Ministry of Education and Science of the Russian Federation
Federal State Autonomous Educational Institution of Higher Education
«Ural Federal University named after the first President of Russia B. N. Yeltsin»

Engineering School of Information Technologies, Telecommunications and Control Systems

MASTER THESIS

Genetic algorithms for route planning of bank employee

Supervisor: Svyatoslav Solodushkin

Student of the group: РИМ-281229

Name: Albahadili Murtdha Sadoon

Ekaterinburg
2020

Abstract:

Evolutionary algorithms (EAs) are machine-learning techniques that can be used in many applications of optimization problems in various fields.

Bank route planning is nowadays combinatorial optimization problem, in which customers request services from a bank. Each service is composed of dependent tasks, which are executed by employees of varying skills along one or more days. The objective is to schedule and route teams so that the distance and cost are minimized, In this thesis we present a solution model tackle this problem using genetic algorithm approach.

Keywords: EAs, combinatorial optimization problem, Bank route planning

Content:

Tables and figures:

Acronyms:

GA Genetic Algorithms

TSP Traveling Salesman Problem

VRP Vehicle Routing Problems

MTSP Multiple Traveling Salesman Problem

NP-Hard Non-Deterministic Polynomial-Time Hard (No Algorithm Can Solve This Problem in Polynomial Time)

NP-Complete Non-Deterministic Polynomial-Time Complete

1. Introduction:

In recent decades, the world has witnessed a growing interest in intensive modern technology. At present, the subject of technological innovation and its integration into the economy in general, as well as in the banking system, has great importance as it affects the relationship with the clients. The revolution in communications and technology in the banking sector increases local and global competition. It provides the necessary services and promotions to be able to thrive, compete, and meet the needs of its potential clients. It also enables customers to choose between services that suit their personal preferences.

Recently, banks have witnessed major changes in their operations and services, and these new changes pose major challenges to the banking sector. To establish a strong relationship with the customers modern technology is one of the most effective tools due its competitive advantage. Modern technology has the potential to coordinate these relationships through direct and effective communication with clients on an ongoing basis to achieve their satisfaction and ensure loyalty.

Maintaining clients and attracting newcomers takes the bank to a higher level. Banks seek to achieve their goals (i.e. profit and customer service), and introduction of modern technologies in banking operations is a mean to achieve these goals because it enhances the trust level between banks and customers.

The association between banks and business owners is another challenge in the banking sector. The bosses and employers think about using external offices or companies to solve problems and develop their services not only at the short period but in the long run as well.

On the other hand, customers generally visit the bank with their problems related to withdraws and/or deposits etc. that can be solved with the help of technological development which consequently led to a significant decrease the number of complains. It doesn’t reduce just the pressure on employers but provides an opportunity for them

to develop a new customer service to help the bank as well as the clients. the idea is Reverse this process where the employee must visit the customer

Such tasks are usually handled by a special team where the CEO or project manager distributes the employees according to a specific algorithm. The development of such algorithm programmatically is very important to anticipate many customers. Which reflects the importance of technology in the distribution process, organization of visits and customer identification.

The new service also requires a comprehensive study to design a plan by specialists to develop the appropriate solution for this service, where the number of clients, time of their visit and distance between them is well managed. It is one of the biggest problems bank managers are facing to get the best outcome for the bank and clients at the same time which are almost impossible to tackle without the interference of modern technology.

Account managers are bank employees who come to meet with a client to open a current account. The impression that these meetings leave directly affects the image of the bank in the eyes of the client, therefore it is important that the manager's route be drawn up correctly and not be late for the meeting. In addition, do not forget that managers - living people with their own preferences - would also like to participate in planning (for example, indicating a convenient start and finish point for the route, or setting the days on which they can work). This assignment proposes to develop a route planning system that takes into account the wishes of both employees and customers of the bank.

The task is similar to the well-known traveling salesman problem and, if properly implemented, will positively affect the work of several hundred bank employees.

1.1. Objective of Dissertation:

The main objective of this dissertation is to develop a genetic algorithm which manages the distances from employs to customers and the times required for each process.

Chapter 1

1. Related Works (Literature review):

The various applications of MTSP arise in real world problems such as school bus routing, printing press scheduling, interview scheduling, crew scheduling, hot rolling scheduling, mission planning and design of global navigation satellite system (GNSS). Due to its diversified applications, the MTSP has been extended to many practical variants such as MTSP with multiple depots, fixed charges, fixed number of salesmen, and time windows [5]. Since the MTSP is an exceptional variant of TSP, the solution procedures available for TSP can also be applicable for MTSP. Additionally, the MTSP can be extended to countless practical situations like distribution system in transportation, particularly in vehicle routing problems (VRP). This study keeps much attention on MTSP than the usual TSP. The solution approaches used to solve MTSP can be categorized into heuristics, meta-heuristics, and exact approaches. Different heuristic algorithms have been presented in the literature to solve MTSP and its variants. The first heuristic algorithm for min-sum MTSP was appeared in [6], where it utilizes an extension of prominent Lin and Kernighan heuristic. A two-phase heuristic algorithm has been proposed to solve no-depot min-max MTSP, where m tours are established in the first phase, and these tours are explored in phase two. A neural network-based solution procedure [7] has been developed for solving MTSP. A competition based neural network approach [8] for MTSP with minmax objectives has been projected. (Soylu, 2015) presented a general variable neighborhood search algorithm (VNS) for MTSP and which was then applied to a real-life problem raised in traffic signalization network of Kayseri province in Turkey [9]. The exact solution procedures for different models of MTSP can be found in [10][11][12]. Apart from the heuristics and exact algorithms, bio-inspired methods like genetic and evolutionary algorithms have been developed to tackle MTSP and its variants in the literature. (Yousefikhoshbakht et al, 2013) [13] recommended a modified version of ant colony

optimization (ACO), which exploits an effectual method to overcome the local optimum. A genetic algorithm based novel approach [14] has been improved to tackle MTSP. (Larki and Yousefikhoshbakht, 2014) gave an efficient evolutionary optimization approach [15], which includes the composition of modified imperialist competitive algorithm and Lin-Kernigan heuristic. A new steady-state grouping genetic algorithm (GGA-SS) (Singh & Baghel, 2009) has been built for MTSP. A genetic algorithm utilizing new crossover operator known to be two-part chromosome crossover (TCX) [16] (Yuan et al., 2013) has been recommended for solving MTSP [17]. (Sarin et al, 2014) studied the multiple asymmetric travelling salesmen problem with and without effect of precedence constraints [18]. (Venkatesh and Singh, 2015) presented two meta-heuristics such as artificial bee colony (ABC) and invasive weed optimization (IWO) algorithms to tackle MTSP [19]. (Wang et al, 2015) proposed an improved non-dominated sorting genetic algorithm II (NSGA-II) by applying the set of experience of knowledge structures (SOEKS) to tackle MTSP [20]. (Bolanos et al, 2016) developed an effective genetic algorithm (GA) to solve MTSP [21]. (Changdar et al, 2016) studied the solid MTSP in the fuzzy environment and proposed a hybrid algorithm based genetic and ant colony optimization approach [22].

There is wide research on TSP, including TSP with time windows [23], TSP with minimum ratio [24]. Most of the existing solutions consider different constraints, whereas finding a minimum Hamiltonian cycle. At present, the approaches of TSP are divided into exact and approximate algorithms. The former mostly includes dynamic programming, branch and bound, [25], integer linear programming, etc. But, if the scale of the TSP becomes too large, its overall computational time and solution space will increase exponentially.

By biological activities or natural phenomena Inspiration, some well-known heuristic algorithms have been developed to solve large-scale TSPs, including ACO, PSO, and GA. There are important research opportunities in the improvement of such

heuristic algorithms and their combination. [26] suggest two new crossover operators to improve the global ergodic property of GA, which is a better key for classical TSP, but not for complex TSP with multiple constraints. [27] present an improved dynamic programming algorithm to deal with large-scale data, used as crossover and mutation operator in GA. [28] integrate K-means algorithm, [29] with the greedy algorithm and Lin Kernighan's algorithm [30] to create an improved solution for large-scale TSP. Though, this method is significantly affected by the scale of the subset partition.

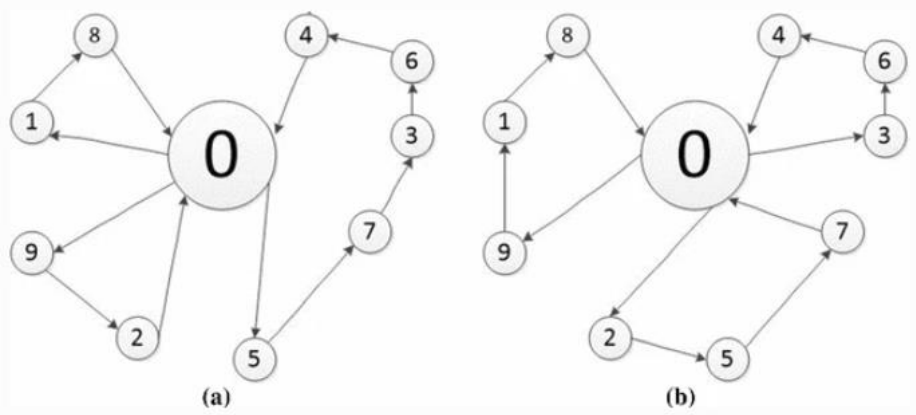


Figure 1: (a) MTSP, (b) MTSP with balance of workload

2. Banking system:

The banking system is the union of a large number of companies or entities together. They carry out their specific occupation of raising funds and lending resources in the financial and economic markets [1]. A network of COMMERCIAL BANKS and other more dedicated BANKS (such as INVESTMENT BANKS, MERCHANT BANKS, SAVINGS BANKS) receives deposits and savings from the general public, companies, and other institutions, and provide money transfers and other types of financial services for clients, operating loans and credit facilities for borrowers and investing in corporate and government securities. The banking system is part of a wider financial system and has a major impact on the country's "monetary economy". Bank

deposits occupy a central place in the supply of funds in the countries, and thus the banking system is closely regulated by the monetary authorities [2].

2.1. Types of Banking Systems:

The group of banks in the economic system is compatible with the banking or banking system. However, there are various types of banks that are interested in the sector to which this entity is oriented and the size of its activity. Here are five various types of banking systems, which are currently used all over the world:

2.1.1. Private banking:

It is a highly professionalized and management global of a client's assets. It aims to meet the investment, financial, wealth, and tax planning needs of individuals or groups of families with high equity. Private banking is therefore devoted to financial counsel and asset management. Therefore, many variables are considered, for which it is crucial to make the best profile of the client [2].

2.1.2. Home banking:

It is called Home Banking Services for all of those resources, tools, and allocations that aim to bring banking services closer to customers as possible. Through this, we can find several forms of banking services depending on the connection routes.

Thus, through digital applications, like as online banking, through telemetric means, through the telephone to execute various operations and checks, digital banking, which is a wider term that gathers all the above, [2].

2.1.3. Branch banking:

Branch Banking is a system where banking business is performed on by a single bank with a network of branches across the country and their offer. The bank will have a main office in one city and branches in different portions of the country. The manager of the branch in accordance with the regulations and policies of the head office guides the affairs of the branch. Each bank is a single entity owned by the shareholder's group

and managed by a group of directors. A bank can decide to create a branch banking organization [2].

2.1.4. Mixed Banking:

Mixed banking is a banking system where a bank gathers both investment banking as well as deposit banking. This means the bank will supply short-term loans for trade, commerce, and long-term finance for industrial units. Whereas this type of banking encourages rapid industrialization, the mixed banking system decreases the liquidity of funds of commercial banks [2].

2.1.5. Fractional Reserve Banking:

Fractional reserve banking is a system of banking in which banks save a portion of their clients' deposits in reserves.This portion is known as the ratio of cash. Under a fractional reserve banking system, banks are not required to keep 100% of their clients 'deposits in their reserves. In this manner, they can lend the bit of the deposits that they are not compelled to keep in reserves, which permits them to obtain gains and remunerate the deposits. This system presupposes that depositors will never pull out all their money at the same time. Fractional reserve banking permits a phenomenon named a bank multiplier to happen.

2.2.Function banking:

Banking was defined as "Accepting for the purpose of lending & investment, of deposit of money from the public, repayable on demand order or otherwise and withdrawable by cheque, draft or otherwise". Banking means dealing business with a bank such as depositing or withdrawing funds or demand a loan see **figure 2**.

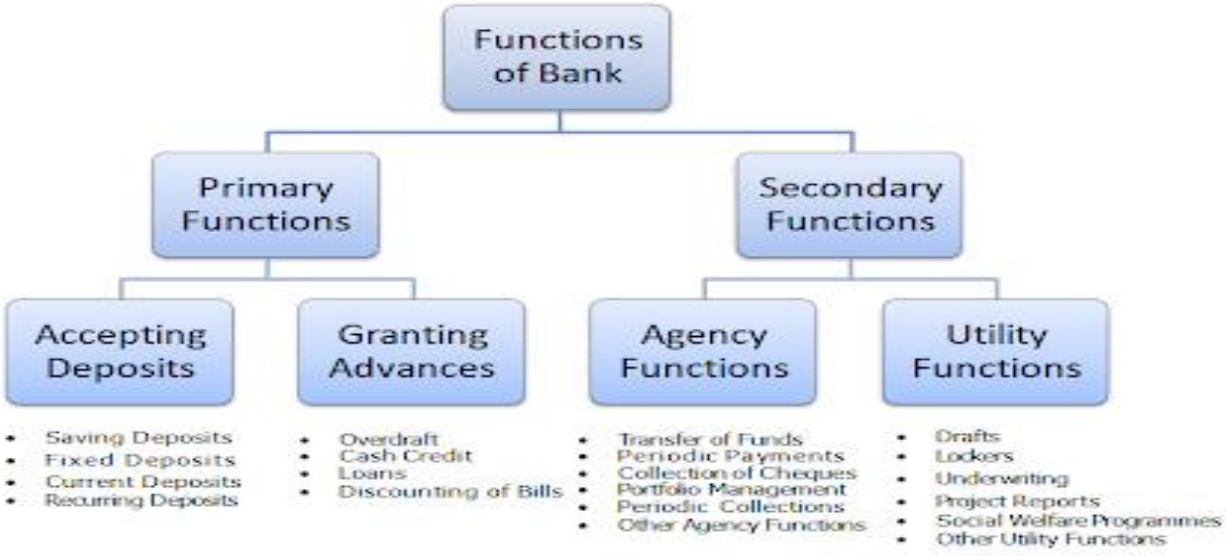


Figure 2: flowchart of functions banks in briefly.

2.2.1. Fundamental Functions of Banks:

The fundamental functions of a bank are also called as banking functions. They are the major functions of a bank [3]. These fundamental functions of banks are clarified below:

2.2.1.1. Accepting Deposits:

The bank gathers deposits from the public. These deposits can be of varieties forms, such as:

- Deposits for Saving.
- Deposits for Fixed.
- Deposits for Current.
- Deposits for Recurring.

2.2.1.2. Granting of Loans and Advances:

The bank provides loans to the business community and other individuals of the public. The ratio charged is greater than what it pays on deposits. The difference in the rates of interest (rate of lending and the rate of deposit) is its profit.

The kinds of bank loans and advances are:

- Loans
- Discounting of Bill of Exchange
- Overdraft
- Cash Credits

2.2.2. Secondary Functions of Banks:

The bank executes a number of secondary functions, also named as non-banking functions. These significant secondary functions of banks are clarified below:

2.2.2.1. Functions of Agency:

The function of the bank as an agent of its customers.

The bank performs a number of agency roles which includes:

- Funds Transfer
- Cheques Collection
- Periodic Payments
- Management of Portfolio
- Periodic Collections
- Other Agency works.

2.2.2.2. Public utility functions:

The bank also performs public utility functions, such as:

- Issuing drafts, letters of credit, and others.
- Treasury facility
- Subscription of shares
- Dealing in foreign currencies
- Project reports
- Social Care Programmers
- Other utility functions.

2.3. Banking and Appointment Scheduling:

With all the developments in the field of electronic banking, people still want a face to face experience. One area that successfully reconciled traditional and digital channels and renewed user experience is scheduling appointments. There is a positive response among clients who use the software of appointment scheduling to book meetings at local banks.

2.3.1. Clients and Schedule Appointments:

The current survey shows that scheduled appointments are the most prevalent of hiking visits in major business hours. Scheduled appointments reached their peak in the morning between 10:00 AM and 11:00 AM, and at 4:00 PM. In contrast, walking visits at that time were much lower and generally peaked from 12:00 p.m. to 1:00 p.m. (lunch hour dates). When it comes to traffic per day, walking visits were the highest on generally low traffic days (Monday, Tuesday, and Wednesday). However, they failed to exceed the number of appointments scheduled at the end of the week, when most account holders visited the local branch office [4].

2.3.2. Maximizing Appointment Scheduling:

- Take an Omnichannel Approach:

Due to their attract, account owners gladly to test new systems, like mobile apps, online, and other digital applications. But at the same time, they are reluctant to give up these that are currently using it. They want new features to add to their experience, not replace existing applications. An appointment scheduling app emphasizes the intent to provide convenience and availability with a full range of accessible channels. However, even the most receptive consumers of digital channels prefer the option to book appointments in a conventional way. It is common when they are dealing with complicated affairs and require expert advice from the finance professionals [4].

- Optimize Available Resources:

Demanding account owners to schedule appointments is less time-consuming for them. It also gives administrators the opportunity to collect data to evaluate and improve scheduling. Knowing who they are meeting beforehand, enables them to better organize available resources, so staff can better serve each client for best performance. Consequently, it allows employees to meet with more clients and facilitate operations and use resources to their maximum possibility. And ultimately that reduces costs, improves performance, and increases sales output and efficiency [4].

- Data-Driven Decision Making:

Appointment scheduling software allows employees to require a data-backed approach to service. These tools make it easy for employees to know what clients want. They also pinpoint the time after they make visits that lead to sales. Decision-making becomes data-driven and allows employees to customize each decision to suit demand. this could be done at every branch office for optimum profit across the board. the employees can maximize the success of their appointments right by leveraging the facility of appointment scheduling software [4].

Chapter 2

1. Evolutionary Algorithms:

Evolution is the theory postulating that all the various types of living organisms have their origin in other pre-existing types, and that the differences are due to modifications inherited through successive generations. Evolutionary computation is a branch of computer science concentrating on algorithms inspired by the theory of evolution and his internal mechanisms. The definition of this field in computer science is not clear, but it could be considered as a branch of computational intelligence and may be involved in the broad framework of bio-inspired heuristics.

1.1. Natural and artificial evolution:

Fundamentally, the original theories regarding evolution and natural selection were almost proposed concurrently and independently by Charles Robert Darwin and Alfred Russel Wallace in XIX century, combined with selectionism of Charles Weismann and genetics of Gregor Mendel, are accepted in the scientific community, and widespread among general public.

This theory (called Neo-Darwinism) offers the basis for the biologists: through it, the whole process of evolution is described, requiring notions such as reproduction, mutation, competition, and selection. Reproduction is the process of creating an offspring where the new copies inherit traits of the old ones. Mutation is the unpredicted alteration of a trait. Competition and selection are the inevitable strive for survival caused by limited resources environment.

The evolution process is a mechanism which progresses as a sequence of steps, some are deterministic and some mostly random [31]. Such an idea of random forces formed by deterministic pressures is inspiring, and not surprisingly, has been exploited to explain phenomena quite unrelated to biology. Important examples include alternatives conceived during learning [32], ideas striving to survive in our culture [33], or even possible universes.

Evolution may be seen as an improving process that makes raw features perfect. in fact, this is a mistake that all biologists warn us not to do. However, if evolution is seen as a force pushing toward a goal, another awful misunderstanding, it must be granted that it worked quite well: in some million years, it turned unorganized assembles of cells into wings, eyes, and other amazingly complex structures without requiring any a-priori design. The whole Neo-Darwinist paradigm may hence be regarded as a powerful optimization tool, able to produce great results starting from scratch, not requiring a plan, and exploiting a mix of random and deterministic operators.

Dismissing all complains of biologists, evolutionary computation practitioners loosely mimic the natural process to solve their problems. Since they don't know how their goal could be reached, at least not in details, they exploit some neo-Darwinian principles to cultivate sets of solutions in artificial environments, iteratively modifying them in discrete steps. Indirectly, the problem defines the environment in which solutions strive to survive. The process has a defined purpose. The simulated evolution is simplistic if not even implausible. Yet, successes are routinely published in the scientific literature. Solutions in a given stage inherit qualifying traits from solutions in the previous ones, and optimal results gradually emerge from the artificial primeval soup.

In evolutionary computation, a single candidate solution is named **individual**; the set of all candidate solutions is termed **population**, and each step of the evolution process called **generation**. The ability of an individual to solve the given problem is measured by the **fitness function**, that ranks how possible one solution to propagate its characteristics to the next generations. Most of the jargon of evolutionary computation mimics the terminology of biology. The word **genome** denotes the whole genetic material of the organism, whereas its actual implementation differs from one approach to another. **The gene** is the efficient unit of inheritance, or, practically, the smallest

fragment of the genome that may be modified during the evolution process. Genes are positioned in the genome at specific positions so-called loci, the plural of locus. The alternative genes that may occur at a given locus are termed allele.

Biologists distinguish between the genotype and the phenotype: the former is all the genetic constitution of an organism; the latter is the observable properties that are formed by the interaction of the genotype and the environment. Various evolutionary computation practitioners don’t stress such a precise distinction. The fitness value which associated to an individual is sometimes assimilated to its phenotype.

To produce the offspring for the next generation, evolutionary algorithms implement both sexual and asexual reproduction. The former is generally named recombination; it necessitates two or more participants and implies the possibility for the offspring to inherit different characteristics from different parents. The latter is called replication, to indicate that a copy of an individual is created, or more commonly mutation, to stress that the copy is not exact. In some applications, mutation takes a place after the sexual recombination. Practically no evolutionary algorithms take into account, gender; whereas, individuals don’t have distinct reproductive roles. All operators which modify the genome of individuals can be cumulatively so-called genetic operators.

Mutation and recombination introduce variability in the population. Parent selection is also frequently a stochastic process, while biased by the fitness. At each generation, the population broadens and contracts rhythmically. First, it widens then generates the offspring. Then, it shrinks when individuals are discarded. The deterministic pressure regularly takes the form of how individuals are chosen for survival from one generation to the next. This step may be termed survivor selection.

Evolutionary algorithms are local search algorithms because they only explore a defined region of the search space, where the offspring define the concept of neighbourhood. They are heuristic algorithms, as they are based on the trial and error

paradigm. They are not usually able to guarantee mathematically an optimal solution in a finite time, while interesting mathematical properties have been proven over the years.

If the present boundary of evolutionary computation may seem unclear, its inception is much more hazy. The field does not have a single recognizable origin. Some scholars identify its starting point in 1950, when Alfred Turing reported the similarities between learning and natural evolutions [34]. Others pinpoint that the inspiring ideas appeared in the end of the decade [35] [36], even though, the lack of computational power significantly impairs their diffusion in the broader scientific community. More commonly, 1960s was the birth of evolutionary computation with the appearance of three independent research lines, namely: evolutionary programming, genetic algorithms, and evolution strategies. Despite the slight disparity, the pivotal importance of these researches is unquestionable.

1.2. The classical paradigms:

The most common concept in evolutionary computation is Genetic algorithm. It is abbreviated as GA, and it is so popular that in the non-specialized literature it is often used to denote any kind of evolutionary algorithm. The reputation of the model is related to the name of John Holland and his 1975 book, however the methodology was used and described during the course of the previous decade by several researchers, including many Holland own students. Genetic algorithms have been proposed as a step-in classification system, a technique also proposed by Holland. Though, it may be maintained that they have been exploited more to study the evolution mechanisms itself, rather than solving actual problems.

Quite basic test benches were used to evaluate various strategies and schemes, as trying to set a number of bits to a specific value. Many variations have been proposed. So, even in this pioneering epoch, is not sensible to describe a canonical genetic algorithm.

1.3. Genetic programming

The last evolutionary algorithm outlined in this introduction is genetic programming, abbreviated as GP. Whereas µGP shares more with it its name than its essence, the approach presented in this research owes a deep debit to its underlying ideas. Genetic programming was developed by John Koza, who described it after applying for a patent in 1989. the methodology's ambitious goal is to create computer programs in a fully automated way, using Neo-Darwinism as an optimization tool. The original version was written in Lisp, an interpreted computer language dating back to the end of the 1950s. The Lisp language has a unique ability to handle fragments of code as data, allowing a program to build up its subroutines before they are evaluated. Everything in Lisp is a prefix expression, except variables and constants. lisp programs were Genetic programming individuals; thus, they were prefix expressions too. Since the Lisp language is as flexible as inefficient, in the following years, researchers switched to alternative implementations, generally using compiled language. Indeed, the need for computational power and the quest for efficiency have been constant pushes in the genetic programming research since its origin. Although the distinction between an expression and a program was subtle in Lisp, it became more apparent in later implementations. Most of the algorithms presented in the literature clearly tackle the former, while hardly applicable to the latter

2. Combinatorial Optimization Problems:

Combinatorial Optimization Problems are very common in industrial processes and planning activities. They are problems where a solution is composed by a set of fundamental discrete decisions or assumptions. Every decision may influence the global cost and the feasibility of the solution. The trivial way to solve a combinatorial optimization problem is to list the elements of the corresponding feasible solutions set and pick up the best one. However, because of the combinatorial nature of the

considered problems, in real cases the number of solutions to be enumerated (feasible or unfeasible) for a given problem is intractable even for very powerful computers.

To deal with the difficulty in solving Combinatorial Optimization Problems, some techniques have been proposed:

• Branch and Bound approaches in which the solution space is systematically divided, and its subsets of solutions are evaluated on the objective function value, according to their limits;

• Heuristic methods, where the problem is solved through the application of experience-based techniques. When these techniques dirive from other generic or natural problems rather than the original problem, we call them Meta-heuristic;

• Methods based on Integer/Mixer Programming and Hybrid methods that combine some of previously mentioned approaches.

The most common optimization problems are:

- Fair Layout Optimization Problem (FLOP).
- Bin Packing Problem with Conflicts (BPPC).
- Generalized Traveling Salesman Problem (GTSP).

2.1. Fair Layout Optimization Problem (FLOP):

Fairs and expositions are now essential tools for providing industrial exhibits and demonstrations. According to the International Association of Fairs & Exposition (IAFE), over 3 200 fairs are currently held in North America each year. The Association of the European Major Exhibition Centres (EMECA) reports that more than 36 million visitors and upwards of 330 000 exhibitors take part in approximately 1000 EMECA exhibitions. A relevant logistical issue in the organization of a fair concerns the way in which the stands must be placed in the exhibition space so as to satisfy all constraints (security, ease of access, services, to mention just a few) arising in this kind of event,

and to maximize the revenues coming from the exhibitors. Such issue is frequently manually solved by the organizers on the basis of experience and common sense.

We are given:

- a non-convex two-dimensional surface that may contain holes (exhibition area);
- an axis-aligned minimal rectangle which encapsulates the exhibition area and touches it on the borders;
- an unlimited number of identical rectangular stands;
- a minimum width needed for the aisles.

The Fair Layout Optimization Problem we consider consists in orthogonally allocating the maximum number of stands, without rotation, to vertical strips parallel to the vertical edges of the rectangle, by ensuring left and/or right (see below) side access to each stand.

Concerning the access constraint, we will consider two variants of the problem, that are frequently encountered in practice:

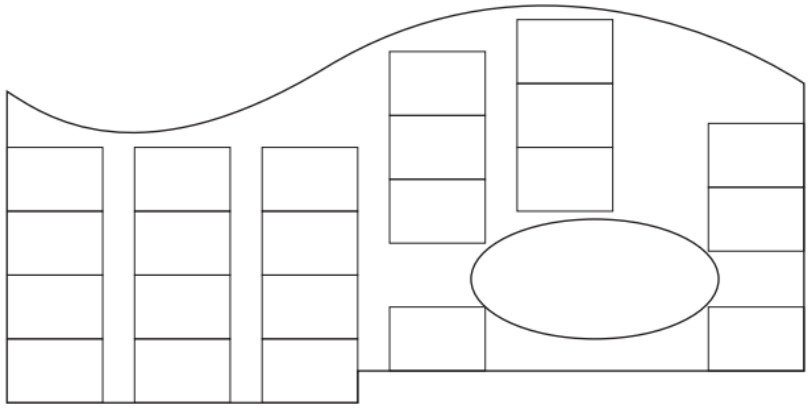


Figure 3: A single strip solution.

- FLOP1: it is required that each stand (i.e., each strip) can be accessed from both sides, as in the solution depicted in **Figure 3**;
- FLOP2: it is allowed to place pairs of strips with no space between them, thus obtaining stands that can be accessed from one side only.

2.2. Bin Packing Problem with Conflicts (BPPC):

In the Bin Packing Problem with Conflicts (BPPC), we are given a set $V = \{1, 2, \ldots, n\}$ of items, each item i having a non-negative weight wi, and an infinite number of identical bins of weight capacity C. We are also given a conflict graph $G = (V, E)$, where E is a set of edges such that $(i, j) \in E$ when items i and j are in conflict. Items in conflict impossible to be assigned to the same bin. The goal of the BPPC is to assign all items to the minimum number of bins, ensuring that the total weight of the items assigned to a bin doesn't exceed the bin weight capacity and that no bin contains items in conflict.

The BPPC is important because of the high number of real-world applications, and because it generalizes other important problems in combinatorial optimization. Some BPPC real-world applications include examination scheduling [37], the assignment of processes to processors and the load balancing of tasks in parallel computing [38]. Other applications concern particular delivery problems, such as food distribution, where some items cannot be placed in the same vehicle [39].

2.3. Generalized Traveling Salesman Problem (GTSP):

The traveling salesman problem (TSP) is a well-studied combinatorial optimization problem with applications ranging from routing to scheduling. It can be classified based on several factors. For example, the TSP can be classified based on the objective function: minimizing the tour cost, maximizing the profit, or a combination of both; and it can be classified based on vehicle capacity, etc.

One real-life application of this problem can be in planning a tour for doctors in an underdeveloped region or at a time of crisis to serve the maximum number of patients. In addition, in health economics, registrars visit different hospitals to collect patient documents while maximizing the total number of served hospitals with a limited budget [40]. In disaster management problems, an objective can be to maximize the

number of served injured people by trained medical staff with limited time and budget constraints who must travel to different locations affected by the disaster, assuming that injured people could be transferred to locations on the route [41].

The traveling salesman and vehicle routing problems have been well studied. Several extensions of each problem can be modelled using different objectives and/or additional constraints.

In a regular TSP, the objective is to find a tour that covers all cities while minimizing the total cost of traveling. A variation of the TSP that has a profit associated with each node can be seen as a bi-criteria problem, where the objective is to maximize the collected profit from visiting each node while minimizing the cost of traveling. The solutions include a set of non-inferior feasible solutions, which none of the objectives can improve except by deteriorating the other objective, the TSP can be different based on the following:

- Number of travellers: If there is more than one traveller who can serve the nodes, then more than one tour can exist in the problem. This problem is called a multiple traveling salesman problem (MTSP). Some variations of this problem are where the number of travellers is not fixed. In the case of a varying number of travellers, there is a cost associated with each route (i.e., to each traveller). If there is a capacity associated with each traveller, then the problem is called a vehicle routing problem.
- Number of depots: If there is more than one depot having a fleet of vehicles, then the problem is called a multiple depot vehicle routing problem (MDVRP). In one variation of the problem, vehicles departing from a depot must return to the same depot (i.e., this is a fixed-destination problem). Vehicles departing from a depot can return to any other depot in a different variation (i.e., a non-fixed-destination problem).
- Time windows: The TSP with time windows has a time window associated with each node, which requires that each node be served during its corresponding time interval.

The Generalized Traveling Salesman Problem (GTSP) is a variant of the Traveling Salesman Problem (TSP). We are given an undirected graph G = (V, E), where $V = \{1, \ldots, n\}$ is the set of vertices and E is the set of edges, each edge (i, j) having an associated cost c_{ij} . The set of vertices V is partitioned into m clusters $V_1, \ldots, V_m$. GTSP is to find an elementary cycle visiting at least one vertex for each cluster, and minimizing the sum of the costs of the travelled edges. If a directed graph is considered, the problem is denoted as Asymmetric GTSP (AGTSP). We focus on the commonly considered version of the problem, i.e. the so-called Equality GTSP (E-GTSP), in which the cycle must visit exactly one vertex per cluster. Both the GTSP and the E-GTSP are generalizations of the TSP: we obtain the TSP in the particular case where each cluster is composed by just one vertex. Consequently, both problems are NP-Hard since the TSP is NP-Hard.

3. Genetic Algorithms:

Proposed in 1975 by J. Holland [42], the genetic algorithm (GA) is an optimization method that was inspired by the evolution concept. GA does not guarantee to find the optimal solution of the problem, however there is empirical evidence [43] that solutions are between acceptable levels, in a competitive time with the rest of combinatorial optimization algorithms, i.e. simulated annealing, sequential search methods, hyper-climbing, etc. Burjorjee offered an explanation for the remarkable GA adaptive capacity [44]. Furthermore, Burjorjee presents evidence that strongly proposes that GA can implement hyper-climbing extraordinarily efficiently for complex optimization problems. Moreover, GAs does not make any presumptions about the search space for the optimization problem. These are some of the reasons why GAs have been applied to solve a wide range of engineering and scientific optimization problems [43].

To understand GA functionality, it is better to first explain how the optimization problem variables must be encoded and then recombined. The theoretical foundation of the GA requires the optimization problem variables to be encoded into a string of either (1) binary bits, (2) real numbers or (3) characters. Each bit, real number, or character in GA, the string is called gene or parameter, and they form as an ensemble a chromosome, also called string or individual. In this work we refer to the variables in the string as parameters and the ensemble of parameters as chromosome. Every different combination of the parameters in the chromosome represents a different variable in the optimization problem search space.

For example, let us consider a simple case in which we want to find the X and Y values to maximize the next equation: $sin(X^2)log(XY)$. One possible encoding solution is to have a string of two real numbers (two parameters forming a chromosome) for X and Y . Chromosomes with different X and Y values represent a different variables to the problem.

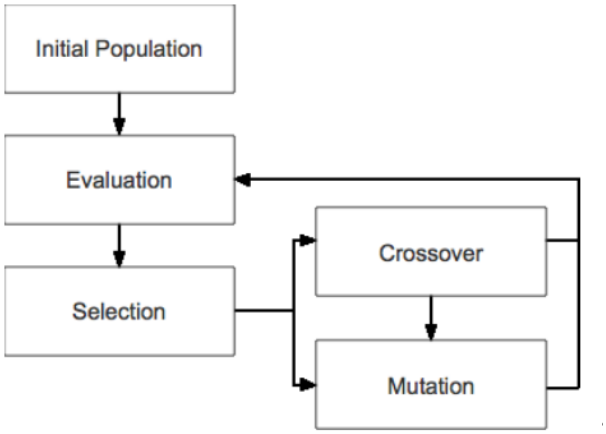
The recombination or crossover needs two or more chromosomes (parents) to generate a new chromosome (offspring). The objective of the crossover operator is to find a new set of parameters that produces an optimum value in the variables to the optimization problem.

Once variable encoding is decided, the first step in GA is creation of a random initial population (set of variables with random encoded parameter values to form a population of n chromosomes). Next, each chromosome in the population is evaluated to have a measurable value that shows how well the set of parameters perform as solution for the problem at hand.

The GA then executes the selection operator to choose the chromosomes to be recombined. The selection operator offers more opportunities to the chromosomes that performed the best in the optimisation problem; however, it is important not to discard weaker chromosomes completely, in order to avoid premature convergence. Finally,

crossover is applied to the selected chromosomes. Selection and crossover operators try to preserve the combination of the parameter values that obtained a better result for the optimization problem.

The GA repeats the selection and crossover operators in order to provide a better set of parameters (chromosome) after every iteration. **Figure 4** shows the GA procedure.



In GA terminology, the optimisation function is called objective function. Moreover, the value that indicates the appropriateness of the chromosomes is known as fitness value and it is calculated by the the fitness function.

The selection, crossover and mutation operators are defined in more detail in the following subsections.

3.1. Selection:

The selection operator selects chromosomes from the entire population for later recombination. The most frequently used selection algorithms are tournament selection and roulette wheel.

Tournament selection algorithm consists of taking k random from population chromosomes. The chromosome having the highest fitness value is then used as parent for the crossover. The tournament size k determines the probability of selection the best

chromosome from the population, known as selection pressure. Weak chromosomes have more probabilities to be selected when k is small (low selection pressure), in the extreme case when k = 1 the chromosome selection is a random process. Moreover, if the value of k is close to the number of chromosomes in the population the probability of selecting the best chromosome increases. Stone and Smith [45] observed that high selection pressure causes low diversity in the population. Thus, the value of k is GA critical factor which depends on the number of chromosomes in the population.

Tournament selection algorithm adds k as an extra parameter to the GA, therefore roulette wheel is often preferred. In roulette wheel selection, the chromosomes have a probability p of being chosen depending on their relative fitness value. For a chromosome i in the population, the selection probability pi is calculated by equation:

$$p_i = \frac{fitness_i}{\sum_1^n fitness_k}$$

Where $fitness_i$ is the fitness value for chromosome i, and n is the number of chromosomes in the population.

The algorithm may be seen as a real roulette wheel, in where after the wheel spinning, the pivot indicates the selected pattern. **Figure 5** an example of a population of four chromosomes and their probability to be chosen.

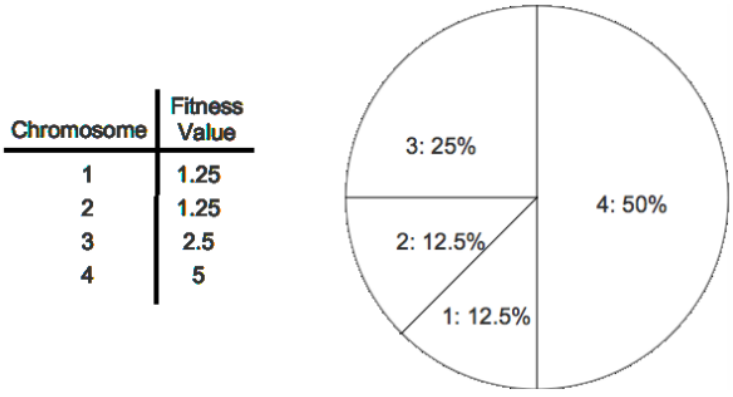


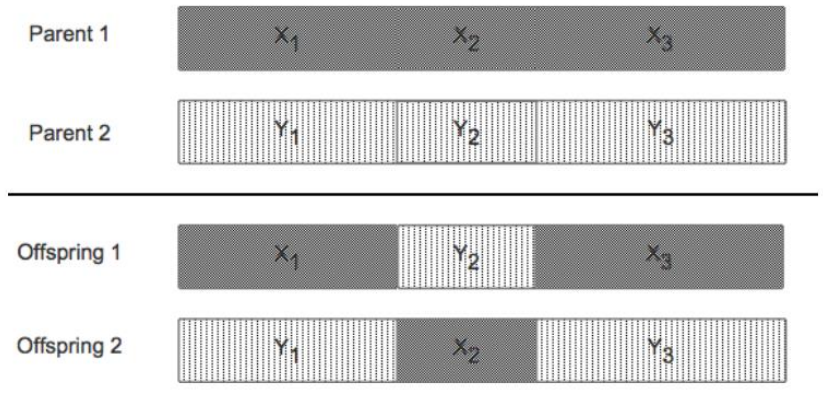
Figure 5: Roulette wheel fitness-based selection.

3.2.Crossover:

As in every optimization algorithm, in order to improve the current value of criteria function, a new set of parameters must be chosen. In a GA the crossover operator was inspired by mix of genes in reproduction. In this operator, the strings of parameters representing the chromosomes for the two parents are cut and mixed to generate the new offspring. There are three different crossover techniques: one-point, two-point and uniform. One-point crossover choses a random point in the genome from both parents and swap them to generate two offspring **Figure 6**.

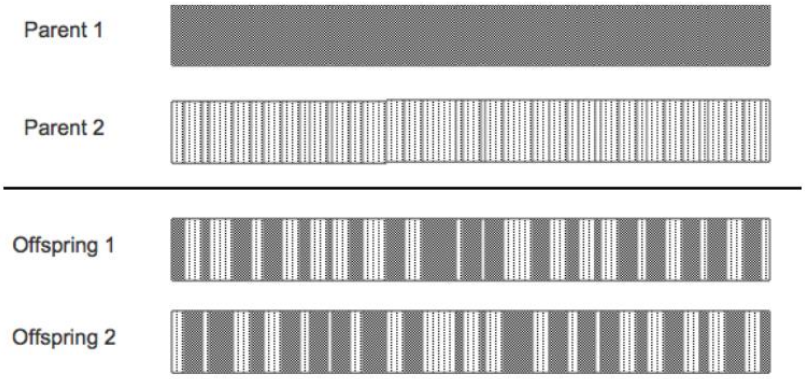
Figure 6: One-point crossover.

Two-point crossover selects two random splitting points from both parents which generate three splices, **Figure 7** shows this process. Uniform crossover varies from the last two



techniques, her ıe for exchange with a probabi ıg p is 0.5 then

around half of the chromosome for the new offspring belongs to parent one and the other half to parent two, **Figure 8** shows this technique.



3.3.GA Simplex:

GA Simplex was proposed by Seront and Bersini [46] proposing that the search process can be effectively improved by taking into crossover multiple parents. This techniques takes into account the relative fitness of the parents. It works on three chromosomes of the population i_1, i_2, and i_3 to generate a new offspring i_4. If $i_{1.fitness} \geq i_{2.fitness} \geq i_{3.fitness}$, the algorithm is as follows:

Algorithm 1 GA Simplex Algorithm

for each $k-th$ parameter in chromosome **do**
  if $i_{1k} = i_{2k}$ **then**
    $i_{4k} \leftarrow i_{1k}$
  else
    $i_{4k} \leftarrow negate(i_{3k})$
  end if
end for

3.4.Mutation:

Mutation is an optional operator used to discover a wider solution space in order to avoid converging in a non-adequate local minima. The mutation alters the value of all parameters in an offspring's chromosome with a probability p_m. The probability p_m is fixed throughout the whole GA implementation and it should be small enough to just

slightly alter the chromosome as otherwise GA would behave very much as a random search. However, the mutation operator could be improved to cover a wider solution space while avoiding getting stuck into a local minima. This enhancement is done by having an evolutive mutation. In evolutive mutation p_m is no longer fixed, instead pm for the offspring increases along with the similarity of the parents. For example, supposing xxxYYxx and xxxYYxY are the encoded variable for both parents. The resulting p_m for the offspring would be high as the string of parameters only differ in one parameter.

One common technique to determine the similarity between the parents is to calculate the distance between them. Whitley et al. [43] calculated the Hamming distance for binary parent strings to calculate p_m.

4. Bootstrap Aggregation (Bagging):

Bootstrap aggregation or bagging was first proposed by Breiman [47]. Bagging is a method that produces multiple predictors with different structures or models in order to get a new predictor referred to as a bagged predictor. Consider a learning set L consisting of xi input samples and the corresponding set of y_i targets, where i = 1, ..., n. The standard, single predictor approach uses the set L to train a single prediction model. Bagging however, contains of using a set of m learning sets, or bootstrap samples L_k, k = 1, ..., m obtained by random independent sample draws with replacement from the original set L. The bootstrap samples L_k are used to train m different predictor models. As a results, for each input bootstrap sample, there is one prediction model. If the predictor is a regression problem where the targets y are numeric, the final prediction $y = pred(L_k, x)$ is the average of y_k over m. When using classification predictors, the bagged output is defined as the majority vote among the bootstrap predictors $pred(L_k, x)$ Figure II.10 shows the bootstrap principle. Studies done by Breiman show that bagged predictors can substantially enhance the accuracy when using unstable predictors, where a small modification in the learning set L

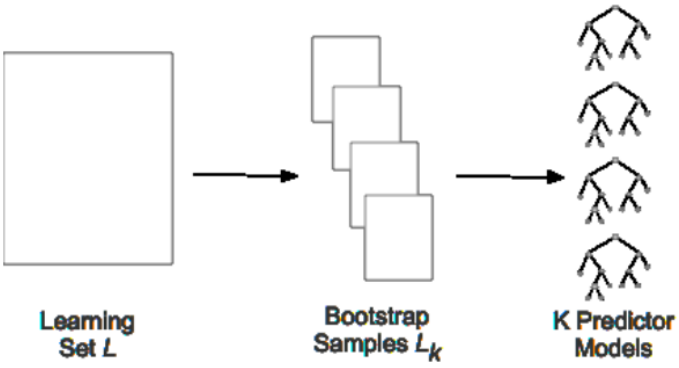


Figure 9: Illustration of bagging process.

considerably alters the predictor model [47],[48]. The improvement in the accuracy resulting from using bagging technique depends greatly in the stability of the predictor. Research has proved that decision trees, naïve Bayes and ANN are unstable classifiers [49], [48]. Grandvalet published a study shows that bagging applied to DTs also improves the estimation process stability. Experimental and theoretical results have shown that bagging reduces the non-linear variation by stabilizing the estimation process [49].

4.1. GA-Bagging:

As explained in the last section, the bootstrap samples are obtained by random independent sample draws with replacement from the original set L. The new method proposed in this work uses a GA to optimize the composition of each of the training data subsets L_k. Fu et al. [50] used a GA to select a single optimal training subset. The rationale behind their idea is to use a portion of the dataset to construct a robust prediction model when it not feasible to use the entire dataset due to the big size. Moreover, they state that even when the dataset is small, the best model might not be found using the entire dataset. This approach aims to ignore the outliers and noise samples in the learning set. They demonstrate that the performance of the tree trained with the learning subset found by the GA achieved better results than the tree using the complete training set.

In this work, we apply a similar GA approach to select m optimal learning subsets that were used to built m models to form the bagged predictor. The principle of the GA Bagging System is shown in Fig. II.11. Implementation details are described in

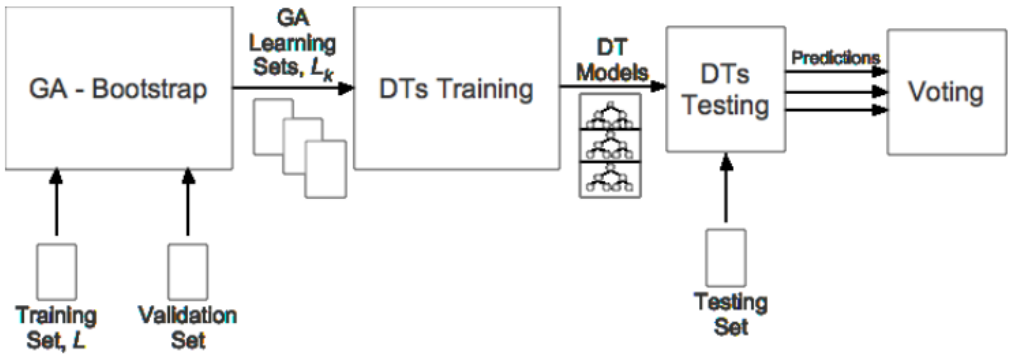


Figure 10: The principle of the GA bagging procedure.

Chapter 3

This chapter presents discuss the research concept and the research process which will explain about the presented heuristic method, the testing of heuristic with a sample of problem the comparison and selection of the searching for answer starting from heuristic method as well as the improvement of answer.

Our program describes a method that assigns customers to employees by partitioning them into employees, then by genetic algorithm the customers within each formed employee are routed using the cheapest route costs.

First and foremost, we select our model's parameters:

Population size = the number of routes generated from each employee client list.

Elite size = the small proportion of the fittest routs unchanged copied to the next generation.

Number of clients = simply means, our number of clients.

Number of employees = bank employees

The first process is run for one time and the second is processed iteratively a finite number of times (generations) to improve the solution quality. The search begins by introducing the clustering customers according to their distances from bank to each other.

1. First process:

   1.1. Clients demonstration:

We first create a clients' list that will allow us to handle our problem. These are simply our (x, y) coordinates. This can be done using a random initialization points in the 100×100 coordinate-system measure.

```
clientList = []
for i in range(num_clients):
    clientList.append((int(random.random() * 100), int(random.random() * 100)))
```

1.2. Initial graph plot:

Using tsp package (traveling salesman problem), we calculate the shortest path passing through all the clients including **Bank** with coordinates (**0**, **0**):

```
t = tsp.tsp(clientList)
clientlist = [clientList[i] for i in t[1]]
```

Then we plot our initial graph using this code:

```
x, y = zip(*s)
plt.figure(figsize=(10,10))
plt.plot(x, y, '--ob')
plt.legend(loc= 'best')
plt.annotate('BANK', (0, 0))
plt.title('Initial clients distribution')
plt.savefig('graph.png')
plt.show()
```

1.3. Distances calculation:

Within the Clients' list, we form a distance calculation graph matrix (making use of the Pythagorean theorem:

```
def euc_dis(a,b):
    return np.sqrt((b[0]-a[0])**2+(b[1]-a[1])**2)

def graph(v):
    a = np.zeros((len(v), len(v)))
    for i in range(len(v)):
        for j in range(len(v)):
            if v[j][0]==v[i][0]:
                a[i,j] = round(abs(v[j][1]-v[i][1]), 2)
            elif v[j][1]==v[i][1]:
                a[i,j] = round(abs(v[j][0]-v[i][0]), 2)
            else:
                a[i,j] = round(euc_dis(v[j],v[i]), 2)
    return a
```

1.1. Clients clustering over employees:

Based on tsp shortest path, we devise clients over employees including BANK point to all clusters, this can be done by:

```
def divide_client_for_employee(clientlist, em):
    n = math.ceil(len(clientList)/em)
    chunks = [clientList[i:i + n] for i in range(0, len(clientList), n)]
    for i in range(len(chunks)):
      chunks[i].insert(0, (0,0))
    return chunks
```

2. Second process (genetic algorithm):

The second process is based on two parameters:

Generations = iterations of the process

Mutation rate = the proportion of

2.1. Population creation:

Selecting the matrix of population for each employee randomly, which contains of " **Population size**" routes possible formed by clients.

2.2. Population ranking:

Ranking the routes of population of each employee from lower distance to higher, depending on distances mentioned in graph matrix.

2.3. Mating pool selection:

Selecting the parents that will be used to create the next generation. The method used here is distance proportionate selection; we keep the first "**Elite size**" routes from the ranked population matrix of each employee, then we fill the rest roads depending on those who have high proportion of low distance.

2.4. Breeding:

With our mating pool created, for each employee, we can create the next generation in a process called "crossover", we save the same "**Elite size**" routes that they have the lowest distances from the mating pool and then we apply crossover breeding to the roads that chosen randomly from the mating pool.

2.5. Mutation:

We'll use a method called "swap mutation". This means that, with specified low probability "**Mutation rate**", two cities will swap places in our children matrix. We'll do this for one individual to avoid local convergence by introducing novel routes that will allow us to explore other parts of the solution space.

In this step we consider the output matrix as a population and then we apply the same algorithm many times "**Generations**".

Chapter 4

In this chapter we introduce the experiments and results obtained by our model.

1. First experiment:

Let's say we have 12 clients, 3 employees, which means each employee will have a list of 4 clients, we will try to implement our algorithm based on these parameters:

Population size = 10, Elite size = 5, Mutation rate = 2, Generations = 100

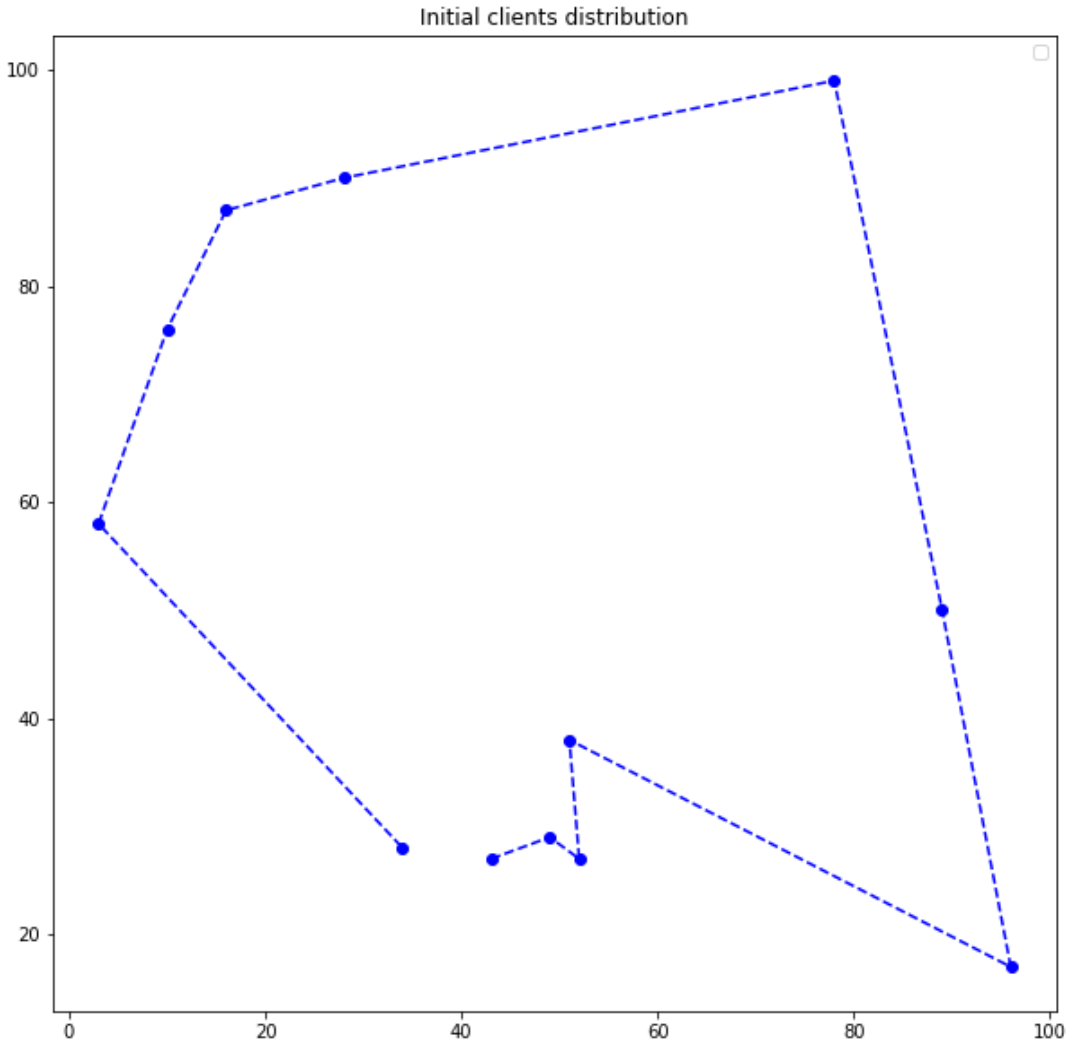


Figure 11: Initial banking route

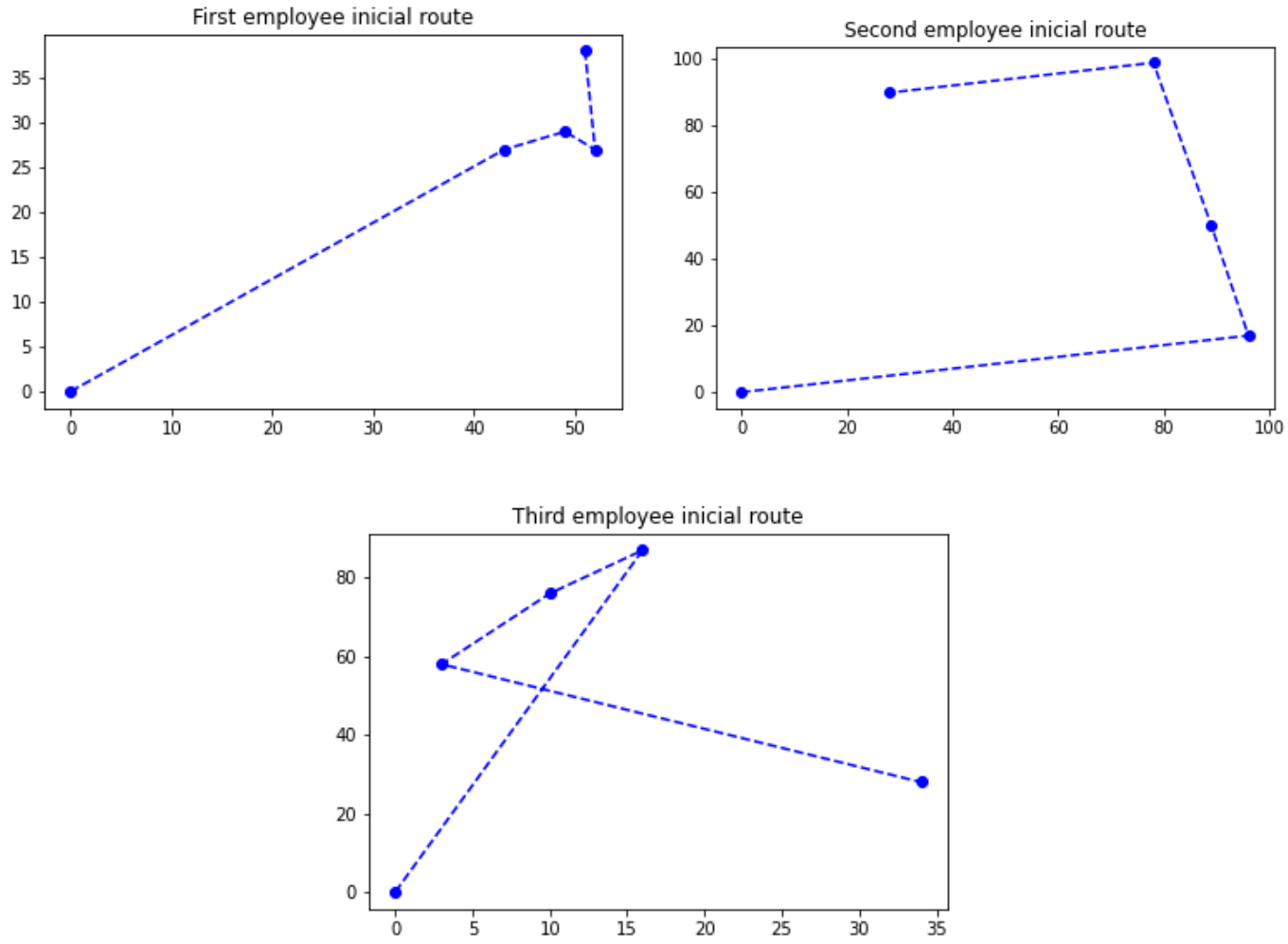


Figure 12: Employees inicial routes

	1st employee	2nd employee	3rd employee
initial distance (m)	135.79	326.49	200.11

Table 1: Employees' initial distances

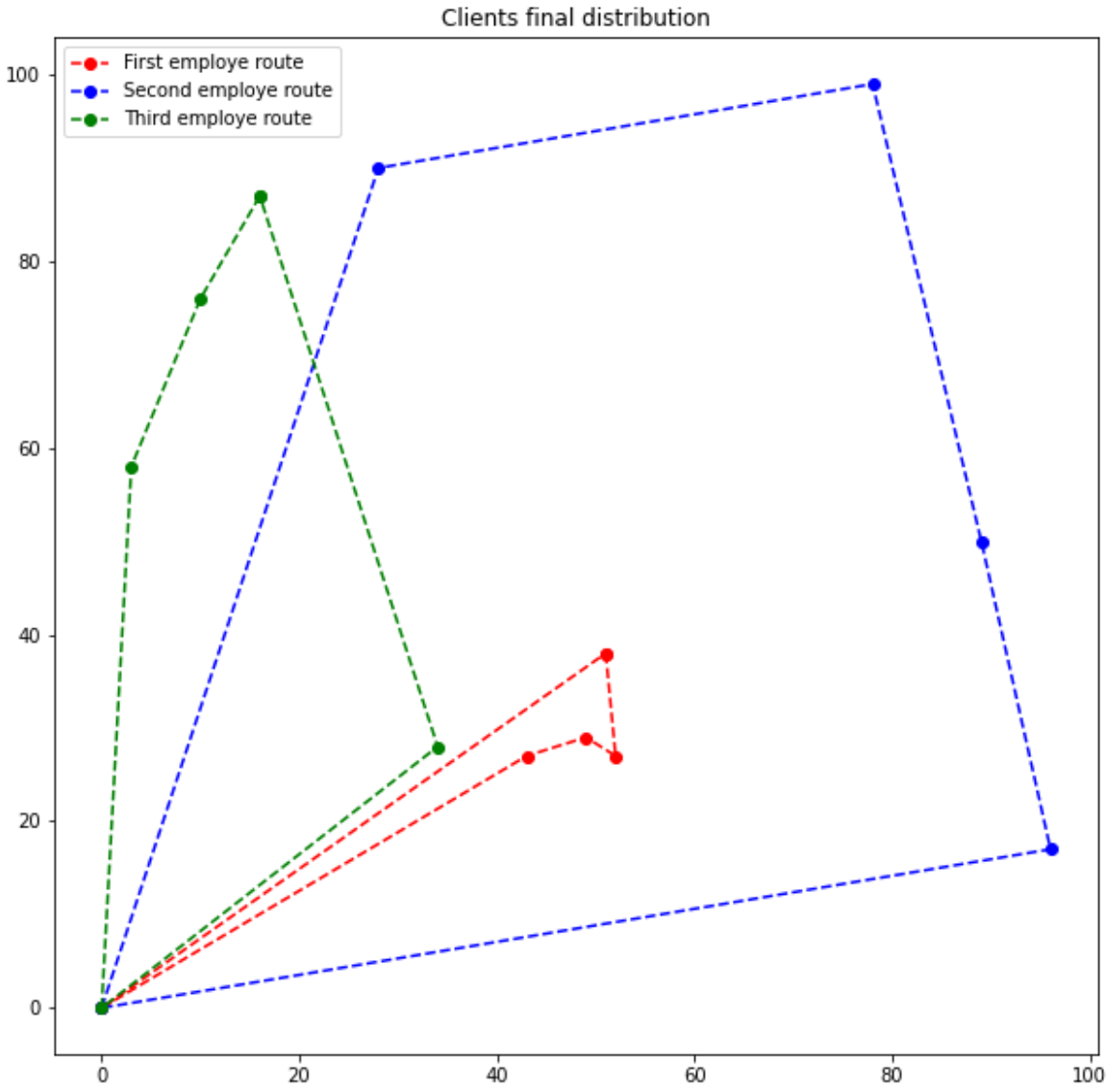


Figure 13: Final employees' routes

1.1.Results:

Execution time 0.16 seconds
1st employee final shortest route: [(51, 38), (52, 27), (49, 29), (43, 27), (0, 0)]
2nd employee final shortest route: [(0, 0), (28, 90), (78, 99), (89, 50), (96, 17)]
3rd employee final shortest route: [(16, 87), (10, 76), (3, 58), (0, 0), (34, 28)]

Distance curves:

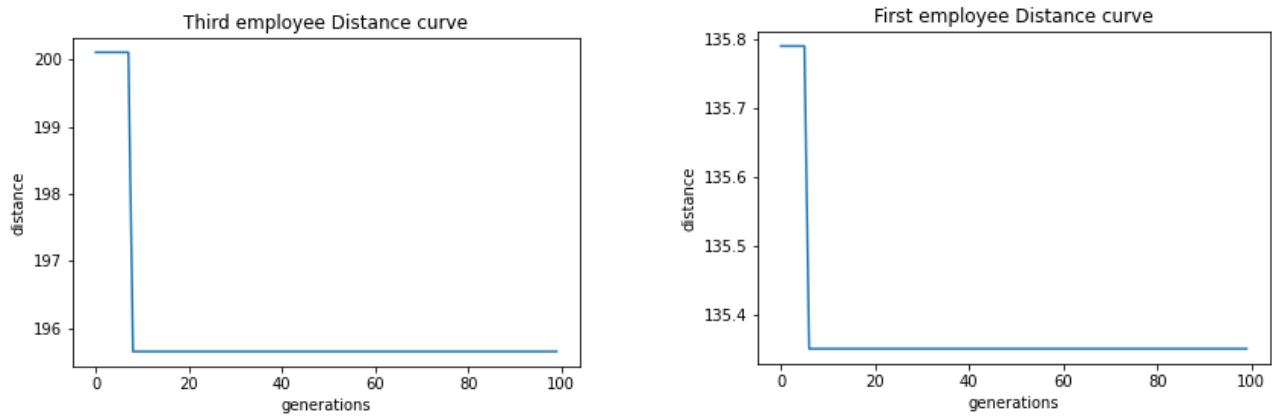


Figure 14: 1st experiment employees' curves

The second employee distance stays the same.

Suppose that each 1 meter takes 2 seconds time, results are summarized in this table:

Employee	Final distance (m)	Time (s)
1st employee	135.35	270.7
2nd employee	326.49	666.14
3rd employee	195.65	391.3

Table 2: first experiment summary

2. Second experiment:

Let's say we have 20 clients, 3 employees, which means each employee will have a list of at least 6 clients, we will try to implement our algorithm based on these parameters: Population size = 10, Elite size = 3, Mutation rate = 2, Generations = 500

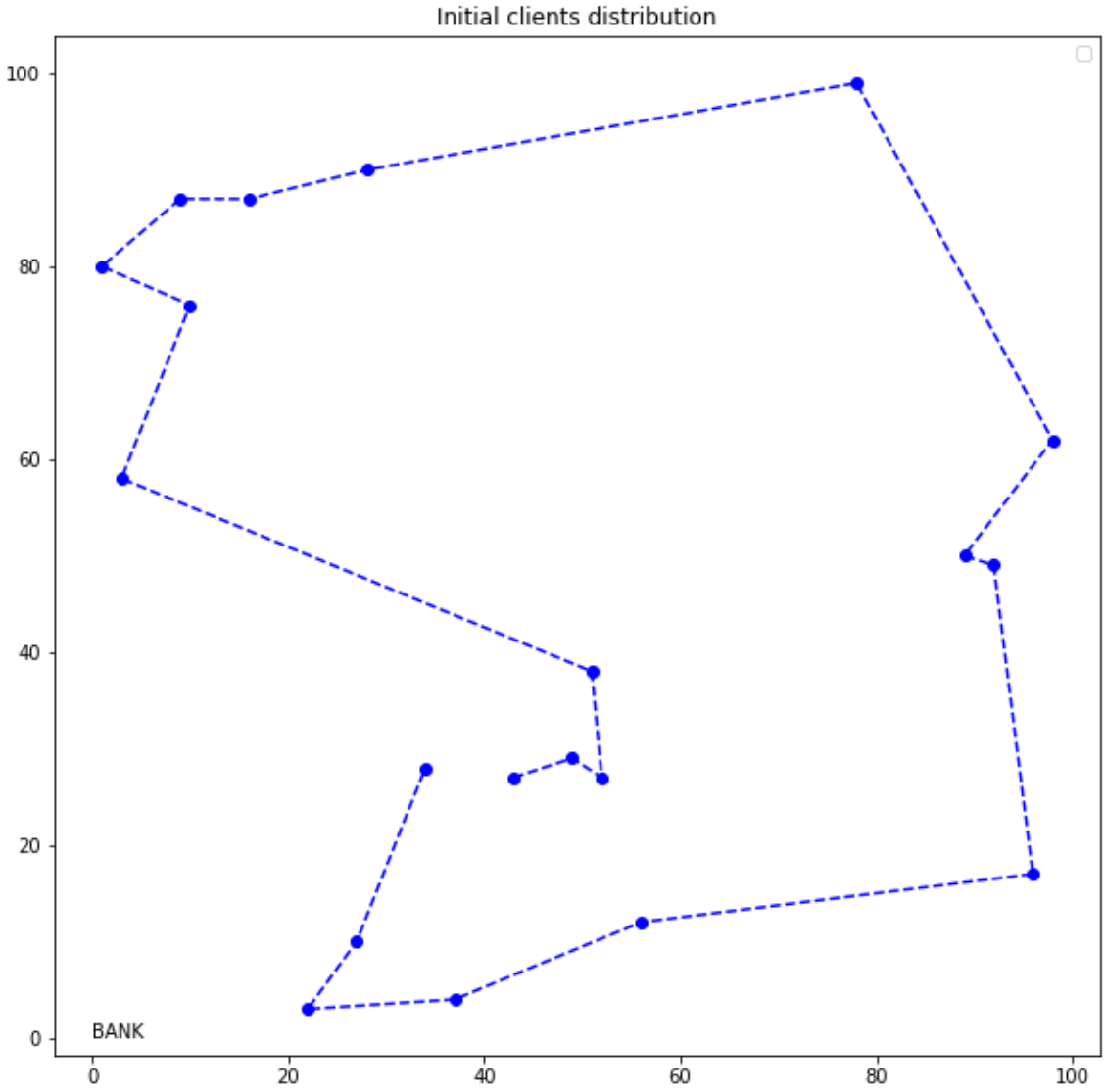


Figure 15: Initial graph plot

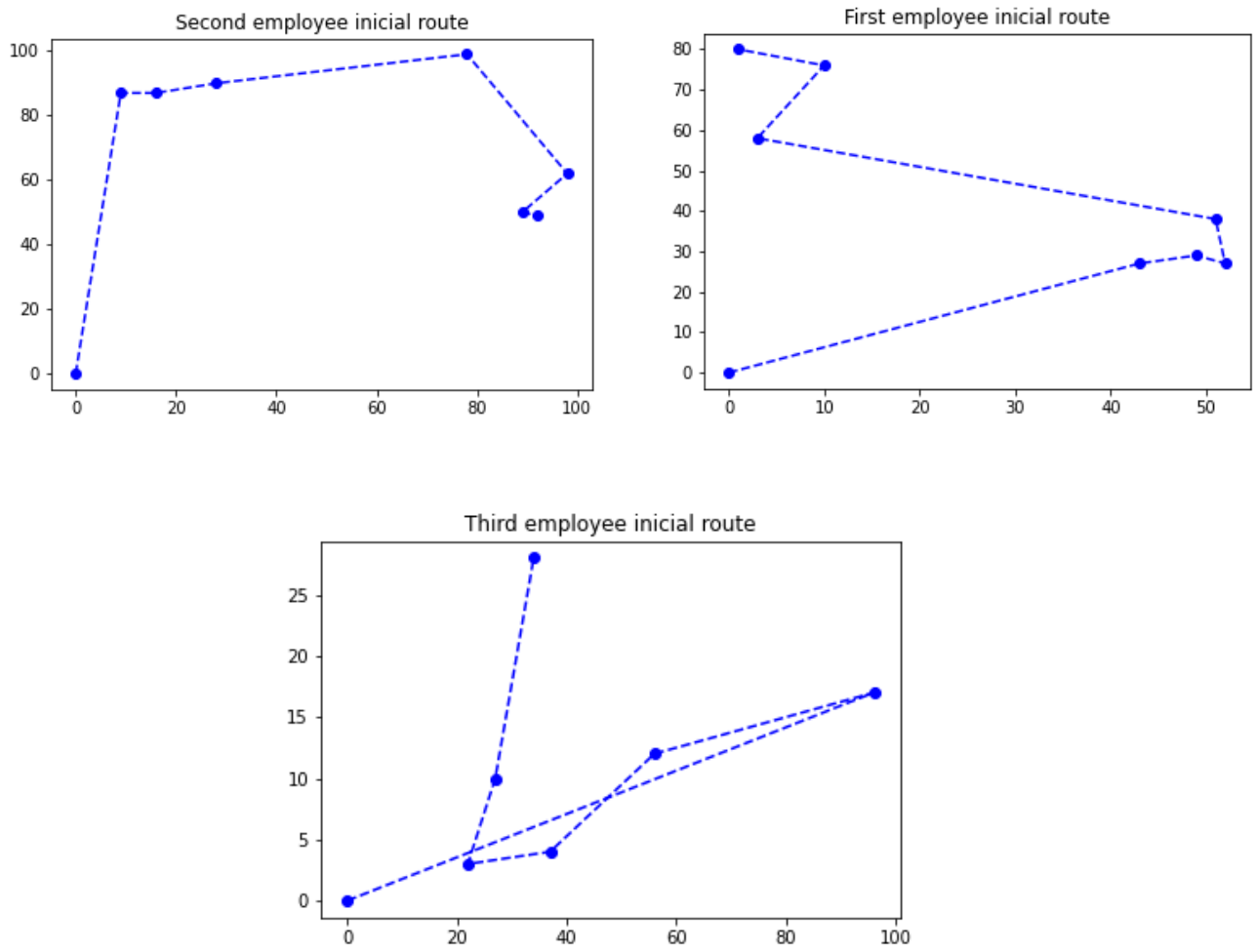


Figure 16: Employees initial routes

	1st Employee	2nd Employee	3rd Employee
Initial distance (m)	336.47	401.42	225.29

Table 3: Employees initial distances

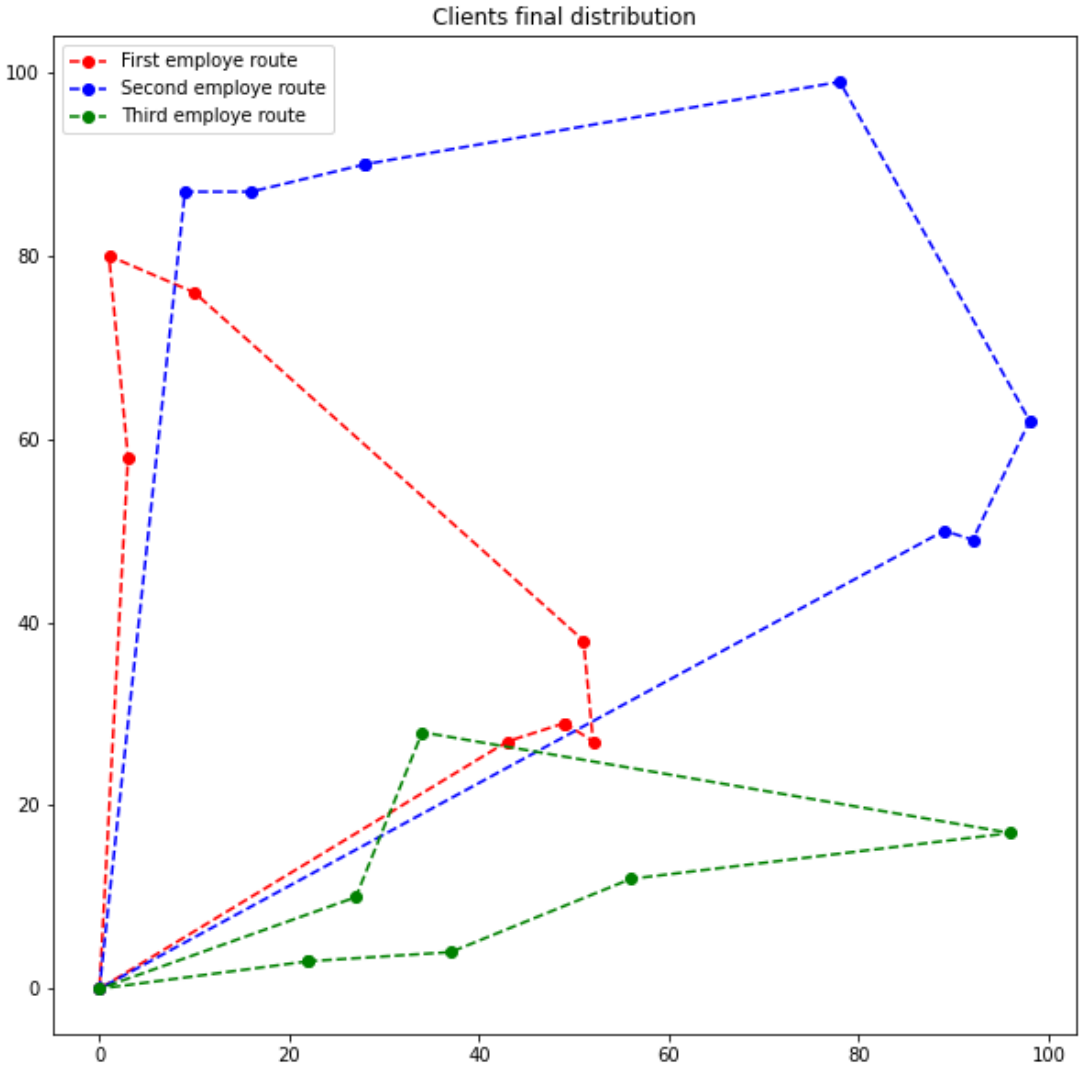


Figure 17: Final employee's routes

2.1. Results:

Execution time 1.02 seconds
1st employee final distance: his shortest route:
[(49, 29), (43, 27), (0, 0), (3, 58), (1, 80), (10, 76), (51, 38), (52, 27)]
2nd employee final distance: his shortest route:
[(28, 90), (16, 87), (9, 87), (0, 0), (89, 50), (92, 49), (98, 62), (78, 99)]
3rd employee final distance: his shortest route:
[(22, 3), (0, 0), (27, 10), (34, 28), (96, 17), (56, 12), (37, 4)]

Suppose that each 1 meter takes 2 seconds time, 2nd experiment results are summarized in this table:

Employee	Final distance (m)	Time (s)
1st employee	217.67	435.34
2nd employee	319.25	638.5
3rd employee	209.23	418.46

Table 4: Second experiment summary

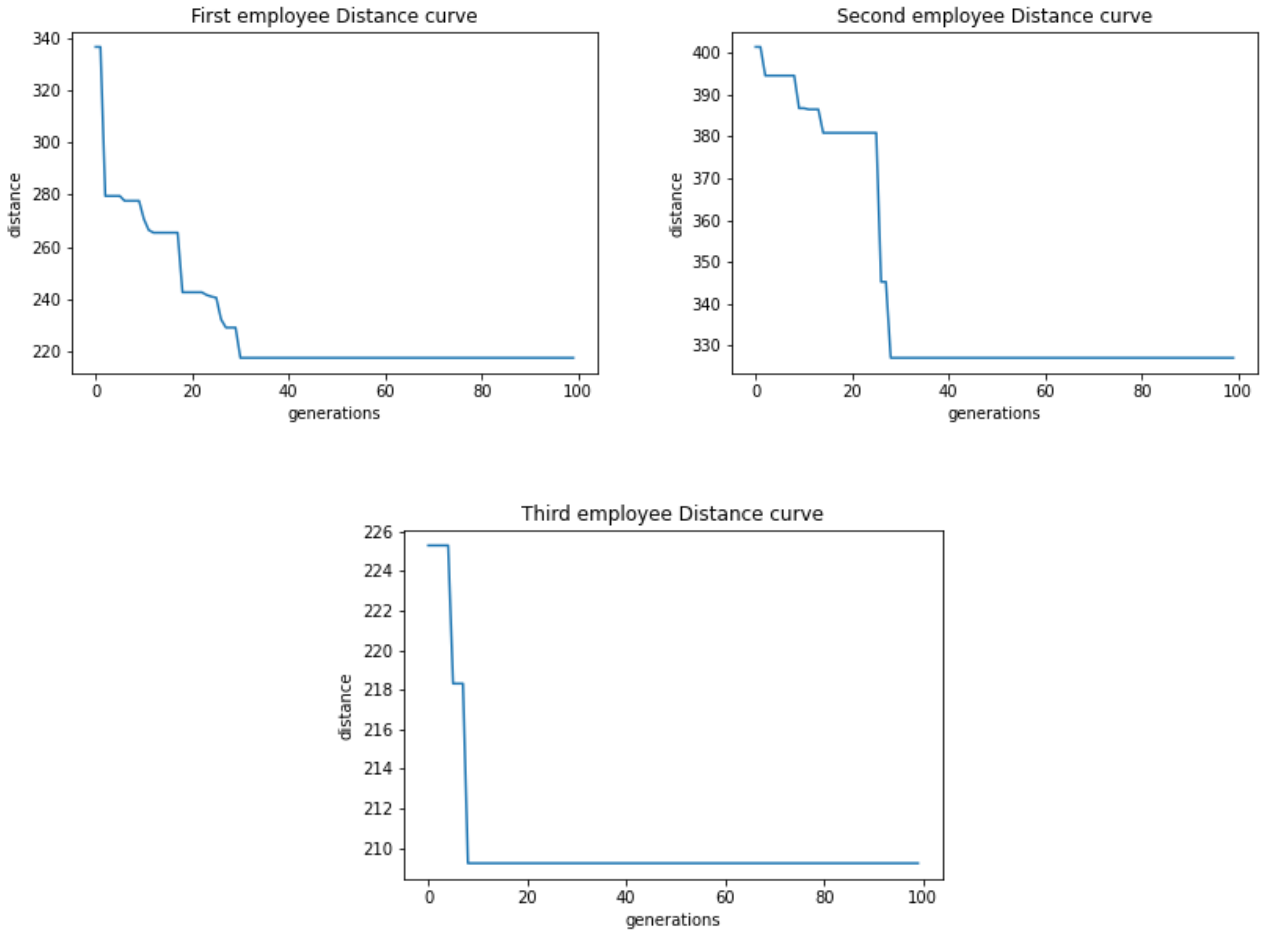


Figure 18: 2nd experiment employees' curves

Conclusion:

The goal of this thesis is to design and implement a genetic algorithm In order to organize bank employees movements, in which, a number of employees travel throughout a set of clients given, the goal is to get the optimal solution which is the minimum cost and distance of each employee's route.

Results above show that the optimal solution is definitely related to the GA parameters, as much as we increase "Generations", "Population size", and "Mutation rate" values as much as we get to the better solution.

References:

1. Umar Farooq, July 25, 2017, What is Banking System | Types of Banking Systems | Banking Systems in Future from https://www.businessstudynotes.com/others/banking-finance/banking-systems-types-and-future-of-banking-systems/
2. banking system. (n.d.) *Farlex Financial Dictionary*. (2009). Retrieved May 27 2020 from https://financial-dictionary.thefreedictionary.com/banking+system.
3. Gaurav Akrani, 4/20/2011, Functions of Banks - Important Banking Functions and Services from https://kalyan-city.blogspot.com/2011/04/functions-of-banks-important-banking.html
4. 30 November 2018, BANKING AND APPOINTMENT SCHEDULING from https://www.appointmentcare.com/industries/banking-and-appointment-scheduling/
5. Kara, I., & Bektas, T. (2006). Integer linear programming formulations of multiple salesman problems and its variations. European Journal of Operational Research, 174(3), 1449-1458.
6. Russell, R. A. (1977). An effective heuristic for the m-tour traveling salesman problem with some side conditions. Operations Research, 25(3), 517-524.
7. Wacholder, E., Han, J., & Mann, R. C. (1989). A neural network algorithm for the multiple traveling salesmen problem. Biological Cybernetics, 61(1), 11-19.
8. Somhom, S., Modares, A., & Enkawa, T. (1999). Competition-based neural network for the multiple travelling salesmen problem with minmax objective. Computers & Operations Research, 26(4), 395-407.
9. Soylu, B. (2015). A general variable neighborhood search heuristic for multiple traveling salesmen problem. Computers & Industrial Engineering, 90, 390-401.
10. Gavish, B., & Srikanth, K. (1986). An optimal solution method for large-scale multiple traveling salesmen problems. Operations Research, 34(5), 698-717.

11. França, P. M., Gendreau, M., Laporte, G., & Müller, F. M. (1995). The m traveling salesman problem with minmax objective. Transportation Science, 29(3), 267-275.
12. Bektas, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. Omega, 34(3), 209-219.
13. Yousefikhoshbakht, M., Didehvar, F., & Rahmati, F. (2013). Modification of the ant colony optimization for solving the multiple traveling salesman problem. Romanian Journal of Information Science and Technology, 16(1), 65 80.
14. Király, A., & Abonyi, J. (2010). A novel approach to solve multiple traveling salesmen problem by genetic algorithm. Computational Intelligence in Engineering, 141-151.
15. Larki, H., & Yousefikhoshbakht, M. (2014). Solving the multiple traveling salesman problem by a novel meta-heuristic algorithm. Journal of Optimization in Industrial Engineering, 7(16), 55-63.
16. Singh, A., & Baghel, A. S. (2009). A new grouping genetic algorithm approach to the multiple traveling salesperson problem. Soft Computing-A Fusion of Foundations, Methodologies and Applications, 13(1), 95-101.
17. Yuan, S., Skinner, B., Huang, S., & Liu, D. (2013). A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms. European Journal of Operational Research, 228(1), 72-82.
18. Sarin, S. C., Sherali, H. D., Judd, J. D., & Tsai, P. F. J. (2014). Multiple asymmetric traveling salesmen problem with and without precedence constraints: Performance comparison of alternative formulations. Computers & Operations Research, 51, 64-89.
19. Venkatesh, P., & Singh, A. (2015). Two metaheuristic approaches for the multiple traveling salesperson problem. Applied Soft Computing, 26, 74-89.

20. Wang, P., Sanin, C., & Szczerbicki, E. (2015). Evolutionary algorithm and decisional DNA for multiple travelling salesman problem. Neurocomputing, 150, 50-57.
21. Bolanos, R. (2016). A population-based algorithm for the multi travelling salesman problem. International Journal of Industrial Engineering Computations, 7(2), 245-256.
22. Changdar, C., Pal, R. K., & Mahapatra, G. S. (2017). A genetic ant colony optimization based algorithm for solid multiple travelling salesmen problem in fuzzy rough environment. Soft Computing, 21(16), 4661-4675.
23. Falcon R, Nayak A (2010) The one-commodity traveling salesman problem with selective pickup and delivery: an ant colony approach. In: Proceedings Barcelona, Spain, CEC, pp 4326–4333Return
24. Ghadle KP, Muley YM (2014) An application of assignment problem in traveling salesman problem (TSP). J Eng Res Appl 4(1):169–172
25. Mahfoudh SS, Khaznaji W, Bellalouna M (2015) A branch and bound algorithm for the porbabilistic traveling salesman problem. In: Proceedings Takamatsu, SNPD, pp 1–6
26. Pham DT, Huynh TTB (2015) New mechanism of combination crossover operators in genetic algorithm for solving the traveling salesman problem, in knowledge and systems engineering, 2nd ed., vol 326, Switzerland, pp 367–379
27. Ye C, Yang ZC, Yan TX (2014) An efficient and scalable algorithm for the traveling salesman problem. In: Proceedings Beijing, ICSESS, pp 335–339
28. tif AK, Muhammad UK, Muneeb I (2012) Multilevel graph partitioning scheme to solve traveling salesman problem. In: Proceedings Las Vegas, NV, ITNG, pp 458–463
29. Hu CQ (2014) A K-means algorithm. J Changchun Univ Technol 35(2):139–142

30. Helsgun K (2014) Solving the equality generalized traveling salesman problem using the Lin–Kernighan–Helsgaun algorithm. Math Program Comput 7(3):269–287
31. E. W. Mayr, Toward a new Philosophy of Biological Thought: Diversity, Evolution and Inheritance, Belknap, Harvard, 1982.
32. W. D. Cannon, The Wisdom of the body, W.W.Norton, 1932.
33. R. Dawkins, The Selfish Gene, Oxford University Press, 1982.
34. A. M. Turing, Computing Machinery and Intelligence, Mind 9 (1950) 433-360.
35. R. M. Friedberg, A learning machine: Part I, IBM Journal of Research and Development 2 (1) (1958) 2-13.
36. G. E. P. Box, Evolutionary operation: A method for increasing industrial productivity, Applied Statistics VI, no. 2 (1957) 81-101.
37. G. Laporte and S. Desroches. Examination timetabling by computer. Computers and Operations Research, 11:351–360, 1984.
38. Klaus Jansen. An approximation scheme for bin packing with conflicts. J. Comb. Optim., 3(4):363–377, 1999.
39. N. Christofides, A. Mingozzi, and P. Toth. The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, editors, Combinatorial Optimization, pages 315–338. Wiley, Chichester, 1979.
40. Rahoual, M., Kitoun, B., Mabed, M. H., Bachelet, V., & Benameur, F. (2001, July). Multicriteria genetic algorithms for the vehicle routing problem with time windows. In 4th Metaheuristics International Conference (pp. 527-532).
41. Iori, M., Salazar-González, J. J., & Vigo, D. (2007). An exact approach for the vehicle routing problem with two-dimensional loading constraints. Transportation Science, 41(2), 253-264.
42. Elissee A, Evgeniou T, and Pontil M. Stability of randomized learning algorithms. J. Mach. Learn. Res., 6:55-79, December 2005.

43. Mora AM, Ashoor H, Awara K, Jankovic BR, Kamau A, Chowdhary R, Archer JAC, and Bajic VB. Conservation of and recognition of translation initiation sites in arabidopsis thaliana. Submitted to BMC Bioinformatics.
44. Cheng B and Titterington DM. Neural networks: A review from a statistical perspective. Statistical Science, 9(1):2-30, 1994.
45. Hassibi B, Stork DG, andWol GJ. Optimal Brain Surgeon and General Network Pruning. Technical Report CRC-TR-9235, RICOH California Research Centre, 1992.
46. Carr C. The mit encyclopedia of the cognitive sciences, edited by robert Wilson and frank keil. Artificial Intelligence, 130:183-184, 2001.
47. Stone C and Smith J. Strategy parameter variety in self-adaptation of mutation rates, 2002.
48. Bishop CM. Neural networks and their applications. Review of Scientific Instruments, 65(6):1803-1832, 1994.
49. Whitley D, Starkweather T, and Bogart C. Genetic algorithms and neural networks: optimizing connections and connectivity. Parallel Computing, 14(3):347-361, 1990.
50. Montana DJ and Davis L. Training feedforward neural networks using genetic algorithms. In Proceedings of the 11th international joint conference on Artificial intelligence - Volume 1, pages 762-767, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

Appendices:

Our notebook:
https://colab.research.google.com/drive/1UtdEOviOaHSdOy6sgWmgEDSWM4yNtp3r?usp=sharing