Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN INGENIERÍA

EN SISTEMAS DE TELECOMUNICACIÓN

Trabajo de Fin de Grado

SOFT COMPUTING TECHNIQUES APPLIED TO THE MILITARY DOMAIN

Autor: Ana María Saiz García
Tutor: José Antonio Portilla Figueras

2022

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

GRADO EN INGENIERÍA

EN SISTEMAS DE TELECOMUNICACIÓN

Trabajo Fin de Grado

SOFT COMPUTING TECHNIQUES APPLIED TO THE MILITARY DOMAIN

Autor: Ana María Saiz García

Tutor/es: José Antonio Portilla Figueras

TRIBUNAL:

Presidente: Silvia Jiménez Fernández

Vocal 1º: Roberto Gil Pita

Vocal 2º: José Antonio Portilla Figueras

FECHA: << Fecha de depósito >>

SOFT COMPUTING TECHNIQUES APPLIED TO THE MILITARY DOMAIN

Ana María Saiz García

ACKNOWLEDGEMENTS

Gracias a mi familia, por haber estado a mi lado en todo momento y haberme apoyado siempre. Sin vosotros esto no hubiese sido posible. A mis abuelos, a mis tíos, a mis primos, a mis padres y a mis hermanos, gracias.

A las bonitas amistades que he hecho a lo largo de la carrera, gracias por haber estado siempre ahí. Hemos compartido muchos momentos de sufrimiento, pero más importante aún, muchos momentos inolvidables. Gracias.

A mi tutor Portilla, gracias por tu ayuda y paciencia.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT IN SPANISH

El objetivo de este trabajo es proporcionar un algoritmo que optimice el posicionamiento de depósitos de suministros para uso militar dentro de un área geográfica determinada. La logística militar debe lidiar continuamente, en sus modelos, con una demanda y unos suministros de características cambiantes. A medida que los intereses políticos varían y las amenazas se modifican en el contexto mundial, las decisiones sobre cómo ajustar o ubicar las instalaciones en cada periodo de tiempo se vuelven más relevantes.

Este trabajo se propone en el marco de la colaboración que la Universidad de Alcalá de Henares mantiene con el Observatorio ISDEFE en TIC (Tecnologías de la Información y las Comunicaciones) y Avance Digital, concretamente en la aplicación de mecanismos de algoritmos bioinspirados a la logística militar.

Palabras clave: Algoritmo, Depósitos de Suministro, Aplicaciones Militares, Algoritmos Bioinspirados, Logística Militar.

ABSTRACT

The purpose of this study is to provide an optimal algorithm for the correct location of supply depots for military use inside a geographical area. Military logistics must continually deal with changing request and supply characteristics inside their models. As political interest and threats alter within the worldwide environment, the elements of how to adjust or locate facilities over different time periods become of interest.

This work is proposed in the framework of the collaboration that the University of Alcalá de Henares maintains with ISDEFE Observatory ICT (Information and Communications Technology) and Digital Advancement, specifically in the application of bioinspired algorithm mechanisms to military logistics.

Keywords: Algorithm, Supply Depots, Military Applications, Bioinspired Algorithm, Military Logistics.

EXTENDED ABSTRACT

Soft computing is the use of inexact calculations to provide usable but not precise arrangements to complex computational issues. The approach enables solutions for problems which are either unsolvable or too time-consuming to solve with current hardware. Soft computing gives an approach to problem-solving using means other than raw power from computers. With the human intellect as a role model, soft computing is tolerant to fractional truths, vulnerability, imprecision, and guess, not at all like conventional computing models. The resilience of soft computing allows analysts to approach a large set of issues that conventional computing cannot handle.

In recent decades, location science has pulled in numerous analysts and specialists from distinctive disciplines. Area science incorporates a few sorts of location issues, solution methods as well as significant sum of theoretical modeling systems and arrangement procedures. Its association and interaction with other disciplines such as mathematics, geology, logistics, and financial matters are the main driving force behind its advancement. Facility location problems have pulled in much consideration among analysts and specialists from diverse disciplines. Among these problems, location models observed in military organizations have a noteworthy effect on the execution of many military activities since they require huge sums of financing, resources, and men power. In addition, an effective arranging of military assets frequently leads to a great direction to triumphs. For this reason, its useful to look at this limited literature on military facility location issues and report the most common problem classes and solution approaches executed in this domain. The challenge of accurately situating military units and assets inside a geographical area has vexed commanders and their staffs for many years.

GLOSSARY OF ACRONYMS AND ABBREVIATIONS

GA	Genetic Algorithm
MCLP	Maximal Covering Location Problem
PD	Potential Depot
DS	Demand Site
PM	Population Matrix
TC	Transportation Cost

Chapter 1

INTRODUCTION

1.1. Motivation

Over time, military commanders' decision-making needs have had a solid impact upon the advancements in the field of operations investigation and analytical problem solving. The challenge of accurately situating military units and assets inside a geographical zone has vexed commanders and their staff for a long time. The advancements in the field of military logistics have benefited along the way to solving these narrowly defined military problems, as generalized strategies have been extended beyond their origins to countless non-military applications.

During World War II, for example, military analysts used mathematics to solve complex problems such as ship convoy routes, code breaking, and material allocation, which led to the development of linear and mathematical programming techniques by wartime scientists very useful in many civilian domains [1]. The War also benefited location analysis knowledge, as commanders needed to be able to spatially position munitions to annihilate a target and cover a search area to find and track the enemy. However, the benefit is reciprocal, as the research in the field of mathematics and computing and the improved ability to solve larger and more complex problems has literally redefined military strategy and planning since World War II.

The main objective of this study is to, through engineering investigating methods, provide an algorithm able to optimize the positioning of supply military depots inside a geographical area to allow the most efficient distribution to the desired destinations.

1.2. Goals

To achieve the objective stated above, the following steps are to be followed:

Step 1 (S1) Study and understanding of three papers based on different location algorithms. Selection of the most appropriate one from the set of algorithms previously studied.

Step 2 (S2) Challenge the selected algorithm by programming and implementing it in a specific environment designed to test it in deep.

Step 3 (S3) Gather all the information obtained through the previous investigation process, relevant data to our purpose, and conduct the subsequent analysis.

The investigation process has been initialized by the analysis of the three documents related to algorithms for solving location and allocation problems. To fulfill **S1**, the correct understanding of the documents will help us to choose the most appropriate algorithm to be implemented. In the execution of **S2,** the selected algorithm is programmed and tested in a specific environment with the goal of reaching the most precise solution possible to achieve our initial purpose.

1.3. Structure of the Memory

The layout of this project is the following:

1. Chapter 2. Study and selection of the most appropriate Location Algorithm.
    - Study of Paper 1
    - Study of Paper 2
    - Study of Paper 3
    - Selection of Location Algorithm
2. Chapter 3. Implementation of the selected Algorithm (Genetic Algorithm).
    - Description
    - Population
    - Cost Function
    - Crossover and Mutation
3. Chapter 4. Experimental Analysis in four different “restriction policies”
    - Experiment with Policy 1. No restrictions applied.
    - Experiment with Policy 2. Radius restriction.

- Experiment with Policy 3. Capacity restriction.
- Experiment with Policy 4. Radius and capacity restriction.

4. Chapter 5. Conclusions.
    - Conclusions of the project.
    - Future Work.

Chapter 2

STUDY AND SELECTION OF THE LOCATION ALGORITHM

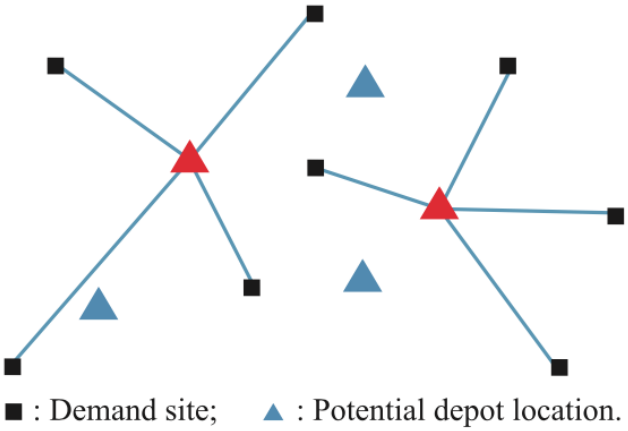
This chapter will include a brief synopsis of the process of study of each one of the papers evaluated, in order to choose the algorithm to be used to complete this investigation project.

2.1. Analysis of article nº 1

This first paper [2], "Location and allocation problem for spare parts depots on integrated logistics support", addresses a depot location model for spare parts. The main objective is to find an optimal position for the centralized depot that allows it to provide supply to the rest of the depots that require service, seeking the lowest possible cost. In this paper, these sites which have spare parts demands are referred to as Demand Sites (DS). Each DS can only be served by one depot that must be located within a certain radius to guarantee service times kept to a minimum, and that the efficiency is maximized.

The purpose of this first paper is to ensure that the expected Transportation Cost (TC) for support system spare parts is kept to a minimum throughout long-term operation. As a result, the authors come to the conclusions that each site's demand for spare parts is unique and that the Potential Depots (PD) have a limited capacity. Maintenance, inventory, and construction costs are disregarded.

Figure 2.1-1 Demands distribution schematic diagram.



This study uses the "Maximum Covering Location Problem" (MCLP) which focuses on maximizing resource utilization in the specified scope, as well as the Genetic Algorithm (GA) to find the optimal place with the lowest TC.

The conclusions that are reached by using this method is that it improves system availability and cost is reduced, a reasonable maximum supply is obtained, and the GA is used to solve the model. A suitable location can help save money and improve the level of service.

2.2. Analysis of article nº 2

This second paper [3], "Integrating simplified swarm optimization with AHP for solving capacitated military logistic depot location problem", tackles a two-level facility location issue in a military logistic system. The goal is to maximize the average utility of requisitioned structures, and a building's utility is determined by its features. This paper proposes an integer programming model and a two-stage solution to the problem.

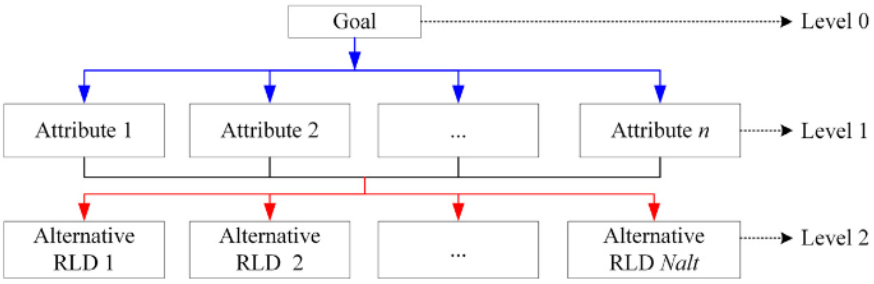
This study makes use of Analytic Hierarchy Process (AHP), a method for resolving problems involving many criteria for decision-making; and Simplified Swarm Optimization (SSO Optimization), a population-based approach to compensate the Particle Swarm Optimization algorithm's shortcomings in tackling discrete problems (it simulates particle behavior in nature).

The key issue is determining where the installation should be placed, being the optimal location chosen from a list of potential sites by the use of the distribution method. This is done by

considering specified attributes of a potential building that are selected with greater or lesser weight. These characteristics include capacity, state of the road connecting the building to the location where they must arrive, a strong structure, the size of the operation area, and the distance.

The first step is to create comparison matrices, in which the attributes that have been determined to have the largest weight are entered as input and the relative weights of these qualities are calculated using the AHP. The SSO is then utilized in the second stage to find the best location for the deposits. In this second stage, the goal is to identify global maximums and minimums; the operation is inspired by each individual's movement, which is the result of merging individual decisions from a variety of actions.

Figure 2.2-1 Decomposition of the problem into a hierarchy process.



2.3. Analysis of article nº 3

This third research, "Emergency material scheduling optimization model and algorithms: A review", compares different algorithms with the final goal of improving EMS (Emergency Material Scheduling). Its key objective is to cut down on time, distance and expenses while also providing as mush assistance as possible [4]. The amount of supply depots, relief vehicles, product varieties, budget, and demand are some of the constraints to consider.

More efficient algorithms are required to identify the optimal solution, such as Genetic Algorithm (GA), Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO), this last one is comparable to SSO Algorithm mentioned before **¡Error! No se encuentra el origen de la referencia.**.

Ant Colony Optimization (ACO) is a probabilistic algorithm for determining the best course of action, positive information feedback and heuristics search are two features of this method. This algorithm has the advantages of having a large global search capacity, being resilient, and being easily integrated with other algorithms. On the other hand, the shortcomings include a lengthier search time as compared to competing algorithms; the path's benefit will become evident but after a long time. Due to this, the method has a high likelihood of stalling.

The (PSO) Particle Swarm Optimization is similar to the SSO (Simplified Swarm Optimization) Algorithm mentioned before. This one has the benefit of a quick convergence time, usually producing a high-quality solution, and optimization robustness. Furthermore, PSO makes few or no assumptions about the problem to be solved, therefore being able to find a lot of different answers not necessarily differentiable to solve the optimization problem, as it is the case with other optimization methods. This method does not ensure that the best solution will be reached.

2.4. Selection of the Location Algorithm to be implemented

The main conclusions of our analysis of the three documents can be schematized as follows:

Article number one makes use of the GA. This evolutionary algorithm is the basis of many other algorithms, and it has a large number of advantages comparing to other methods. The decision was made, not only based on the study of the algorithms utilized by each article, but also on the basis of the overall. One of the reasons was the MCLP. The goal of the MCLP is to locate a number of facilities on a network so that the population covered is maximized [6]. There is a pre-defined radius and if the distance between a facility and a demand node is less than this radius, the facility will be able to cover this demand node.

Article number two looks for an optimal site to locate a facility. It does so by defining some attributes and assigning them greater or lesser importance, helping to decide between one location or another working with a comparison matrix. The main flaw of this approach is that the outcome could vary greatly depending on the demands of each individual, as well as on what each one perceives to be the most significant attributes.

Emergency Material Scheduling optimization model mentioned in article number three improved the accuracy and efficiency, but the limitation between model complexity and search speed did not work effectively. Instead, it compares and combines alternative approaches.

As a result of all of this, the selected paper to be used as the basis of this Degree Project is number one: “Location and allocation problem for spare part depots on integrated logistics support”.

Chapter 3

IMPLEMENTATION OF THE GENETIC ALGORITHM

This chapter covers the implementation of the selected algorithm trying to reach the best solution for our work's main objective: Identifying the lowest cost among the starting population subjected to different iterations.

Following the selection of the paper number two, as described in the previous chapter, we proceed to develop the GA trying to optimize our objective. The implementation of the algorithm is carried out in MATLAB, a mathematical package with its own programming language, and to attain the optimal result, the technique outlined in this chapter will be repeated multiple times.

3.1. Description

GA is a search and optimization technique based on Darwin's natural selection and genetic reproduction principles [7]. The principles of nature that inspires the AG's are quite basic. The selection principle, according to C. Darwin's hypothesis, favors the fittest individuals, those with a more noteworthy longevity and, subsequently, those with the highest likelihood of continuing the species. Individuals with a larger number of descendants have a greater chance of passing on their stronger genetic codes to future generations. These genetic codes, which are represented on chromosomes, make up the individual's identity.

As an optimization method, GAs have proved to be a powerful and effective tool for solving complicated problems illustrating the validity of evolutionary principles. The capacity of GAs

to optimize results in complex situations is further increased when the problem can be evaluated using combination approaches, since it is efficient in terms of reaching an approximate global maximum.

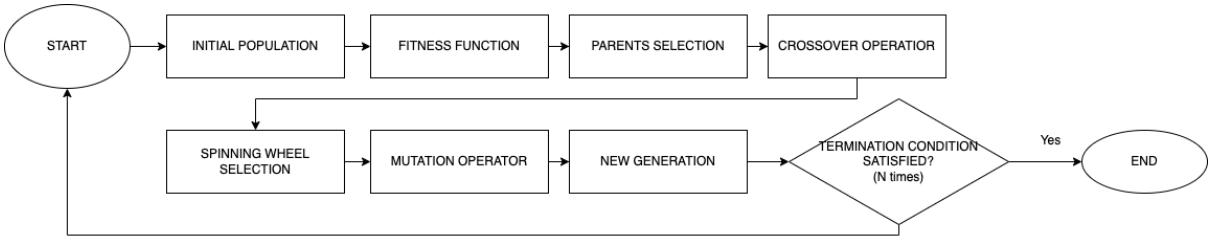
From a computer standpoint, a GA is a programming technique that replicates biological evolution as a problem-solving method. The input is a group of potential solutions to a problem, defined in a specific manner, and with a cost (fitness) that allows each candidate to be quantitatively evaluated, given a specific problem to be addressed. Candidates who are most adapted to the environment are protected by the algorithm and given the opportunity to reproduce.

Furthermore, during the copying process, random alterations are introduced. The elements that have not improved are removed. The expectation is that the population's average fitness will improve with each round, and therefore by repeating this process many times, better solutions of the problem will emerge.

GA's have been used to solve extremely complicated engineering issues in a range of sectors. Planning, process control, pattern recognition, image processing, robotics, power electronics, chemistry (protein classification), and signal processing are only a few of the applications [8].

The initial purpose of this project was to determine the proper location of military supply depots within a geographical area. The first step in achieving the main goal has been to generate the population. This will be a group of potential solutions to our situation. Each potential solution will have its cost (fitness), this will lead us to the second step, determining the cost function using the specified paper as a basis. Finally, to achieve the optimum outcome, we will proceed to build generations and possible changes in specific individuals. To all of this we are going to impose some restrictions to the process. These restrictions will consist on distance limitations, setting a radius that will reduce the range of service from one PD, and capacity limitations meaning that each PD will have a maximum number of resources to deliver to each DS asking for a certain amount of supply. Figure 3.1-1shows the flow diagram that we are going to follow.

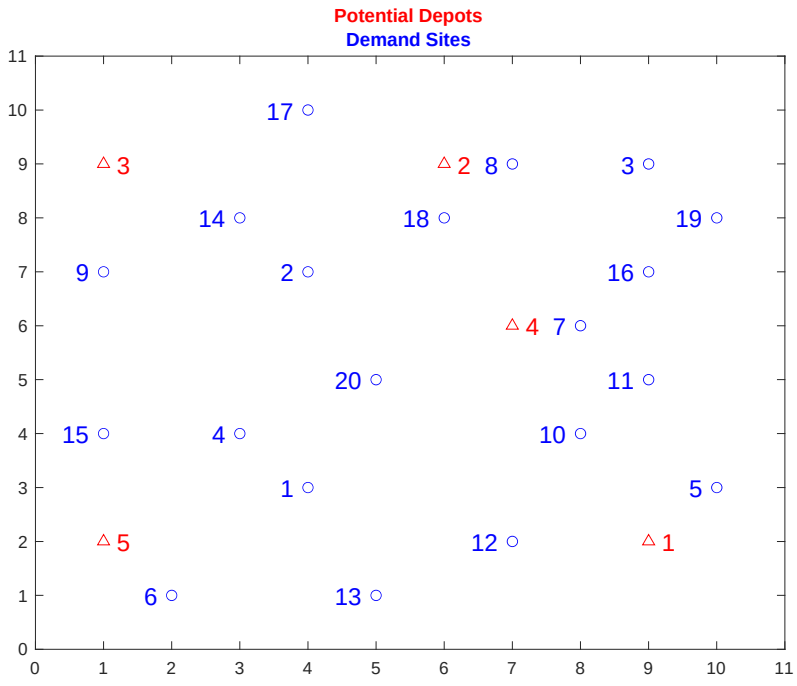
Figure 3.1-1 Flow Diagram



Our final situation will consist of 100 DS and 10 PD. At the beginning of the MATLAB code that we are going to program, we must place the sites on the coordinates we want each DS or PD to be located. To accomplish so, a "txt" file containing information on each site's x and y coordinates, as well as information about the depot's capacity (which will be detailed later), is created.

Before tackling this situation in the full range, we started the process with a smaller number of depots to ensure that calculations and results we obtain are accurate. We extrapolated everything to our final situation after confirming it with the lower amount. The first small situation is shown in Figure 3.1-2, where we have 20 DS and 5 PD.

Figure 3.1-2 Initial Small Situation



3.2. Population

The first phase is to generate the initial population, which represents the random set of individuals that appear at the beginning of the execution of the algorithm. Our individuals will be strings with the same length as the number of DS we have. Each position corresponds to that DS and will be randomly assigned to a PD. The population will consist of a number M of individuals. As a result, we seek a *MxN* matrix. M rows, which will be 50 individuals in our scenario. And N columns, which reflect the assignment of each DS to its PD.

A function has been programmed in MATLAB that generates the initial population matrix (PM) *MxN*. This initial PM will evolve following the laws of the GA. In Table 3.2-1 we have the first individual of this scenario. DS 1 has been assigned to PD 3, DS 2 has been assigned to PD 4, and so on with each of the DS. This table represents the first row of our PM.

Table 3.2-1 Assignment PD to DS of Small Scenario

1	2	3	4	5	6	7	8	9	10
3	4	5	4	4	3	5	4	3	3
11	12	13	14	15	16	17	18	19	20
3	4	2	2	1	5	3	1	3	5

This same procedure is done to our final scenario where we have 100 DS, random allocation of each PD to each DS. In this same section, the distance between each DS and its assigned PD is calculated.

3.3. Cost Function

The next step was to program a function to determine the TC. We did this based on the selected paper, number 1. Here, the depots to satisfy the DSs and the amount that each DS requires the distance between our potential and DSs, the amounts allocated the factors we had to consider. With all this, we can calculate the expected TC. To carry out this task we will use the data collected in the "txt" file.

3.3.1. Distance

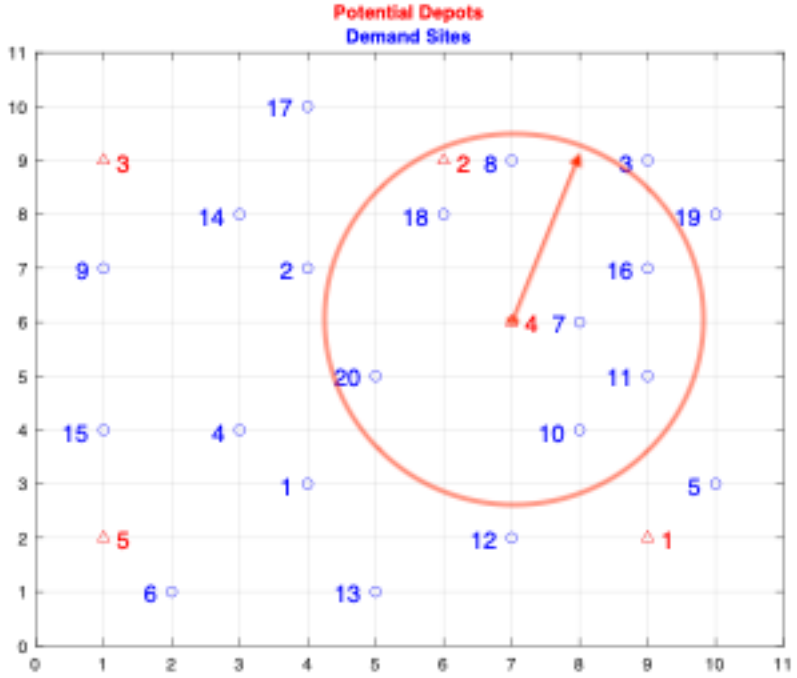
The first stage in determining the costs function is to know the distance between each PD and the DS. To determine the distance between two different points, the squares of the differences between their coordinates were calculated and then the root of the sum of said squares was found. The implementation of this is expressed below:

$$d_{ij} = \sqrt{\left(x_{demand} - x_{potential}\right)^2 + \left(y_{demand} - y_{potential}\right)^2}$$

The distances are saved in a matrix with as many rows as there are PD. The distance between the PD and each of the DS is recorded in each row. The notation for this variable will be $\mathbf{d_{ij}}$.

One of the restrictions indicated at the beginning of this chapter is addressed in this section, is the radius restriction, where there will be a maximum radius that will limit which DS each PD can service and which cannot. We can see an example in Figure 3.3.1-1 where the service scope of site 4 is sketched. Only when a DS is in the set radius, it can be supplied. These are numbers 7, 8, 10, 11, 16, 18 and 20, while all the other sites won't be able to receive service from PD 4.

Figure 3.3.1-1 Radius Restriction PD 4



3.3.2. Capacity

After knowing the distance between the sites, the other variable that we need for determining the cost function is the capacity. Our PD are limited in capacity, so when the capacity that the DS requires is higher than the maximum capacity that the assigned PD has, this will not be able to serve the site.

The function programmed in MATLAB for this section consisted of comparing the values of the amount of capacity that the DS require with the amount that the PD can offer. We will store the values of the capacity in a matrix that will have the notation of $\mathbf{a_{ij}}$. In addition, a matrix of ones and zeros is produced to indicate whether the service sought from that site can be provided or not. This matrix will be named after $\mathbf{y_{ij}}$.

3.3.3. Cost Function Calculation

Once we have fixed the variables that we need to calculate the cost, we will need to develop the function that returns an associated cost for each vector in the population we have created. This function is based on paper one.

With the intention of arriving to the cost function of each individual, we have to determine the total TC, which is obtained by multiplying the transport distance by the capacity required [2]. The total TC will then be expressed as:

$$C_{ij} = \sum_{i \in I} \sum_{j \in J} a_i d_{ij} y_{ij}$$

This will give us the cost of each individual, and we will have to repeat the process for the whole population. Continuing with our example of Figure 3.1-2, we can show the procedure of the calculation of the cost as follows:

First, we determined the distances from the DS to the assigned PD as said in section 3.3.1. , Table 3.3.3-1 shows these values for the first individual of the population. Also, Table 3.3.3-2 shows the information about the capacity.

Table 3.3.3-1 Distance of first Chromosome

1	2	3	4	5	6	7	8	9	10
6,7082	3,1623	10,6301	4,4721	4,2426	8,0623	8,0623	3	2	8,6023
11	12	13	14	15	16	17	18	19	20
8,9443	4	8,0623	3,1623	8,2462	9,4340	3,1623	6,7082	9,0554	5

Table 3.3.3-2 Capacity of the first Chromosome

1	2	3	4	5	6	7	8	9	10
10	23	5	6	11	23	9	22	9	8
11	12	13	14	15	16	17	18	19	20
8	10	15	14	18	46	34	42	15	37

As shown on the cost formula stated above, to achieve the cost value we multiply the distance between PD and DS by the capacity requested from each DS. This will give us the cost of Table 3.3.3-3 as shown below.

Table 3.3.3-3 Cost of the first Chromosome

1	2	3	4	5
67,082	72,7324	53,1507	26,8326	46,669
6	7	8	9	10
185,4319	72,5603	66	18	68,8186
11	12	13	14	15
71,5542	40	120,9339	44,2719	148,4318
16	17	18	19	20
433,9631	107,5174	281,7446	135,8303	185

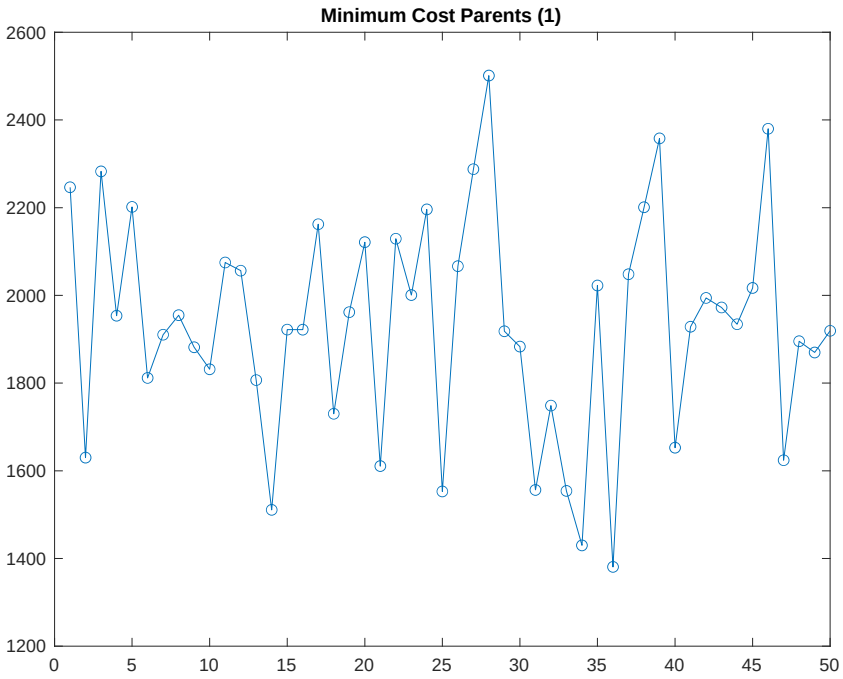
We can see the cost of each DS to the assigned PD (of each gene). The sum of all these costs will lead us to the total cost of this chromosome. This procedure is done to all the PM, arriving to a total cost for each individual of the population. Table 3.3.3-4 and Figure 3.3.3-2 shows this example.

Table 3.3.3-4 Cost and assignment of the whole Population

M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	COST
1	3	4	5	4	4	3	5	4	3	3	3	4	2	2	1	5	3	1	3	5	2246,5
2	3	5	3	3	5	1	4	2	1	2	3	2	2	3	5	2	2	2	3	1	1630
3	1	4	3	2	4	2	5	1	5	4	2	5	2	5	5	3	1	3	1	1	2282,7
4	2	3	2	4	3	4	2	5	2	2	2	5	2	4	1	2	4	2	5	1	1953,5
5	3	5	2	2	4	3	2	1	2	5	2	3	3	5	5	2	5	5	1	2	2201,7
6	1	2	3	4	4	1	5	1	2	3	5	5	1	1	3	1	3	4	4	1	1811,7
7	2	5	3	1	4	4	3	5	2	2	3	2	5	2	4	1	2	4	1	3	1910,2
8	5	4	4	5	2	3	4	4	4	5	5	5	2	1	2	5	2	4	4	3	1954,7
9	3	2	2	4	2	1	4	3	5	1	2	2	1	3	3	3	1	4	1	4	1881,6
10	4	5	1	1	4	1	5	5	4	2	3	5	1	3	5	1	2	3	3	4	1831,5
11	2	5	1	1	3	1	3	1	3	1	2	3	4	2	5	4	5	5	1	2	2074,9
12	5	4	2	1	2	5	1	2	1	3	4	2	5	1	5	3	5	5	2	3	2056,2
13	5	3	2	5	5	5	1	2	3	4	3	2	1	1	2	3	3	1	2	2	1806,8
14	1	4	2	3	2	5	3	4	3	4	1	1	5	2	5	1	5	3	4	4	1510,9
15	4	3	1	5	5	3	4	2	2	3	5	2	2	3	5	3	2	1	2	5	1921,9
16	4	4	5	1	3	4	2	3	3	1	1	5	2	3	1	5	2	4	5	4	1921,8
17	3	1	5	5	4	3	3	1	4	5	3	3	1	4	1	3	3	4	2	3	2162,2
18	3	5	1	5	4	5	2	3	2	5	3	1	4	1	1	2	4	3	4	4	1729,9
19	4	4	2	5	3	5	5	4	4	3	1	4	3	2	4	2	1	1	1	5	1961,6
20	3	4	2	3	3	2	1	5	4	3	3	2	2	3	4	2	1	3	4	4	2121,3
21	5	3	1	5	3	3	2	4	1	1	4	5	4	3	5	4	1	2	2	1	1610,6

22	4	4	5	4	2	1	3	1	4	1	2	1	2	5	3	2	5	5	2	1	2129,1
23	2	4	3	3	5	3	5	5	1	5	5	4	4	3	5	4	1	4	5	2	2000,6
24	4	1	4	2	1	4	1	5	5	1	5	4	4	5	3	2	1	1	3	5	2196
25	3	5	1	3	2	5	5	2	4	3	4	5	2	2	5	4	3	5	4	4	1552,8
26	2	3	3	2	3	5	3	3	3	2	4	4	3	1	1	2	3	5	5	3	2066,5
27	2	4	2	3	4	4	4	1	5	4	3	2	5	5	2	5	1	1	4	2	2287,7
28	2	5	4	4	4	4	5	3	1	3	5	1	3	1	5	5	1	3	3	3	2501,1
29	5	4	1	1	3	1	1	2	2	5	3	1	1	4	4	4	5	1	3	2	1918,1
30	2	1	3	2	2	4	2	5	5	5	5	3	1	5	1	2	2	4	4	2	1883,1
31	3	2	1	1	4	4	3	3	2	4	2	4	5	4	3	4	3	4	1	5	1556,3
32	3	2	1	1	2	5	5	5	4	2	1	2	1	5	2	3	2	4	4	4	1748,6
33	4	4	3	5	4	1	4	4	2	4	1	5	5	3	5	4	1	4	1	3	1554
34	5	5	2	3	1	4	2	2	3	2	3	4	5	2	4	2	3	2	1	1	1429,9
35	4	1	5	2	1	3	2	4	3	3	2	3	3	5	1	1	1	2	2	1	2022,4
36	5	4	1	5	5	5	4	2	3	4	2	3	3	4	4	2	2	4	2	5	1380,8
37	1	2	2	5	5	5	3	5	4	5	3	5	1	3	1	3	5	4	2	1	2048,2
38	3	5	2	2	1	5	1	2	4	3	5	4	1	3	4	5	1	5	3	1	2200,9
39	5	3	1	2	2	3	1	1	5	5	5	3	4	3	2	5	4	5	1	1	2357,9
40	1	1	3	1	1	4	3	4	2	3	2	1	3	3	2	4	4	2	4	3	1652,6
41	2	3	4	2	4	3	2	4	2	3	3	5	5	2	1	2	1	4	5	2	1928,5
42	2	4	3	3	5	1	1	4	2	5	4	3	2	2	5	5	3	3	2	1	1994,1
43	4	5	2	3	2	4	4	2	3	2	2	3	5	2	4	5	3	1	4	1	1972,4
44	1	1	1	5	2	3	1	4	3	2	3	2	3	5	4	3	2	2	4	3	1934,1
45	3	5	1	3	5	5	2	4	5	3	4	2	3	4	1	1	4	1	3	2	2016,8
46	4	5	4	2	4	1	5	1	2	3	5	2	5	4	5	3	1	5	4	3	2379,9
47	2	4	4	4	5	5	3	3	2	5	2	5	4	1	5	2	2	1	2	4	1624
48	1	2	5	4	5	4	2	2	4	1	5	4	3	2	4	5	4	2	4	3	1895,3
49	2	1	3	4	3	1	3	3	3	2	2	4	3	5	2	2	3	2	3	2	1869,7
50	3	3	1	1	1	5	2	2	1	2	4	5	3	2	4	5	1	4	2	5	1919,1

Figure 3.3.3-2 Cost of the first Population



Likewise, there are a couple of additional issues that will have to be solved. The first one is the radius restriction that we mentioned in section 3.3.1. and the other one is the limitation of depots in terms of capacity, brought up in section 3.3.2. . These two issues will be solved with the same strategy. In the event of having any of these two scenarios, the following steps will be taken:

The cost of that DS assigned to that PD will be set to 0 whenever the radius is less than the distance between the sites and/or the desired capacity is larger than the capacity that can be offered. After all the final expenditures have been determined, the individual who has experienced one of these issues will be penalized. The penalty is computed by adding the maximum cost that exists in that already estimated population to the cost that the individual already has from the rest of the sites that could be served.

For example, the maximum cost value of the population of Table 3.3.3-5 is 2501,1, if we had an individual with one of these two cases, to its final cost, this value will be added. Let's take, for instance, the first individual and impose that DS 4 cannot be served by the PD assigned, this will be set to 0, arriving to a cost of 2219,6921. To this cost, we will add the maximum

cost value of the whole population, 2501,1, arriving to the final cost of that individual of 4720,7921.

Table 3.3.3-5 Cost of Chromosome with one Restriction

1	2	3	4	5	COST	FINAL COST
67,082	72,7324	53,1507	0	46,669	2219,6921	4720,7921
6	7	8	9	10		
185,4319	72,5603	66	18	68,8186		
11	12	13	14	15		
71,5542	40	120,9339	44,2719	148,4318		
16	17	18	19	20		
433,9631	107,5174	281,7446	135,8303	185		

3.4. Crossover and Mutation

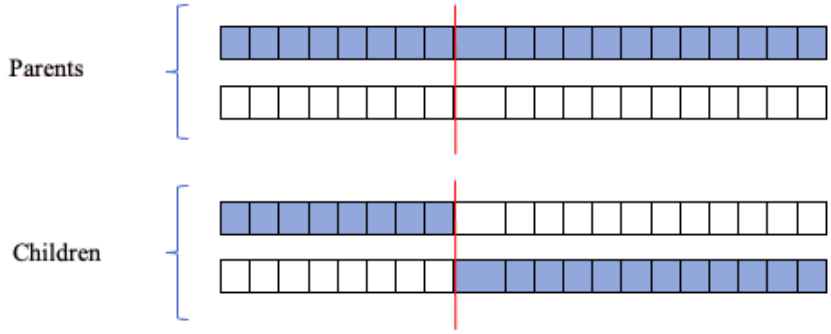
We are now facing one of the most important parts of the GA which is the creation of future generations via the process of crossover and the mutation of selected genes of some individuals to arrive at the greatest, most cost-effective solution.

3.4.1. Crossover

The first phase to generate the next generations is to select those individuals that are going to be the parents. To make this decision, we gave each individual a distinct probability and we compared this value to the crossover operator **Pc**, which will determine whether or not we star with that individual for the crossover. This **Pc** has a value of 80%, any individual that has a higher or equal probability to that value stays in the process.

Once we have 2 selected parents, the crossover takes place. To do this, a cross position variable is defined. This variable will tell us about the exact position in which the crossover is going to take place [9].

Figure 3.4.1-1 Schematic diagram of crossover operation



In our case, this **crossPosition** has a value of 6. Continuing with our small example, if the two parents selected are individual 1 and individual 12 with probabilities 86% and 94%, the resulting children will be the one shown on Table 3.4.1-1.

Table 3.4.1-1 Example crossover operation of Chromosome 1

1	3	4	5	4	4	3	5	4	3	3	3	4	2	2	1	5	3	1	3	5
12	5	4	2	1	2	5	1	2	1	3	4	2	5	1	5	3	5	5	2	3

C1	3	4	5	4	4	3	1	2	1	3	4	2	5	1	5	3	5	5	2	3
C2	5	4	2	1	2	5	5	4	3	3	3	4	2	2	1	5	3	1	3	5

This process is repeated until we have *M* number of children. After all these have been identified, the process of determining their TC can begin. We will end with a total number of individuals of *M*+*M* (parents+children). Now, from this *M*+*M* matrix, we must only stay with half of them, those with lower cost will be more likely to stay. To do this, we employed the Spinning Roulette Wheel method for identifying potentially useful recombination solutions [11].

Roulette selection is a method based on assigning each individual a portion of the wheel proportional to their fitness. The wheel is spun *M* times. At each draw, the individual marked

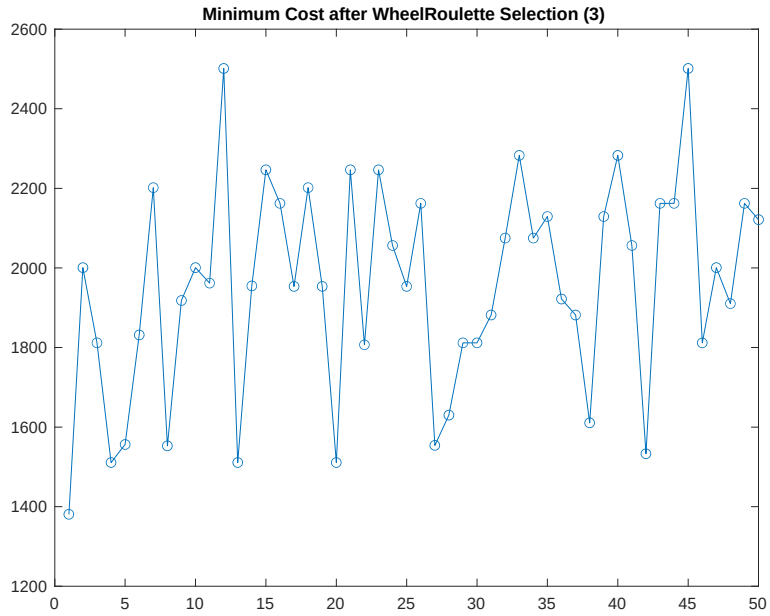
by the roulette is selected and saved. The first step is to order each chromosome of the population based on its cost calculated from the fitness function. Subsequently, we assign a probability of selection to each individual, based on the order settled down. Chromosomes with lower cost will have a higher probability to be selected [9]. To implement this, we followed the steps states below:

The first step was to arrange all the individuals in ascending order, saving the first one that had the lowest cost (elitism). We used the spinning roulette wheel characterized by the evaluation module of paper 1 [2], the evaluation function used is defined as:

$$Eval = \alpha(1-\alpha)^{n-1}$$

Where *n* is the total amount of individuals (*M+M*), and α (alpha) has a value of 0,05. All the individuals were evaluated and each one is assigned a probability, the best solutions will have a bigger probability, while the worst ones, smaller probability, and so, smaller chance of being selected. With this in mind, the process of selection begins to create the population. The process is done *M-1* times, because the first individual of the generation will be the one with lower cost we saved before. Figure 3.4.1-2 shows the costs of the population after the wheel roulette selection.

Figure 3.4.1-2 Cost of population after Wheel Roulette Selection



3.4.2. Mutation

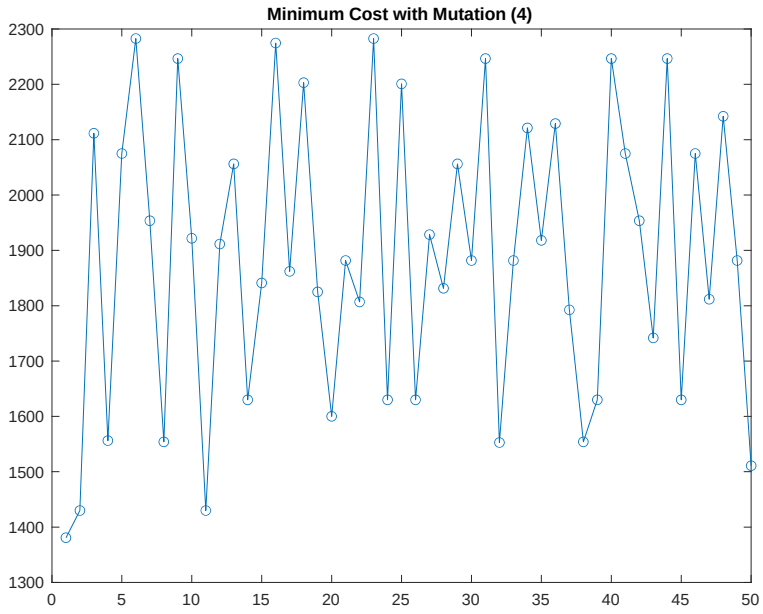
With the intention of arriving to the best solution, the process of mutation is carried out [11]. In this part, we decided to modify 20% of the chromosomes and 10% of the genes of the whole population. As we have 50 chromosomes, 10 will be modified, and 2 of the 20 genes will also be changed.

The identical procedure from section 3.4.1. of assigning a probability to each chromosome and generating a random probability that will make us choose that chromosome or another one was used in this part, and equally genes were manipulated. Once we have the individual and the genes that are going to be changed, a random number from 1 to the number of PDs we have is assigned excluding the one that it already had. For example, if we change the genes 15 and 17 from chromosome 19, we get the individual shown in Table 3.4.2-2. The cost is recalculated for these newly generated individuals.

Table 3.4.2-2 Chromosome 19 with mutation

CHROMOSOME 19 BEFORE MUTATION																			
M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
19	2	3	2	4	3	4	2	5	2	2	2	5	2	4	1	2	4	2	5	1
CHROMOSOME 19 AFTER MUTATION																				
M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
19	2	3	2	4	3	4	2	5	2	2	2	5	2	4	4	2	2	2	5	1

Figure 3.4.2-3 Cost of Population after Mutation



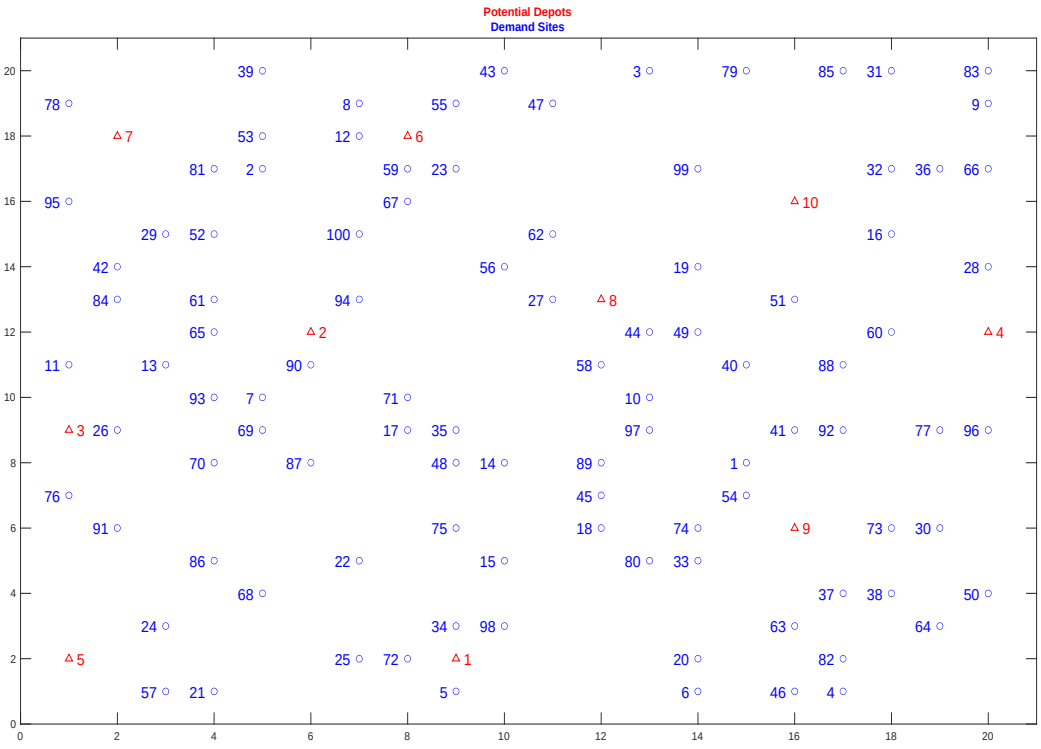
As a result, we arrive at a final generation with final cost values, which will serve as our input matrix for repeating the process with the primary goal of achieving the greatest possible outcome. In order to examine the cost trend and see how the final result improves, two variables will be saved in each iteration. These two variables are Average fitness and Best fitness.

Chapter 4

EXPERIMENTAL ANALYSIS

The results of what was developed in Chapter 3 are shown in this chapter. The goal is to see how well the pre-programmed algorithm can arrive at the best solution extrapolating the process to our final situation. This scenario is shown in Figure 3.4-1.

Figure 3.4-1 Final Scenario



This chapter will be divided into 4 sections. In the first we will complete the process without applying any restrictions, in the second will be completed with radio restrictions but no capacity constraints, the third will be performed only with capacity restrictions, and the fourth and final process will be done with both constraints altogether.

4.1. Experiment with Policy 1

We will start with the scenario given above, which includes 100 DS and 10 PD. The capacity requested from the DS is a random array from 6 to 60. As in this stage there are no restrictions, the capacity of the PDs is high enough so that every DS can be served. Table 4.1-1 shows the values of the capacity of each DS.

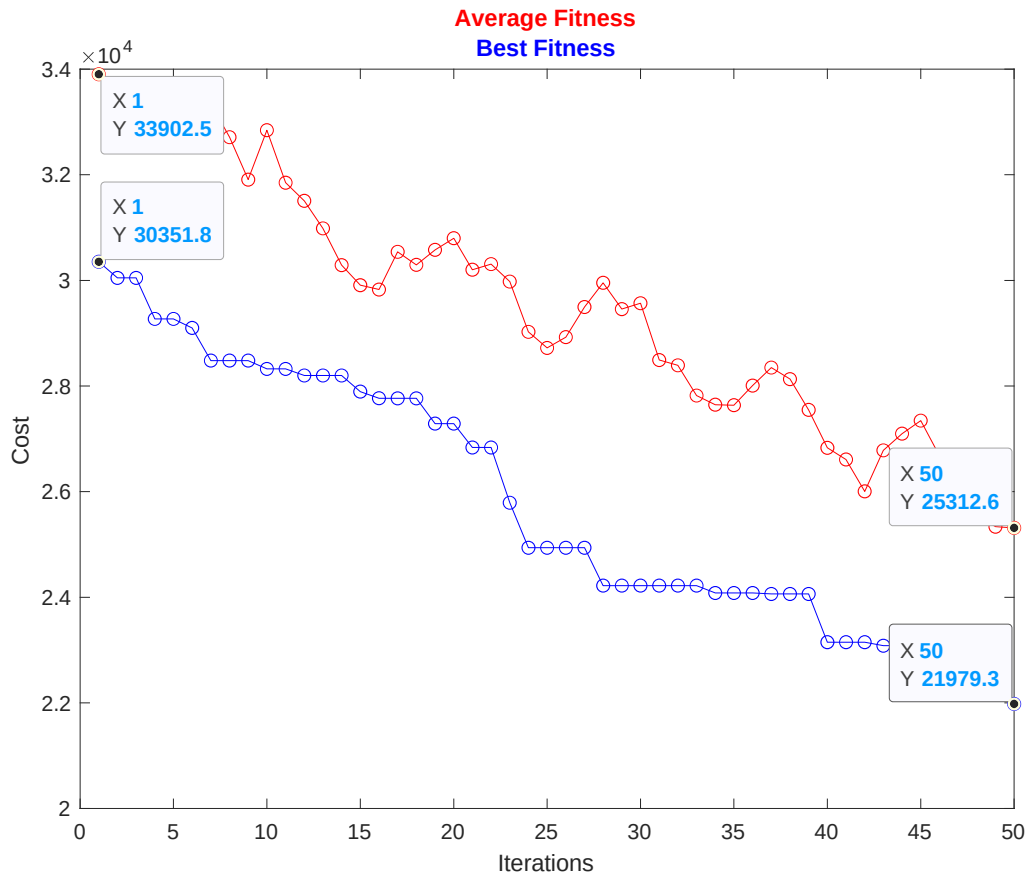
Table 4.1-1 Capacity of each SD

Id	C	Id	C	Id	C	Id	C	Id	C
1	38	21	51	41	34	61	20	81	36
2	56	22	40	42	55	62	50	82	29
3	32	23	39	43	38	63	6	83	17
4	20	24	13	44	24	64	32	84	13
5	23	25	18	45	23	65	21	85	7
6	41	26	20	46	36	66	27	86	56
7	27	27	21	47	40	67	56	87	53
8	17	28	50	48	10	68	13	88	12
9	36	29	31	49	57	69	46	89	24
10	55	30	11	50	25	70	12	90	38
11	54	31	48	51	21	71	50	91	41
12	11	32	39	52	12	72	23	92	53
13	15	33	57	53	47	73	16	93	12
14	12	34	50	54	52	74	17	94	55
15	56	35	52	55	35	75	24	95	17
16	12	36	23	56	10	76	25	96	43
17	59	37	41	57	51	77	6	97	51
18	17	38	15	58	43	78	27	98	40
19	36	39	20	59	21	79	32	99	35
20	43	40	29	60	25	80	21	100	41

In the same way, as there is no radius restriction, we set a radius large enough so that all depots can reach all DS. All things considered, we put the code to work, repeating the procedure 50 times.

Figure 4.1-1 depicts the improvement of the cost as the 50 iterations of the process advance. We can see that the average fitness at the beginning has a value of 33902,5 with a minimum value of 30351,8, while after the 50 iterations, we have an average fitness value of 25312,6 and the lowest fitness is 21979,3.

Figure 4.1-1 Cost after 50 iterations Policy 1

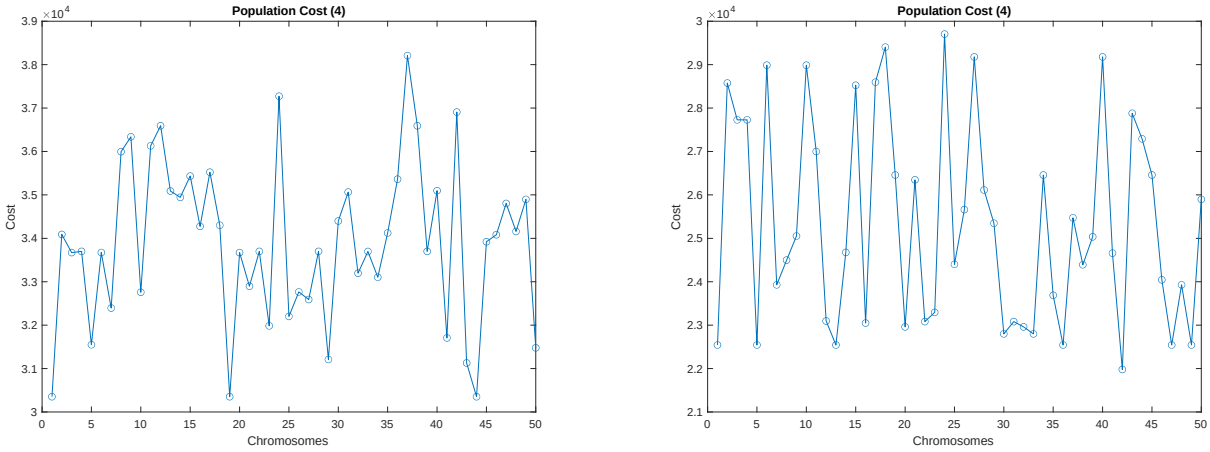


This graph compares the costs of the first generation to the most recent one, indicating a decline of 8589,9 for the average fitness and 8372,5 for the best one of each generation. As it can be seen, the graph's trajectory follows an exponential decay function. This means that we are able to achieve our goal of lowering the TC. We manage to minimize the cost or maintain it constant in each cycle if a better option is not found. It is worth noting that the average cost graph is

trending downwards, despite some punctual rises (that do not reach the maximum cost in any iteration of the process). These increases suggest that, despite the fact we are improving the cost, as we run mutations across the full test space, we occasionally reach a place where the cost is higher, resulting in cost increases. However, if the spot we have discovered has a low cost, we will be able to achieve our goal. This indicates that the process is being carried out well and that all the range of possibilities are being analyzed.

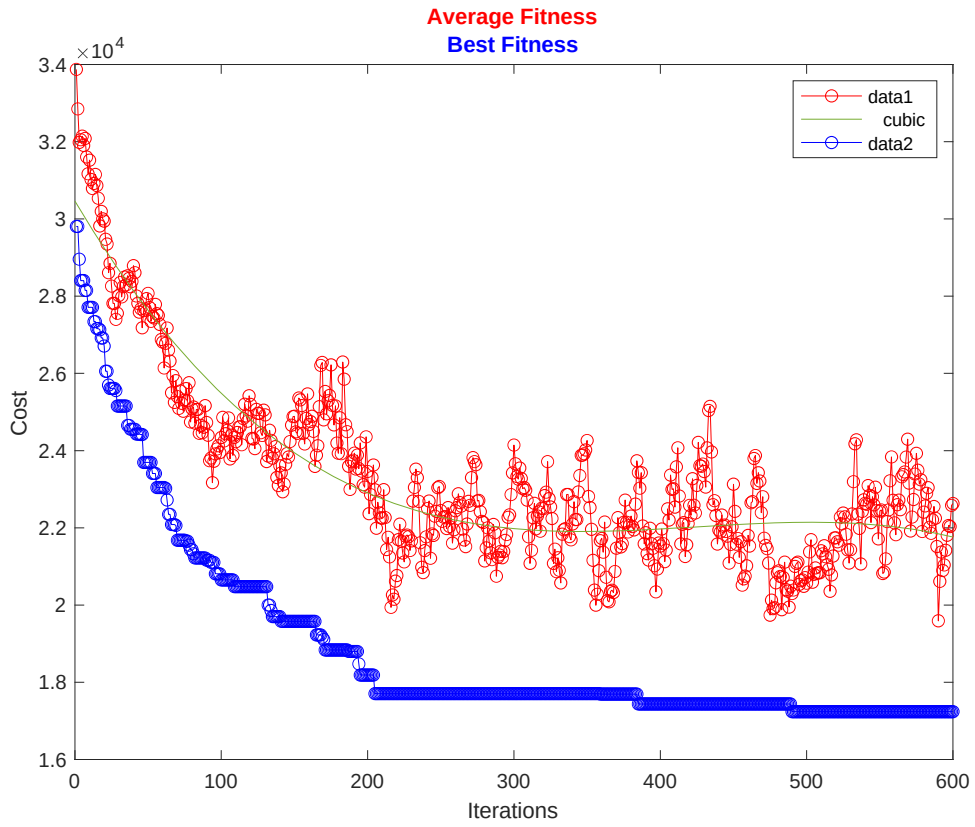
Figure 4.1-2 shows, on the left side the costs for the first generation and on the right side for the last one. We can see that the cost values of the first generation are between $3 \cdot 10^4$ and $3{,}9 \cdot 10^4$ while, on the last generation, these values descend to $2{,}15 \cdot 10^4$ and $3 \cdot 10^4$.

Figure 4.1-2 Costs of first and last generations Policy 1



In addition, we set the code to work by repeating the operation 600 times to assess the graph's trend with a larger number of iterations. We can see that, in lieu of continuing the exponential descending pattern we discussed earlier giving results successively closer to the desired optimum result, at one point the graph eventually become constant in this desired result. The green line depicts the mean fitness data's trend.

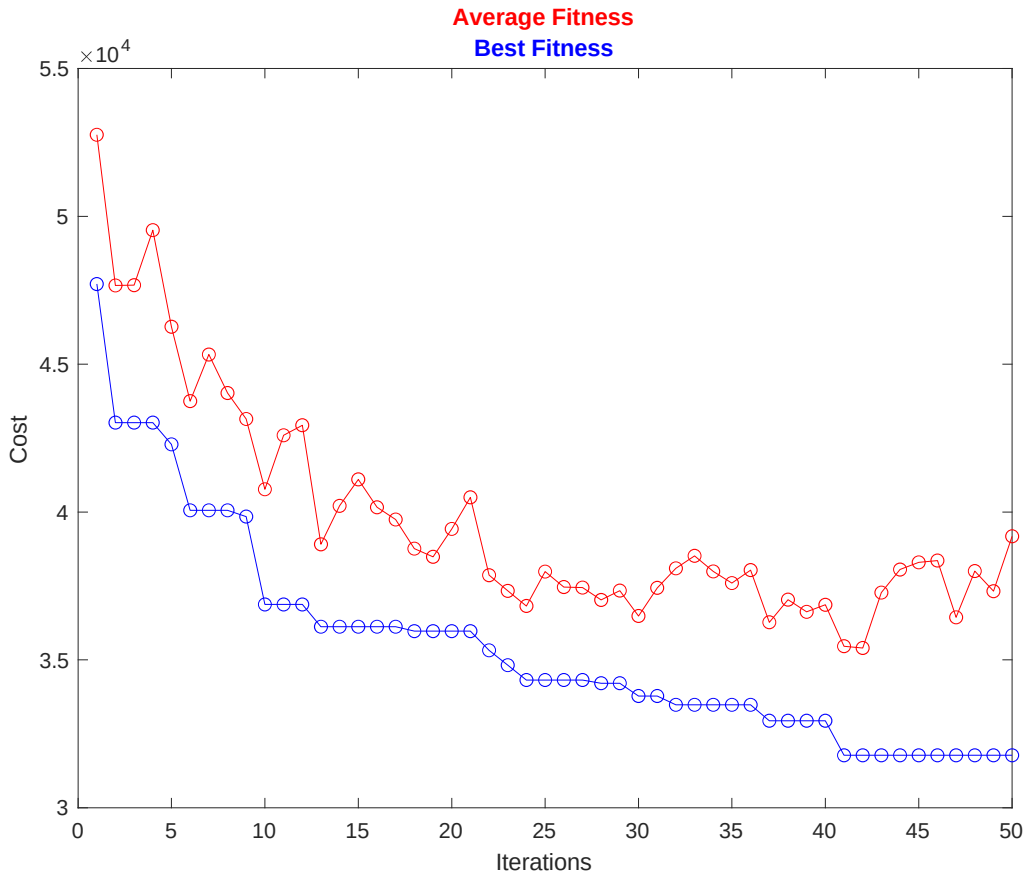
Figure 4.1-3 Cost with 600 iterations



4.2. Experiment with Policy 2

In this second instance, we set a radius of 16 as a maximum distance to be covered and run the algorithm to see the results under this restriction. Doing this, some DS will not be served by the PD assigned to them. As it was explained in Chapter 3, this DS not served will be charged with the penalty of adding the highest cost of the individual in that population. With this in mind, the following graphs are obtained.

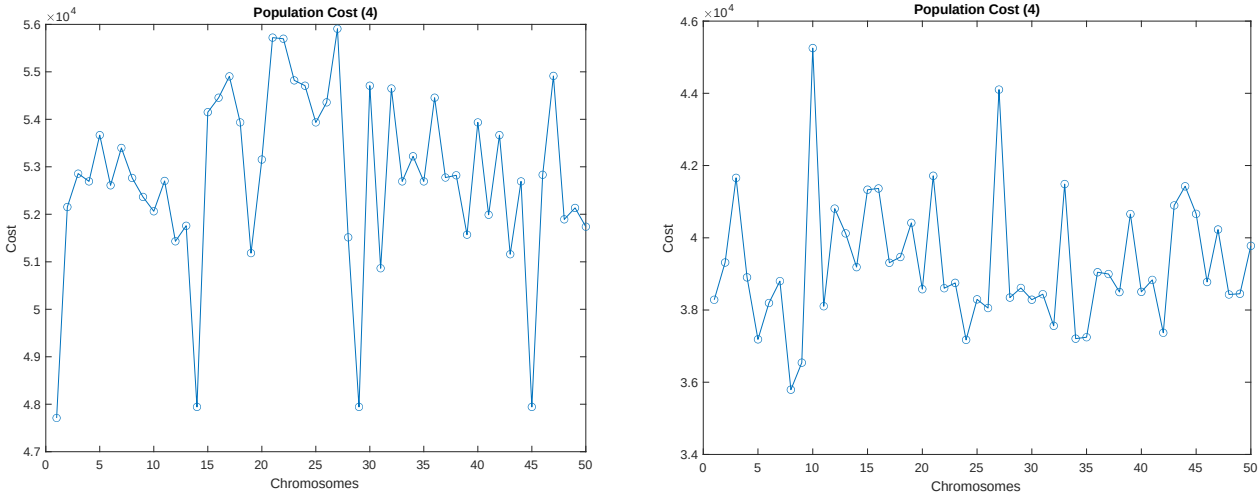
Figure 4.2-1 Cost after 50 iterations Policy 2



As in the first example, we can infer that we are able to lower the cost as iterations augment. However, if there are more individuals in that generation that are not able to get the service they require because it does not match the radius requirement, there will be more penalties, which means that the costs will increase considerably. Here, the minimum cost has values between $4{,}7 \cdot 10^4$ and $3{,}2 \cdot 10^4$, while, when we have no restrictions, the values of the best fitness are between $3{,}1 \cdot 10^4$, and $2{,}2 \cdot 10^4$.

The following figure depicts the cost of the first generation on the left and the last generation on the right. We can see that the range of values has been reduced as well.

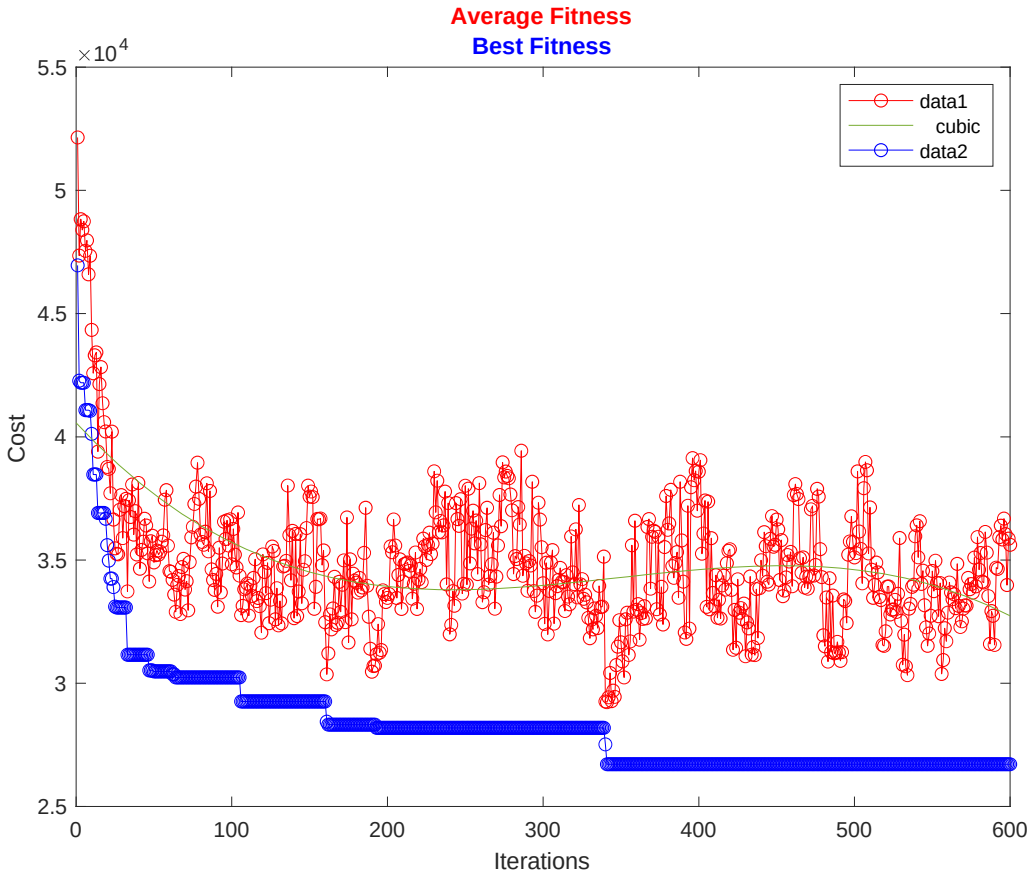
Figure 4.2-2 Costs of first and last generations Policy 2



It is worth noting that the cost is higher in this circumstance than in the scenario one, when there were no restrictions, despite the fact that we are not providing service to those depots that do not satisfy the established condition.

We repeated the process 600 times also in this scenario to see the trend of the graph, depicted in Figure 4.2-3. Same as before, at one point the graph the tendency eventually become constant. The green line depicts the mean fitness data's trend.

Figure 4.2-3 Cost for 600 iterations and radius restriction



As an additional experiment, we have carried out the same study but this time, the penalty in each generation is going to be the same. We impose a non-variable assignment, this is: twice the diagonal of our grid multiplied by the number of DS we have. The following Figure 4.2-4 and Figure 4.2-5 show the results obtained.

Figure 4.2-4 Cost after 50 iterations Policy 2 with constant Penalty

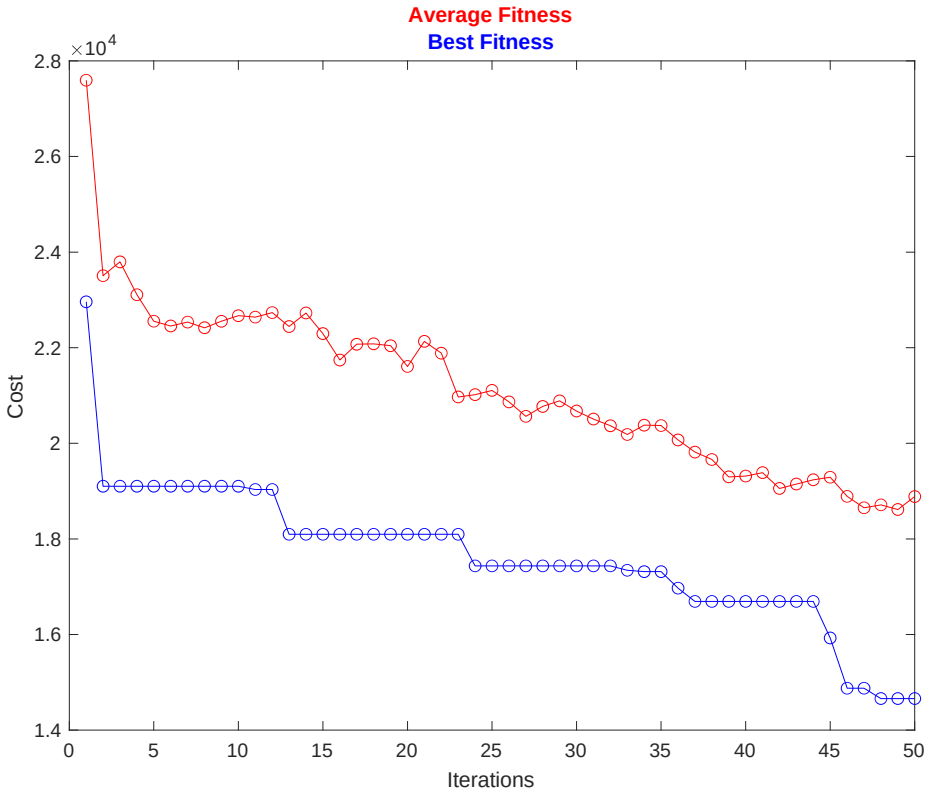
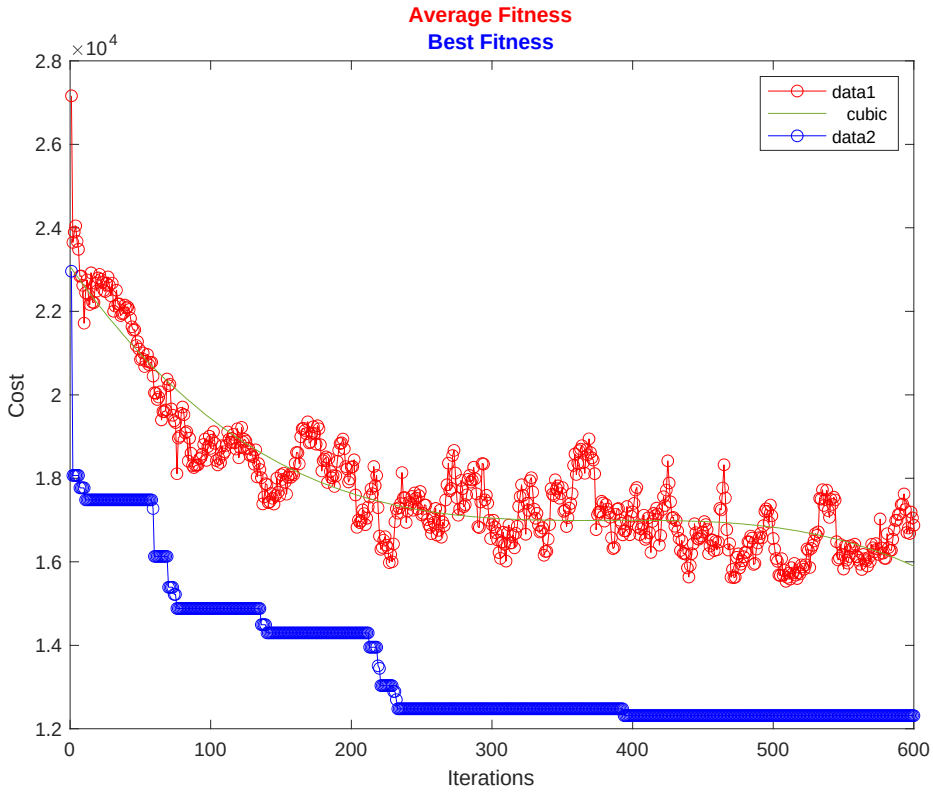


Figure 4.2-5 Cost after 600 iterations Policy 2 with constant Penalty



We can see how, doing this, the costs are lower than in the case of having a variable penalty, depending on the maximum value of the cost on each generation.

The following Table 4.2-1 shows the difference between applying variable or non-variable penalties.

Table 4.2-1 Fitness Comparison for the 2 Penalties of Policy 2

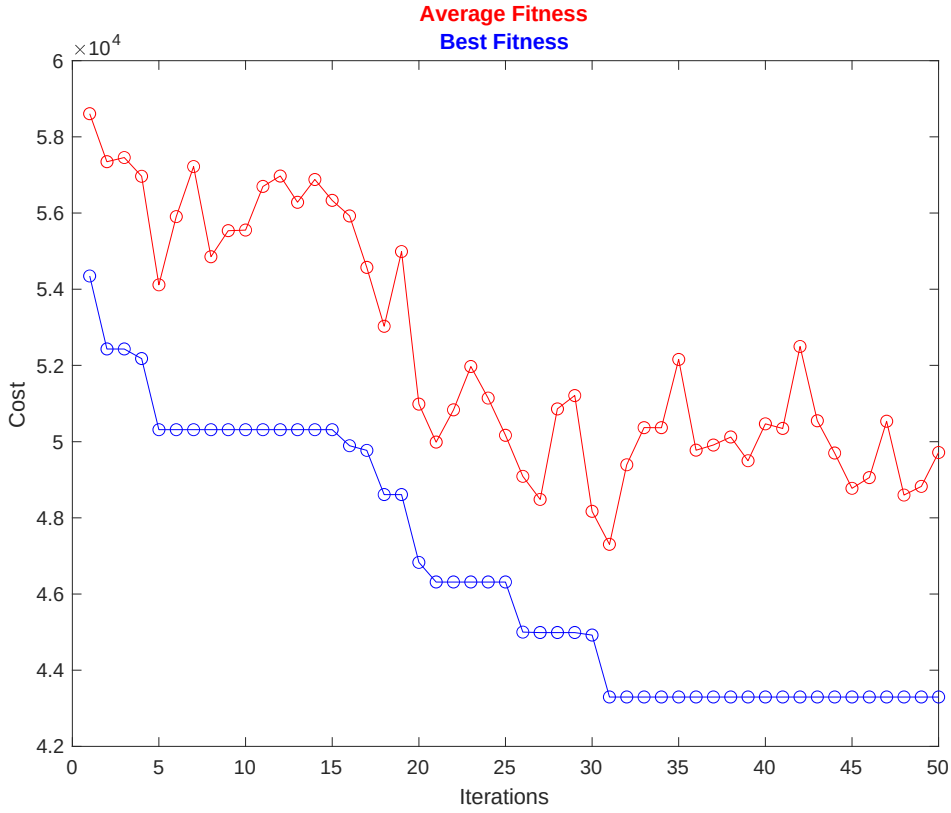
Penalty	Average Fitness 50 Gen	Best Fitness 50 Gen	Average Fitness 600 Gen	Best Fitness 600 Gen
Variable	42364	32100	35000	27650
Non-variable	22783	14500	17821	12160

We can again see how, the TC values are lower in the case of applying a non-variable penalty.

4.3. Experiment with Policy 3

The third stage has no radius restriction, but capacity restrictions instead. Until now, the PDs had a capacity large enough to service each one of the DM. Now things are different, the PDs will have a limited capacity implying that they will not be able to serve all of the DM, only those who demand less than the maximum capacity that the PD has to give. In Figure 4.3-1 we see the consequences of this restriction.

Figure 4.3-1 Cost after 50 iterations Policy 3



The same thing happens as in the previous radius restriction scenario. We see a decrease in cost, but, then again, the costs are also higher in this scenario than in the first one where we had no restrictions at all.

The cost of the first and last generations are sketched below, showing that first generation has higher costs than the last one.

Figure 4.3-2 Costs of first and last generations Policy 3

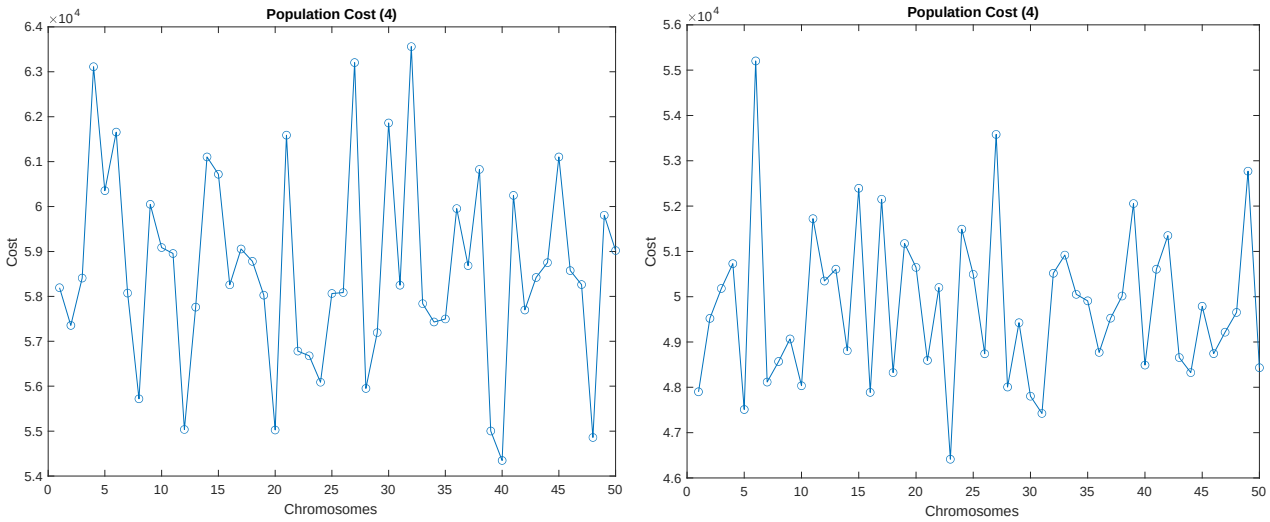
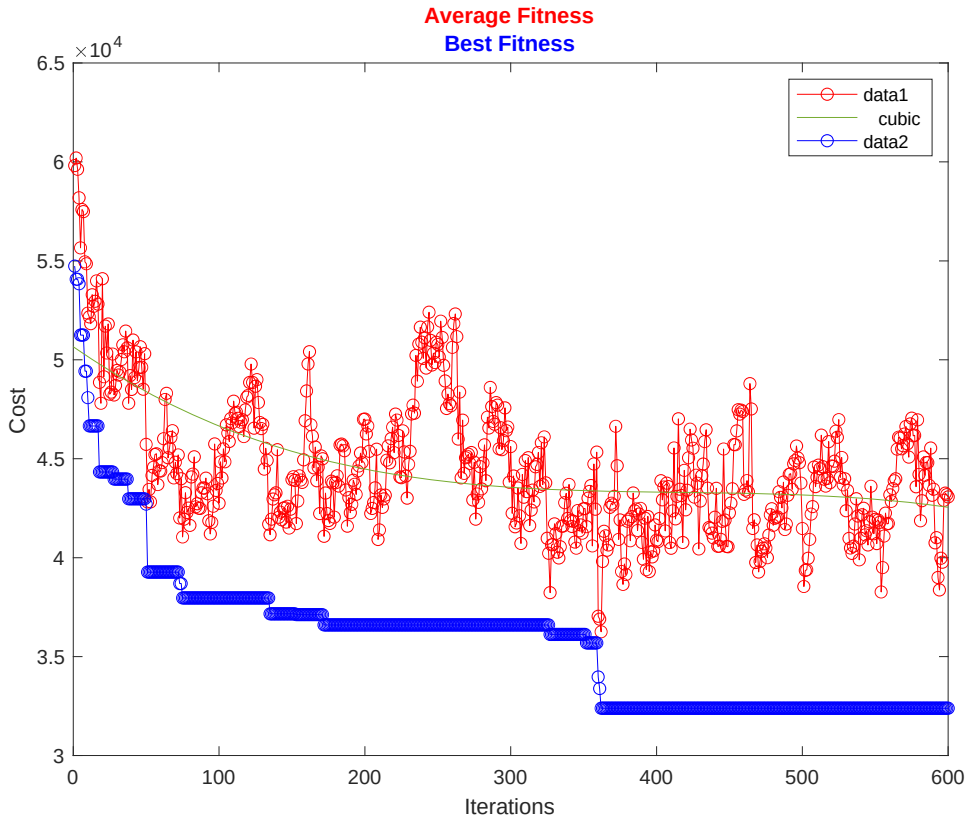


Figure 4.3-3 shows the tendency of the graph repeating the process 600 in this third scenario.

Figure 4.3-3 Cost for 600 iterations and capacity restriction



Again, we repeated the process applying the same penalty in each iteration to see what we obtain. It is observed that also in this case the costs are lower than before.

Figure 4.3-4 Cost after 50 iterations Policy 2 with constant Penalty

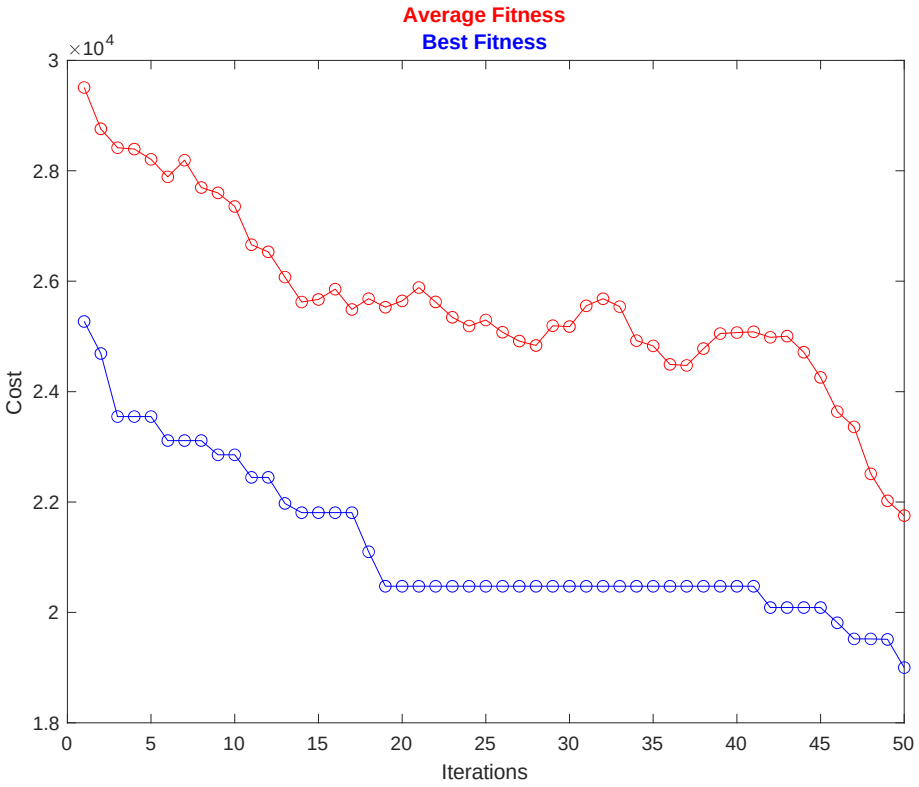


Figure 4.3-5 Cost after 600 iterations Policy 2 with constant Penalty

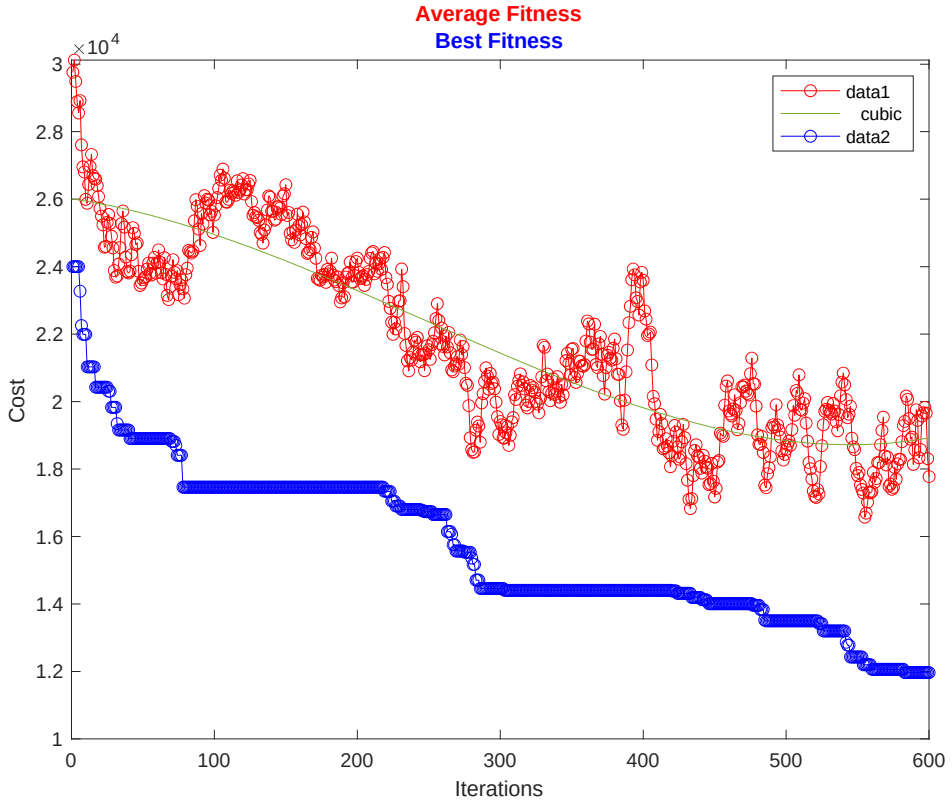


Table shows the values of the costs in which we can see how, as in Policy 2, in the case of having a non-variable penalty, the cost is lower.

Table 4.3-1 Fitness Comparison for the 2 Penalties of Policy 3

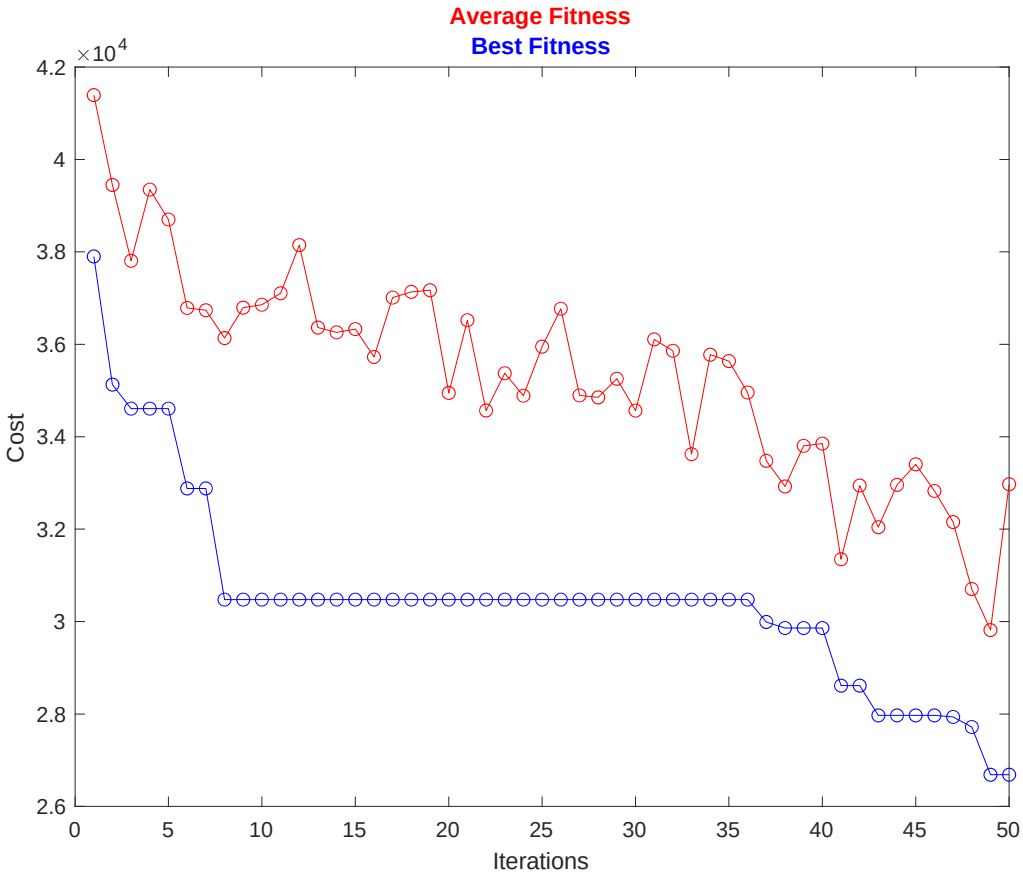
Penalty	Average Fitness 50 Gen	Best Fitness 50 Gen	Average Fitness 600 Gen	Best Fitness 600 Gen
Variable	52000	43863	45000	32000
Non-variable	25754	19243	22172	13000

4.4. Experiment with Policy 4

In this last case, we applied both capacity and radius restrictions (

Figure 4.4-1). We get the same outcomes as before. With each repetition of the process, the cost decreases and follows a decaying exponential form.

Figure 4.4-1 Cost after 50 iterations Policy 4



We still have greater cost than in the first case without restrictions, but it is remarkable that costs are, for the same number of iterations, lower than in the cases where we only apply one constraint. Reason for this is that when more limits are imposed, less DS will be served, reducing costs (TC will be lower) and therefore reducing individual's maximum cost, resulting in a lower penalties than when only one restriction is applied. To see this clearly,

Table 4.4-1 shows the maximum values for 20 generations with 2 restrictions compared to these same values but only with one restriction.

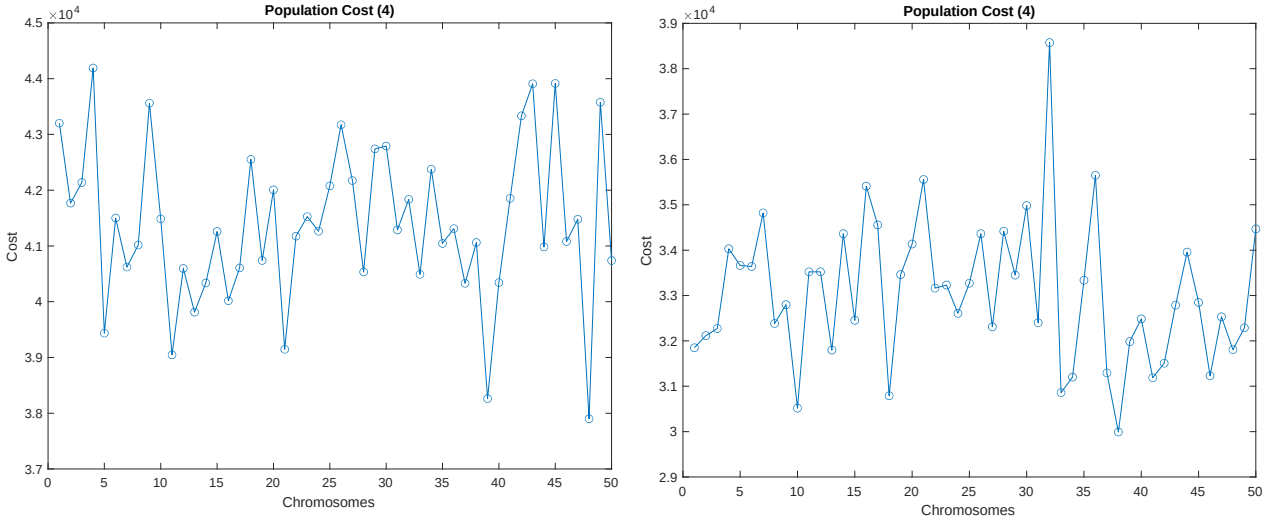
Table 4.4-1 Maximum Cost values with 1 and 2 restrictions

2 restrictions			1 restriction	
1	21713		1	33434
2	21193		2	31849
3	21785		3	31265
4	20959		4	31718
5	20534		5	31748
6	20076		6	29003
7	21045		7	31369
8	21739		8	32002
9	21550		9	30850
10	20840		10	34089
11	19344		11	33683
12	19490		12	30159
13	21842		13	28613
14	21134		14	28448
15	21591		15	28808
16	21845		16	28480
17	22752		17	28046
18	22441		18	29099
19	19498		19	29867
20	20239		20	27987

It is also worth noting that the values of the minimum and average costs are closer in this situation than they are in the others. This is because we concentrate the maximum and minimum values, leaving out the DS that are at a greater distance or have higher demand (those with higher costs).

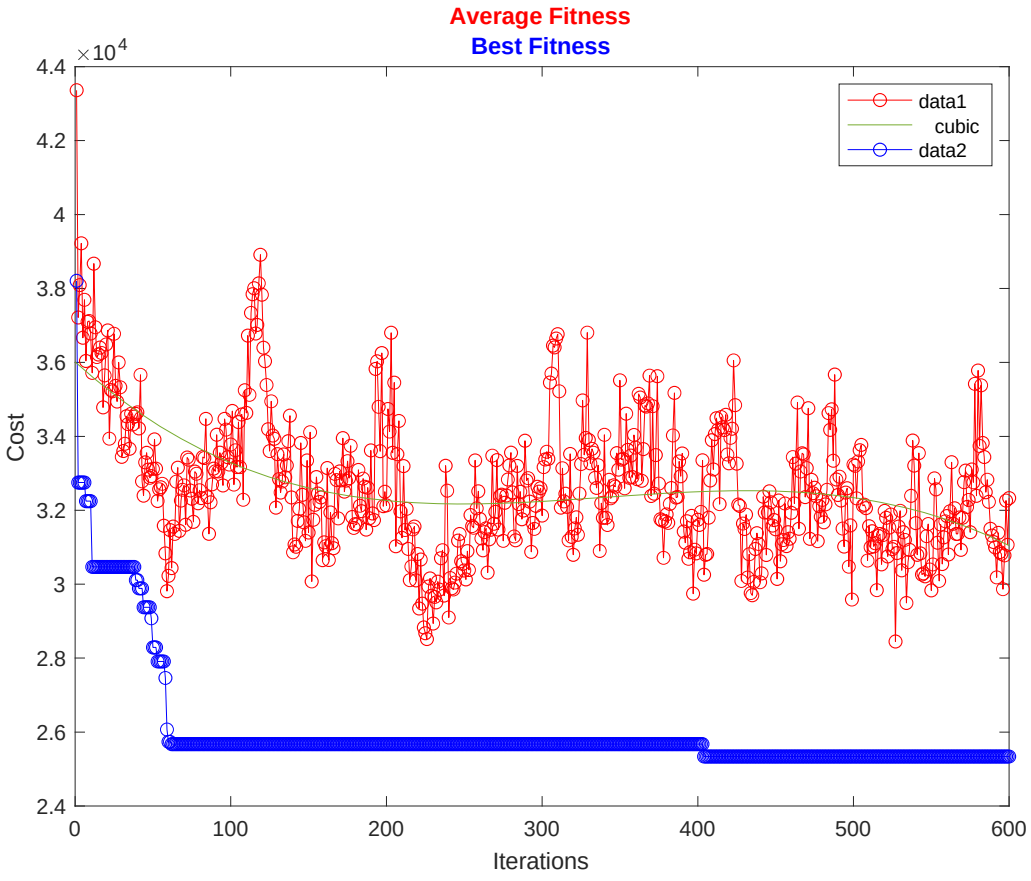
Again, we have the cost of the first generation on the left and the cost of the last one on the right in the next figure.

Figure 4.4-2 Costs of first and last generations Policy 4



The following Figure 4.4-3 shows the tendency of the graph after repeating the process 600 in this fourth scenario will both radius and capacity restrictions.

Figure 4.4-3 Cost for 600 iterations and both restrictions



The study applying the same penalty in each generation is shown in Figure 4.4-4 and Figure 4.4-5, where we can see that, once again, the costs are lower than in the case of having a different penaly in each generation.

Figure 4.4-4 Cost after 50 iterations Policy 2 with constant Penalty

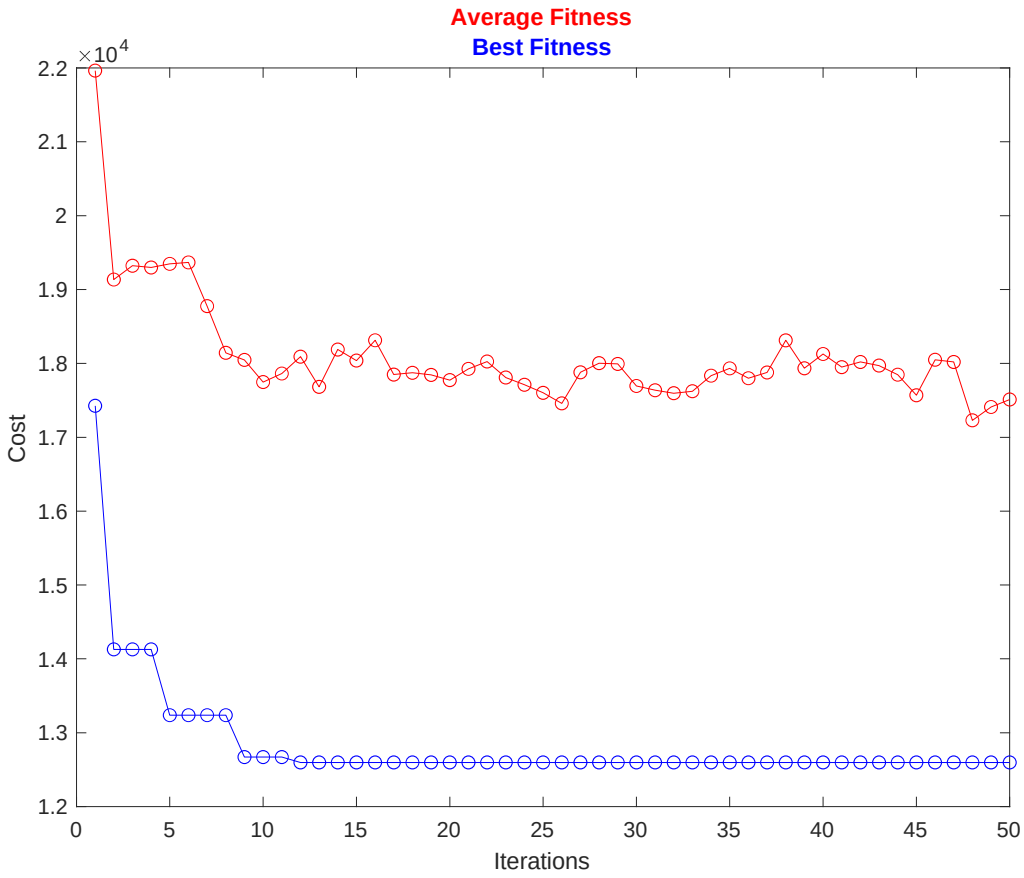
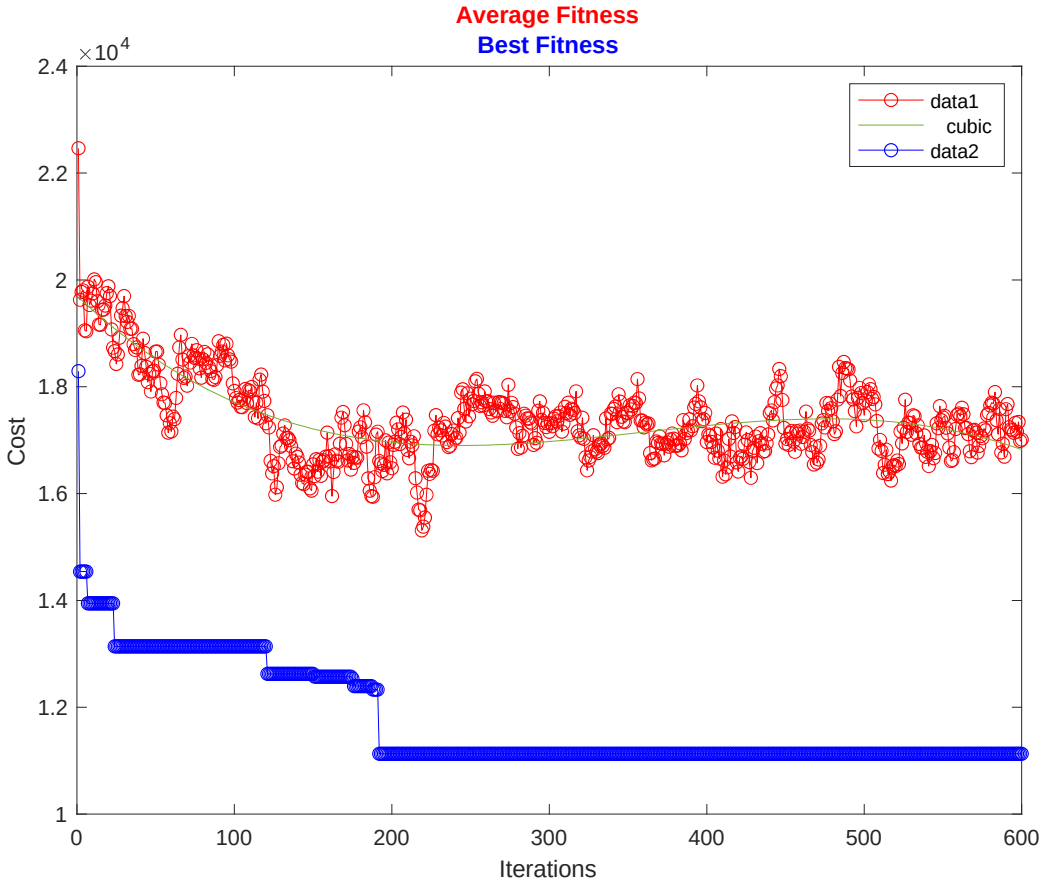


Figure 4.4-5 Cost after 600 iterations Policy 2 with constant Penalty



The Table 4.4-2 compares the values of each TC, and we can see that the values with non-variable penalty are lower once again.

Table 4.4-2 Fitness Comparison for the 2 Penalties of Policy 4

Penalty	Average Fitness 50 Gen	Best Fitness 50 Gen	Average Fitness 600 Gen	Best Fitness 600 Gen
Variable	34721	27532	32340	25891
Non-variable	18134	12563	17152	11621

Chapter 5

CONCLUSIONS

This last chapter shows us the key conclusions reached while working on this project after the analysis of the implementations of the genetic algorithm we performed. Its purpose is to provide a broad overview of the study, highlighting the research's lessons learned and accomplishments achieved, as well as to suggest future way ahead.

5.1. Conclusions

First, we must consider the potential associated with genetic algorithms. During the project, we were able to verify that the algorithm implementation is relatively simple and that the results obtained are adequate. In most cases, the algorithm's convergence was satisfactory, with fewer than 100 iterations capable of covering the answers to practically all the situations. It has been necessary to repeat the simulation on some occasions because the solution sought was dependent on the initial population, the probabilities to choose parents and the genes and chromosomes manipulated. We observed that the solution could be improved by repeating these simulations.

In the world of business, economies of scale are cost advantages reaped by companies when production becomes efficient. Companies can achieve economies of scale by increasing production and lowering costs when these are spread over a larger number of goods.

In our case study, the results shown in Chapter 4 led us to a number of interesting conclusions. Our objective, broadly speaking, is the optimization of our model to make our business model

more efficient, by means of adjusting the existing processes to increase the occurrence of favorable outcomes and decrease the occurrence of undesirable ones.

In the four scenarios we see how, as we expand the number of iterations, the implicit costs decrease. In the first one (no penalizations) this reduction of costs is lineal up to one point where it stabilizes (Figure 4.1-1), which is not the case in the other three instances. Here, the consequences of not attending certain DS, either due to excessive distance (Figure 4.2-1), demand (Figure 4.3-1), or both (

Figure 4.4-1) produce relevant increments in terms of costs where these events occur.

In any case, the study shows that when we tend to a relevant number of cases, the results are better if we meet all the demand sites, including situations where the distance or volume demanded might lead us to believe that covering certain demand sites would be inefficient. This is due to the penalties we incur by proceeding in this way. It is worth highlighting the results obtained in the last case in which we impose two restrictions. The costs are here lower than in the scenarios with just one restriction, even though more sites are left without service.

Therefore, if our objective, as stated above, is the optimization of our model and to reach the most efficient business model, the solution is to cover the maximum number of demand sites from our depots, reducing the penalizations to a minimum.

An observed disadvantage is the fact that one solution is “better” only in contrast to another, an evolutionary algorithm like our GA will finish when we decide, either because the time consumed or the number of iterations, never on its own when the optimum result is reached.

5.2. Future Work

Some of the possible future lines of research arising from the project are detailed below.

On the first place, complementing this study with the Coral Reefs Optimization (CRO) algorithm will possibly lead to better solutions. CRO algorithm is based on a reef of solutions to a given optimization problem (corals), that reproduce in a similar way as corals do in nature (broadcast spawning, brooding and budding operators are implemented) [12].

Second, once studied the design for transport costs for potential depots, the study process can continue with the same transport costs calculation for specific resources, such as medical, material, personnel.

Finally yet importantly, conducting this research in more defined environments would be quite beneficial. This is, determine the costs and time required based on the orography of the terrain, for instance, or to extend the study by defining and characterizing the use of different means of transport, will allow us to a more accurate and down to real life conclusions.

BIBLIOGRPHY

[1] Mumtaz Karatas, Ertan Yakici, Nasuh Razi, “Military Facility Location Problems: A Brief Survay,” July 2018, DOI: 10.4018/978-1-5225-5513-1.ch001, In book: Operations Research for Military Organizations, pp. 1-27, Publisher: IGI GLOBAL.

[2] WEN Meilin, LU Bohan, LI Shuyu, and KANG Rui, “Location and allocation problem for spare parts depot on integrated logistic support,” Journal of Systems Engineering and Electronics, Vol. 30, No. 6, December 2019, pp.1252-1259

[3] Chyh-Ming Lai, “Integrating simplified swarm optimization with AHP for solving capacitated military logistic depot location problem,” Applied Soft Computing, Vol. 78,May 2019, Pages 1-12.

[4] Hui Hu, Jing He, Xiongfei He, Wanli Yang, Jing Nie, Bin Ran, “Emergency material scheduling optimization model and algorithms: A rewiew,” Journal of Traffic and Transportation Engineering (English Edition), Vol. 6, Issue 5, October 2019, Pages 441-454.

[5] Xin Yao, Yong Liu and Guangming Lin, "Evolutionary programming made faster," in IEEE Transactions on Evolutionary Computation, vol. 3, no. 2, pp. 82-102, July 1999

[6] M.H. Fazel Zarandi, S. Davari, S.A. Haddad Sisakht (2011) The large scale maximal covering location problem, Scientia Iranica, Volume 18, Issue 6, 2011, Pages 1564 1570.

[7] Sanz, S. S. (2020). UNA INTRODUCCIÓN A LOS ALGORITMOS DE COMPUTACIÓN EVOLUTIVA. Departamento de Teoría de la Señal y Comunicaciones. Universidad de Alcalá.

[8] C. A. Coello y G. B. Lamont (2007). Evolutionary Algorithms for Solving Multi-Objective Problems, Springer, 2007.

[9] R. L. Haupt y D. H. Werner. (2007). Genetic Algorithms in Electromagnetics IEEE Press.

[10] G. Smith, "*Essential Statistics, Regression, and Econometrics* ", Handbook of Statistics, 2020. ScienceDirect

[11] Marczyk, A. (2004). Algoritmos genéticos y computación evolutiva. the-geek.org. http://the-geek.org/docs/algen/

[12] S. Salcedo-Sanz, J. Del Ser, I. Landa-Torres, S. Gil-López, J. A. Portilla-Figueras, "The Coral Reefs Optimization Algorithm: A Novel Metaheuristic for Efficiently Solving Optimization Problems", The Scientific World Journal, vol. 2014, Article ID 739768, 15 pages, 2014. https://doi.org/10.1155/2014/739768

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá