate the objective function in this issue, MP is utilized to solve the mathematical model. The initial model was changed to accommodate the mathematic approach. The limitations that are part of the solution representation and those that have been verified by the most recent feasibility test were all eliminated. Additionally, all the associated remaining limitations were changed appropriately.

3.2.4) Selection and cross-over

Pick any two parent strings from the original population P (0) and perform a sequence from the parent (1) to the parent (2), and from the parent (2) to the parent (1). To create 2 brand new substrings

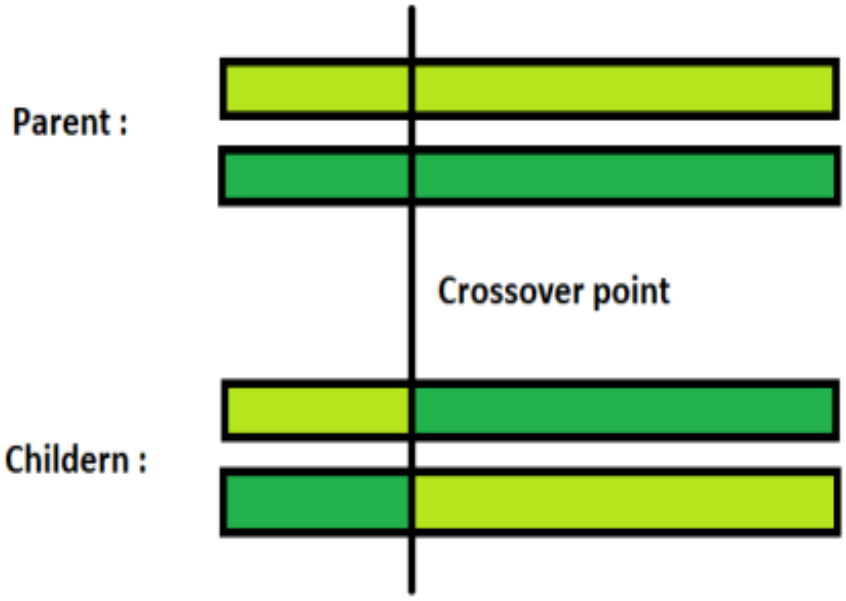


Figure 3: Illustration of GA cross-over

For this problem case, the inversion of the distribution center and customer chains will be different.

- For the distribution center chain, a new chain will be created due to the reversal of the parent chain and the parent chain, then two new chains will be created as described above.
- For the customer chain, the parent chain or parent chain will invert independently based on any position, to form 2 new chains.

3.2.5) Mutation

A chromosome's random change, known as a mutation, produces a novel outcome. The "exploration" of the search space is a function of the GA component mutation. To maintain genetic variety in a population from one generation to the next, a genetic operator is used. Bit Flip Mutation, Random Resetting, Swap Mutation, Scramble Mutation, and Inversion Mutation are a few strategies for a mutation operator.

- Bit Flip Mutation: We choose one or more genes (array indices) and flip their values in bit flip mutation.
- Random Resetting: When performing a random resetting mutation, we choose one or more genes (array indices) and change their values to another random
- Swap Mutation: We choose two genes from our chromosome and swap their values in Swap Mutation.
- Scramble Mutation: We choose a subset of our genes and scramble their values in Scramble Mutation. The chosen genes might not be contiguous.
- Inversion Mutation: We reverse the order of a fraction of our genes in Inversion Mutation. In this situation, the genes must be contiguous.

In this study, Bit Flip Mutation will be used for the Distribution center chain, the location to apply Bit Flip Mutation will be random. For the customer string, permutation will be applied, permutation is the mutation method by transposing all positions on a string to create a new string.

3.2.6) Elitism, Replacement and Forming new population

The children's set is regarded to be added to the population once it has been formed. Elitism selection is a popular way to pick solutions from the children set to replace solutions from the population set. In the Elitism selection, the best individuals from the previous generation are transferred (unaltered) to the following generation. Because the matheuristic has a strong mutation operator to avoid the local optimum trap, Elitism selection is chosen as the technique for producing the new population.

In this study, we will have initialization populations consisting of 10 sets, so after cross-over, and mutation we must have 6 sets, and using Elitism to select 4 sets which has most optimal solution to replace to 4 worst cases of initialization population, to perform new population for next calculation. Continuing to calculate the fitness function till the iteration that has the solution better than solver solution.

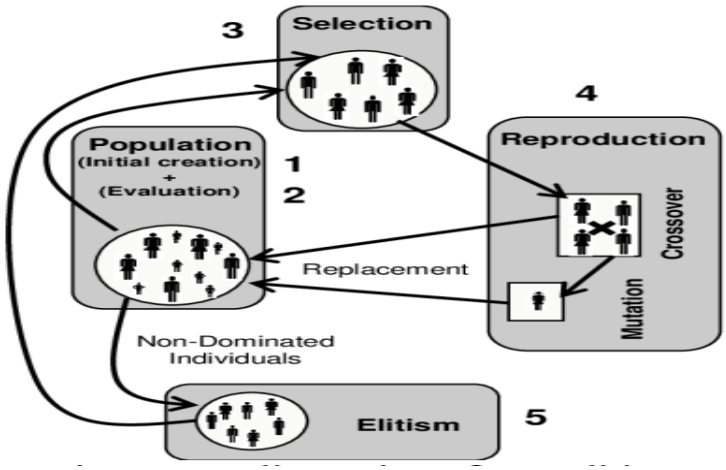


Figure4: Illustration for Elitism, replacement, and perform new population.

CPLEX and MATHLAB

CPLEX

The MP (Mathematical Programming) model is solved using CPLEX IBM software in this paper. The CPLEX system is a sophisticated modeling tool that covers the entire optimization modeling lifecycle, from design to testing, deployment, and maintenance. CPLEX provides a modeling language for describing optimization data, variables, objectives, and constraints, as well as a command language for exploring models and assessing results and a scripting language for data collection and manipulation and iterative optimization techniques.

MATHLAB

MATLAB is a high-performance mathematical computing, visualization, and programming environment software package. It offers an interactive environment with hundreds of built-in technical computing, graphics, and animation functions. MATHLAB is an easy-to-use programming package, as well as how to use and approach the matrix here, which is suited for manipulation and use for GA programming.

Chapter IV. PROTOTYPE DEVELOPMENT AND IMPLEMENTATION

1. Prototype Development

Because of the problem's vast scope, a few assumptions are made to simplify the problem-solving procedure and swiftly derive an estimation result.

- Fresh products e-commerce firms prefer to rent distribution centers from existing candidate distribution centers rather than build ones. The advantage of using existing distribution centers is that the fixed and operational expenses are cheaper than when building a new center.
- Customer demand is deterministic, and any prospective distribution center may fully meet a certain customer's demand.
- Fresh commodity vendors' supply capacity can fulfill the majority of demand.
- Firms do not include fresh goods inventory expenses, loading and unloading losses, vehicle maintenance costs, equipment maintenance costs, and employee bonuses in the overall cost estimate.
- The cost of unit delivery is known and constant.
- The rate of decomposition of fresh items during the distribution process is constant.
- When fresh commodities leave the distribution facility, they remain fresh.

- The freshness of fresh product is always detectable during delivery.
- Although delivery times may vary, drivers are fined based on the degree of deviance from the service requirement.
- This model is only applicable for a specific time and static location assumptions, regardless of future cost and earnings changes.

2. Data Collection

We collect data from local authorities in Bac Lieu province and other sources. Historical data is provided by the Bac Lieu Department of Industry and the Bac Lieu Department of Commerce, Department of Agriculture and Rural Development. Some secondary data was collected by accessing authorized websites, reports, publications, and newspapers. Another data collection method is interviewing farmers and field experts. Some information is collected by direct observation and referencing coordinates and distances from the Google simulation application.

Data

Supplier = H	5
Distribution center = I	8
Customer = J	12
Average speed = v	45 km/h
Largest number of DC rent	3
Fresh food	2

- **Our customers are mainly concentrated in the area around Bac Lieu city where the population is most densely concentrated and has the highest consumption potential center and a few small markets and retail stores in the suburbs.**

Numerical Order	Market	Location
1	Dau Moi Market	Ward 1, Hoang Dieu Street, Bac Lieu City
2	Nha Long Market	Ward 5, Duy Tan Street, Bac Lieu City
3	Ward 1 Market	Ward 1, Cao Van Lau Street, Bac Lieu City
4	Vinmart	Ward 3, Tran Phu Street, Bac Lieu City
5	Coopmart	Ward 7, Tran Huynh Street, Bac Lieu City
6	Thuan Huy Company	Ward 8, 23/8 Street, Bac Lieu City
7	Kim Anh Company	Ward 2, Nguyen Thi Minh Khai Street, Bac Lieu City
8	TRUONG PHU SEAFOODS CO.	No. 1/399, South Hau River Highway, Vinh An Hamlet, Vinh Trach Commune, Bac Lieu City
9	Bau Sang Market	Provincial Road 2, Vinh Binh, Hoa Binh, Bac Lieu
10	Thuy Ngan Seafood	National Highway 1A, Vinh My B, Hoa Binh
11	Dau Kinh Market	Lo Ren, Hiep Thanh, Bac Lieu
12	Vinh Trach Dong Market	Vinh Trach Dong, Bac Lieu

Google simulation application we calculate the distances between:

- **Supplier *h* and distribution center *i*:**

	i1	***i2***	***i3***	***i4***	***i5***	***i6***	***i7***	***i8***
h1	39.3	10.1	13.4	12.7	22.1	28.6	22	20.5
h2	30.4	15.3	37.3	21	13.2	25.2	18.6	16.6
h3	20.9	28.5	39.1	22.8	15.7	28	30.8	30.5

- **Distribution center *i* and customer *j*:**

Distance between distribution center *i* and customer *j* (km)								
	i1	***i2***	***i3***	***i4***	***i5***	***i6***	***i7***	***i8***
j1	22.9	19.3	8.8	20.9	22.5	7.2	10.2	12
j2	21.5	19.8	8.7	21.7	21.9	8.7	11.7	8.2
j3	24.5	16.9	8.5	27.7	20.9	8.2	11.2	12.2
j4	22.6	18.6	9	20.4	20.3	9	12	9.1
j5	24.1	17.9	10	19.8	19.5	9.8	12.7	10.4
j6	24.4	17.1	10.5	17.6	17.8	11	14	11.7
j7	22.1	19.1	8	22.1	21.9	7.3	10.3	9.4
j8	26.3	22.9	12.5	25.7	25.3	3.6	6.5	13.1
j9	29.4	22.6	31.2	25	15.8	35.6	38.6	36.1
j10	20.6	12.2	21.6	14.7	3.5	25.3	28.3	31.8
j11	22.2	18.8	12.9	25.8	28.4	10.1	14.7	5.6
j12	23.2	19.8	15.6	33.7	33.5	7.8	1.7	6.6

- **Based on statistics on human costs, energy costs, space rents as well as costs for equipment and occupational safety in the workplace, we have obtained a cost information sheet fixed annual fees for distribution centers.**

i	fixed cost	Maximum capacity dc	Operation cost
1	1907290000	3500	235315000
2	1622435000	3200	200637000
3	1758670000	3000	193206000
4	1832980000	5000	220453000
5	1849080000	2500	221615000
6	1985635000	4400	186426000
7	1744080000	3500	180453000
8	1895867000	4000	210453000

- **Based on the optimal distances between suppliers and distribution centers to get the best source of goods to ensure both costs as well as maintain the best freshness and product quality.**

Quantity shipped from supplier to DC								
Supplier/DC	1	2	3	4	5	6	7	8
Vinh Thinh	0	2450	3000	1900	0	0	0	0
Vinh Hau	500	750	0	880	1700	2000	1600	2000
Vinh My A	3000	0	0	0	1300	1500	1100	1200

Quantity shipped from DC to Customer								
Customer/DC	i1	i2	i3	i4	i5	i6	i7	i8
1	0	0	558	0	0	550	0	0
2	0	0	400	0	0	400	0	467
3	0	0	485	0	0	550	0	0
4	0	0	0	0	0	0	0	392
5	0	0	0	273	0	0	0	0
6	0	0	750	1350	0	2000	0	1000
7	2000	1500	800	0	0	0	550	200
8	0	0	0	0	0	0	700	0
9	0	0	0	0	1570	0	0	0
10	1500	500	0	970	1430	0	0	0
11	0	1200	0	0	0	0	0	1100
12	0	0	0	0	0	0	1350	0
Sum	3500	3200	2993	2593	3000	3500	2600	3159

Mathematical model

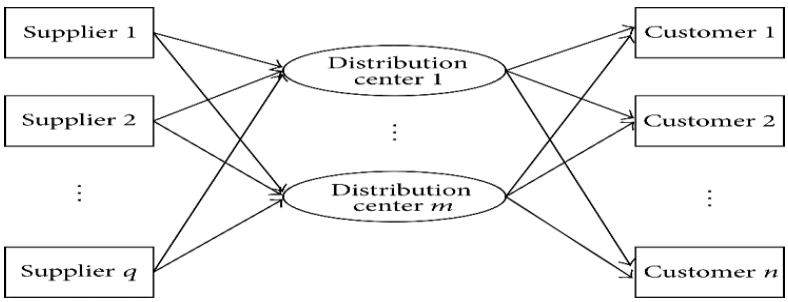


Figure5: Illustration Logic diagram of locating distribution centers.

Parameter

H: Set of suppliers $H = \{h \mid h = 1, 2, \ldots, q\}$

I: Set of distribution centers $I = \{i \mid i = 1, 2, \ldots, m\}$

J: Set of customers $J = \{j \mid j = 1, 2, \ldots, n\}$

K: Set of fresh food types K = {k| k = 1,2…, n}

U_h: Maximum supply capacity of supplier h to the enterprise

N_i : Maximum inventory capacity of distribution center i

θ_k : Corruption rate coefficient of fresh food k

l_{hi} : Distance between supplier h and distribution center

l_{ij} : Distance between distribution center i and customer j

v: Average speed of vehicle

C_{hi} : Transportation cost per unit from supplier h to distribution center i

C_{ij} : Transportation cost per unit from distribution center i to customer j

A_i : Fixed cost of distribution center i

Bi: Operating cost of distribution center i

f_{ij} : Penalty costs of vehicle arriving late to customer j

w: The largest number of new distribution centers available to rent

r_k: The average selling price of the variety k of fresh goods

t_{ij} : Time of vehicle from distribution center i to customer j

T_j : The serve time that customer j requires from the delivery center.

$T_{j\,max}$: The shortest waiting time that customer j is not dissatisfied.

$T_{j\,min}$: The longest waiting time that customer j is dissatisfied.

α_{jk} : The freshness satisfaction threshold for customer j of variety k of fresh goods

β_j : The time satisfaction threshold for customer j

x_{max}: $\sum_h^H \sum_i^I \sum_j^J [\lambda_1 (l_{hi}/v + t_{ij}) + \lambda_2 (l_{hi} + l_{ij})]\, z_{ij}$

$\lambda_1 = 0.7$: weight for time satisfaction

$\lambda_2 = 0.3$: weight for distance satisfaction

d_j : The demand of customer j

Decision Variables

d_{hi} : The transport quantity of goods from supplier h to distribution center i

d_{ij} : The transport quantity of goods from distribution center i to customer j

y_i : Binary variable that takes the value 1 if distribution center i is selected

z_{hi} : Binary variable that takes the value 1 if distribution center i is supplied by supplier h

z_{ij} : Binary variable that takes the value 1 if customer j is supplied by distribution center i

z_{jk} : Binary variable that takes the value 1 if fresh food k is supplied to customer j.

Mathematicals model

Location Mathematical Model under Determinate Demand: Location Mathematical Model under Determinate demand is a type of optimization problem that seeks to minimize the cost of placing a certain number of facilities in different locations based on the needs of consumers. The model is based on the assumption that the demand for each facility is predetermined and known, and the goal is to determine the optimal locations for the facilities that result in the lowest possible total cost while still meeting demand. This model involves various parameters such as the distance between locations, the size of the facilities, the capacity of the facilities, the demand at each location, and the cost associated with each location, fixed costs, operating expenses, transportation costs, delayed penalty costs, and fresh goods corruption costs for distribution centers fixed costs, operating expenses, transportation costs, delayed penalty costs, and fresh goods corruption costs for distribution centers. Additionally, constraints need to be considered, such as:

Fixed costs and operating expenses

$$S1 = \sum_i^I A_i y_i + \sum_i^I B_i * y_i$$

Transportation costs

$$S2 = \sum_h^H \sum_i^I C_{hi} l_{hi} d_{hi} + \sum_i^I \sum_j^J C_{ij} l_{ij} d_{ij}$$

Delayed penalty costs

$$S3 = \begin{cases} f_{ij}(t_{ij} - T_j) & t_{ij} > T_j \\ 0 & t_{ij} \leq T_j \end{cases}$$

Fresh good corruption costs

$$x = \sum_h^H \sum_i^I \sum_j^J [\lambda_1 (l_{hi}/v + t_{ij}) + \lambda_2 (l_{hi} + l_{ij})] z_{ij}$$

$$\emptyset_k = (1 - \theta_k)^x$$

$$= \begin{cases} \frac{x_{max}}{3}(\theta - 1) - x\,(\theta_{\frac{x_{max}}{3}} - 1), & 0 \le x \le \frac{x_{max}}{3} \\ \frac{x_{max}}{3}(\theta - \theta_{\frac{x_{max}}{3}}) - (x - \frac{x_{max}}{3})(\theta_{\frac{2x_{max}}{3}} - \theta_{\frac{x_{max}}{3}}), & \frac{x_{max}}{3} < x \le \frac{2x_{max}}{3} \\ (\theta - \theta_{\frac{2x_{max}}{3}})(x_{max} - \frac{2x_{max}}{3}) - (x - \frac{2x_{max}}{3})(\theta_{x_{max}} - \theta_{\frac{2x_{max}}{3}}), & \frac{2x_{max}}{3} < x \le x_{max} \end{cases}$$

$$S4 = \sum_k^K r_k (1 - \emptyset_k)$$

The freshness of fresh goods

$$\tau_{jk} = \sum_j \sum_k^K z_{jk} \emptyset_k$$

The distribution center service time

$$\mu_j = \sum_i \sum_j^J z_{ij} \omega(t_{ij})$$

With

$$\omega(t_{ij}) = \begin{cases} 1 & t_{ij} \leq T_{jmin} \\ \dfrac{T_{jmax} - t_{ij}}{T_{jmax} - T_{jmin}} & T_{jmin} < t_{ij} < T_{jmax} \\ 0 & t_{ij} \geq T_{jmax} \end{cases}$$

Objective function:

$$\text{Minimize } S = 0.6*(s1 + s2 + s3 + s4) - 0.2* \sum \mu_j - 0.2* \sum \tau_{jk}$$

Constraint 1:

Supply constraint of the supplier, ensure that the total amount of products transported from supplier h does not exceed the maximum supply capacity.

$$\sum_i^I d_{hi} \leq U_h \; \forall h \in H$$

Constraint 2:

Distribution center inventory constraints, and they enforce that the storage of the total fresh product does not exceed the maximum inventory capacity.

$$\sum_h^H d_{hi} \leq N_i y_i \qquad \forall i \in I$$

$$\sum_j^J d_{ij} \leq N_i y_i$$

Constraint 3:

Represents the number of new distribution center constraints.

$$\sum_{i}^{I} y_i \leq \omega,$$

Constraint 4:

Ensures that each customer has only one candidate distribution center to deliver to him or her.

$$\sum_{i}^{I} z_{ij} = 1, \qquad \forall j \in J$$

Constraint 5:

Indicates that the volume of fresh foods delivered to the customer cannot be less than the customer demand, due to possible cargo damage.

$$\sum_{i}^{I} d_{ij} \geq d_j, \qquad \forall j \in J$$

Constraint 6:

Ensures that the selected distribution center can provide delivery.

$$z_{ij} \leq y_i, \qquad \forall i,j \in I,J$$

$$z_{hi} \leq y_i, \qquad \forall h,i \in H,I$$

The business goal is to minimize the total cost and the freshness, and the distribution center service time only needs to achieve a consumer satisfaction threshold, then consumer satisfaction does not have to be maximized, but consumer satisfaction is transformed into a constraint. In this paper, the main objective function is chosen to be the minimum total cost, and the satisfaction of the consumer is set as a constraint.

Result and Sensitive Analysis

RESULT

For this model, the desired output will be the component costs, freshness, and DC uptime to serve all customers.

Because of the many nonlinear limitations in CPLEX, the expected results will be in error relative to the non-linearized model. In parallel, there are some problems with the model's scale data that can be met. Here are some cases where the model will work in CPLEX, but for a larger scale, it gives infeasibility.

Scale 1: The smallest scale, to show that the logic of the problem has no problem when giving the result, then gradually increase the scale.

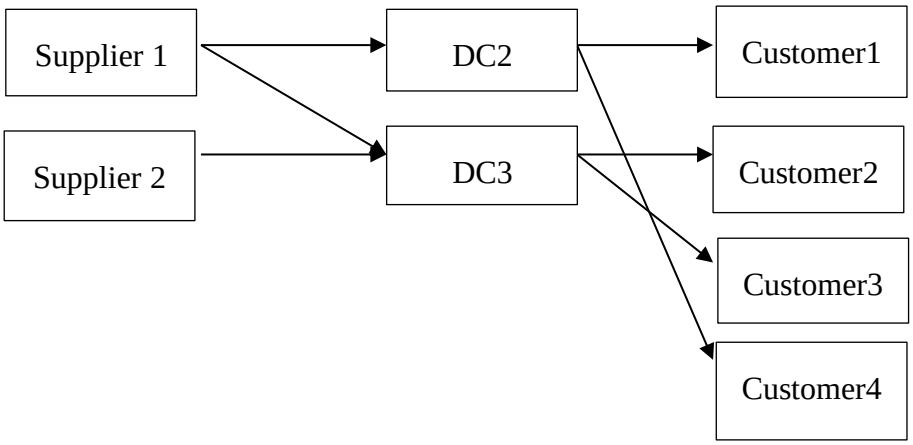
Supplier	2
Distribution center	3
Allow renting	2
Customer	4

The distribution centers were being selected are 2nd distribution, 3rd distribution

Operation cost and fixed cost	3,774,900,000 VNĐ
Transportation cost	17,545,000,000 VNĐ
Penalty cost	21,114 VNĐ
Corruption cost	259,259.72 VNĐ
Distribution center service time	11.3104 hours
Freshness	1.538387588

Give the objective function: 12,792,222,237.

Routing:



Scale 2:

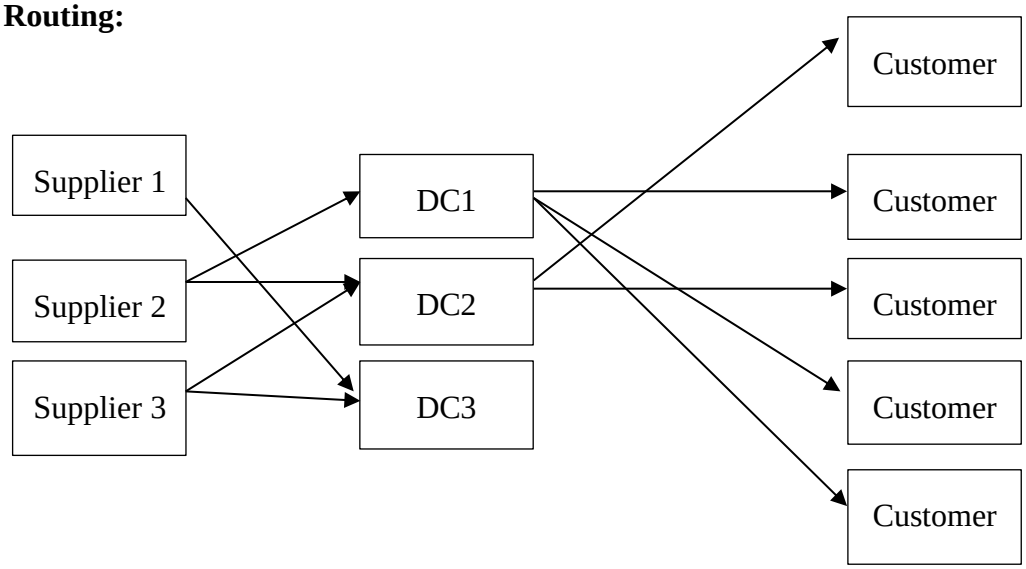
Supplier	3
Distribution center	4
Allow renting	3
Customer	5

The distribution centers were being selected are 2nd distribution center, 3rd distribution center, 4th distribution center.

Operation cost and fixed cost	5,828,400,000 VNĐ
Transportation cost	24,052,000,000 VNĐ
Penalty cost	20,950 VNĐ
Corruption cost	212,709.262 VNĐ
Distribution center service time	12.32 hours
Freshness	3.373543238

Give the objective function: 17,929,624,742.

Routing:



Scale 3:

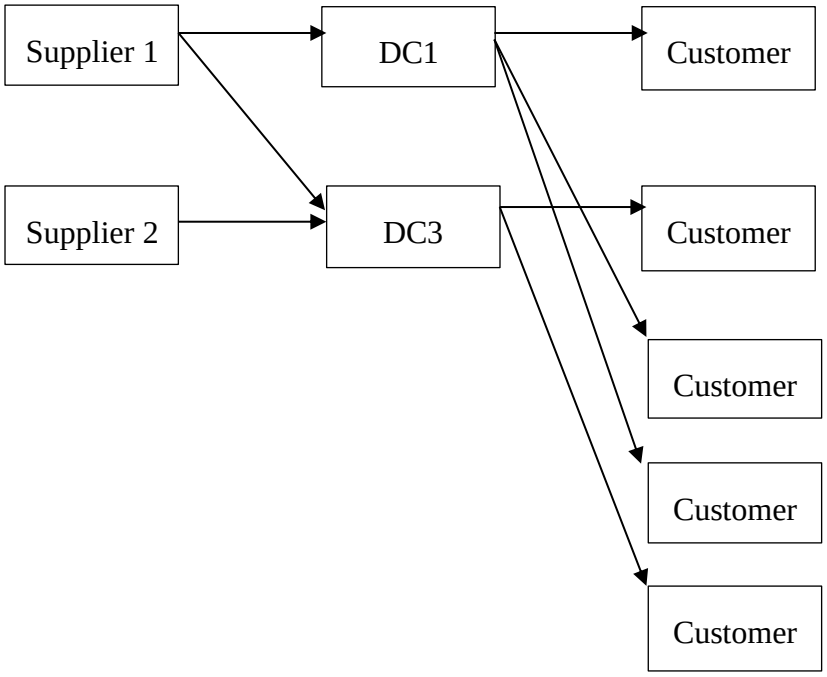
Supplier	2
Distribution center	5
Allow renting	2
Customer	5

The distribution centers were being selected are 1st distribution center, 3rd distribution center.

Operation cost and fixed cost	6,147,900,000 VNĐ
Transportation cost	32,106,000,000 VNĐ
Penalty cost	67,278 VNĐ
Corruption cost	217,193.5526 VNĐ
Distribution center service time	12.68 hours
Freshness	3.880614854

Give the objective function: 22,952,510,680.

Routing:



We have tried scaling it up to 3 suppliers, renting 3 over 6 distribution center and serve for 7 customers, and the results on CPLEX are not showing up yet. This confirms that the model is limited to this when used on CPLEX.

IMPROVEMENT

To increase the scanning range of the model, I have applied the Genetics Algorithm to apply this model, but due to not fully understanding my own knowledge, it is still unfinished in the process. Elitism. We will report and complete the algorithm on defense day. Under the Appendix is the process we are implementing the algorithm.

When applying GA to this model, the scale of the model will increase, solving problems such as solving time, objective function will improve because there is no longer a nonlinear problem, and will not appear. error, so it will be completely optimal than CPLEX's answer.

SENSITIVE ANALYSIS

I. Mathematical model

We will do sensitive analysis on case 2 to see how the model changes when we increase and decrease a unit amount of demand and operation cost. Here are the results:

	Default	**Increasing 10^6 units in operation cost**	**Dreasing 10^6 units in operation cost**	**Increasing 1000 units in demand**
Operation cost and fixed cost	5828400000	5831400000	5825400000	5828400000
Transportation cost	24052000000	24052000000	24052000000	25103000000
Penalty cost	20,952	20,952	20,952	66,893
Corruption cost	212709.262	212709.262	212709.262	227164.8638
Distribution center service time	12.32	12.32	12.32	8.4668
Freshness	3.373543238	3.373543238	3.37354324	2.923212962

Objective function	17928368880	17930168880	17927000000	18558980310

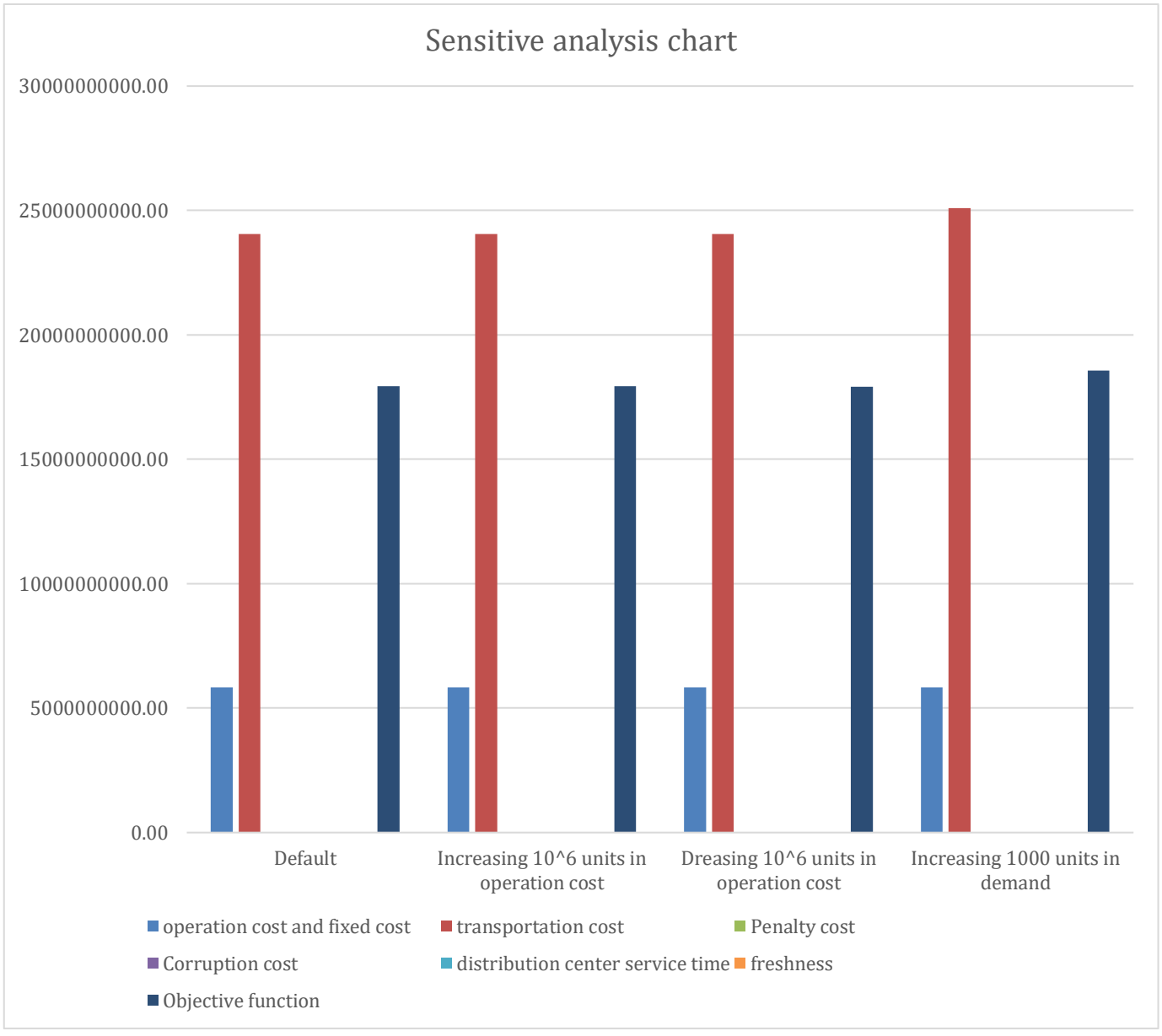


Figure 6: Sensitive Analysis Chart

II. Meta-heuristics (Genetics Algorithm)

Here are the results of Genetics Algorithm. Due to the nature of the method, it is a random calculation method from the time the population is formed until the time when children are created. Therefore, for each iteration, the model will give different calculation methods, because when changing the iteration from 50 to 100, the model also randoms the population, so a large iteration does not mean good results. than. Better results may appear on lower iterations or may appear on higher iterations. Here with such calculation method, it is not possible to give the best optimal result for the whole model but only the best result for each population that can be random, so increasing and decreasing the number of arrays in the population and increasing and decreasing the number of children generated instead also affects the run time of the model and the result. Here are some illustrations.

a) Perform with 10 arrays in population and take 4 in 6 children was created.

Array solution:

Iteration	1st DC			2nd DC			3rd DC			Customer													
50	1	1	0	1	0	1	1	1	1	12	6	5	1	2	0	11	9	10	4	0	7	8	3
100	1	1	0	1	1	1	1	0	1	5	6	0	3	8	1	4	12	7	2	0	9	11	10
200	1	1	0	1	0	1	1	1	1	1	2	6	11	9	8	7	0	10	0	12	3	4	5
400	1	1	1	1	1	0	1	0	1	0	12	2	3	6	7	11	5	4	8	0	9	10	1
500	1	1	1	1	1	0	1	0	1	0	8	5	11	2	1	6	12	7	4	3	0	9	10
800	1	1	0	1	1	1	1	0	1	12	2	8	7	4	3	6	11	0	1	0	5	10	9
1000	1	1	1	1	1	0	1	0	1	1	4	8	12	2	0	7	6	11	5	0	9	3	10

5000	1	0	1	0	1	0	1	1	0	5	10	9	0	0	2	3	8	7	4	6	1	12	11

Total cost solution:

Iteration	Total cost	Iteration	Total cost
50	18,908,000,000	500	13,843,000,000
100	17,960,000,000	800	14,181,000,000
200	16,567,000,000	1000	13,872,000,000
400	15,590,000,000	5000	13,969,000,000

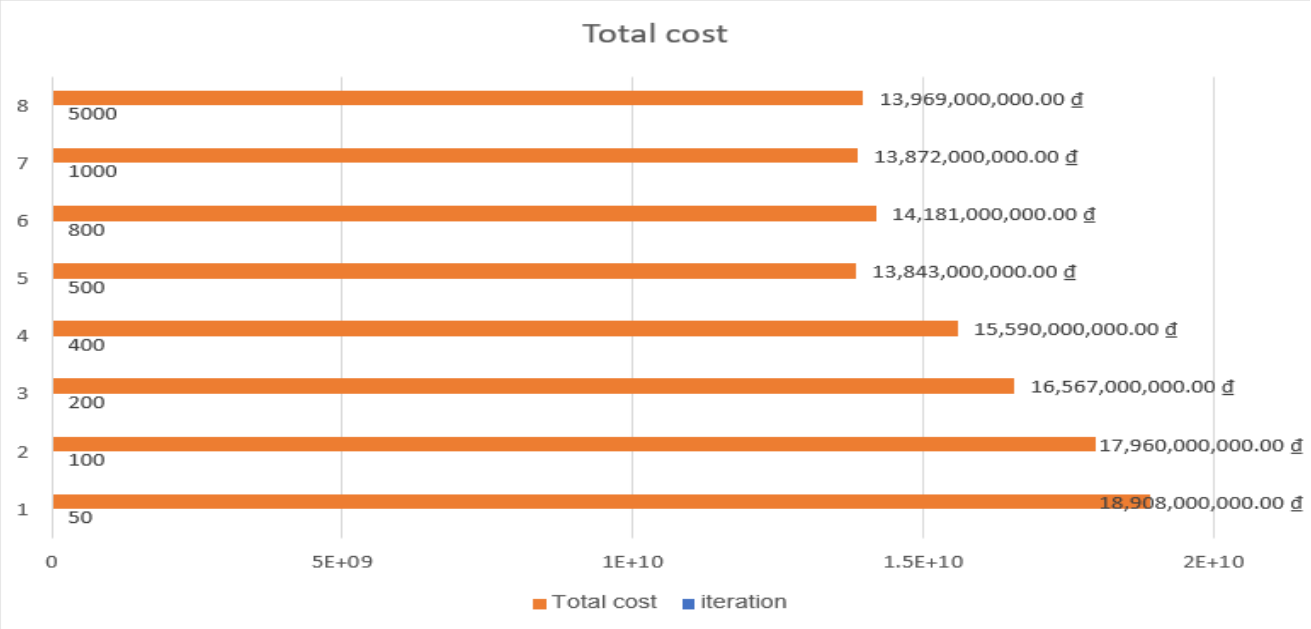


Figure 7: Illustrate total cost during the iterations.

The results through the above graph show that, at iteration 500, the results are better than all the remaining iterations.

b) Perform with 12 arrays in population and take 6 in 8 children was created.

Array solution:

Iteration	1st DC			2nd DC			3rd DC			Customer													
50	0	1	0	1	1	0	1	1	1	10	9	0	2	7	12	5	6	8	1	11	4	3	0
100	1	0	1	1	1	0	0	0	1	10	9	8	0	12	6	3	7	11	2	4	5	1	0
200	1	1	1	1	0	1	1	1	0	7	12	0	1	9	10	0	2	11	3	4	6	5	8
400	0	1	0	1	1	0	1	0	1	0	1	6	2	11	5	8	4	7	3	12	0	10	9
500	1	1	0	1	0	1	0	1	0	6	7	4	12	3	11	2	8	1	0	10	0	5	9
800	1	0	1	1	1	0	0	1	0	9	10	5	0	1	7	8	3	11	6	4	12	2	0
1000	1	1	1	1	1	0	1	0	1	8	12	0	11	2	3	7	1	5	6	0	9	10	4
5000	1	0	1	1	1	0	1	1	1	9	10	0	5	6	11	7	12	4	0	3	1	8	2

Total cost solution:

Iteration	Total cost	Iteration	Total cost
50	16,629,000,000	500	15,150,000,000
100	16,278,000,000	800	13,969,000,000
200	15,254,000,000	1000	12,714,000,000

400	13,739,000,000	5000	13,805,000,000

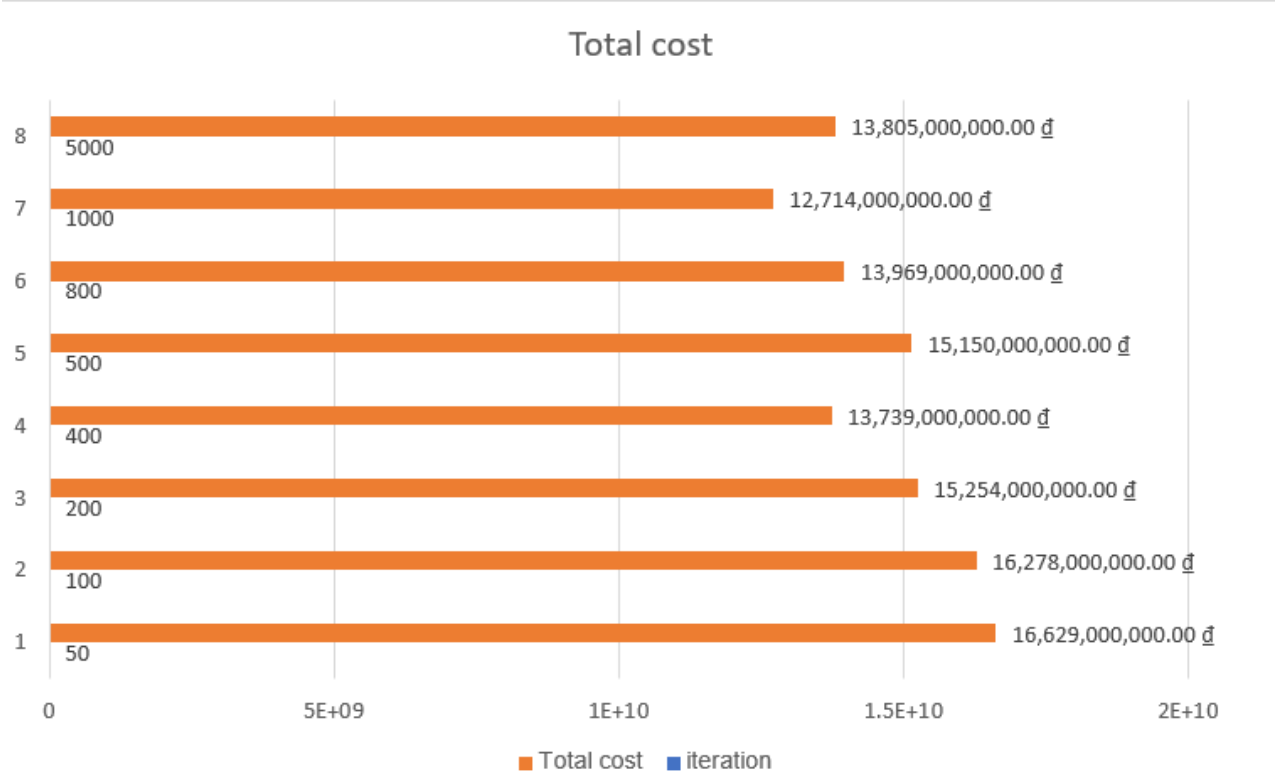


Figure 8: Illustrate for total cost during the iterations.

The results through the above graph show that, at iteration 1000, the results are better than all the remaining iterations.

c) Perform with 14 arrays in population and take 7 in 10 children was created.

Array solution:

Iteration	1st DC			2nd DC			3rd DC			Customer												
50	1	1	1	1	0	1	1	1	0	6	11	8	7	12	0	5	4	10	2	9	3	0	1

100	0	1	0	1	1	1	1	1	0	9	8	10	0	3	0	11	12	7	4	1	2	5	6
200	1	0	1	1	1	1	1	1	0	4	10	5	9	3	0	7	11	2	8	0	6	1	12
400	1	1	1	1	0	1	1	1	0	2	1	12	7	0	10	5	4	9	0	8	11	3	6
500	1	1	0	0	1	0	1	0	1	4	11	2	7	8	5	3	6	12	0	1	0	9	10
800	1	1	0	1	0	1	1	1	1	1	11	6	0	5	10	9	0	7	2	8	3	4	12
1000	1	1	1	1	0	1	1	1	0	3	4	7	0	10	9	0	6	5	8	11	12	2	1
5000	1	1	0	1	1	1	1	0	1	3	2	1	6	7	11	8	5	0	4	12	0	10	9

Total cost solution:

Iteration	Total cost	Iteration	Total cost
50	17,904,000,000	500	14,941,000,000
100	18,550,000,000	800	13,664,000,000
200	16,263,000,000	1000	14,720,000,000
400	14,466,000,000	5000	12,943,000,000

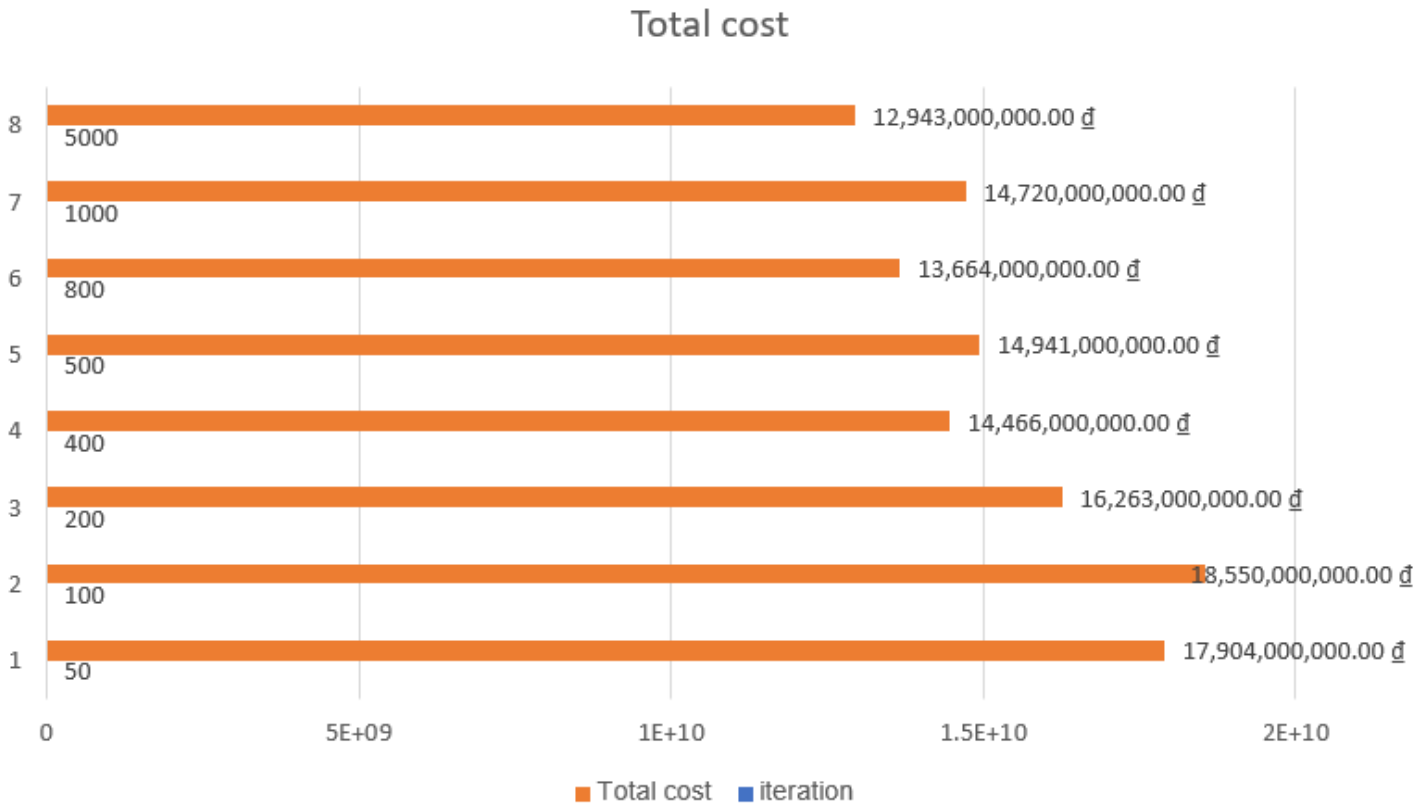


Figure 9: Illustrate total cost during the iterations.

The results through the above graph show that, at iteration 5000, the results are better than all the remaining iterations.

Summary:

Case	Iteration	Best solution
10 arrays in population and take 4 in 6 children	500	13,843,000,000
12 arrays in population and take 6 in 8 children	1000	12,714,000,000
14 arrays in population and take 7 in 10 children	5000	12,943,000,000

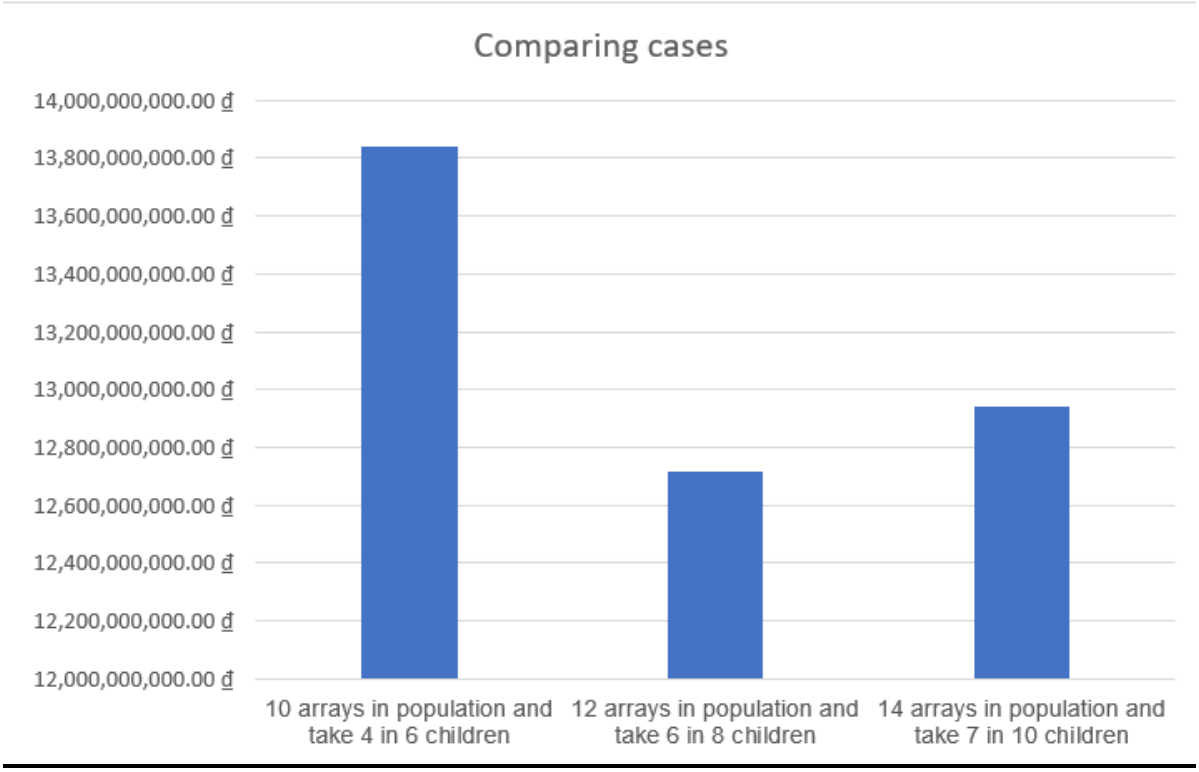
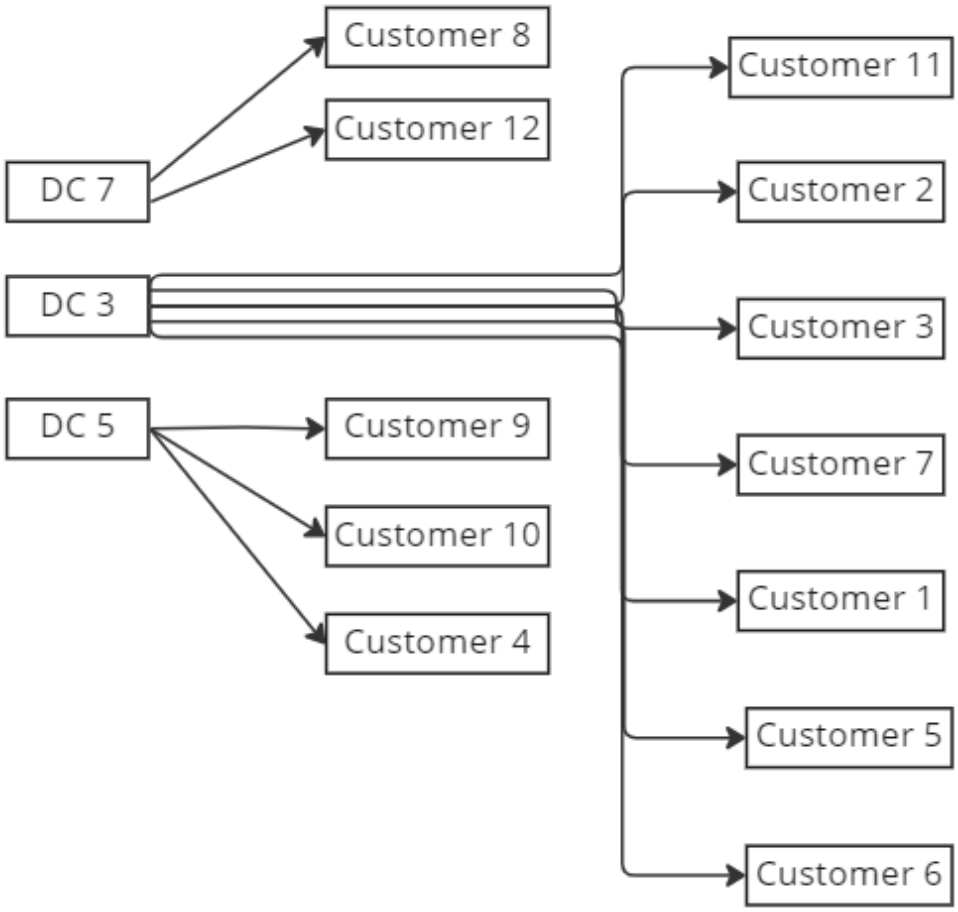


Figure 10: Illustrate for 3 cases of model in GA.

In this case, the best solution during all case is cases “12 arrays in population and take 6 in 8 children was created”, has the solution is 12,714,000,000.

The order of DCs serving customers will be shown in the array below.

1	1	1	1	1	0	1	0	1	8	12	0	11	2	3	7	1	5	6	0	9	10	4



Chapter V. CONCLUSIONS

1. Effect on economic and environmental aspects.

- ***Environmental impact:***

Transportation is a major energy consumer in the world and consumes large amounts of gas and oil, so the environmental impact of transportation is huge. In the transport sector, road transport, a major contributor to carbon dioxide and other greenhouse gas emissions, is the largest contributor to global warming.

A well-located supply chain network can help reduce transport distances and the severe environmental impacts of road transport, such as air pollution, noise pollution, and power consumption.

- ***Social impact:***

A standardized supply chain network can have a positive impact on society by assisting in the management of issues like transportation, land use planning, production and company management, economic development, quality enhancement, and other related residents' quality of life in the area.

- ***Economic impact:***

By lowering operating and transportation costs, the supply chain boosting the profitability of goods, their competitiveness in the global market, ensuring the quality of goods helps goods to be consumed more, contributing to the development of the economy and allowing the supply chain to be adaptable

enough handle recent changes and uncertainties, like rising demand, material import cost and quality (freshness, safety, cost, ...).

2. Conclusion.

The area of conveyance centers may be a complex optimization issue. To illuminate the issue of outlandish, arrange format, destitute client request, and tall conveyance costs, this article optimizes the area of dispersion centers, minimizing the full cost of dissemination. Set up a organize of dispersion centers rapidly and completely meet the requirements of clients. The conclusions of this article are summarized as follows. In this paper, a dissemination center demonstrate is built up to diminish the overall taken a toll of the endeavor. The show is consolidated with the inborn characteristics of new merchandise e-commerce, taking under consideration the standards of conveyance center choice, benefit levels and freshness confinements. Due to the increment within the number of new merchandise e-commerce businesses with fiercer competition, customer request is progressively dubious. Effective optimization is utilized to optimize the dispersion center demonstrate, making the show more powerful.

In this research, a Genetic Algorithm (GA) optimization algorithm is used. The addition of the weight function allows for dynamic adjustment of the search step size. GA will help to address the existing pattern. GA, on the other hand, has a wider and more exact scope of application.

The impact of specific demand on the location of distribution centers for fresh produce is examined in this article. It enhances the concept of fresh goods e-commerce distribution centers and offers the theoretical underpinnings for a methodical approach to choosing the sites of fresh e-commerce

distribution centers online. theoretically and practically related to a different issue like ambiguous client demand. The proposed methodology can be applied to make meaningful and practical decisions on where to locate a new e-commerce distribution facility.

References

[1] Zhang, H., Xiong, Y., He, M., & Qu, C. (2017). Location Model for Distribution Centers for Fulfilling Electronic Orders of Fresh Foods under Uncertain Demand. Hindawi, 2017. https://doi.org/10.1155/2017/3423562

[2] Han, L. T. N. (2020). Design supply chain for agricultural productions of Vietnam: A case study of white leg shrimp in Hoa Binh district, Bac Lieu province (Doctoral dissertation, International University-HCMC).

[3] (n.d.). Occupational Safety and Health Administration. UNITED STATES DEPARTMENT OF LABOR. https://www.osha.gov/

[4] ANSI Webstore (n.d.). Manufacturing and Production Standards. https://webstore.ansi.org/industry/manufacturing-production

[5] Wikipedia (n.d.). Genetic algorithm - Wikipedia. https://en.wikipedia.org/wiki/Genetic_algorithm#

[6] Sai Chandana, K. (2020). A New Approach for Solving the Disruption in Vehicle Routing Problem During the Delivery: A Comparative Analysis of VRP Meta-Heuristics. Faculty of Computing, Blekinge Institute of Technology.

[7] Hassanzadeh Amin, S., & Baki, F. (2016). A facility location model for global closed-loop supply chain network design. ScienceDirect, 41(0307-904X), 316-330. https://doi.org/10.1016/j.apm.2016.08.030

[8] Hassanat AB, Prasath VBS, Abbadi MA, Abu-Qdari SA, Faris H. An Improved Genetic Algorithm with a New Initialization Mechanism Based on Regression Techniques. Information. 2018; 9(7):167. https://doi.org/10.3390/info9070167

[9] Kora, P., & Yadlapalli, P. (2017). Crossover operators in genetic algorithms: A review. International Journal of Computer Applications, 162(10).

[10] Lambora, A., Gupta, K., & Chopra, K. (2019, February). Genetic algorithm-A literature review. In 2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon) (pp. 380-384). IEEE.

Table of Group Members

Name	Student ID	Contribution
Nguyễn Hải Nam	IEIEIU19008	100%
Phạm Ngọc Huy	IELSIU19166	100%
Trần Bảo Sơn	IELSIU19256	100%

APPENDIX

CPLEX File.mod

```
/*********************************************
 * OPL 12.8.0.0 Model
 * Author: USER
 * Creation Date: 25-01-2023 at 22:47:07
 *********************************************/
/*********************************************
 * OPL 12.10.0.0 Model
 * Author: nghai
 * Creation Date: Nov 19, 2022 at 2:45:53 PM
 *********************************************/
//parameters
execute
{
  cplex.optimalitytarget=3;
}
int H = ...; //Set of suppliers
range rangeh = 1..H;
int I = ...; //Set of distribution centers
range rangei = 1..I;
int J = ...; //Set of customers
range rangej = 1..J;
int K = ...; //Set of fresh food types
range rangek = 1..K;
int U[rangeh] = ...; //Maximum supply capacity of supplier  to the enterprise
int N[rangei] = ...; //Maximum inventory capacity of distribution center
float R[rangek] = ...; //Corruption rate coefficient of fresh food , constant
float l1[rangeh][rangei] = ...; //Distance between supplier  and distribution center
float l2[rangei][rangej] = ...; //Distance between distribution center  and customer
int v = ...; //Average speed of vehicle
int c1[rangeh][rangei] = ...; //Transportation cost per unit from supplier  to
distribution center
float c2[rangei][rangej] = ...; //Transportation cost per unit from distribution center
to customer
int A[rangei] =...; //Fixed cost of distribution center
float B[rangei] = ...; //Operating cost of distribution center
float f[rangei][rangej] =...; //Penalty costs of vehicle arriving late to customer
int w=...; //The largest number of new distribution centers available to rent
float r[rangek] =...; //The average selling price of the variety  of fresh goods
int t[rangei][rangej] =...; //Time of vehicle from distribution center  to customer
int T[rangej] = ...; //The serve time that customer  requires from the delivery center
int Tmax[rangej] = ...; //The shortest waiting time that customer  is not dissatisfied
```

```
int Tmin[rangej] = ...; //The longest waiting time that customer  is dissatisfied
float a[rangej][rangek] = ...; //The freshness satisfaction threshold for customer  of
variety  of fresh goods
float b[rangej] =...;  //The time satisfaction threshold for customer .

float A1 = 0.5;
float A2 = 0.5;
float xmax = …;
float xmax13 = xmax*(1/3);
float xmax23 = xmax*(2/3);
int d[rangej] = ...;
float fixmax[rangek] = ...;
float fixmax3[rangek] =...;
float fi2xmax3[rangek] = ...;



//decision variables
dvar float+ d1[rangeh][rangei];
dvar float+ d2[rangei][rangej];
dvar boolean y[rangei];
dvar boolean z1[rangeh][rangei];
dvar boolean z2[rangei][rangej];
dvar boolean z3[rangej][rangek];
dvar float+ s3[rangei][rangej];
dvar float+ pit[rangei][rangej];
//
//dvar int+ freshness[rangej][rangek];

dexpr float x = sum(h in rangeh, i in rangei, j in rangej)(A1*((l1[h][i]/v) + t[i][j]) +
A2*(l1[h][i] + l2[i][j]))*z2[i][j];;
dvar float+ freshness[rangek];
//auxiliary variable
//dvar boolean r[rangei][rangej];

//
dexpr float s1 = sum(i in rangei)(A[i]*y[i]) + sum(i in rangei)(B[i]*y[i]);
dexpr float s2 = sum(h in rangeh, i in rangei)(c1[h][i]*l1[h][i]*d1[h][i]) + sum( i in
rangei, j in rangej)(c2[i][j]*l2[i][j]*d2[i][j]);
dexpr float s4 = sum(k in rangek) (r[k]*(1- freshness[k]));
dexpr float u[j in rangej] = sum(i in rangei, j in rangej)(z2[i][j] * pit[i][j]);
dexpr float t2[j in rangej][k in rangek] = sum(j in rangej, k in
rangek)(z3[j][k]*freshness[k]);
// Objective functions:
minimize 0.6*(s1 + s2 + sum(i in rangei, j in rangej)s3[i][j] + s4) - 0.2* sum(j in
rangej)u[j] - 0.2* sum(j in rangej, k in rangek)t2[j][k];

subject to{
```

```
//s3
forall(i in rangei, j in rangej)
{
t[i][j] >= T[j] && t[i][j] != T[j] => s3[i][j] == f[i][j]*z2[i][j]*(t[i][j] - T[j]);
t[i][j] <= T[j] => s3[i][j] == 0;
}


forall(k in rangek)
  {

    0 <= x && x <= xmax13 => freshness[k] == (3*x*(fixmax3[k] -1 )/xmax) + 1;
    (xmax13+ 0.0001) <= x && x <= xmax23 => freshness[k] == (3*(x -
(1/3)*xmax)*(fi2xmax3[k] - fixmax3[k])/xmax) + fixmax3[k];
    (xmax23 + 0.0001) <= x && x <= xmax => freshness[k] == (3*(x - (2/3)*xmax)*(1 -
fi2xmax3[k])/xmax) + fi2xmax3[k];
  }



//mean[j] - distribution center service time
forall(i in rangei, j in rangej)
  {
    t[i][j]<=Tmin[j]=>pit[i][j]==1;
    t[i][j]>=Tmin[j] && t[i][j]<=Tmax[j] && t[i][j] != Tmin[j] && t[i][j] != Tmax[j] =>
pit[i][j]==(Tmax[j]-t[i][j])/(Tmax[j]-Tmin[j]);
    t[i][j]>= Tmax[j]  => pit[i][j]==0;
}

// CONSTRAINTS:
// constraint 1
forall(h in rangeh){
  sum(i in rangei) d1[h][i] <= U[h];
 }
// constraint 2 & 3
forall(i in rangei){
  sum(h in rangeh) d1[h][i] <= N[i]*y[i];
  sum(j in rangej) d2[i][j] <= N[i]*y[i];
 }
 // constraint 4&5
forall(j in rangej){
sum( i in rangei) y[i] <= w;
sum( i in rangei) z2[i][j] ==1;
}
// constraint 6
forall(j in rangej){
  sum( i in rangei) d2[i][j] >= d[j];
```

```
}
// constraint 7&8
forall( i in rangei, j in rangej, h in rangeh){
  z2[i][j] <= y[i];
  z1[h][i] <= y[i];
}
// constraint 9&10
forall( i in rangei, j in rangej, h in rangeh){
  d1[h][i] >= 0;
  d2[i][j] >= 0;
}
}
```

CPLEX.dat

```
/*********************************************
 * OPL 12.8.0.0 Data
 * Author: USER
 * Creation Date: 25-01-2023 at 22:47:07
 *********************************************/
SheetConnection excelsheet("C:\\Users\\USER\\opl\\Capstone GA\\data.xlsx");
H from SheetRead(excelsheet,"Sheet1!B2");
I from SheetRead(excelsheet,"Sheet1!B3");
J from SheetRead(excelsheet,"Sheet1!B4");
K from SheetRead(excelsheet,"Sheet1!E3");
U from SheetRead(excelsheet,"Sheet1!B32");
N from SheetRead(excelsheet,"Sheet1!D25:D26");
l1 from SheetRead(excelsheet,"Sheet1!B9:C9");
l2 from SheetRead(excelsheet,"Sheet1!B18:C19");
v from SheetRead(excelsheet,"Sheet1!B5");
c1 from SheetRead(excelsheet,"Sheet1!L13:M13");
c2 from SheetRead(excelsheet,"Sheet1!J34:K35");
A from SheetRead(excelsheet,"Sheet1!H25:H26");
B from SheetRead(excelsheet,"Sheet1!F25:F26");
f from SheetRead(excelsheet,"Sheet1!J50:K51");
w from SheetRead(excelsheet,"Sheet1!E2");
r from SheetRead(excelsheet,"Sheet1!N21:N22");
t from SheetRead(excelsheet,"Sheet1!J42:K43");
T from SheetRead(excelsheet,"Sheet1!C50:C51");
Tmax from SheetRead(excelsheet,"Sheet1!B50:B51");
Tmin from SheetRead(excelsheet,"Sheet1!D50:D51");
a from SheetRead(excelsheet,"Sheet1!J3:K4");
b from SheetRead(excelsheet,"Sheet1!N3:N4");
R from SheetRead(excelsheet,"Sheet1!O21:O22");
d from SheetRead(excelsheet,"Sheet1!B40:B41");
fixmax from SheetRead(excelsheet, "Sheet1!R21:R22");
fixmax3 from SheetRead(excelsheet, "Sheet1!Q21:Q22");
fi2xmax3 from SheetRead(excelsheet, "Sheet1!P21:P22");
```

Metaheuristics Genetics Algorithm:

```
%data input
clear
clc
A = [1907290000 1622435000 1758670000 1832980000 1849080000 19856350000 1744080000
1895867000];
B = [235315000 200637000 193206000 220453000 221615000 186426000 180453000 210453000];
C1 = [117900 30300 40200 38100 66300 85800 66000 61500; 91200 45900 111900 63000 39600
75600 55800 49800; 62700 85500 117300 68400 47100 84000 92400 91500];
L1 = [39.3 10.1 13.4 12.7 22.1 28.6 22 20.5; 30.4 15.3 37.3 21 13.2 25.2 18.6 16.6; 20.9
28.5 39.1 22.8 15.7 28 30.8 30.5];
C2 = [68700 64500 73500 67800 72300 73200 66300 78900 88200 61800 66600 69600; 57900
59400 50700 55800 53700 51300 57300 68700 67800 36600 56400 59400; 26400 26100 25500
27000 30000 31500 24000 37500 93600 64800 38700 46800;62700 65100 83100 61200 59400 52800
66300 77100 75000 44100 77400 101100;67500 65700 62700 60900 58500 53400 65700 75900
47400 10500 85200 100500; 21600 26100 24600 27000 29400 33000 21900 10800 106800 75900
30300 23400; 30600 35100 33600 36000 38100 42000 30900 19500 115800 84900 44100 5100;
36000 24600 36600 27300 31200 35100 28200 39300 108300 95400 16800 19800];
L2 = [22.9 21.5 24.5 22.6 24.1 24.4 22.1 26.3 29.4 20.6 22.2 23.2; 19.3 19.8 16.9 18.6
17.9 17.1 19.1 22.9 22.6 12.2 18.8 19.8; 8.8 8.7 8.5 9 10 10.5 8 12.5 31.2 21.6 12.9
15.6; 20.9 21.7 27.7 20.4 19.8 17.6 22.1 25.7 25 15.7 25.8 33.7; 22.5 21.9 20.9 20.3 19.5
17.8 21.9 25.3 15.8 3.5 28.4 33.5; 7.2 8.7 8.2 9 9.8 11 7.3 3.6 35.6 25.3 10.1 7.8; 10.2
11.7 11.2 12 12.7 14 10.3 6.5 38.6 28.3 14.7 1.7; 12 8.2 12.2 9.1 10.4 11.7 9.4 13.1 36.1
31.8 5.6 6.6];
demand = [1358 1267 1035 392 273 5100 5050 1590 1570 4400 2300 1350];
f = [173000 173000 173000 173000 173000 173000 173000 173000 173000 173000 173000
173000;173000 173000 173000 173000 173000 173000 173000 173000 173000 173000 173000
173000;173000 173000 173000 173000 173000 173000 173000 173000 173000 173000 173000
173000;173000 173000 173000 173000 173000 173000 173000 173000 173000 173000 173000
173000;173000 173000 173000 173000 173000 173000 173000 173000 173000 173000 173000
173000;173000 173000 173000 173000 173000 173000 173000 173000 173000 173000 173000
173000;173000 173000 173000 173000 173000 173000 173000 173000 173000 173000 173000
173000];
s = [1 2 3]; %supplier
v = 45;
t = L2/v;
T = [0.42 0.33 0.33 0.42 0.38 0.33 0.25 0.38 0.58 0.42 0.42 0.25];
Tmax = [0.58 0.5 0.5 0.58 0.55 0.5 0.42 0.55 0.75 0.58 0.58 0.42];
Tmin = [0.25 0.17 0.17 0.25 0.22 0.17 0.08 0.22 0.42 0.25 0.25 0.08];
fi = [0.05 0.04 0.08]; %contaminate rate of freshfood
alpha1 = 0.7;
alpha2 = 0.3;
r = [126000 195000 215000]; %price of freshfood

%% initialization
pmax = 12;
population = initialization(pmax);
```

```
for a = 1:pmax
    ar1 = population(a,(1:23));
    [d4,consumer] = output3(ar1);
    [k1,k2,k3] = splitout3(d4);
    [dbn01,dbn02,dbn03] = intonumarr1(k1,k2,k3);
    dcp = [dbn01,dbn02,dbn03];
    % fitnessfunction
    op1 = myf1(A,B,dcp);
    op2 = myf2(C1,C2,s,L1,L2,dcp,demand,consumer);
    op3 = myf3(f,dcp,t,T,consumer);
    [op4,corruptp] = myf4(t,dcp,alpha1,alpha2,L1,L2,v,fi,consumer,s,r);

    totalPop = op1 + op2 + op3 + op4;
    freshnessPop = sum(corruptp);
    DCservicetimePop = myf5(consumer,t,Tmin,Tmax,dcp);
    DCtimePop = sum(DCservicetimePop);

    finalobjPop = 0.6*totalPop + 0.2*freshnessPop + 0.2*DCtimePop;
    cstPop(a,1) = totalPop; %,freshnessPop,DCtimePop,finalobjPop];
end
%% sorting for population
[x5,I1] = sort(cstPop,"descend");
sortpop = population(I1,:);
%% solution for each array
iter = 0;
while iter <= 1000
if(iter >= 0 && iter <= 1000)
%% Main loop
for c = 1:8
    out1 = 0;
    check = 0;
    feasible = 0;
    %crossover DC
        while check == 0
            [r1,r2] = randm();
            check = choosing(r1,r2);
        end
        [parent1,parent2] = choosing2(population,r1,r2);
        [x,y] = splitintodcarray(parent1,parent2);
        [childrendc1,childrendc2] = crs(x,y);
    %muitation DC
        while out1 == 0 || feasible == 0
            ran = randomize(10);
        while ran > 0
            m1 = round(random('Uniform',1,10)-0.5);
            m2 = round(random('Uniform',1,3)-0.5);
            if(m2 == 1)
                if (childrendc1(m1) == 0)
                    childrendc1(m1) = 1;
```

```
                else
                    childrendc1(m1) = 0;
                end
            end
            if(m2 == 2)
                if (childrendc2(m1) == 0)
                    childrendc2(m1) = 1;
                else
                    childrendc2(m1) = 0;
                end
            end
            ran = ran - 1;
        end
        [dc1,dc2,dc3] = splitdc(childrendc1,childrendc2);
        [dcn1, dcn2, dcn3] = dctonum(dc1,dc2,dc3);
        feasible = myfunc4(dcn1,dcn2,dcn3);
        out1 = myfunc3(dcn1,dcn2,dcn3);
        dcnchil = [dc1,dc2,dc3];
        dcn = [dcn1,dcn2,dcn3];

        end

%crossover customer
[p1,p2] = splitcus(parent1,parent2);
ran2 = randomized(15);
v1 = cross(p1,p2,ran2);
%muitation customer
r3 = randperm(14);
v5 = transfering(v1,r3);
%fitnessfunction
o1 = cost1(A,B,dcn);
o2 = cost2(C1,C2,s,L1,L2,dcn,demand,v5);
o3 = cost3(f,dcn,t,T,v5);
[o4,corrupt] = cost4(t,dcn,alpha1,alpha2,L1,L2,v,fi,v5,s,r);

totalcost = o1 + o2 + o3 + o4;

freshness = sum(corrupt);
DCservicetime = sv(v5,t,Tmin,Tmax,dcn);
DCtime = sum(DCservicetime);

seq = array(dcnchil,v5);

finalobj = 0.6*totalcost + 0.2*freshness + 0.2*DCtime;
%building population
chil(c,(1:23)) = seq(1,(1:23));
cst(c,1) = totalcost; %,freshness,DCtime,finalobj];
end
%% sorting children
```

```
[y5,I] = sort(cst,"ascend");
sortchil = chil(I,:);
%% replace and perform new population
j = 1;
while j <= 6
      if(x5(1) > y5(j))
         x5(1) = y5(j);
         sortpop(1,(1:23)) = sortchil(j,(1:23));
      end
      [temp,I2] = sort(x5,"descend");
      sortpop1 = sortpop(I2,:);
      x5 = temp;
      sortpop = sortpop1;
      j = j + 1;
end
population = sortpop;
bestcost = x5(10);
bestsolution = population(10,(1:23));
iter = iter + 1;
disp(x5);
end
end
%% final array of DC - customer
function seq = array(dcn,v5)
    seq = [dcn,v5];
end
%% forming right form sequence from population
function [d4,consumer] = output3(ar1)
    d4 = ar1(1:9);
    consumer = ar1(10:23);
end
function [k1,k2,k3] = splitout3(d4)
        k1((1:3)) = d4((1:3));
        k2((1:3)) = d4((4:6));
        k3((1:3)) = d4((7:9));
end
function [dbn01,dbn02,dbn03] = intonumarr1(k1,k2,k3)
dbn01 = 0;
dbn02 = 0;
dbn03 = 0;
    for i = 0:2
            dbn01 = dbn01 + (2^i)*k1(i+1);
            dbn02 = dbn02 + (2^i)*k2(i+1);
            dbn03 = dbn03 + (2^i)*k3(i+1);
    end
end

%% population
function population = initialization(pmax)
```

```
sas = 0;
feasiblepop = 0;
p = 1;
    while p <= pmax || sas == 0 || feasiblepop == 0
            population(p,(1:23)) = [randi([0 1],1,9),randperm(14,14)];
            for j = 10:23
                if(population(p,j) == 13 || population(p,j) == 14)
                    population(p,j) = 0;
                end
            end
            [d,cus] = splitarr(population,p);
            [db1,db2,db3] = splitout2(d,p);
            [dbn1,dbn2,dbn3] = intonumarr(db1,db2,db3,p);
            feasiblepop = checkingout2(dbn1,dbn2,dbn3);
            sas = checkingout(dbn1,dbn2,dbn3);
            if(feasiblepop == 1 && sas == 1)
                p = p + 1;
            else
                p = p;
            end
    end
end
%% split into cus and dc array
function [d,cus] = splitarr(population,p)
    d(p,(1:9)) = population(p,(1:9));
    cus(p,(1:14)) = population(p,(10:23));
end
%% split DC for population for population
function [db1,db2,db3] = splitout2(d,p)
        db1(p,(1:3)) = d(p,(1:3));
        db2(p,(1:3)) = d(p,(4:6));
        db3(p,(1:3)) = d(p,(7:9));
end
%% changing into dc-number for population
function [dbn1,dbn2,dbn3] = intonumarr(db1,db2,db3,p)
dbn1 = 0;
dbn2 = 0;
dbn3 = 0;
    for i = 0:2
            dbn1 = dbn1 + (2^i)*db1(p,i+1);
            dbn2 = dbn2 + (2^i)*db2(p,i+1);
            dbn3 = dbn3 + (2^i)*db3(p,i+1);
    end
end
%% checking for population
function sas = checkingout(dbn1,dbn2,dbn3)
            if(dbn1 == dbn2)
                sas = 0;
            elseif(dbn2 == dbn3)
```

```
            sas = 0;
        elseif(dbn1 == dbn3)
            sas = 0;
        else
            sas = 1;
        end
end
function feasiblepop = checkingout2(dbn1,dbn2,dbn3)
        if(dbn1 == 0)
            feasiblepop = 0;
        elseif(dbn2 == 0)
            feasiblepop = 0;
        elseif(dbn3 == 0)
            feasiblepop = 0;
        else
            feasiblepop = 1;
        end

end
%% fitness function for population
%% operation cost
function op1 = myf1(A,B,dcp)
k = 0;
k1 = 0;
     for i = 1:3
         k = k + A(1,dcp(i));
         k1 = k1 + B(1,dcp(i));
         op1 = k + k1; %total fixed and operation cost
     end
end
%% transportation cost
function op2 = myf2(C1,C2,s,L1,L2,dcp,demand,consumer)
trans1 = 0;
for h = 1:3
    for i = 1:3
       trans1 = trans1 + C1(s(h),dcp(i))*L1(s(h),dcp(i));
    end
end
trans2 = zeros(3,length(consumer));
pos = 1;
i = 1;
while pos <= length(consumer)
    while consumer(pos) == 0
        if(pos < length(consumer))
            i = i + 1;
            pos = pos + 1;
        else
            break
        end
```

```
    end
    if(pos == length(consumer) && consumer(pos) == 0)
        i = i + 1;
        pos = pos;
        trans2(i,pos) = 0;
        break
    end
    trans2(i,pos) =
C2(dcp(i),consumer(pos))*L2(dcp(i),consumer(pos))*demand(consumer(pos));
    pos = pos + 1;
end
trans2 = sum(sum(trans2));
op2 = trans1 + trans2; %total of transportation cost
end
%% Delayed penalty cost
function op3 = myf3(f,dcp,t,T,consumer)
i = 1;
pos = 1;
p1 = zeros(3, length(consumer));
while pos <= length(consumer)
    while consumer(pos) == 0
        if(pos < length(consumer))
            i = i + 1;
            pos = pos + 1;
        else
            break
        end
    end
    if(pos == length(consumer) && consumer(pos) == 0)
        i = i + 1;
        pos = pos;
        p1(i,pos) = 0;
        break
    end
    if t(dcp(i),consumer(pos)) > T(consumer(pos))
        p1(i,pos) = f(dcp(i),consumer(pos)) * (t(dcp(i),consumer(pos)) -
T(consumer(pos)));
    else
        p1(i,pos) = 0;
    end
     pos = pos + 1;
end
op3 = sum(sum(p1));
end
%% freshness cost
function [op4,corruptp] = myf4(t,dcp,alpha1,alpha2,L1,L2,v,fi,consumer,s,r)
lnew = zeros(3,length(consumer));
pos = 1;
i = 1;
```

```
h = 1;
while pos <= length(consumer)
    while consumer(pos) == 0
        if(pos < length(consumer))
            h = h + 1;
            i = i + 1;
            pos = pos + 1;
        else
            break
        end
    end
    if(pos == length(consumer) && consumer(pos) == 0)
        i = i + 1;
        h = h + 1;
        pos = pos;
        lnew(i,pos) = 0;
        break
    end
        lnew(i,pos) = (alpha1*((L1(s(h),dcp(i))/v) + t(dcp(i),consumer(pos))) +
alpha2*(L1(s(h),dcp(i)) + L2(dcp(i),consumer(pos))));
        pos = pos + 1;
end
lnew = sum(sum(lnew));
op4 = 0;
for k = 1:3
     corruptp(k) = (1 - fi(k)).^lnew; % total corruption cost
     op4 = op4 + r(k)*(1 - corruptp(k));
end
end
%% Distribution center service time
function DCservicetimePop = myf5(consumer,t,Tmin,Tmax,dcp)
i = 1;
pos = 1;
omega = zeros(3,length(consumer));
    while pos <= length(consumer)
        while consumer(pos) == 0
            if(pos < length(consumer))
                i = i + 1;
                pos = pos + 1;
            else
                break
            end
        end
        if(pos == length(consumer) && consumer(pos) == 0)
            i = i + 1;
            pos = pos;
            omega(i,pos) = 0;
            break
        end
```

```
        if(t(dcp(i),consumer(pos)) <= Tmin(consumer(pos)))
            omega(i,pos) = 1;
        elseif (t(dcp(i),consumer(pos)) > Tmin(consumer(pos)) && t(dcp(i),consumer(pos))
< Tmax(consumer(pos)))
            omega(i,pos) = ((Tmax(consumer(pos)) -
t(dcp(i),consumer(pos)))/(Tmax(consumer(pos)) - Tmin(consumer(pos))));
        else
            omega(i,pos) = 0;
        end
        pos = pos + 1;
    end
    DCservicetimePop = sum(omega);
end



%% random row
function [r1,r2] = randm()
    r1 = round(random("Uniform",1,10)-0.5);
    r2 = round(random("Uniform",1,10)-0.5);
end

%% choosing
function check = choosing(r1,r2)
    if(r1 == r2)
        check = 0;
    else
        check = 1;
    end
end
function [parent1,parent2] = choosing2(population,r1,r2)
    parent1 = population(r1,(1:23));
    parent2 = population(r2,(1:23));
end
%% split Dc from array parrent
function [x,y] = splitintodcarray(parent1,parent2)
    for i = 1:9
        x(i) = parent1(i);
        y(i) = parent2(i);
    end
end
%% crossover for DC
function [childrendc1,childrendc2] = crs(x,y)
    for i = 1:5
        childrendc1(i) = x(i);
        childrendc1(9 + 1-i) = y(9 + 1-i);
        childrendc2(i) = y(i);
        childrendc2(9 + 1-i) = x(9 + 1-i);
```

```
    end
end
%Muitation for DC
%randomize
function ran = randomize(ub)
    ran = round(random('Uniform',0,ub)-0.5);
end
%% split DC
function [dc1,dc2,dc3] = splitdc(childrendc1,childrendc2)
ran3 = round(random('Uniform',1,3)-0.5);
if (ran3 == 1)
    for i = 1:3
      dc1(i) = childrendc1(i);
      dc2(i) = childrendc1(i + 3);
      dc3(i) = childrendc1(i + 6);
    end
end
if(ran3 == 2)
    for i = 1:3
      dc1(i) = childrendc2(i);
      dc2(i) = childrendc2(i + 3);
      dc3(i) = childrendc2(i + 6);
    end
end
end
%% changing dc to num dc
function [dcn1, dcn2, dcn3] = dctonum(dc1,dc2,dc3)
dcn1 = 0;
dcn2 = 0;
dcn3 = 0;
    for i = 0:2
     dcn1 = dcn1 + (2^i)*dc1(i+1);
     dcn2 = dcn2 + (2^i)*dc2(i+1);
     dcn3 = dcn3 + (2^i)*dc3(i+1);
    end
end

function out1 = myfunc3(dcn1,dcn2,dcn3)
             if(dcn1 == dcn2)
                out1 =0;
            elseif (dcn2 == dcn3)
                out1 = 0;
            elseif (dcn1 == dcn3)
                out1 = 0;
            else
                out1 = 1;
             end
end
```

```
function feasible = myfunc4(dcn1,dcn2,dcn3)
        if(dcn1 == 0)
            feasible = 0;
        elseif (dcn2 == 0)
            feasible = 0;
        elseif (dcn3 == 0)
            feasible = 0;
        else
            feasible = 1;
        end
end

%% split customer array
function [p1,p2] = splitcus(parent1,parent2)
    for i = 1:14
      p1(i) = parent1(i + 9);
      p2(i) = parent2(i + 9);
    end
end
%% randomize position
function ran2 = randomized(ub)
         ran2 = round(random('Uniform',1,ub)-0.5);
end
%% crossover
function v1 = cross(p1,p2,ran2)
         r4 = round(random('Uniform',1,3)-0.5);
            if (r4 == 1)
                v1 = cat(2,p1(ran2 :end),p1(1:ran2 -1));
            elseif (r4 == 2)
                v1 = cat(2,p2(ran2 :end),p2(1:ran2 -1));
            end
end
%% muitation
function v5 = transfering(v1,r3)
    for i = 1:14
        v5(i) = v1(r3(i));
    end
end
%% fixed cost and operation cost
function o1 = cost1(A,B,dcn)
k = 0 ;
k1 = 0;
     for i = 1:3
         k = k + A(1,dcn(i));
         k1 = k1 + B(1,dcn(i));
         o1 = k + k1; %total fixed and operation cost
     end
end
```

```
%% transportation cost
function o2 = cost2(C1,C2,s,L1,L2,dcn,demand,v5)
trans1 = 0;
for h = 1:3
    for i = 1:3
       trans1 = trans1 + C1(s(h),dcn(i))*L1(s(h),dcn(i));
    end
end
trans2 = zeros(3,length(v5));
pos = 1;
i = 1;
while pos <= length(v5)
    while v5(pos) == 0
        if(pos < length(v5))
            i = i + 1;
            pos = pos + 1;
        else
            break
        end
    end
    if(pos == length(v5) && v5(pos) == 0)
        i = i + 1;
        pos = pos;
        trans2(i,pos) = 0;
        break
    end
    trans2(i,pos) = C2(dcn(i),v5(pos))*L2(dcn(i),v5(pos))*demand(v5(pos));
    pos = pos + 1;
end
trans2 = sum(sum(trans2));
o2 = trans1 + trans2; %total of transportation cost
end
%% Delayed penalty cost
function o3 = cost3(f,dcn,t,T,v5)
i = 1;
pos = 1;
p1 = zeros(3, length(v5));
while pos <= length(v5)
    while v5(pos) == 0
        if(pos < length(v5))
            i = i + 1;
            pos = pos + 1;
        else
            break
        end
    end
    if(pos == length(v5) && v5(pos) == 0)
        i = i + 1;
        pos = pos;
```

```
        p1(i,pos) = 0;
        break
    end
    if t(dcn(i),v5(pos)) > T(v5(pos))
        p1(i,pos) = f(dcn(i),v5(pos)) * (t(dcn(i),v5(pos)) - T(v5(pos)));
    else
        p1(i,pos) = 0;
    end
     pos = pos + 1;
end
o3 = sum(sum(p1));
end

%% Freshgood corruption cost
function [o4,corrupt] = cost4(t,dcn,alpha1,alpha2,L1,L2,v,fi,v5,s,r)
lnew = zeros(3,length(v5));
pos = 1;
i = 1;
h = 1;
while pos <= length(v5)
    while v5(pos) == 0
        if(pos < length(v5))
            h = h + 1;
            i = i + 1;
            pos = pos + 1;
        else
            break
        end
    end
    if(pos == length(v5) && v5(pos) == 0)
        i = i + 1;
        h = h + 1;
        pos = pos;
        lnew(i,pos) = 0;
        break
    end
        lnew(i,pos) = (alpha1*((L1(s(h),dcn(i))/v) + t(dcn(i),v5(pos))) +
alpha2*(L1(s(h),dcn(i)) + L2(dcn(i),v5(pos))));
        pos = pos + 1;
end
lnew = sum(sum(lnew));
o4 = 0;
for k = 1:3
     corrupt(k) = (1 - fi(k)).^lnew; % total corruption cost
     o4 = o4 + r(k)*(1 - corrupt(k));
end
end
%% distribution center service time
function DCservicetime = sv(v5,t,Tmin,Tmax,dcn)
```

```
i = 1;
pos = 1;
omega = zeros(3,length(v5));
    while pos <= length(v5)
        while v5(pos) == 0
            if(pos < length(v5))
                i = i + 1;
                pos = pos + 1;
            else
                break
            end
        end
        if(pos == length(v5) && v5(pos) == 0)
            i = i + 1;
            pos = pos;
            omega(i,pos) = 0;
            break
        end
        if(t(dcn(i),v5(pos)) <= Tmin(v5(pos)))
            omega(i,pos) = 1;
        elseif (t(dcn(i),v5(pos)) > Tmin(v5(pos)) && t(dcn(i),v5(pos)) < Tmax(v5(pos)))
            omega(i,pos) = ((Tmax(v5(pos)) - t(dcn(i),v5(pos)))/(Tmax(v5(pos)) -
Tmin(v5(pos))));
        else
            omega(i,pos) = 0;
        end
        pos = pos + 1;
    end
    DCservicetime = sum(omega);
end
```