VEHICLE ROUTING FOR WASTE COLLECTION IN BAGUIO CITY USING PSO-GA

BY

Lance Oliver Licnachan

A special problem submitted to the

Department of Mathematics and Computer Science

College of Science

The University of the Philippines

Baguio, Baguio City

as partial fulfillment of the

requirements for the degree of

Bachelor of Science in Computer Science

June 2018

This is to certify that this Special Problem entitled **"Vehicle Routing for Waste Collection in Baguio City Using PSO-GA"**, prepared and submitted by **Lance Oliver Licnachan** to fulfill part of the requirements for the degree of **Bachelor of Science in Computer Science**, was successfully defended and approved on June 25, 2018.

JOEL M. ADDAWE, PH.D.
Special Problem Adviser

The Department of Mathematics and Computer Science endorses the acceptance of this Special Problem as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science .

JERICO B. BACANI, PH.D.
Chair
Department of Mathematics and
Computer Science

Table of Contents

Acknowledgments

I would also like to acknowledge my family and friends for their support in my pursuit to accomplish this study. I would also like to acknowledge the Solid Waste Management Division of Baguio City for entertaining my requests and questions about the current state of waste management in the city.

Abstract

Vehicle Routing for Waste Collection in Baguio City Using PSO-GA

Lance Oliver Licnachan
University of the Philippines, 2018

Adviser:
Joel M. Addawe, Ph.D.

Waste collection services are the key to proper development in any country. In Baguio City, waste collection has been a main concern for the past years as population increases. The waste collection problem was modeled as a Vehicle Routing Problem and a hybrid PSO-GA algorithm was employed to obtain the set of routes that give the minimum amount of travel distance. It was found that the hybrid PSO-GA algorithm was indeed able to solve vehicle routing problems and that the optimal set of routes give the minimum distance of $1,284.830000$ kilometers using 74 vehicles.

List of Tables

List of Figures

Chapter 1

Introduction

Municipal solid waste management is one of the key services held as a foundation in developing urban cities around the world. It is without a doubt that urbanization and development comes with the production of large volumes of waste. Moreover, in densely populated cities, waste management becomes challenging as it is met with increasingly larger amounts of municipal solid waste generated by the increasing population size. Development and industrialization ensues the improvement of the standards of living and an increase in the total amount of disposable income, hence, individuals become encouraged to consume goods and services, thereby resulting in an increase in the amount of waste generated. Couple this with the fact that major economies run on an unyielding cycle of production and consumption, waste generation is likely to gain speed as cities become more industrialized and populated. According to the study conducted by the World Bank, called "WHAT A WASTE: A Global Review of Solid Waste Management", back in 2012, global urban population annually produced about 1.3 billion metric tons of Municipal Solid Waste (MSW) which is expected to grow to about 2.2 billion metric tons in 2025.[**?**]

According to the WHAT A WASTE,[**?**] waste management in underdeveloped countries are worse than that of developing ones. This is because there are few to no proper facilities or vehicles that are needed to efficiently handle waste collection. Poorly managed waste can result to the destruction of the environment, and endangerment of public health. If waste is left uncollected, it may lead to sewer flooding, different kinds of pollution (such as soil, air, water etc.), and road blockades. Methane, a highly flammable gas, is produced when garbage is decomposed, hence, uncollected waste can become a source of residential fires. Moreover, toxic chemicals from household materials may seep out of garbage bags and contaminate both soil and water. This endangers both flora and fauna living in the environment. In addition to the effects to public health, mismanaged

waste can impact an area's economy. Businesses that do not comply with the regulations imposed by public health and safety get shut down. Uncollected waste can result to a hazardous work environment that can endanger employers, employees, and customers. Garbage can be a breeding ground for diseases which can lead to epidemics that may cripple not only the work flow but also foreign interest. In contrast, a city that has efficient waste management would be able to develop faster as its focus shifts to other community needs such as transportation, health, and education. Given the effects of waste collection mismanagement, all units of the community should improve their own waste management to cope with the problems that are brought about by large amounts of solid waste.

The waste management services industry is a growing business. Developed nations highly depend upon their waste management service providers to move waste out of urban districts and industrial sectors. These services help reduce public health risks and provide a clean atmosphere for a conducive area of business. Normally, people would rather work in a safe and clean environment than at a filthy and odorous one however, people that do work on waste are being paid a generous amount of money. This is because on-site workers risk their lives working with different kinds of waste which may cause them to be exposed to toxic and highly dangerous substances, hence, there must be a good amount of compensation. Not only that but the construction and maintenance of facilities needed for safely dealing with waste are expensive. This is the reason governments allocate a large portion of their annual budget for waste management. According to WHAT A WASTE[**?**], solid waste management costs will increase from an annual $205.4 billion in 2012 to about $375.5 billion in 2025 to compensate for the projected increase in the amount of waste generated.

Waste, in this context, is defined as matter that is unwanted or unusable. This refers to matter which are discarded after primary use or is deemed defective, or worthless. There are various types of waste but our focus is on municipal solid waste. This is the type of waste that is produced daily from residential, commercial, institutional and industrial sources. The term 'municipal' comes from the fact that it is the duty of the municipality to collect and manage these kinds of waste. A list of what can be considered as municipal solid waste is as follows:

- **Biodegradable waste** which is any form of organic matter that is found in the trash. This type of solid waste are usually composed of leftover food, by products of cooking, agricultural waste (such as lawn clippings, dead leaves, etc.) and paper-based materials.
- **Recyclable materials** which are objects that can be re-used in an alternative way such as glass, bottles, jars, clothes, fabrics, rubber etc.
- **Residual waste** is the type of solid waste that is neither recyclable nor reusable. This may include items that are beyond repair such as broken instruments, shattered glass and ceramics, and used fireworks.
- **Inert** or **nonreactive waste** such as dirt, rocks, construction debris, etc. These items do not react, chemically or physically, and hence, do not decompose.
- **Electrical and electronic waste** by its name, are discarded electrical and electronic devices or components such as appliances, light bulbs, mobile phones, television sets, etc.
- **Composite waste** which are composed of two or more constituent materials such as toys, tetra packs, clothing, fiber glass, etc.
- **Hazardous waste** which poses health risks such as paints, batteries, aerosol sprays, and fertilizers
- **Toxic waste** which are poisonous such as pesticides, herbicides and fungicides
- **Biomedical waste** which may contain infectious materials such as expired pharmaceuticals, used medical equipment, used tissues, etc.

Waste collection includes gathering, transporting, and disposing of solid waste and recyclable materials. Waste collection involves deploying vehicles that collect and transport the waste from communities to facilities that receive, sort, and process the waste. Processing waste may be in the form of incineration, rapid degradation, segregation, resource recovery, energy recover, etc.

Waste can be collected in several ways. House-to-house collection is done by individually visiting each house and collecting the garbage straight from the source. Communities can opt for gathering their garbage at certain designated locations or community bins in the neighborhood. Another way is through curbside pick-ups where households leave

their garbage directly in front of their houses to be picked-up by waste collection vehicles at a particular time. Households can also volunteer to personally deliver their garbage directly to disposal sites. The local government can opt for waste collection services with private companies that have specialized facilities and vehicles that can handle the job.

Before collection, households can be asked to segregate their garbage into specified categories. These categories might include, wet and dry, biodegradable and non-biodegradable, reusable, recyclable, paper, glass, plastics, aluminum etc. The quality of waste segregation can determine efficiency and effectiveness of waste processing. Certain waste can be re-segregated by a machine-sorter or by designated personnel called 'pickers'. The reusable quality of an object can be determined by the machine or by personally inspecting the object. Waste such as bear bottles and plastic-containers can be taken out and deemed reusable. Waste such as broken plastic frames or metals can undergo secondary use through remelting and remolding. Papers and plastics can be recycled and re-purposed for a different niche. Biodegradable waste can be converted to compost and degraded through anaerobic digestion. The rest can be piled into landfills or disintegrated using incinerators. Landfills, however, come with other sets of problem such as health-care, contamination, pollution, land use etc. On the other hand, incinerating waste produces greenhouse gases which may raise health care issues. There even exists materials that are immune to incineration such as soil and rocks. Toxic waste disposal is also an arduous task since harmful chemicals are involved. One cannot just dump them in a field and hope for the best. The more developed the facilities and methods are, the better the waste management.

Each of the different collection options presented above have different ways of computing cost. Costs in waste collection may involve staff salaries, vehicle purchases, fuel costs, construction of buildings and infrastructure, fuel expenditure, maintenance, land use and purchase, business contracts, environmental and health care expenditures, social acceptability, etc. We have seen that waste collection services is an expensive service to develop and maintain. It is therefore crucial to find ways to minimize the costs involved whilst not compromising efficiency. Therefore, we look to finding he most efficient set of waste collection routes in order to minimize waste collection expenses.

The routes of waste collection vehicles are, conventionally, manually determined by

drivers on-site. Data collection and surveys can help analyze the conditions that affect vehicle routing such as traffic severity and road inclination. However, the use of such methods produce inflexible routes because they are instance-based data. In the recent years, with the development of commercial interactive navigational tools such as Google Maps, Waze and Global Positioning Systems, vehicle routing has become a hot topic. Researchers have used similar kinds of systems to improve delivery and collection services. In order to solve these kinds of problems, a vehicle routing problem model is patterned after the specific problem. There are various methods that can be used to solve this kind problem.

The **Vehicle routing problem (VRP)** involves the deployment of a fleet of vehicles which are expected to service a given set of customers. Solutions to the vehicle routing problem are a set of routes for the fleet of vehicles wherein the set gives the minimum amount of traveling cost. VRP is a generalization of the **Traveling Salesman Problem (TSP)**. The traveling salesman problem is description is as follows:

1. A salesman needs to visit every city in a set of cities.

2. He/she does not care about the order of visitation as long as he/she gets to visit each one along the way.

3. He/she must begin and end at the same city

4. Each city is connected to other close by cities, by airplanes, or by road or railway. Hence, each of the connections between the cities has one or more costs attached depending on the availability of transportation means.

5. The cost describes how "expensive" it is to travel from one city to another. This may be in the form of the cost of an airplane ticket or train ticket. Perhaps the cost may be given by the length of the distance or span of time needed to travel from one city to another

6. The salesman wants to keep both the travel cost and the length of the distance he travels to a minimum.

The aim is to generate a path or a sequence of cities that lets the salesman pass through all cities exactly once before returning to the starting city, spends the minimum amount of travel expenses and the travels shortest possible distance. The problem is mostly concerned about generating the best sequence of N cities among the $(N-1)!$ permutations. As we can see, the amount of permutations rapidly increases as the number of cities is increased. $N-1$ because we always start at a given city therefore, we can say that the salesman has already visited that city.

The vehicle routing problem is typically the same problem however, there are more than one salesman. The load of the single salesman is distributed in order to save other costs (i.e. time) and maximize man power. VRP is a combinatorial problem that deals with arranging multiple specified locations along a single or multiple path(s) that gives the smallest amount of transportation cost while satisfying known constraints. VRP is usually concerned with the minimization of the temporal and/or geographical aspects of traveling along road networks while accommodating the most amount of customer demands along the way. Customers are usually distributed at different locations in the real world. In a capacitated VRP, each customer has a certain amount of demand that utilizes the maximum capacity of the vehicles servicing them. Vehicles neither collect nor delivery more than their capacities. VRP models may also include minimizing the number of vehicles needed to satisfy the total customer demand.

Municipal waste collection VRP involves vehicles collecting municipal waste at customer locations or community bins, and disposing the load at disposal sites. Vehicles start at a depot where they are parked. They are then deployed and travel along their respective routes while collecting waste. When full, the vehicle then moves to a disposal site to unload the waste collected. The vehicle may then either continue along it's path or return to the depot. Waste Collection Vehicle Routing Problems may impose that waste must either be completely or partially collected by a vehicle. Complete collection invokes the rule in VRP that customers are only serviced once by any vehicle. On the other hand, partial collection implies that customers can be serviced more than once by one or more vehicles.

The next sections of this paper are as follows. Chapter 2 involves the discussion of studies conducted by researchers in the previous years. Chapter 3 is a discussion of the

method used in order obtain the set of routes. Chapter 4 is where the results from the tests done with the method in chapters 3 is analyzed. Chapter 5 is where the results are summarized and conclusions are stated.

1.1 Background of the Study

According to the National Solid Waste Management Commission, about 37,000 tons of municipal solid waste (MSW) are produced in the Philippines. Based from the available data from 2008 to 2013, most of the total municipal solid waste in the Philippines comes from the residential sector at 56.7%, while the contributions of the commercial, institutional, and industrial sectors are 27.1%, 12.1% and 4.1% respectively. The municipal solid waste is mostly composed of biodegradable waste at 52.31% while recyclables, residual and special wastes contribute 27.78%, 17.98% and 1.93% respectively. Biodegradable waste consists of kitchen or food waste as well as yard or garden waste. Recyclable wastes consists of plastic packaging, paper and cardboard, metals, glass, textile, leather and rubber. Special waste consists of household health-care waste, waste electrical and electronic equipment, bulky waste and other hazardous materials. Residual waste is composed of the waste that is neither biodegradable, recyclable, nor special waste. This is the type of waste that is sent to landfills. It was projected that the amount of MSW is to increase to about 40,000 tons in 2016 from 37,000 in 2012. Out of the 16 regions, the Cordillera Administrative Region (CAR) contributed about 1.66% in 2012.

In 2015, the City of Baguio in the Cordillera Administrative Region conducted a Waste Analysis and Characterization Survey (WACS)[12] where they investigated the output of garbage in the city. It was found that the residents of the city produced 402 tons of mixed waste daily, 41.67% being biodegradable, 33.78% recyclables, 21.41% residuals for recycling, 2.74% residuals for disposal and 0.41% special wastes. Most of the generated waste came from the commercial sector at 60.44%, the contribution of the residential, institutional and industrial sectors are 35.16%, 3.53% and 0.86% respectively. Baguio City, at its core, is a place where farmers from both the surrounding mountains and valleys take their crops to be sold hence, the reason for the commercial sector generating a majority of its waste.

In relation to the WACS, the city drafted a 10 year solid waste manage ment plan as required by Republic Act 9003 or the Ecological Solid Waste Management Act of 2000. Using the WACS, it was projected that the population of Baguio City would climb to about $398,215$ in 2025 from $337,798$ in 2015 and the daily generated waste would rise to about 522 tons from 402 tons. Inclusive of this 10 year solid waste management plan, several facilities are to be constructed for the recovery of resources from municipal solid waste. The plan was approved on 2017, the city is now en route to establishing and developing several waste collection facilities for the next ten years, namely, a centralized materials recovery facility, an engineered sanitary land-fill, an anaerobic digester, a waste-to-energy plant, Environmental Recycling System machines, a health and medical waste treatment plant, and a special waste treatment plant.[25] As of 2018, Baguio City has 14 functioning waste collection trucks, two of which are used as a quick response team in cases of emergencies. The city has purchased 4 more vehicles this last January. This move of purchasing vehicles was said to boost efficiency and to keep-up with the growing tourist influx during the weekends, holidays and the incoming summer vacation.[26] The 14 waste collection vehicles are responsible for servicing the 129 barangays (villages) inclusive of the Central Business District. Vehicle compartments are partitioned such that there is segregation between residual and biodegradable waste. Recyclable materials are usually dealt with by the barangays (villages) who hire personnel to sort the garbage and take out reusable, recyclable materials. The drivers follow a 5-day schedule, the other two days of the week are given as rest days. In each of the five days, they are to service a set of 2-5 barangays. The drivers are set to work 9 hours each working day. Currently, the drivers are the ones who select their routes; they try to avoid traffic in order to attend to each designated collection site where the residents of each barangay pool their waste. All vehicles start at the Eco-Waste Recovery Services-Material Recovery Facility (ERS-MRF) at Barangay Irisan where the drivers sign in for the day. Garbage is collected until vehicle capacities are reach. Waste is returned to the ERS-MRF for final sorting. The residual waste is then transported to the Garbage Transfer Station at Barangay Dontogan. Biodegradable waste is composted while recyclable and reusable materials are sold. It is important to note that there are no specific time windows when each collection site is visited because there are too much variables that can affect collection time such as

traffic conditions, weather conditions, amount to be collected, etc.

Indeed, the city is doing it's part to reduce the carbon footprint and employ better waste management by employing the no plastic policy or the "Plastic and Styrofoam-Free Baguio Ordinance" of 2017. This city ordinance regulates the sale, distribution and use of plastic bags and styrofoam in the city. Instead of plastics and styrofoam containers, vendors are encouraged "to provide or make available to customers for free or for a cost, paper bags or reusable bags or containers made of paper or materials which are biodegradable, for the purpose of carrying out goods or other items from the point of sale.[22]

In line with these city policies and activities, we study the current efficiency of the routing and scheduling of waste collection vehicles. Our motivation is to help the community.

1.2 Statement of the Problem

We want to find a way to reduce traveling expenses of waste collection vehicles in Baguio City. It was observed that the cost of waste collection is large for any developing country. Baguio City, for over 10 years, has spent over PHP 1 billion for hauling the city's solid waste to the sanitary landfill in Capas, Tarlac. This involves the operation of collection, transportation, and disposal. In 2017, City Budget Officer Leticia Clemente estimated the annual garbage disposal expense at PHP 100 million, where PHP 80 million of which is spent for hauling and tipping fees in Capas, Tarlac; and for personnel, garbage trucks, and other operating expenses.[17]. The city council wants to reduce annual solid waste disposal expenses so that it can be allocated elsewhere. We approach the problem by modeling the waste collection problem into a vehicle routing problem wherein the goal is to obtain the minimum travel distance required to collect waste and transport it back to the city's Eco-Waste Recovery Services-Material Recovery Facility (ERS-MRF) at Barangay Irisan for sorting before it is transported to Capas, Tarlac. A hybrid Particle Swarm Optimization - Genetic Algorithm (PSO-GA) proposed by Harish Garg[13] will be used to solve the vehicle routing problem.

1.3 Objective of the Study

1.3.1 General Objective of the Study

The general objective of this study is to identify which processes and sections of Baguio City's waste management can be optimized in order to reduce the cost involved in waste management.

1.3.2 Specific Objective of the Study

We identify that both sorting and waste collection are processes that can affect the cost of waste management. We know that the city has implemented its own policies on waste segregation therefore, we look to waste collection. Specifically, we find a way to reduce the travel cost of waste collection vehicles. Moreover, we aim to obtain a set of vehicle routes that give the minimum amount of distance while also determining the minimum number of waste collection vehicles needed for the job.

1.4 Significance of the Study

The results from this study will allow the determination of the possible routes that may minimize the cost of travel among waste collection vehicles in the City of Baguio. Hence, we directly reduce the cost of travel expenses in waste collection. Determining the number of vehicles required to accomplish the job may give incite as to the scale and severity of the waste collection problem in the city. The robustness of the algorithm used in this study will also be determined by the results obtained. Since the hybrid PSO-GA[13] is designed to solve constraint optimization problems, we expect that the algorithm will produce good results for the vehicle routing problem which is another kind of constraint optimization problem compared to that of the engineering design problems Harish Garg used to test the algorithm.

1.5 Scope and Limitation

The study is limited to generating a set of routes involved in waste collection in Baguio City. In order to model the waste collection problem, each of the 129 barangays (villages) were taken as collection sites since there was no available data on the specific locations of the community bins in each of the barangays. This is because the specific collection site locations vary depending on accessibility, road orientation, and public health acceptability. The ERS-MRF at Barangay Irisan was identified as the location where each waste collection vehicle starts and returns when the vehicle is full. Given that there exists no exact time table as to when vehicles are required to visit each barangay, we opt only for a Capacitated Vehicle Routing Problem to model the waste collection problem. The real distances between the ERS-MRF and each barangay as well as the distances between each barangay were obtained through Google Maps©. A map of all 129 barangays are shown on figure 1.1. The red house marker indicates the Irisan ERS-MRF, the yellow flag indicates the Dontogan Transfer Station and the blue markers are the 129 barangay markers. A list of the barangays and their respective markers are seen on table A.1

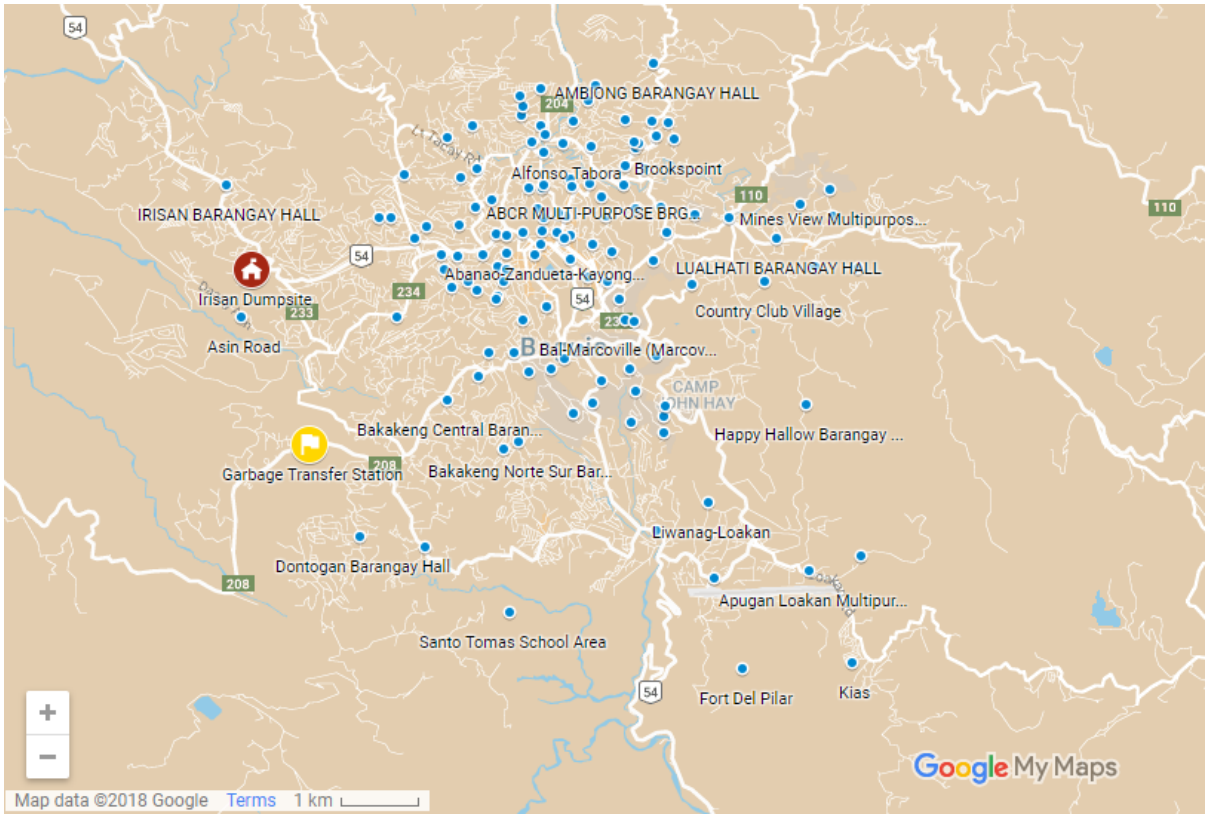

Figure 1.1: A Map Showing the Locations of the 129 Barangays and the Irisan ERS-MRF

Chapter 2

Preliminaries

We first define some terms and notations we will be using throughout this document.

2.0.1 Definitions

1. An **algorithm** is a sequence of unambiguous instructions for solving problems. Every step is clear and concise, no instruction should be interpreted more than one way.

2. **Optimization** is a mathematical technique used for solving the maximum or minimum value of a function or system of equation. In a broader sense, it is a technique used to solve for the optimum solution to a particular problem. Optimality refers to obtaining the best possible form or functionality in the sense that it is more than sufficiently efficient given a set of resources. This involves meeting an expected result with high accuracy and precision such that specifications and limitations are also achieved.

3. An **optimization algorithm** is a process followed in finding the best or most efficient solution to a given problem.

   In order to solve a problem, we must create a model that embodies the essence of the problem. In this sense, the model must be created in such a way that we can approach it through computable means.

4. An **Objective Function** or **Fitness Function** is the mathematical equation that is modeled after the problem such that, satisfying the function will satisfy the given problem. The objective function is important because it will determine the computability and complexity of the problem as well as the approach taken, in this

case, the algorithm and its implementation. Optimization problems aim to obtain the minimum and/or maximum of certain properties related to some object. The output of the objective function dictates whether or not a specific input is not only a solution but also the most optimal one. We say, it is a 'fitness function' because it measures the capability and efficiency of the input in solving the problem.

5. **Design Variables** are the input to objective functions. We say 'design variables' because these sequence of numbers are being used to test and determine the quality of the output. The algorithm is tasked to manipulate the values of these variables in order to get the optimal solution. In tackling real world problems, design often involves a huge amount of data collection through trial-and-error. Our variables are associated to the factors which undergo changes in values during the trial-and-error processes. The data collected should give the efficiency or numerical score of the given design variables.

6. A **heuristic** is a technique designed for solving a problem when classical methods are too slow for finding approximate solutions or when classical methods fail to find any exact solution at all. These classical methods are those that use mathematical identities, properties, and theorems to prove, show, derive or systematically find solutions to problems. The objective of a heuristic is to produce a solution within a reasonable amount of time such that the solution is acceptable enough to the implementor. Although time may not be the only factor that may be taken in consideration, it is the most commonly used factor in differentiating the quality of heuristic approaches.

7. **Metaheuristic** is a high-level procedure to find, generate or select a heuristic that may provide a sufficiently good solution to an optimization problem. Since we are dealing with optimization, finding the fastest and most efficient way to solve the problem is considered to help in finding solutions.

8. An **evolutionary algorithm** is a generic population-based metaheuristic optimization algorithm. It uses mechanisms inspired by biological evolution such as reproduction, mutation, recombination and selection. Candidate solutions to the

optimization problem play the role of individuals in a population, and the fitness function determines the quality of these solutions. We say 'candidate solutions' because all of these individuals may give an acceptable solution but not all of them give the best solution. Evolution of the population takes place after repeated applications of the mentioned operators. We say 'evolution' because members of the population changes or are somewhat different as time progresses or as the population shifts from one generation to another.

9. A **simulation** is a computational model that imitates real world situations and processes. These usually involved an implementation of mathematical equations that employ stochastic variables for a more 'lifelike' appearance.

10. A **stochastic variable** is a variable whose value is a random number usually taken from a uniform distribution from 0 to 1. That is, every number between 0 and 1 has an equal chance to be selected.

11. **Natural Selection** is the process by which organisms with better attributes adapted to the environment tend to increasingly survive and transmit their genetic characteristics through generations. **'Survival of the fittest'** is a phrase by Charles Darwin that describes the mechanism of natural selection. It is best understood as survival through reproductive multiplicity. That is, the more survivability an individual has, the more it is likely to reproduce, hence it's genes are more likely to be transmitted to the next generation.
    In natural selection, there is a variation on traits that is to say that individuals have differences in certain attributes such as height, length, shape etc. It is important to note that not all individuals reproduce to the full potential because the environment has a certain limit to the number of creatures it can sustain. The passing of characteristics or traits from one generation to another is called **heredity**. The more advantageous traits is more commonly passed on and retained because they help the individual or group to survive.

12. Gregor Mendel is known as the father of modern genetics. He discovered the mechanics of heredity or how traits are being passed down from parents to offspring.

During cell division, thread-like structures located inside the nucleus of animal and plant cells called **chromosomes** are replicated. These chromosomes contain the genes which dictate the attributes of the individual. They tell how the body is to be built and how it functions.

13. **Recombination** or **Crossover** is the rearrangement of the genetic material by exchanging the same gene subsegments of two chromosomes (one from each parent) which allows for the creation of a new individual that has characteristics similar to those of the father and of the mother. Note that the exchanging process may occur in multiple areas of the strands.

14. **Mutation** on the other hand is the alteration of genes resulting from an error during replication. This results in unique characteristics that may be new from the gene pool of the previous generation. Mutation may be good or bad for the individual but this phenomenon has a low chance of occurring naturally for every generation.

15. **Robustness** is the balance between efficiency and efficacy necessary for the survival in many different environments. For Algorithms, this translates to consistent efficiency under different problems areas such that there is little to no change in the process. This means that there is less cost for redesigns. Note that nature is the best example in terms of robustness. It tries to maintain and cope with the many different changes that occur everyday. Hence, we have evolutionary algorithms as stated above.

16. **Exploration** is the capability of the algorithm to search solutions in parts of the subspace it has not yet taken into consideration. **Exploitation** is the capability of the algorithm to utilize known data in searching for solutions in the search space.

17. A **set** is a collection of well defined objects. In this document, we will talk about sets as a collection of numbers that represent objects. A set is usually denoted by braces ('{' and '}') and capital letters (A,B,C,D,...) (ex. $A = \{1, 2, 3\}$). In a set, the order of enumeration and repetition of numbers do not matter. That is, $A = \{2, 3, 3, 2, 1, 1\}$ is equal to $A = \{1, 2, 3\}$.

18. An object is considered an **element** (denoted by $\in$) of a set if it belongs to the set. Using our previous example, we say that 1 is an element of A ($1 \in A$) but 4 is not an element of A ($4 \notin A$). There are two ways of declaring membership of sets,

    (a) (a) **roster method** where we define all the elements included in a set by listing or enumerating all of them; and

    (b) (b) **rule method (set-builder notation)** where we define all the elements included in a set using their properties.

    An example of the rule method is $A = \{x \text{ is a natural number}, x < 4\}$ which can also be written as $A = \{x | x \in \mathbb{N}, x < 4\}$, to be pronounced as "the set of all x, such that x is an element of the natural numbers and x is less than 4". The vertical bar ('|') is usually pronounced as "such that", and it comes between the name of the variable you're using to stand for the elements and the rule that tells you what those elements are.

19. **Cardinality** of a set is the number of unique appearances of elements in a set. Cardinality is denoted by two vertical bars ('|') separated by the set name such as '$|A|$'. That is, using our example, the cardinality of A written as $|A|$ is 3 because A has unique elements $1, 2$ and 3.

20. A set without elements is called the **null** or **empty set** (denoted by $\varnothing$) that is, $\varnothing = \{\}$. Therefore $|\varnothing| = 0$.

21. A set with infinite elements is called an **infinite set**, $F = \ldots, 1, 2, 3 \ldots$. and $|F| = \infty$.

22. A **countable** set is a set with the same cardinality as some subset of the set of natural numbers $\mathbb{N}$. A countable set is either a **finite** set or a **countably infinite** set, nevertheless, the elements of a countable set can always be counted one at a time and, although the counting may never finish, every element of the set is associated with a unique natural number.

23. A **Venn Diagram** is a visual representation of the relationships of sets.

24. We say that A is a **subset** of B (written as $A \subseteq B$). If all elements of A are also elements of B. If $A = \{1, 2, 3\}$ and $B = \{1, 2, 3, 4, 5\}$ then $A \subseteq B$. However if we have the set $C = \{1, 2, 3, 6\}$, $C \nsubseteq B$ because $6 \notin B$ but $A \subseteq C$. A venn diagram of the relationships of A, B and C are shown on figure 2.1.

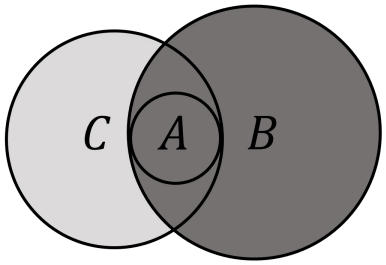

Figure 2.1: A Venn Diagram Showing the Relationship of A, B and C

25. If we have $A = \{1, 2, 3\}$ and $D = \{3, 4, 5, 6\}$, then the **Union** of A and D (written as $A \cup D$) is the set containing all elements of A and D. That is, $E = A \cup D = \{1, 2, 3, 4, 5, 6\}$. A venn diagram showing $A \cup D$ shaded in gray is shown on figure 2.1.

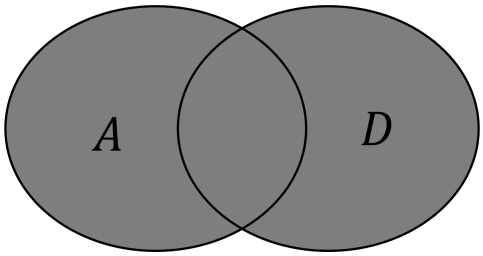

Figure 2.2: A Venn Diagram Showing $A \cup D$

26. If we have the same sets A and D, then the **Intersection** of A and D (written as $A \cap D$) is the set containing all the common elements of A and D. That is, $A \cap D = \{3\}$. A venn diagram of showing $A \cap D$ shaded gray is shown on figure 2.3.

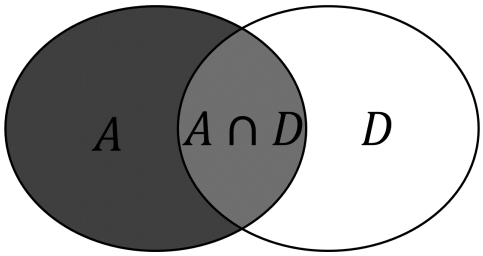

Figure 2.3: A Venn Diagram Showing $A \cap D$

27. If we have the same set A, then the **Complement** of A written as A' or $\bar{A}$ is the set containing all the elements that are not in A. That is $\bar{A} = \{x | x \in \mathbb{N}, x > 3\}$. A venn diagram of showing $\bar{A}$ shaded gray is shown on figure 2.4.

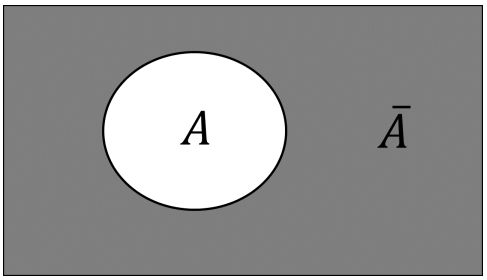

Figure 2.4: A Venn Diagram Showing $\bar{A}$

28. If we have the same sets A and D, then set **Difference (subtraction)** is defined as $A - D$ or $A\ D$ which consists of elements in A but not in D. That is, $A - D = 1, 2$. A venn diagram of showing $A - D$ shaded gray is shown on figure 2.5.

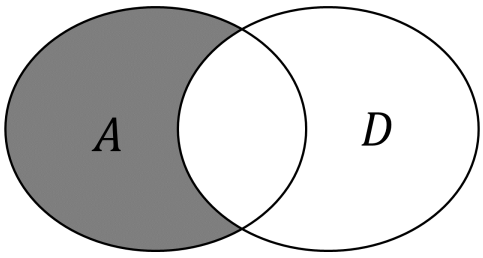

Figure 2.5: A Venn Diagram Showing $A - D$

29. In mathematics, numbers are grouped in sets and subsets.

(a) We first have the smallest subset, the set of **Natural** or **Whole Numbers** ($\mathbb{N}$) which is the set of counting numbers, {0, 1, 2, 3, 4, 5...}.

(b) The next subset is the set of **Integers** ($\mathbb{Z}$) which is the set of natural numbers and their negatives {...-4, -3, -2, -1, 0, 1, 2, 3, 4, ...}.

(c) Next are the **Rational numbers** ($\mathbb{Q}$) are the ratios of integers, also called fractions, such as $\frac{1}{2}$, $\frac{-10}{56}$ etc.

(d) Next are the **Irrational Numbers**, numbers that are not included in the rational number set such as radicals or roots (ex. $\sqrt{5}$) and numbers having infinite non-repeating decimal places such as π.

(e) Finally, the set of **Real Numbers** ($\mathbb{R}$) which consists of both rational and irrational numbers.

(f) Other than the real numbers, we have the **Imaginary numbers** ($\mathbb{I}$) which are the numbers that have negative squares. These numbers are involved with the number $i = \sqrt{-1}$.

(g) The set containing all numbers is called the **Complex Number** ($\mathbb{C}$). This set is the union of both real and imaginary numbers. These numbers are usually represented by the sum of a real and an imaginary number (ex. $1 + i$).

A Venn Diagram of the number sets is given by figure 2.6 .

30. A **solution space** the set of all possible values of an optimization problem that satisfy the problem's constraints, potentially including inequalities, equalities, and integer constraints. This is the initial set of candidate solutions to the problem, before the set of candidates has been narrowed down.

31. **candidate solutions** are potential solutions to problems.

32. A **sequence** is a collection of objects wherein the oder of enumeration is important (ex. a list). Unlike a set, the same elements can appear multiple times at different positions in a sequence, and order of which the elements are enumerated matters, that is if we have two sequences $(1, 2, 3)$ and $(3, 2, 1, 1)$, $(1, 2, 3) \neq (3, 2, 1, 1)$. A

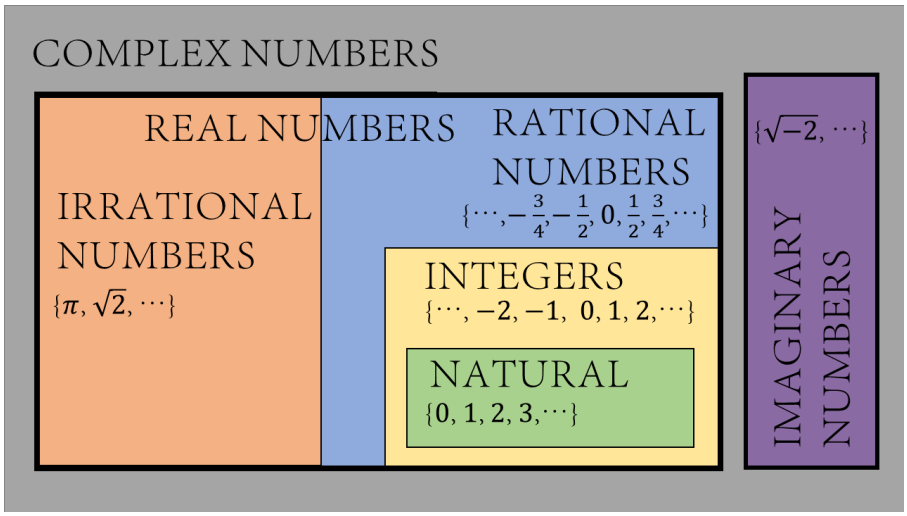

Figure 2.6: A Venn Diagram Showing the Relationship of Number Sets

sequence is usually denoted by parentheses ('(' and ')'), for example, the famous Fibonacci sequence is given as $(0, 1, 1, 2, 3, 5, 8, \dots)$. Mathematical objects, functions or relations are usually described as sequences.

33. The elements of a sequence are called **terms**.

34. The number of elements of a sequence is called the **length** of that sequence.

35. A sequence may be **finite** in length (ex. $(1, 2, 3, 4, 5)$) or **infinite** (ex. $(1, 2, 3, \dots)$) as in sets.

36. Similar to sets, we can define inclusion to a sequence by:

    (a) The **roster** method, generating all its elements, we must be sure that the sequence is finite.

    (b) In case that the sequence may be infinite or has too many elements to list, then we use a **rule**. An example is 'the sequence of alternating 0's and 1's, starting with a 0', $(0, 1, 0, 1, 0, 1, \dots)$.

    (c) We can also use a **formula**. For example, the sequence generated by $(a_n)_{n\in\mathbb{N}} = 2n + 1$ is the sequence of odd numbers starting from 3, $(3, 5, 7, 9, \dots)$.

37. In order to specify which element is being called, we say "**the** n^{th} **term**" of a sequence. For example, given the same sequence $(a_n)_{n\in\mathbb{N}} = 2n + 1$ if we want to know the 3rd element of the sequence, we write '$a_3 = 7$', we say "the third term of the sequence is the number 7".

38. A **permutation** is related to the act of arranging items of a set into some sequence or order. The number of all possible arrangements of a set of N items is given by $N!$. If we have the set $A = 1, 2, 3$, the permutations of set A is given as follows:

    - $(1, 2, 3)$
    - $(1, 3, 2)$
    - $(3, 1, 2)$
    - $(3, 2, 1)$
    - $(2, 3, 1)$
    - $(2, 1, 3)$

39. A **point** is a location. It has neither width nor length, even though it is visually represented as a dot for reference.

40. Locations are usually made up of a sequence of numbers called **coordinates**.

41. A **line** is one-dimensional, having length but no thickness. A line is composed of infinite points as it extends infinitely in both directions however, two points are enough to define a line. For example, if we are given two connected points A and B, then make-up the line $\overleftrightarrow{AB}$.

42. A **real number line** is a line wherein each point is associated to some real number $r \in \mathbb{R}$ This makes sense because the set of real numbers is infinite. Since each point is represented as a real number, the coordinate of any point on the line is given by a real number. A visual representation of a real number line is shown on figure 2.7. As previously stated, if we want to know where a point is on the line, we simply tell what number the point represents. Hence, we also know the distance from which the point is located from our reference point, 0.

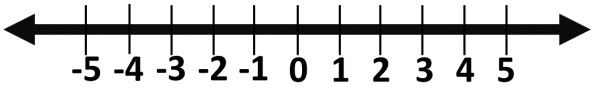

Figure 2.7: A Real Number Line

43. A part of a line that has defined endpoints is called a **line segment**. A line segment as the segment between A and B is written as: $\bar{AB}$. Two lines that meet at a point are called **intersecting lines**. Perpendicular lines are two line that form a 90 degree angle.

44. We say that a set of points are **collinear** if there is a line that passes through all the points.

45. A **plane** is a two-dimensional surface. Ruled and spanned by two independent perpendicular lines. A plane is defined by three non-collinear points.

46. A **coordinate plane** is a plane that is spanned by the real number lines, x-axis and y-axis hence, it is also known as the space R^2. Each point on this plane represents a pair of coordinates (x, y). We usually assign the first number, x, for the distance on the x-axis and the second number, y, for the distance on the y-axis. A coordinate plane is shown on figure 2.8. As we can see, the black point is said to be located at (1,4) this means that it is 1 unit away from (0,0) on the x-axis and 4 units away from (0,0) on the y-axis.

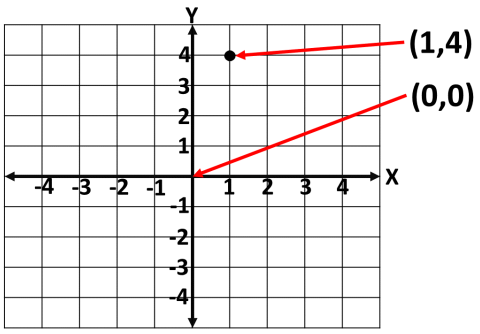

Figure 2.8: A Coordinate Plane

47. A **Vector** is a quantity having both magnitude and direction. In a coordinate plane, it is represented by an arrow as shown in figure 2.9. We can see that vector $a =< 1, 1 >$ is 1 unit to the right of the point (0,0) and 1 unit above the point (0,0). A vector is mainly composed of two points in N dimensions, represented by the points on its tail and its head but it these two points are arbitrary because vectors are only concerned with magnitude and distance but not location. Magnitude is visually represented in length. Direction, one the other hand, is visually represented by the arrowhead. A vector is usually given in the form $< x_1, x_2, x_3, \ldots x_n >$ where each component x_i is the absolute numerical distance between two points in dimension $i \in (1, 2, \ldots, n)$. When representing vectors in two dimensions, it is broken down into two parts, x and y components. The x component is the horizontal length while the y component is the vertical length. The vector's magnitude ($|a|$) is given by the 2D Pythagorean theorem: $|a| = \sqrt{x^2 + y^2}$ where x and y are its x and y components. In higher dimensions, the same representations follow and the Pythagorean theorem for higher dimensions are used. $|a| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$ where each x_i is the component of the vector in dimension $i \in (1, 2, \ldots, n)$.

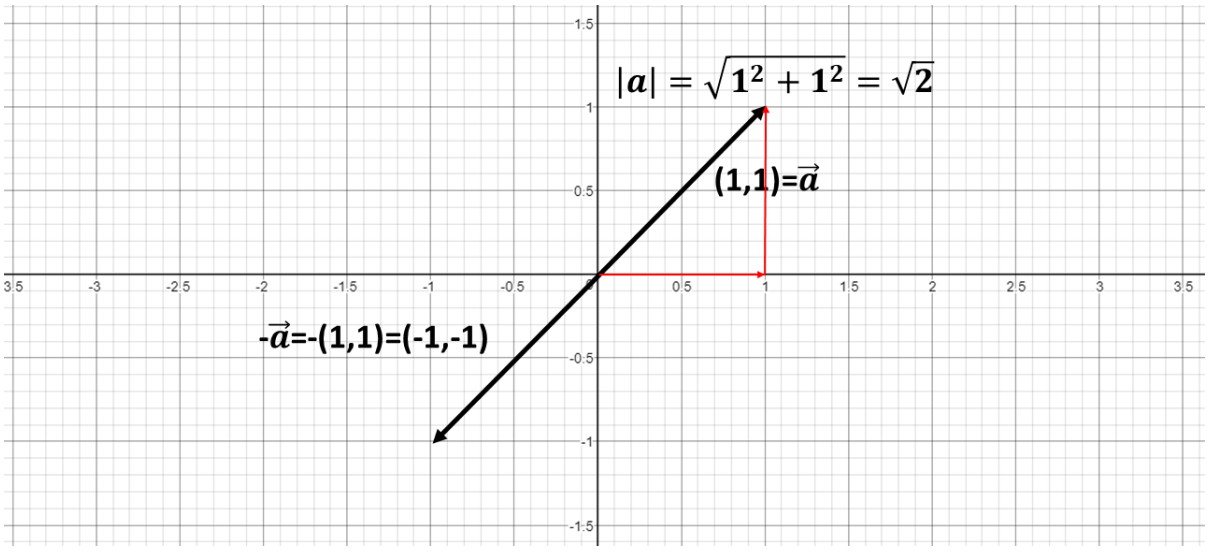

Figure 2.9: Vectors in a Coordinate Plane

48. The number given by the Pythagorean theorem is also known as the **Euclidean**

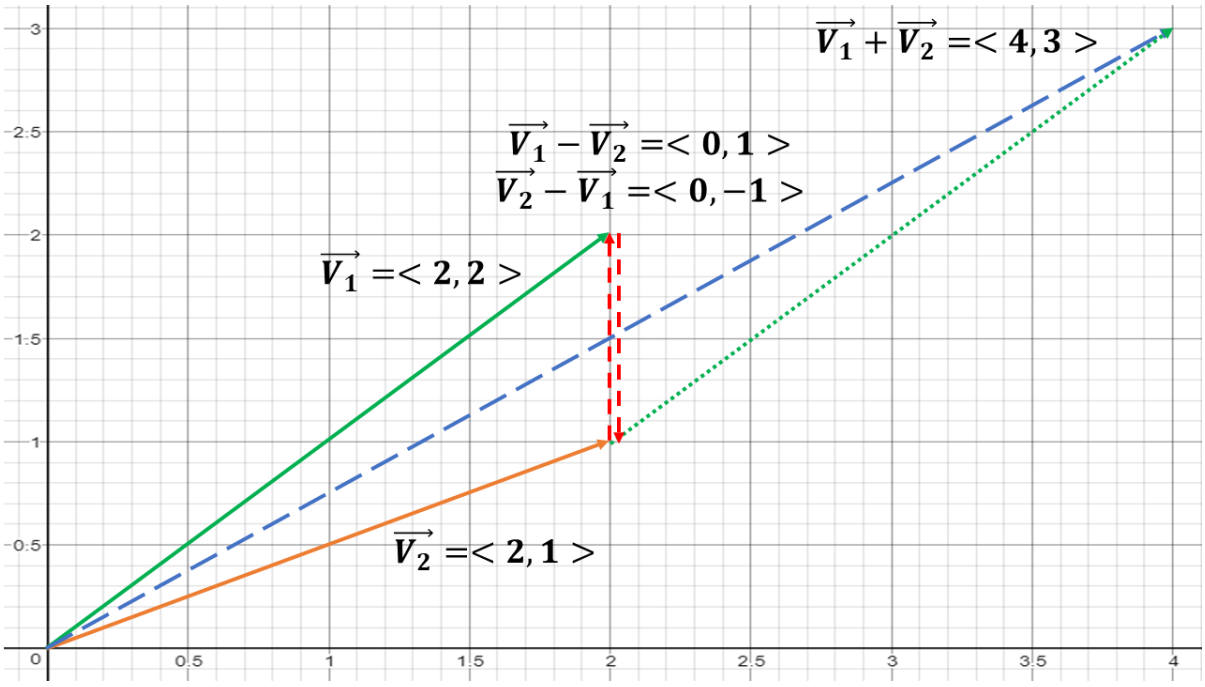

Figure 2.10: Adding and Subtracting Vectors (Coordinate Plane)

distance from two points, $(x, 0)$ and $(0, y)$. Euclidean distance is the length of the shortest possible path through space between two points that could be taken if there were no obstacles in between them.

49. The **negative of a vector** is simply a vector having the same magnitude but of opposite direction as seen on figure 2.9. We can see that the vector $-a$ has the same magnitude but opposite of the direction of vector a. Adding and subtracting vectors are simple in that each component of vector A is added or subtracted to the respective components of vector B. For addition, $C = A + B$ can be written as $(x, y) =< 1, 2 > + < 3, 4 >=< 3, 6 >$. For subtraction, $C = A - B$ can be written as $< x, y >=< 1, 2 > +- < 3, 4 >=< 1, 2 > + < -3, -4 >=< -2, -2 >$. An example is seen on figure 2.10. As we can see, if we add two vectors, $V_1 and V_2$, the resulting vector $< 4, 3 >$ is longer than both vectors if they are both in the same direction. If we subtract the vectors, $V_1 and V_2$ the resulting direction will depend on which vectors are considered as the minuend and subtrahend.

    If we consider a vector in dimension 3, then we will have to add to its components.

Its components are now x, y and z where x is its length, y is its height and z is its width. In general, if we have a vector in dimension n, it is defined with n components.

In this document, we consider the velocity of an object inside a defined virtual space of dimension n.

50. **Velocity** is defined in physics as speed with direction. For example, if an object has a speed of 9 m/s then we can say that the object is simply covering a distance of 9 metric units at each time step but if we state that the object has a velocity of 9 m/s to the right, then we can say that the object is covering a distance of 9 metric units at each time step to the right of its current position. It is important that take note that vectors usually involve two ordered n-tuples that give its original and final positions.

51. An **Array** is a collection of objects, having shared some similar properties, arranged in a particular order. An array is usually contained in rows and columns.

    Arrays are denoted by the syntax ArrayName[Size][Size] wherein every [] denotes a **dimension**.

    For example we have the array MyArray[3] it is an array of one-dimension having 3 elements. Take note that the size is sometimes omitted to represent variability. In simple terms, an array is like a series of boxes that contain elements with some similar properties. If we have an array of dimension 2 (MyArray[X][Y]) then we have X rows of Y boxes. A visual representation is shown on figure 2.11. As we can see, the array A[2][5] has two rows and 5 columns. Each element occupies a single box.

    It is common notation to access elements of arrays by its index. Indexing usually starts from 0. In figure 2.11 the red numbers indicate indices of the elements. For example, if we want to access the first element in A from figure 2.11, we say A[0][0]. If we want to access element 'H' in A, we say A[1][2].

    In this paper we will be dealing with arrays that whose element are vectors and coordinates.

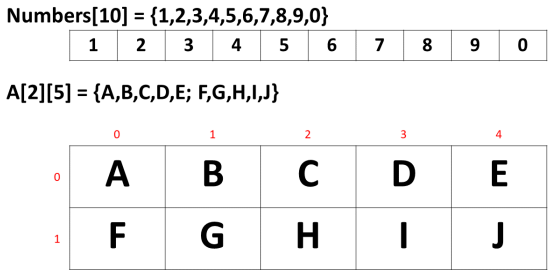

Figure 2.11: Visual examples of arrays

52. A **matrix** is an array of numbers.

The **dimensions** of a matrix is the number of rows and columns of the matrix in that order. A 'two by three' matrix is an array with two rows and three columns. A 'three by two' matrix is an array with three rows and two columns. To show this, we let $M1$ be a 2×3 matrix and $M2$ be a 3×2 matrix.

$$M1 = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}, \; M2 = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}$$

We access the elements of a matrix the same way as we do for arrays. An example is $M1[1][1] = a_{11}$

A matrix whose row and column have the same dimension is called a **square matrix**.

The operations that can be done for matrices are as follows:

[**Matrix Addition**] Adding two matrices means that we add their corresponding elements. We can only add matrices with the same dimensions. Let two matrices $M3$ and $M4$ be matrices of the same size $n \times m$, then matrix addition $M3 + M4$ is done as

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1m} \\ b_{21} & b_{22} & \dots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nm} \end{bmatrix} = \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \dots & a_{1m}+b_{1m} \\ a_{21}+b_{21} & a_{22}+b_{22} & \dots & a_{2m}+b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}+b_{n1} & a_{n2}+b_{n2} & \dots & a_{nm}+b_{nm} \end{bmatrix}$$

[**Multiply by a Constant**] Multiplying a constant number c to a matrix M is done by multiplying the constant to every element of the matrix.

$$c \cdot \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1m} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nm} \end{bmatrix} = \begin{bmatrix} c \cdot a_{11} & c \cdot a_{12} & c \cdot a_{13} & \dots & c \cdot a_{1m} \\ c \cdot a_{21} & c \cdot a_{22} & c \cdot a_{23} & \dots & c \cdot a_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c \cdot a_{n1} & c \cdot a_{n2} & c \cdot a_{n3} & \dots & c \cdot a_{nm} \end{bmatrix}$$

[**Negative of a Matrix**] The negative of a matrix is just the matrix multiplied to the constant $c = -1$ hence, all elements are multiplied to -1.

$$- \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1m} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nm} \end{bmatrix} = \begin{bmatrix} -1 \cdot a_{11} & -1 \cdot a_{12} & -1 \cdot a_{13} & \dots & -1 \cdot a_{1m} \\ -1 \cdot a_{21} & -1 \cdot a_{22} & -1 \cdot a_{23} & \dots & -1 \cdot a_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 \cdot a_{n1} & -1 \cdot a_{n2} & -1 \cdot a_{n3} & \dots & -1 \cdot a_{nm} \end{bmatrix}$$

[**Matrix Subtraction**] Matrix subtraction is just the addition of two matrices where the addend is negative. Note that here, we can only subtract matrices with the same dimensions. Let two matrices $M3$ and $M4$ be matrices of the same size $n \times m$, then matrix subtraction $M3 - M4$ is done as

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix} - \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1m} \\ b_{21} & b_{22} & \dots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nm} \end{bmatrix} = \begin{bmatrix} a_{11} - b_{11} & a_{12} - b_{12} & \dots & a_{1m} - b_{1m} \\ a_{21} - b_{21} & a_{22} - b_{22} & \dots & a_{2m} - b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} - b_{n1} & a_{n2} - b_{n2} & \dots & a_{nm} - b_{nm} \end{bmatrix}$$

[**Hadarmard Product**] The Hadamard Product is a **component/element-wise multiplication** where each element is multiplied to the corresponding element of the other matrix. Note that here, we can only multiply matrices with the same dimensions. Let two matrices $M3$ and $M4$ be matrices of the same size $n \times m$, then their hadamard product is given by $M3 \circ M4$ is done as

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix} \circ \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1m} \\ b_{21} & b_{22} & \dots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nm} \end{bmatrix} = \begin{bmatrix} a_{11} \cdot b_{11} & a_{12} \cdot b_{12} & \dots & a_{1m} \cdot b_{1m} \\ a_{21} \cdot b_{21} & a_{22} \cdot b_{22} & \dots & a_{2m} \cdot b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} \cdot b_{n1} & a_{n2} \cdot b_{n2} & \dots & a_{nm} \cdot b_{nm} \end{bmatrix}$$

[**Matrix Multiplication**] Matrix multiplication is not the Hadamard product. Matrix multiplication involves the sum of products. If A is an $n \times m$ matrix and B is an $m \times p$ matrix, their matrix product AB is an $n \times p$ matrix, in which the m elements across a row of A are multiplied with the m elements down a column of B, the resulting elements are then summed to produce an entry of AB. Let two matrices A and B be matrices of the sizes $n \times m$ and $m \times p$ respectively, then their matrix product is given by $A \times B$ is done as

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix} \circ \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1p} \\ b_{21} & b_{22} & \dots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mp} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1p} \\ c_{21} & c_{22} & \dots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \dots & c_{np} \end{bmatrix}$$

such that

$$c_{ij} = a_{i1} \cdot b_{1j} + a_{i2} \cdot b_{2j} + \cdots + a_{im} \cdot b_{mj} = \sum_{k=1}^{m} a_{ik} \cdot b_{kj}, \forall i = 1, 2, \dots, n \text{ and } j = 1, 2, \dots, p$$

An example is

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \times \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} = \begin{bmatrix} z_{11} \end{bmatrix}$$

$$z_{11} = 1 \cdot 4 + 2 \cdot 5 + 3 \cdot 6 = 32$$

53. The **transpose** of a matrix (denoted as M^T) is a matrix where the rows and columns are swapped. That is

$$M3^T = \begin{bmatrix} a_{11} & a_{21} & \dots & a_{n1} \\ a_{12} & a_{22} & \dots & a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1m} & a_{2m} & \dots & a_{mn} \end{bmatrix}$$

An example is

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \; A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

54. A matrix is said to be **symmetric** if and only if the matrix M is equal to it's transpose M^T. Given by $M = M^T$. An example is

$$O = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix} = O^T = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix}, \therefore O \text{ is symmertric}$$

55. A **graph** G is a mathematical object composed of two sets, a finite set V called the **vertices** and another set E whose elements are pairs of vertices called **edges**, expressed as $G = (V, E)$.

56. A **vertex**, also called a **node**, is the fundamental unit needed to construct graphs. They are visually represented as points in some space S having N dimensions. In this document, they are used to represent real world objects. Later, we will assign numbers to these points to achieve discreteness, (to know what they are and what they are not).

57. **Edges** are visually seen as lines that connect vertices, they show that those vertices are related in some way. Edges usually connect two vertices, they represent and show that there exists a relationship between these vertices. If there are no edges that connect a pair of vertices, then it an be said that there is no direct relationship between those edges.

58. If two vertices $u, v \in V$ are connected by some edge $(u, v) \in E$, and if the edge $(v, u) \in E$ is the same edge, then we say that vertices u and v are connected by the **undirected edge** (u, v) (or (v, u)).

59. We also say that the vertices $u, v \in V$ are **adjacent** because an undirected edge connects them.

60. A graph G is called an **undirected graph** if and only if it is made up of undirected edges.

61. However, if edge $(u, v) \in E$, and $(v, u) \in E$ are not the same edges, then we say that (u, v) is a **directed edge** from vertex u (called the edge's 'tail') to vertex v (called the edge's 'head').

62. If edge $(u, v) \in E$ but $(v, u) \notin E$, then we say that vertex u is **adjacent** to vertex v but vertex v is not adjacent to vertex u.

63. A graph G is called a **directed graph** if and only if it is made up of directed edges.

64. A graph with which every pair of vertices $u, v \in V$ is connected by an edge $(u, v) \in E$ is called a **complete graph**, denoted as $K_{|V|}$. That is, there exists an edge (u, v) in set E for any pair of u and v in set V (expressed as $\exists(u, v) \in E \ \forall u, v \in V$).

65. A graph is said to be a **weighted graph** if numbers are assigned to it's edges. These numbers are called **weights** or **costs**.

66. A **path** from vertex u to vertex v of a graph is defined as a sequence of adjacent vertices (connected by edges) that start from u and end with v.

67. If all vertices of a path are distinct, then the path is said to be **simple**.

68. The **length** of a path is the total number of edges in the path.

69. A **directed path** is a sequence of vertices in which every consecutive pair of the vertices u and v is connected by a directed edge from u to v.

70. A graph is said to be **connected** if for every pair of vertices u and v in set V, there exists a path from u to v.

71. A **cycle** is a path of positive length (at least one edge) that starts and ends at the same vertex and does not traverse the same edge more than once.

72. A graph with no cycles is said to be **acyclic**.

73. An **adjacency matrix** is a square matrix that shows the relationships of vertices in a graph. Each dimension in the matrix is assigned a vertex. The elements of the matrix is from the set $0, 1$. The element $M[u][v] = 1$ if there is an edge that connects vertices u and v, otherwise, is it 0. The unweighted graph in figure 2.12

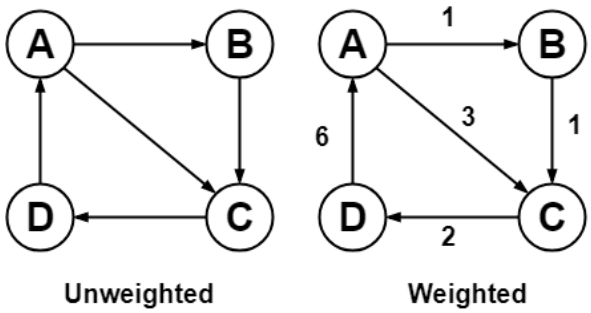

Figure 2.12: Sample Graph

has the adjacency matrix:

$$\begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \left[\begin{array}{cccc} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{array}\right] \end{array}$$

If the graph has weights then we replace the 1's with their respective weights. The adjacency matrix of the weighted graph in figure 2.12 is:

$$\begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \left[\begin{array}{cccc} 0 & 1 & 3 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \\ 6 & 0 & 0 & 0 \end{array}\right] \end{array}$$

Note that for an undirected graph, the adjacency list is symmetric.

74. The **shortest path problem** is the problem of finding a path between two vertices (or nodes) u and v in a graph G such that (a) if G is unweighted, the total length of the path is minimized; (b) if G is weighted, the sum of the weights of the edges in the path is minimized.

    The well known algorithms used to solve the shortest path problem are as follows:

    (a) **Dijkstra's Algorithm** which solves the shortest path problem with non-negative weights. It is an algorithm for solving the single-source shortest

path, which means that it solves the shortest path from any node $u \in V$ to any other node $v \in V$.
The dijkstra's algorithm uses a priority queue.

A **queue** is a list where the elements are inserted at one end and are removed at the other.

A **priority queue** is a queue wherein each element is associated with a value which dictates whether or not that element is highly likely to be selected/removed from the queue. An element with high priority is served before an element with low priority. If two elements have the same priority, they are served according to their order in the queue.
The dijkstra's algorithm is:

i. Select a source vertex s from the set of vertices V

ii. Create an empty priority queue Q

iii. For each vertex v in the Graph, do the following
   - Set the distance from the source s to vertex v as infinity (∞)
   - Set the optimal path from the source s to node v as empty
   - Add vertex v to the priority queue Q

iv. Set the distance of vertex s from itself as 0

v. While Q is not empty, do the following:
   - Select vertex u from the priority queue Q with the minimum distance
   - Remove u from the queue
   - For each vertex w, still in the queue, adjacent to u, do the following:
     - Compute the path from the source vertex s to the node w that passes through u before it reaches w
     - If the newly computed path is shorter than the current one,
       Update the distance from the source vertex s to vertex w
       Add vertex u to the path of w

The flowchart of the algorithm is seen on figure 2.13.

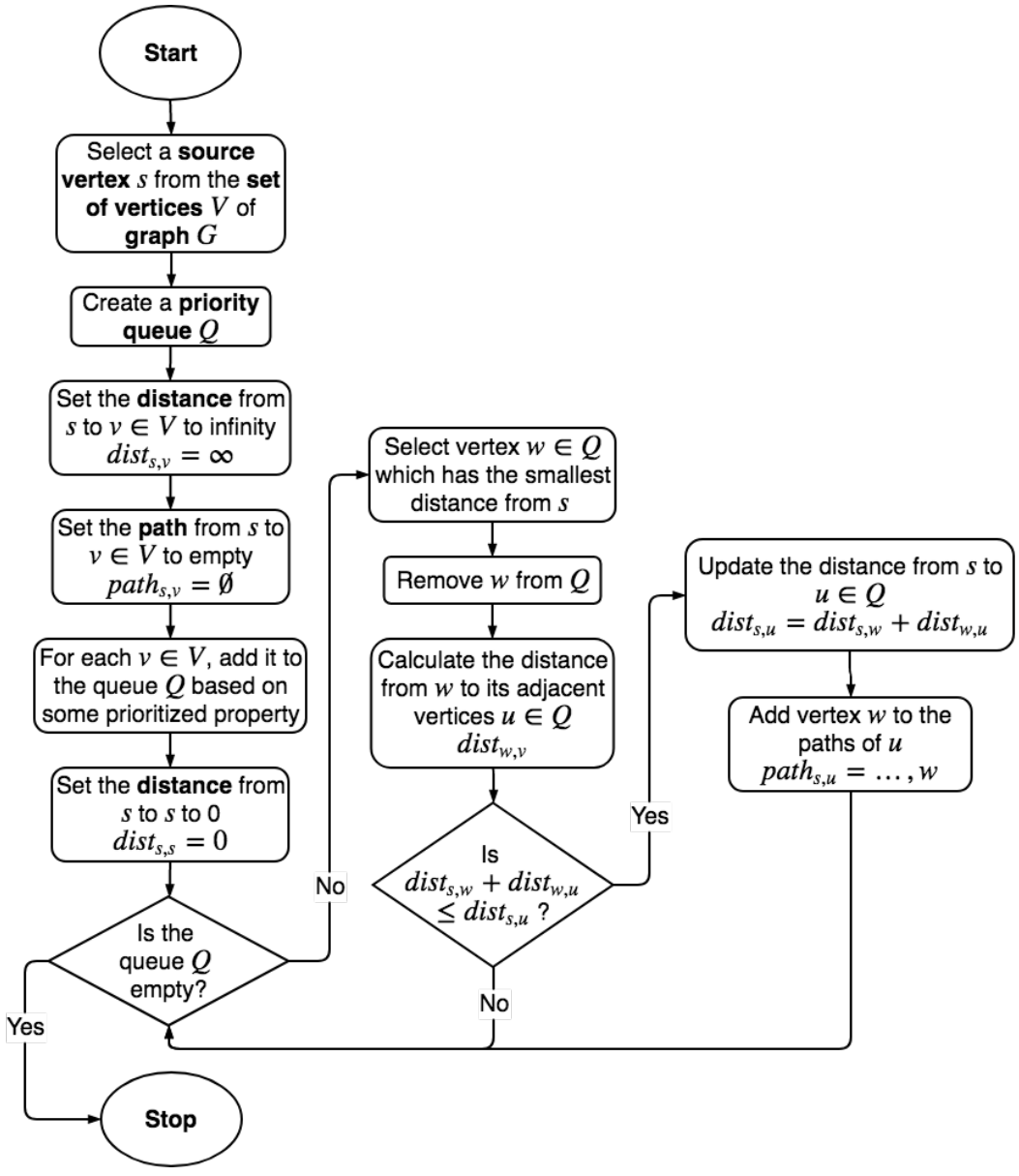

Figure 2.13: Flowchart of the Dijkstra Algorithm

(b) **Floyd-Warshall Algorithm** which solves the shortest path for any two node u and v in V. The floyd-warshall algorithm starts off with the adjacency matrix of the graph G. All non-existent edges have the value of infinity ∞. The algorithm takes advantage of the transitivity in order to replace the infinite values. Transitivity is the relation wherein if a property holds between the first and the second and also holds between the second and the third, then it follows that this property also hold between the first and the third. It can be simplified as "if one can go from a to b and from b to c then one can go from a to c by passing through b."

The Floyd-Warshall algorithm is as follows:

i. Let $dist$ be a matrix of size $|V| \times |V|$ whose values are ∞

ii. For each edge, $(u, v) \in E$, set $dist_{u,v}$ as the weight of the edge (u, v).

iii. For each vertex $v \in V$, set the distance to itself as 0. $dist_{u,v} = 0$

iv. For each vertex $w \in V$, do the following:

- For each pair of vertices $u, v \in V$ do the following:
  - Check if the distance from u to v is greater than the distance from u to w and w to v. That is, check if $dist_{u,v} > dist_{u,w} + dist_{w,v}$
  - If that is true, set $dist_{u,v} = dist_{u,w} + dist_{w,v}$

The flowchart of the floy-warshall algorithm is seen on figure 2.14. An example is shown on figure 2.15.

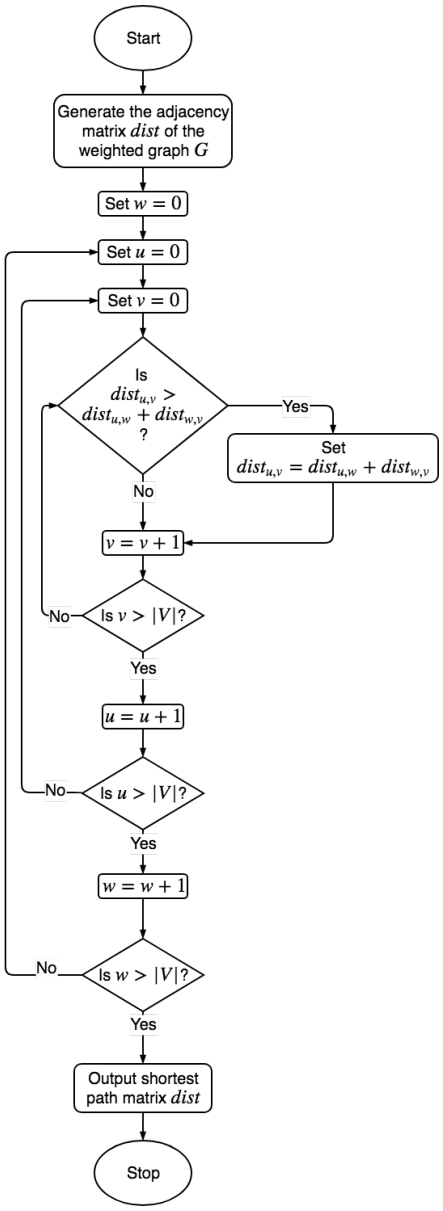

Figure 2.14: Flowchart of Floyd-Warshall Algorithm

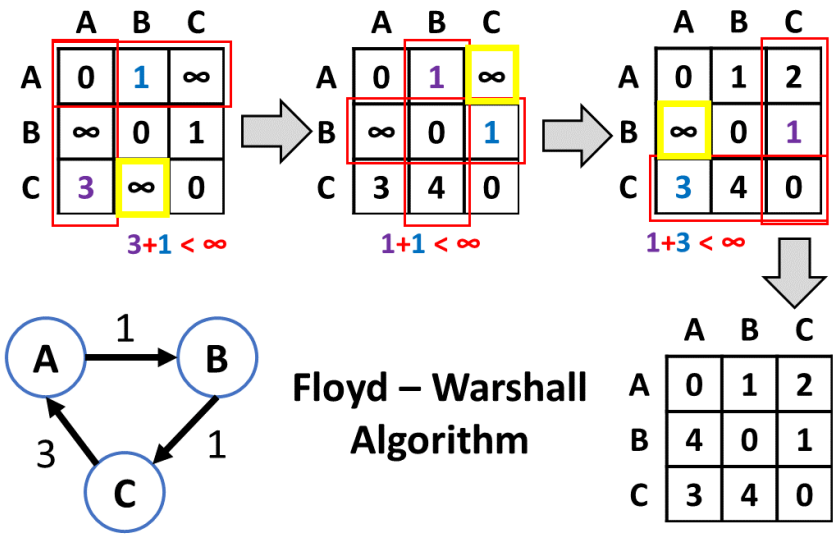

Figure 2.15: Floyd-Warshall Algorithm Example

An application of this algorithm involves finding a sequence of road segments that take a vehicle from a source to a destination using a graph that represents a road network. In this representation, we can let vertices be the source, destination, intersections, land marks, etc. whatever objects that can help split the entire road systems into road segments. We then let edges be the road segments between any two vertices. We assign the costs/weights to the edges based on some information that can help us understand and/or distinguish edges that are favorable to traverse. These costs may be quantified as actual distances, cost of fuel, average amount of travel time, traffic gradient, risks involved (bridge instabilities, accident proneness, etc.) and much more depending on the realism of the model and/or data availability. The model for the amount of cost can be as simple as minimizing the amount of distance traveled or as complex as maximizing the total amount of money gained after subtracting the total money expended on fuel (affected by both distance and time), car maintenance, driver salary, etc.

75. The **Traveling Salesman Problem (TSP)** is a problem involving generating a **Hamiltonian Cycle** from a graph G.

    A hamiltonian path is a path in a graph which contains each vertex of the graph

exactly once. A hamiltonian cycle is a hamiltonian path that starts and ends at the same vertex.

The problem description are as follows:

(a) A salesman needs to visit every city (represented by vertices)

(b) He/she does not care about the order of visiting each city. As long as he/she visits each one.

(c) He/she must start and finish at the same city

(d) Each city is connected to other close by cities, or nodes, by airplanes, or by road or railway. Hence, each of the connections between the cities has one or more weights (or the cost) attached depending on the availability of transportation means.

(e) The cost describes how "expensive" it is to traverse this edge on the graph, and may be given, for example, by the cost of an airplane ticket or train ticket, or perhaps by the length of the edge, or time required to complete the traversal.

(f) The salesman wants to keep both the travel costs, as well as the distance he travels to a minimum.

The aim is to generate a path or a sequence of nodes that lets the salesman pass through all cities at most once before returning to starting city and spends the minimum amount of travel expenses and distance.

The problem is mostly concerned about generating the best arrangement of N cities among $(N-1)!$ permutations. As we can see, the amount of permutations rapidly increases as the number of cities is increased. $N-1$ because we always start with the same given city.

76. The **Vehicle Routing Problem** is a generalization of the Traveling Salesman Problem. VRP is a problem that involves generating the best set of routes for a fleet of vehicles to service all customers in a graph. Here, there are more 'salesmen' (changed into 'vehicles' for formalities when we consider modern delivery services). VRP is concerned with delivering or collecting 'goods' to and/or from customers using a number of vehicles.

A **route** is a hamiltonian cycle which starts and ends at a depot.

A **depot** is where vehicles are stored or parked when they are not in use.

The usual way customers and road networks are set-up is to let vertices represent the depot and customers and let the edges represent road segments that connect the vertices. Another way is to represent clusters or customers as an edge, and let the vertices serve as road intersections. This is simple but it is too simple that it does not capture individuality of customers. Hence, the former is commonly used since most adaptable models are complex.

VRP is defined on a complete undirected graph $G = (V, E)$. The set of vertices $V = 0, 1, 2, \ldots, n$ where each vertex $u \in V - \{0\}$ represents a customer having a nonnegative demand q_u. The demand is usually the amount of goods (in some quantity) to be delivered or collected by the vehicle. The amount of goods can be measured in mass, weight, quantity, volume, bulk, etc. Vertex 0 is usually designated as the depot. Each edge $e \in E = (u, v)|u, v \in V$ is associated with a travel cost c_e or $c_{u,v}$. Travel cost may be in terms of distance (actual, euclidean, circular, manhattan, chessboard), time (travel time, time waiting in traffic), fuel cost (convert distance and time into amount of fuel and convert that number into how much money fuel costs), monetary cost (adding up expenses, salaries, penalties) etc. There are a total of k available vehicles in the depot. The vehicles are assumed to be homogeneous and all have the same carrying capacity Q. Carrying capacity refers to the maximum amount of goods that can be carried by a vehicle at any phase or time during it's traversal of the route. The task is to develop k routes whose total travel cost is minimized such that

- Each customer is visited exactly once by a route
- Each route starts and ends at the depot
- The total demand of customers served by a route does not exceed the vehicle capacity Q
- The length of the route does not exceed a preset limit L

The last item ensures that all the drivers have the same workload.

If we consider a directed graph, then we need only to change the edges and must produce directed cycles.

77. **Waste** is defined as materials that are unwanted or unusable. This refers to matter which are discarded after primary use or is deemed defective, or worthless.

78. There are various types of waste but our focus is one **municipal solid waste** or what we call household trash. This is the type of waste that is produced daily at households and communities. The term 'municipal' comes from the fact that it is the duty of the municipality to collect and manage these kinds of waste. A list of what is considered as municipal solid waste is as follows:

    - **Biodegradable waste** produced from food, cooking, paper etc.
    - **Recyclable materials** such as glass, bottles, jars, clothes, fabrics, rubber etc.
    - **Inert waste** such as dirt, rocks, debris
    - **Electrical and electronic waste** such as appliances, light bulbs, mobile phones, television sets
    - **Composite waste** such as toys, tetra packs, clothing
    - **Hazardous waste** such as paints, batteries, aerosol sprays, and fertilizers
    - **Toxic waste** such as pesticides, herbicides and fungicides
    - **Biomedical waste** such as expired pharmaceutical drugs, used medical equipment

79. **Waste collection** includes gathering, transportation, and delivery for disposal of solid waste and recyclable materials. Waste collection involves vehicles that collect and transport the waste from communities to facilities that receive, sort and process the waste. Processing the received waste may be in the form of incineration, rapid degradation, segregation, resource recovery, energy recover, etc.

80. **Residual waste** is the type of solid waste that is neither recyclable nor reusable.

81. An **Eco-Waste Recovery Services-Material Recovery Facility** is where final sorting of waste is done. Once sorted, the garbage is then moved to designated

areas for recycling, recovery, reuse, re-purposing, composting, etc. This facility reduces the amount of residual waste and also corrects any mis-segregated matter.

82. A **Geographic Information System (GIS)** is a collection of computer software, and data used to view, manage, analyze, and transform geographical information. A GIS provides a framework for gathering and organizing spatial data and related information such as temporal, visual, demographic, economic, etc. Out of the data available, it is able to produce analyses, maps, patterns, predictions, assessments and other forms of usable information. It can create fast and logical decisions, produce and display maps, graphs, charts and perform a vast quantity of calculations. An example is Google Maps which offers satellite imagery, street maps, 360 deg panoramic views of streets (Street View), real-time traffic conditions (Google Traffic), and route planning for traveling by foot, car, bicycle (in beta), or public transportation. These kinds of information was produced through available data and some algorithms which processes the data for generating visuals and graphics, route creation, land mark associations etc. Data that is stored in a database is placed in several layers of maps and graphs that have common properties. These layers can come in the form of roadways, vegetation patterns, layout of buildings and structures, traffic information, physical layout of the environment, temperature and pressure maps, radiation maps, demographics, environmental compositions, sets of images and videos, etc. Informally, a physical map is itself a GIS. Within it, is information which can be read and analyzed to produce observations, inferences, hypotheses, predictions, plans, patterns, etc. Basically, it is anything that can tell you something about a place. It has been used in businesses for needs assessments, sales predictions, discovering patterns of customer interests, discovering trends in purchases and demand.

83. **Deterministic** means that the next procedure/step is known without having any other choice. There is no randomness involved.

84. **Non-deterministic** means that there are multiple available decisions that can be done at a certain circumstance. This helps examine the ability to make decisions based on the statements.

85. **Decision problems** is any yes-or-no question that involves an infinite set of inputs. These inputs are logical objects, be it numbers, graphs, strings, or sets. The input is broken down to its properties and based on those properties, the question is thrown an affirmation or negation. For example, "is 4 an element of $\mathbb{Z}$? Well we can compare all numbers in $\mathbb{Z}$ to 4 and this will of course be true, hence the answer returned is 'yes' 4 is an element of $\mathbb{Z}$.

86. Nondeterminstic Polynomial time **NP** is a set of problems with the same resource-based complexity used to describe certain types of decision problems. NP is the set of all decision problems for which the instances where the answer is "yes" are efficiently verifiable through deterministic computations that can be performed in polynomial time. A problem belongs to the $NP-hard$ or the set of the "hardest" NP problems if there is no known polynomial time algorithm that can provide an optimal solution.

87. A **constrained optimization problem** is a problem that is bounded by some limiting factors. According to Garg[13], Constrained optimization problems are defined as:

$$\text{Minimize } F(x)$$

that is subjected to p equality constraints,

$$h_k(x) = 0; \qquad k = 1, 2, \ldots, p$$

and q inequality constraints,

$$g_j(x) \leq 0; \qquad j = 1, 2, \ldots, q$$

where each set of decision variables x is in D dimensions such that $x = [x_1, x_2, x_3, \ldots, x_D]^T$. Each element of x are bounded as

$$l_i \leq x_i \leq u_i; \qquad i = 1, 2, \ldots, D$$

where l_i and u_i are the minimum and maximum permissible values of each element x_i.

88. **Feasible solutions** are solutions that do not violate any constraint imposed in a constraint optimization problem. The solution space S is also called the 'feasible space' since it composes of all solutions that are within the limitations of the problem.

89. **Infeasible solutions** are solutions that do violate any constraint imposed in a constraint optimization problem.

Chapter 3

Review of Related Literature

As mentioned in the previous chapter, waste collection problems have been solved through the use of the vehicle routing problem. We take a look at some of the studies conducted in solving vehicle routing problems and modeling waste collection into a vehicle routing problem. We then move on to discus the basic PSO and GA algorithms. Finally, we discuss the hybrid PSO-GA approach of Harish Garg.

3.1 Vehicle Routing Problem

As stated, Dantzig and Ramser[8] were the first to state the vehicle routing problem. Their paper focused on routing a fleet of gasoline-powered delivery trucks that deliver fuel from a 'bulk terminal' to a large number of service stations. The bulk terminal is a facility used for storing and/or marketing of petroleum products. Service stations are facilities where gasoline-powered vehicles refill their tanks, usually, vehicle repair is also included in the services offered. They stated that the traveling salesman problem, at its core, is merely concerned with determining the shortest possible route which passes through each of the n given cities exactly once. Assuming that for each pair of cities, there exists some link that directly connects them, then therefore the total number of distinct routes through n cities is given by $\frac{1}{2}n!$. This is because the sequence at which cities are visited is the same as in the revere order since the vehicle returns to the same point. The generalization of the TSP is made by adding more conditions to the problem. They basically inquired about the possible outcome when there was a limit to the number of cities that the salesman can visit before returning to the origin city. The salesman would have to take an increasing number of shorter trips every time the maximum number of cities that can be visited is reduced. The context was changed into a delivery

truck delivering fuel to some service stations. Hence, a limit was introduced by giving each service station i a quantity q_i which the vehicles needs to deliver. The vehicle is imposed to only have the ability to carry a total quantity of C every time it is deployed from the bulk terminal where $C > q_i$ for any service station i. Hence, the number of service stations that can be serviced by the vehicle is determined by how much each q_i is. The vehicle is now forced to make multiple deliveries when the sum of all q_i's is greater than the capacity C. The main goal was still the same, minimize total travel distance covered by the vehicle. Another interpretation made from imposing the limit is that instead of a vehicle taking on multiple trips, each trip is assigned a vehicle hence, there are multiple vehicles with the same capacity C that deliver fuel to the same set of service stations. Dantzig and Ramser solved the truck dispatching problem using integer linear programming. This is a method where the solution is obtained by using the linear relationships of the mathematical models in terms of their graphs. A limited space is usually produced when the equations are plotted, this space is called the solution space. The best solutions are then identified using the points near the boundaries of the space.

In 1987, Solomon[19] proposed some methods of constructing routes. He tested them on some problems sets he created called the "Solomon Benchmark Problems" which are used for benchmarking Vehicle Routing Problems with Time Windows solution methods. The first is the savings heuristics which starts out with each customer having dedicated routes, then the algorithm tries to combine the best pair of routes until the minimum amount of routes are produced. The best pair of routes is decided by some savings equation which gives the amount of cost saved if two routes are combined rather than separate. The next heuristic is a greedy approach, the time-oriented nearest-neighbor heuristic which starts by selecting the "closest" (in terms of travel distance and time) node from the depot and attaching it to the current route. The process is repeated but this time the 'closest' node from the last added node is obtained and attached until the schedule is full. The algorithm proceeds to create the next route if there are still un-routed nodes. The next heuristic introduced is the insertion heuristic which also constructs routes sequentially. The route starts as two depot nodes. Each customer node is then inserted between two consecutive nodes in the route. The best location for insertion

is determined by some function that shows how efficient the route becomes after insertion. There are three proposed ways of evaluating the efficiency of the routes each called the I1, I2, and I3 respectively. I1 evaluates the route by distance and start of service time; I2 evaluates the route by total distance and total time; I3 evaluates the route by a combination of total distance, total travel time, and total time vehicles are late. When the node is inserted, the nodes succeeding it in the path are *pushed forward*, meaning that servicing these nodes are adjusted based on how much time and distance is used to accommodate the inserted node. The last heuristic discussed is the time-oriented sweep heuristic which groups customers into clusters and assigns each cluster to a vehicle. The route construction and scheduling is then done for each cluster of nodes associated to a vehicle. The results show that among the proposed heuristics, the insertion algorithm (specifically the I1) proved to be the most effective is solving the benchmark test cases because it focuses more on correct node sequencing rather than grouping and assigning customers to vehicles.

In 2000, Son[27] utilized a Chaotic Particle Swarm Optimization (CPSO) algorithm to generate routes and schedules of the different waste collection vehicles at Danang City Vietnam. The CPSO obtained data on the roads and waste collection facilities from a Geographic Information System (GIS) that simulates a continuous environment from a model of the road networks and waste collection system of Danag City. The information used by the GIS are a collection of real data obtained through a span of time. From this data, the average amount of waste collected at an area is known and is then simulated to vary based on the average amount. Traffic and other variables taken into consideration are also simulated the same way. There are three different kinds of vehicles available, namely, tricycles, hook-lifts and forklifts which take up different roles in the waste collection system. The objective in this case was to create a schedule that maximizes the amount of garbage collected in the simulation.

In 2005, Nuortio et.al.[20] improved the inflexible and inefficient waste collection scheduling and routing in Eastern Finland by creating a GIS model that is made based on the available road network and waste collection data. They employed a hybrid insertion heuristic for generating the initial population. A guided variable neighborhood thresholding meta heuristic was then used for improving the initial routes. This heuristic

is based on three principles, (1) guided local search, which performs a search on the search space S with the intent of finding the local optima. The decision of selecting which part of the search space to explore is based on a deciding factor that 'guides' the search. (2) variable neighborhood search which explores a particular local search space while executing the same local searching approach on adjacent neighborhoods (local search spaces) and switches the current local search space being explored with the neighborhood that shows a better or promising solution. (3) Threshold accepting is a method of evaluating the solutions found and judging whether or not the solution is a good enough approximation of the best solution. This is done for when obtaining the best solution becomes inefficient in terms or resources so instead, it is better to settle for a close approximate. The result of their experimentation showed that the schedule produced by the heuristic significantly reduced traveling distance of vehicles.

In 2007, Cordeau et.al.[7] compiled and defined general models of the vehicle routing problem and it's extensions (i.e. capacitated, time windows, etc.). They also collated and cited several approaches done by researchers over the years to tackle the VRP and it's extensions. They give a brief description of the process of how each algorithm solves the problems presented.

In 2012, Burhkal et.al.[3] set-up a model for waste collection vehicle routing problem with time windows (WCVRPTW) with lunch breaks based on two test cases, namely that of the (1) Waste Management Inc. which is responsible for waste collection in parts of Northern America and (2) the Henrik Tofteng Company responsible for handling waste collection at Denmark. These two cases have different policies for lunch break hours, limits on the number of customers served per route, and total amount collected at each route. They provided both cases with solutions using an adaptive large neighborhood search heuristic. Neighborhood search is a technique that tries to find good or near-optimal solutions to a combinatorial optimization problem by repeated transformation of a current solution into different solutions in its 'neighborhood'. The neighborhood of a solution is a set of similar solutions obtained by relatively simple modifications to the original solution (i.e. swapping two nodes in the route). For a large-scale neighborhood search, the neighborhood produced of a solution is relatively numerous in count compared to the regular one. The 'adoptive' part stems from the fact that the algorithm

tries to improve the solution by adjusting the neighborhood produced using the current known solutions at each iteration or time-step. They found that the algorithm produced considerable improved routes from those being used by the two companies.

In 2015, Akhtar, Hannan and Basri[2] proposed a method of solving Waste Collection Vehicle Routing Problem by node clustering in order to simplify the problem. They distributed customers into bins and modeled a traveling salesman problem for each cluster/bin. They then applied the Particle Swarm Optimization algorithm to find optimal routes for each TSP. This method is based on the notion of divide-and-conquer however, note that the clustering method used is significantly important since it determines which nodes go to which route. In their approach, they used smart bin technology which sends information about each bin specifically the .

Masrom et.al.[1] developed a hybrid PSO by incorporating the mutation mechanism of GA. Each particle is made up of $n + 2m$ components where n is the number of customers and m is the number of vehicles. The initial population is made through assigning real numbers to each component. The n components associated with customers are distributed to m vehicles using the real numbers. The customers are assigned to the vehicle whose associated real number is closest to the customers' real number. PSO is followed in each iteration and Mutation only occurs when the population's total health is low. A 'healthy' particle is one that changes it's personal best at each iteration. A population with low total health is a population where the majority of particles have not changed their personal best positions therefore we can say that the population might have fallen into a local optima and has stagnated. Mutation maintains that the population keeps moving even if only at a small distance. Both PSO and GA algorithms are discussed later in this document.

Lu et.al [18] also developed a hybrid PSO algorithm however this time, the crossover mechanism of GA was used. Each particle has $n+l$ components where n is the number of customers and l is the number of vehicles. Integers are assigned to each component which allows for the creation of the initial population. The components are arranged using the integers hence the routes are created. A vehicle's route is the sequence of customer components between two vehicle components. Crossover is done using two particles, a subsequence of the same size is selected in each particle and removed from that particle.

The subsequence is then placed at the beginning of the other particle's sequence. This is done for both particles, hence two new sequences are created which replaces the old ones. The results showed that the hyrbid PSO outperformed the basic PSO and basic GA algorithms.

In 1999, Tung and Pinnoi[30] conducted a case study wherein they investigated the refuse collection of a public company (URENCO) in five urban district of Hanoi, Veitnam. The aimed to improve its daily operation, particularly its their vehicle routes and schedule. The collection of waste involved two types of vehicles, motorized vehicles and manually pushed handcarts. The handcarts were used to manually gather refuse from each household or industrial unit. The refuse was then transported to gather sites where the motorized vehicles collect. Each gather site had a set schedule based upon the arrival of vehicles and the time it took for handcarts to deliver the refuse to the site. The motorized vehicles, after having been filled, transport the refuse to a landfill and return to servicing gather sites. The workers were separated into three shifts; morning, afternoon and night. They implemented both route construction and improvement methods. Route construction was done with the I1 insertion heuristic of Solomon. Route improvement manipulates the route constructed by the insertion heuristic in order to obtain better routes. Two route improvement methods were used, either method is invoked in an alternating or random pattern. The Or-opt exchange modification tries to improve a route by removing up-to-three adjacent nodes and reinserts them at different locations within the same route. The 2-opt operation on the other hand removes two edges, one from two selected routes and replaces them with edges wherein the first selected route is connected to the detached segment of the second route and the second route is connected to the detached segment of the first route.

3.2 PSO

Particle Swarm Optimization (PSO) is an optimization algorithm based on a simplified avian social model. PSO was proposed by Kennedy and Eberhart on 1995. [10][9] The PSO algorithm is seen on below. PSO was discovered from the attempts to simulate bird flocking and fish schooling. It has been used to solve a wide array of optimization

problems ranging from simple root finding to complex engineering optimization problems. The flowchart for the algorithm is shown on figure 3.1.

1. Generate an initial population of N members composed of random positions and velocities with d dimensions from the problem space.

2. For each particle x_{id} in the population, evaluate the fitness function values $F(x_{id})$, $i \in (1, 2, 3, \ldots, N)$

3. Compare the fitness value with the current particle's best value (pbest). (That is, $pbest$ compared to $F(x_{id})$)

   If the current fitness value is better than pbest ($pbest > F(x_{id})$), then set the pbest value equal to the fitness value ($pbest \leftarrow F(x_{id})$) as well as the pbest location to the current location of the particle in d-dimensional space ($pbest_{id} \leftarrow x_{id}$).

   else retain the pbest value and location.

4. Compare fitness value with the population's global best value (gbest). (That is, $gbest$ compared to $F(x_{bd})$, b is index of the best particle in the population)

   If the current overall best fitness value is better than gbest ($gbest > F(x_{bd})$), then set the gbest value equal to the overall best fitness value ($gbest \leftarrow F(x_{Bd})$) as well as the gbest location to the current location of the overall best particle in d-dimensional space ($pbest_{gd}, g \leftarrow b$, where $gwasthepreviousbestlocationindex$).

   else retain the gbest value and location.

5. Change the velocity and position of the particles according to the equations:

$$v_{id} = v_{id} + c_1 * rand() * (pbest_{id} - x_{id}) + c_2 * rand() * (pbest_{gd} - x_{id})$$

$$x_{id} = x_{id} + v_{id}$$

6. Loop the process from step 2 until one of the following conditions are met, a sufficiently good fitness is reached or a maximum number of iterations (generations) are reached.

The original algorithm is quite simple. The population is initialized by randomly obtaining some particles within the search space and generating random velocities that are paired to each particle. There are N particles in the population. Each particle's position (x_{id}) and velocity (v_{id})is composed of d numbers where d is the dimension of the search space and $i \in (1, 2, 3, \ldots, N)$. We take note that each dimension of the search space is usually bounded or are in intervals $[a_j, b_j]$, $j \in (1, 2, 3, \ldots, d)$. a_j is the lowest number that each x_ij can be while b_j is the highest number that each x_ij can be.

Each particle's personal best value and location are recorded as *pbest* and $pbest_{id}$. At each loop, the pbest value is compared to the particle's fitness value and updated. This acts as a memory of where the particle was last at its best. Another pair of value and location are recoded which are the overall best particle's fitness value and location. The overall best particle is the particle in the current population that has currently the best fitness value. (Best is usually determined as the lowest or highest fitness value depending on the implementation) These values are known as *gbest* and $pbest_{gd}$. This pair serves as a memory of where the most optimum location is currently at. These recorded values will serve to guide each member to the most optimum location on the search space as seen on the equations at step 5.

The particle's velocity and location are changes in step 5. As we can see, there are many variables involved in the equations. They will be discussed in the next sections.

3.2.1 Background

We first discuss the concepts where the algorithm was based upon. Early computer animations used to simulate a flock of birds by individually giving each bird a script to

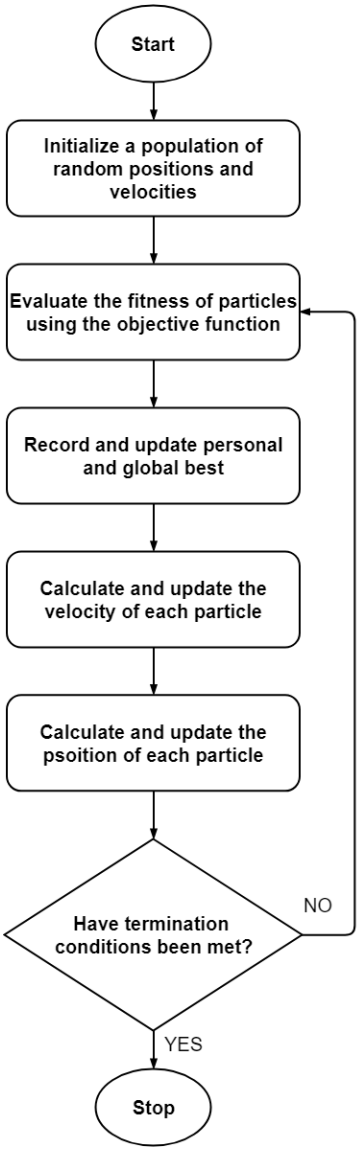

Figure 3.1: Flowchart of the PSO Algorithm

follow, this includes motion, direction, and speed. Each bird was much like an actor in a play, performing actions under a set of instruction. The problem was that it was not scalable. Animators could not possibly give individual scripts to thousands of birds within a short period. This type of approach is too inefficient. This is why, scientists such as Reynolds[24], Heppner and Grenander [15] have tried to simulate movements of birds and fish using the computational power of computers. They tried to simulate where birds would fly to in every time step or frame in the animation. These simulations were using mathematical and physical concepts to mimic the unpredictable movements of birds when they fly in groups. The initial tests were made such that a population of birds were created, each having its own velocity and initial position on a defined space of definite dimension. These birds were "flying" through the virtual space created by simply adding each bird's velocity to its current position at each time step. Their velocities would change each time step according to the velocities of the nearest neighboring birds to avoid collisions. The initial tests showed that direction and speed were not enough to capture the natural flocking of birds this is because after several time steps, the whole flock would unanimously uniformly fly through the defined space in an unchanging direction. This resulted in the introduction of a craziness factor in the form of stochastic variables multiplied to the velocities of each bird. This change resulted to simulations looking much more lifelike.

Let us take, for example, two birds A and B on a real number Cartesian plane. If bird B is flying at a rate of 9 units per second forward and 5 units per second upward and bird B is bird As neighbor, bird A will change its velocity to match bird Bs velocity. Hence, bird A will have a flying rate of 9 units per second forward and 5 units per second upward with each value multiplied to a random number uniformly distributed from 0 to 1. This means, bird A might not fully replicate the velocity that bird B has. This is seen in nature as bird A trying to approximate the velocity of bird B in such a way that they will not collide.

The next step towards development was the introduction of a focal point to which the flock would move towards. This was introduced as a "roost" by Heppner[15], typically it is a point in space that indicated where the flock would finally land. Upon simulating this, the birds already have a "lifelike" appearance which therefore allowed the elimination of

the 'craziness' factor. It was then noted that birds usually land where there is food, hence the roost was replace by a vector called the "cornfield vector" which is a two-dimensional vector of XY coordinates on the Cartesian plane. Given a known position of food, the birds now changed their velocity according to the distance between their current position and the cornfield vector. Each bird now "remembers" the closest position values it was at during that time step. It also took in consideration the closest position values that any bird in the population has been in. Each bird now changed their velocities with the values that they remember.

The algorithm was then extended to spaces with multiple dimensions. The algorithm was tested from the singular dimesion space R, then to the coordinate system R^2 and finally to the 3-dimensional space, R^3. It was generalized that the algorithm would work in any number of dimensions R^N.

The velocity equation underwent some changes until it became:

$$V[i][d] = c1 * rand() * (pbest[i][d] - present[i][d]) + c2 * rand() * pbest[gbest][d] - present[i][d]) \tag{3.1}$$

where $v[i][d]$ is the d^{th} velocity component of particle i in D dimensions, $rand()$ are the randomly generated stochastic variables, $pbest[i][d]$ is the d^{th} component of the particle's best position in D dimensions, $pbest[gbest][d]$ is the d^{th} component of the population's best particle's position ($gbest$) in D dimensions, $present[i][d]$ is the d^{th} component of the particle's current position in D dimensions, $c1$ and $c2$ are constant numbers, $x \in (1, 2, 3, \ldots, n)$

Eberhart and Kennedy [10] adopted the term swarm from Millonas under the circumstance that the behavior of the members of the population satisfies the 5 principles of swarm intelligence as proposed by Millonas. These 5 principles are:

1. proximity principle - members are able to carry out simple space and time calculations

2. quality principle - members respond to the quality factors of the environment

3. principle of diverse response - members do not commit it activities along excessively narrow channels

4. principle of stability - members do no change the mode of behavior everytime the environment changes

5. principle of adaptability - members are able to change their mode of behavior when it is worth the computation price

The members of the population satisfy these principles because

1. The population carries out n-dimensional space calculations over a series of time steps

2. Each member responds to the quality of the personal best and global best variables

3. The allocation of responses between personal best and global best ensures diversity of response.

4. The population changes its overall mode of behavior only when the global best changes.

5. The population is adaptive because it does change when the global best changes.

3.2.2 Further Developments

Eberhart and Shi[11] explains that the terms of the velocity vector seen on step 5 of the original algorithm are all important. The first term ($v_i d$) being the previous velocity value gives 'memory' to the particle. It keeps the particle at a good position until a better position is found. Without it, the particle will fly towards the centroid of the locations

$pbest_{id}$ and $pbest_{gd}$. In addition, without it, the search space will shrink and never grow since it will only move toward the centroid of it's recorded locations $pbest_{id}$ and $pbest_{gd}$. The two terms $c_1 * rand() * (pbest_{id} - x_{id})$ and $c_2 * rand() * (pbest_{gd} - x_{id})$ concerning the personal best and global best comparison with the current position is necessary to keep the particles from flying in the same direction for every iteration and leaving the search space.

Eberhart and Shi[11] further improved the original algorithm proposed by Eberhart and Kennedy[10] by introducing inertia weight. Inertia weight is responsible for balancing global and local exploration. The new velocity equation becomes

$$v_{id} = v_{id} * w_{id} + c_1 * rand() * (pbest_{id} - x_{id}) + c_2 * rand() * (pbest_{gd} - x_{id}) \tag{3.2}$$

where the new variable w_{id} is the inertia weight. Eberhart and Shi[11] states that having a high inertia weight ($w > 1.2$) results in more global exploration but less chances of finding the optima because the particles keep exploring new regions in the space. In contrast, having a low inertia weight ($w < 0.8$) will converge to local optima quickly but will not ensure that the global optimum value will be found. Low inertia weight allows for a fine exploration of a region in the space. Having an inertia weight between 0.8 and 1.2 gives the best chances of finding a global optimum but will take a moderate number of iterations. They theory crafted that it is best to have a high inertia weight in the beginning for extensive global exploration and then reducing the inertia weight gradually through time for a more refined search on local areas. Although the study does give a good background as to the selection of such numbers, in implementing PSO, one must also take in consideration that not all problems are the same hence, implementor must tweak the PSO variables to suit the problems they are trying to solve.

Fixing Convergence

Although PSO is simple in implementation and design, it had certain flaws. It has high computational costs which is given by it slow convergence.[16] Convergence is a problem for PSO because of the restrictions imposed on the velocities of the particle, in addition, although it converges to a point, the particle are ever moving which causes the particles

to be in perpetual oscillation around the optima. The population may converge but due to the perpetual motion, convergence can become a problem if high precision is taken in consideration. The population may not at all converge. Hence, many studies try to solve such problems.

An innovation to the PSO is the introduction of a constriction factor K necessary for ensured convergence introduced by Clerc[5]. The formula then becomes

$$v_{id} = K[v_{id} + c_1 * rand() * (pbest_{id} - x_{id}) + c_2 * rand() * (pbest_{gd} - x_{id})]$$

where $K = \frac{2}{|2-\varphi-\sqrt{\varphi^2-4}|}, \varphi = c_1 + c_2$ and $\varphi > 4$.

Chaos Search

Chaos is a characteristic of non-liner system that includes infinite unstable periodic motions and depends on initial conditions.[23] Due to its uncertainty and stochastic properties, chaotic sequences have been used to replace random generated numbers and to enhance the performance of heuristic optimization algorithms such as GA, PSO and others. There are several chaotic maps available with different properties and characteristics. The piecewise linear chaotic map (PWLCM) is a simple and efficient chaotic map with good dynamic behavior. The simplest PWLCM is defined by Xiang, Liao and Wong [31],

$$x(t+1) = \begin{cases} x(t)/p, & x(t) \in (0, p) \\ \frac{1-x(t)}{(1-p)}, & x(t) \in [p, 1) \end{cases}$$

The PWLCM behaves chaotically in (0,1) when $p \in (0.05) \bigcup (0.5, 1)$. The chaotic variable, x, can be randomly initialized (i.e. $x(0) \bigcup (0, 1)$) as suggested by Xiang et.al [31] who implemented the PWLCM in PSO to perform chaotic search. They implemented the CPSO (Chaotic PSO) by adding the term $r(2cx - 1)$ to the global best $\hat{y}$. cx is the chaotic variable given by PWLCM and r is a random number taken from the uniform distribution of $(0, 1)$. If the resulting vector's objective function value is better, then the global best is replaced, if not, then retain the global best. The velocity function they used for this method is quite different as they have taken inspiration from Clerc and

Kennedy [6]:

$$v_{ij} = \chi(v_{ij} + c_1 r_1(y_{ij} - x_{ij}) + c_2 r_2(y_j - x_{ij}))$$

Although it is not very different from the equation of Kennedy et.al.[10], there is no inertial weight present but the variable χ (a.k.a Constricting Factor) is new. χ is added so that the velocity of a particle is throttled such that it does not fly too fast (not having too high of a magnitude for a single time/generation step). χ replaces the need of having to manually set a bound on the magnitude of the velocity. To recall, the velocity of a particle in PSO is usually set to have a bounded magnitude so that it does not travel too fast through the search space, thereby adding realism and further enhance the ability of each particle to explore the search space thoroughly.

Quantum Mechanics

In Newtonian mechanics a particle has a position and a velocity that determines its trajectory. However, in quantum mechanics, the particle's position and velocity cannot be determined simultaneously according to the uncertainty principle. Hence, the term trajectory is meaningless[28]. Sun et.al.[28] proposed a quantum model of PSO, called QPSO, where particles move according to the following equation,

$$x(t+1) = g \pm \frac{L}{2} ln(\frac{1}{\mu})$$

where g is a local attractor, L is a parameter that must go to zero as $t- > \inf$ to guarantee convergence and $\mu \bigcup(0,1)$. L is a very important parameter of QPSO and different methods have been proposed to determine it.[28, 29]
Uncertainty principle states that the position and the velocity of an object cannot both be measured exactly, at the same time, even in theory. The very concepts of exact position and exact velocity together, in fact, have no meaning in nature. The uncertainty principle implies that there is no exact trajectory since there is no absolute measurement to the position and/or velocity of an object. This is because there will always be an unknown amount in the measurement given by the precision of the instruments used.

PSO has been improved by combining it with other optimization algorithms as well. These hybrids will be explored in the later sections.

3.3 GA

Genetic Algorithm (GA) is an evolutionary algorithm developed by John Holland et. al.[14] It is based on the mechanics of natural selection and natural genetics, that is, it imitates the processes involved in selection, recombination and evolution. It involves randomness due to the fact that it mimics natural processes, but users can control the degree of randomness that GA exhibits.

The goals of optimization seeks to improve performance towards some optimal point or points. However, there is a distinction between the process of improvement and the destination or optimum itself. In this case, GA is the process and is independent of the objective being approached. GA is not focused on solving a single problem. It is a flexible tool used under different circumstances. This robustness makes GA popular among optimization algorithms.

GA was developed by John Holland[14] with the help of his colleagues and students. Their goal was to (1) abstract and rigorously explain the adaptive process of natural systems and (2) design artificial system software that retains the important mechanisms of natural systems. This approach led to important discoveries in both natural and artificial systems.

3.3.1 Components of GA

The basic algorithm for GA is shown below. A flowchart of the algorithm is also shown on figure 3.2.

1. Generate random population of N chromosomes in the solution space of D dimensions.

2. Evaluate the fitness $F(x)$ of each chromosome x_{id} in the population where $i \in (1, 2, 3, \ldots, N)$ and $d = (1, 2, 3, \ldots, D)$

3. Create a new population by repeating the following steps until the new population is complete

   (a) (Selection) Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chances of selection)

   (b) (Crossover) With a crossover probability, perform crossing for selected parents to produce new offspring (children). If no crossover was performed, offspring is an exact copy of parents. (Implementor may choose to select only 1 child to be placed in the population through some selection means)

   (c) (Mutation) With a mutation probability, mutate new offspring at some positions $x_d, d \in (1, 2, 3, \ldots, D)$

   (d) (Accepting) Place new offspring in a new population

4. Replace the old population with the new population for further iterations of GA

5. Loop the process to step 2 until one of the following conditions are met, a sufficiently good fitness is reached or a maximum number of iterations (generations) are reached.

We now discuss what happens at each part of the algorithm.

Initialization of Population

As we can see, the first step is to generate a population sufficient enough to cover our search space and is limited by the resources at hand. Each member of this population is encoded as an array of values. The number of elements in the array will be determined by the problem and the one who creates the GA. The population size N determines how many chromosomes are in one generation. If there are too few chromosomes, GA will not be able obtain diversity during crossover hence, only a small part of the search space is explored depending on the values of the initial population. On the other hand, if there are too many chromosomes, GA slows down and many of the elements of the initial population tend to be repeated, hence overestimation occurs. After several years of research, it was determined that after some limit (which depends mainly on the encoding and the problem) it is not useful to increase population size, because it does not make solving the problem faster.[21] This is because the population size becomes too big for the solution space, or the number of computations needed becomes too large and redundant.

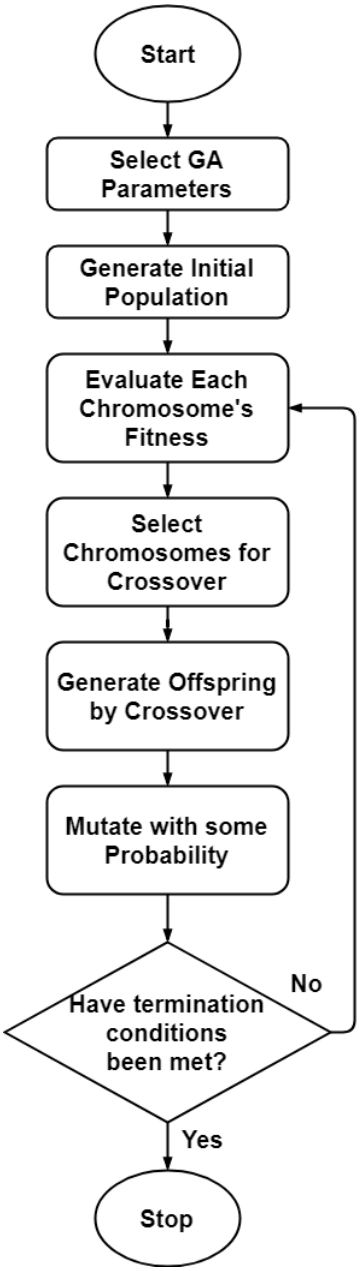

Figure 3.2: Flowchart of the GA Algorithm

Fitness Evaluation

Next is to evaluate the fitness function $f(x)$ for each chromosome x_i in the generation. A fitness function is the function that the algorithm is trying to optimize. The word "fitness" is taken from the evolutionary theory. It tests and quantifies how 'fit' each potential solution is with respect to the problem.[4] It is important to note that the fitness function is a large factor in problem solving using GA. The fitness function must be able to define the numerical complexity and constraints that are present in the problem. Choosing the right fitness function will determine computability and complexity usage of the algorithm.

Selection

Selection allows for persistence and propagation of better genes in the next generation based on the current gene pool. The selection process is 'repeating' if it allows re-selection of already selected members. Selection is 'non-repeating' if it does not allow re-selection of members for cross-over. Non-repeating allows retention of other possibly 'good' genes (genes that might lead to better solutions later on) and a slower convergence rate. Repeating selection can lead to a population of individuals that have the same already good genes but differ in only some features. This allows for local exploration, searching for a good solution in a specific area in the search space. In consequence, since the same parents can be selected numerous times, it can lead to generating a population with a uniform genetic make-up.

Examples for selection process are roulette selection and elimination selection. Roulette selection is done by creating a roulette wheel where individuals that have better genes are provided with a lager portion of the whole wheel. The wheel is spun and the corresponding member mapped to the portion of the wheel where the pointer ends at is selected. The process is repeated until there is a good enough number for generating the next population. An example of the roulette wheel is seen on figure 3.3. Elimination selection is done by selecting a number of individuals and pitting them against each other based on their function values. Individuals that have better function values are selected and the process is repeated until there is a good enough number for generating the next

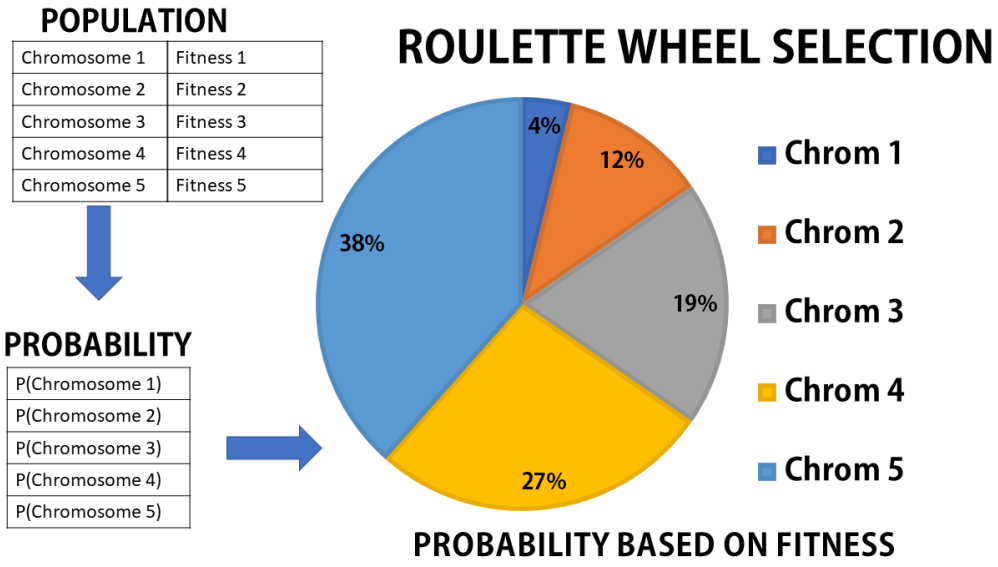

Figure 3.3: Roulette Wheel Selection

population. An example of the elimination selection is seen on figure 3.4.

Recombination or Cross-over

Cross-over is the process of taking two selected individuals and swapping portions of their genetic make-up to create offspring that have genes from both parents (heredity). The number of points where crossing occurs is determined by the implementor. The cross-over chance is the probability that tells whether recombination occurs for a pair of chromosomes. Cross-over chance is determined by the implementor after some tests. Cross-over mechanism is important because it allows the creation of possibly new solutions from the previous gene pool. This allows exploration over a specific area in the search space. The individuals generated by this process are the members of the next generation. It is up to the implementer how much of the newly generated individuals are chosen. A possible implementation is where offspring that have the better function values may be retained. It is also possible to retain chromosomes form the previous generation. Suppose that we have chromosome A having the genetic make-up $< 1, 2, 3, 4 >$ and chromosome B having the genetic make-up $< 6, 7, 8, 9 >$. If we implement a single point cross-over after the second value we get the offspring $< 1, 2, 8, 9 >$ and $< 6, 7, 3, 4 >$. A

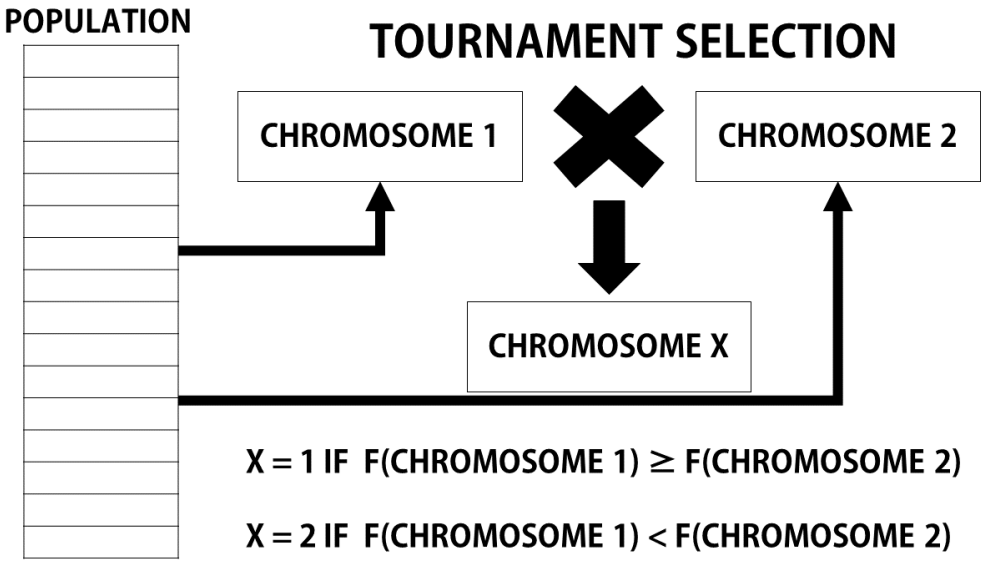

Figure 3.4: Simple Elimination Selection

visual representation is seen in figure 3.5.

Mutation

Mutation mechanism is the process wherein some genes of members in the population are replaced by a completely new value. This allows for exploration on possibly 'unexplored' areas in the search space. It helps get the population unstuck from a local optima (optimum solution for a certain area in the search space but may not the most optimal of solutions in the entire search space). Mutation chance is determined by the implementor after some testing. In nature, however, mutation is a rare occasion hence the mutation chance must be low (usually 0.02). A visual representation is seen in figure 3.6. If the chromosomes are bound to have each gene $x_i \in [1, 9]$ where i is from $[1, 4]$. We can see that 3 is replaced by 9 and that both 3 and 9 are still in $[1, 9]$. Note that the number of genes (elements) to be mutated is not limited to one. You can change a few more genes but the number must be small. Mutation is stated to be some minor change in the genes this means that it is up to the implementor to determine the number genes to be manipulated such that it only brings a minor change. When we take binary numbers in consideration, flipping a few bits still creates a minor change if let's say that

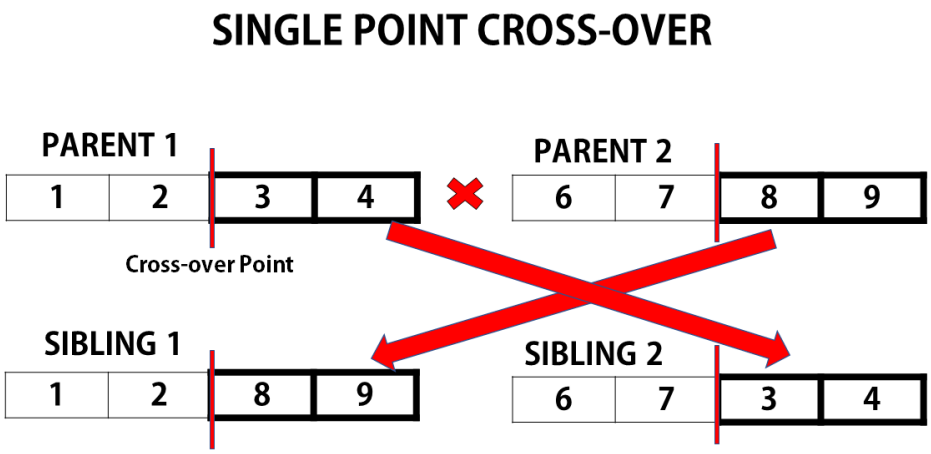

Figure 3.5: Singe Point Cross-Over

the chromosome is made up of 30 or 50 genes, then flipping 2-3 bits will not cause a major change provided that they are less significant bits.

Acceptance

Accepting is just evaluating whether or not the generated member can be added to the new population. This can be done through comparing with the parents' fitness values. If the offspring have better fitness values then accept, otherwise reject. This step us usually done after cross-over to check if the siblings generated will replace members in the population based on their fitness.

Replacement

Replace the old population with the new population. We can also choose to retain some of the old population, some of those that have 'good' genes can be kept in the new population. This is called being 'elitist' since it keeps only the fit members of society to move forward.

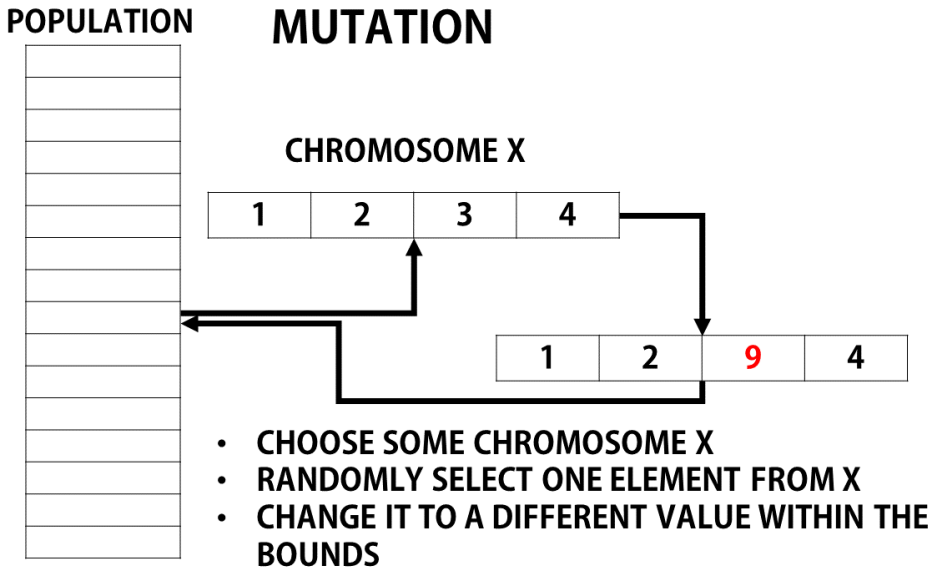

Figure 3.6: Mutation in GA

Termination Conditions

Testing is done through keeping track of the best fitness of each generation. If the fitness is the same for n amount of times and is below a certain acceptable threshold, then we terminate the process. This is considered as a success only if most of the members in the population have the same acceptable fitness, otherwise it is a failure. n is determined by the user. If the population becomes uniform, terminate the process and print out the value. If the fitness is acceptable under a threshold, then it is a success and we say that the population has converged to that point. If the population has become uniform but does not have an acceptable fitness, then it is a convergence but a failure. If a certain number of iterations has been reached and it has not yet converged and has been the same for n times, the process terminates and it is a failure.

3.4 PSO-GA Approach

PSO-GA is a hybrid of PSO and GA. Harish Garg has proposed a PSO-GA[13] which supplements the particular disadvantages of both PSO and GA with the advantages of each. The algorithm attempts to balance the exploration and exploitation ability of both

algorithms. Exploration happens in PSO when particle fly through the search space. It is less applicable to GA since the algorithm only utilizes what is current known in the population. It only occurs for GA through Cross-over and Mutation. Exploitation happens in PSO when a particle flies to or near an area containing a possible solution, every other particle in the population will tend to flock towards that area in order to find the solution. PSO's problem is that local optima may trap the whole population. Exploitation happens in GA during the Selection operator, wherein the members with the fittest values have a higher chance of being chosen for Cross-over and Mutation. Hence, more chances of exploring that particular gene pool.

In GA, if an individual is not selected, the information contained by that individual is lost but in PSO, the memory of the previous best position is always available to each individual. Without a selection operator, PSO may waste resources on poorly located individuals. PSO-GA by Garg[13] combines the ability of social thinking in PSO with the local search capability of GA.

PSO's velocity vector guides the population to a certain solution point while GA's selection and cross-over replaces infeasible solutions with feasible ones by creating an individual from the set of feasible solutions.

3.4.1 Parts of PSO-GA

The algorithm for PSO-GA is shown below

1. Set PSO and GA parameters
    - Set current PSO iteration, $PSO_{CurrIt} = 0$ and max iteration PSO_{MaxIt}
    - Set PSO population size PSO_{PopNum}, cognitive and social bias constants c_1 and c_2, maximum and minimum inertial weights w_{max} and w_{min}
    - Set GA parameters, crossover probability GA_{cross}, mutation probability GA_{mut}
    - Set GA parameters: rate of the number of PSO particles affected by GA γ and rate of increasing GA maximum iterations β, maximum and minimum number of individuals to be selected GA_{NumMax} and GA_{NumMin}, maximum and minimum GA population sizes $GA_{MaxPopSize}$ and $GA_{MinPopSize}$,

maximum and minimum GA iteration numbers GA_{MinItr} and GA_{MaxItr}

- Set the PSO dependent GA parameters, number of individuals affected by GA GA_{Num}, GA population size $GA_{PopSize}$ and GA maximum iteration GA_{MaxItr} using the equations

$$GA_{Num} = GA_{NumMax} - (\frac{PSO_{CurrIt}}{PSO_{MaxIt}})^{\gamma} \times (GA_{NumMax} - GA_{NumMin}) \quad (3.3)$$

$$GA_{PopSize} = GA_{MinPopSize} + (\frac{PSO_{CurrIt}}{PSO_{MaxIt}})^{\gamma} \times (GA_{MaxPopSize} - GA_{MinPopSize}) \quad (3.4)$$

$$GA_{MaxItr} = GA_{MinItr} + (\frac{PSO_{CurrIt}}{PSO_{MaxIt}})^{\beta} \times (GA_{MaxItr} - GA_{MinItr}) \quad (3.5)$$

PSO Section

2. Generate a random population of particles of PSO_{PopNum} members in D dimensions, each with a corresponding random velocity v

3. Increment PSO_{CurrIt} by 1

4. Evaluate each particle's objective function value $F(PSOx)$

5. Update *gbest* and *pbest* positions and values of each $PSOx_i$ in the population ($i \in 1, 2, 3, \ldots, PSO_{PopNum}$)

6. Update each particle's velocity and position with the equations,

$$w = w_{max} - (w_{max} - w_{min}) \times (\frac{PSO_{CurrIt}}{PSO_{MaxIt}}) \quad (3.6)$$

$$v_i = v_i \times w + c_1 \times rand() \times (pbest_i - PSOx_i) + c_2 \times rand() \times (pbest_g - PSOx_i) \quad (3.7)$$

where $i \in 1, 2, 3, \ldots, PSO_{PopNum}$ and g is position/individual in the PSO population that is currently designated as global best (*gbest*) individual

$$PSOx_i = PSOx_i + v_i \quad (3.8)$$

GA Section

7. Set the number of currently selected individuals $GA_{CurrNum} = 0$

8. Increment $GA_{CurrNum}$ by 1

9. Choose a random position/individual $PSOx_s$ from the PSO population.

10. Generate a random population of $GA_{PopSize}$ individuals in the same D dimensions.

11. Set the first individual GAx_1 in the GA population to be a randomly selected individual $PSOx_s$ from the PSO particle population.

12. Set the current GA iteration $GA_{CurrItr} = 0$

13. Increment $GA_(CurrItr)$ by 1

14. Perform elitism

    - set the replacing individual GA_{rep} as the randomly selected PSO particle $PSOx_s$ if $GA_{CurrNum} = 0$
    - otherwise, check each individual in the current GA population, if $F(GAx_i)$ is less fit than $F(PSOx_s)$, then replace GAx_i with $PSOx_s$

$$GAx_i = \begin{cases} PSOx_s & \text{if } F(PSOx_s) < F(GAx_i) \\ GAx_i & \text{otherwise} \end{cases} \quad i \in 1, 2, \ldots, GA_{PopSize}$$

15. Perform selection, crossover and mutation to generate the next GA population

16. Evaluate the penalizing objective fitness values $F(GAx_i)$ for each individual in the GA population

17. Check if maximum GA iterations is reached

    - If reached, proceed to step 18
    - otherwise, go back to step 13

18. Replace the selected PSO particle $PSOx_s$ with the best individual in the GA population

19. Check if the maximum number of replacements have occurred

    - If reached, proceed to step 20

- otherwise, go back to step 9

20. Update the PSO dependent GA parameters using equations (3.3), (3.4) and (3.5)

21. Check if the maximum number of PSO iterations have been reached or if the population has converged
    - If reached, end
    - otherwise, go back to step 3

The flowchart of the algorithm is shown on figure 3.7

As you can see, the algorithm follows the both PSO and GA algorithms in succession. PSO is first done to the population to obtain points across the search space. GA is then applied to some of the best individuals. This is done to replace the worst individuals in the population with those closer to the better ones.

After forming the new population with PSO, some of the individuals in the population will get replaced. Some not all because if we have a huge population, it would take a long time to complete. This number is given by GA_{Num}. After selecting the best individuals from the population, the algorithm aims to create a new population by replacing points in the current population with better points via the genetic principles, selection, cross-over and mutation. After all selected individuals have been processed, we change the GA variables, $GA_{PopSize}$ and GA_{MaxItr} which are for the population size in GA and the maximum iterations done for GA respectively by the equations (3.3), (3.4) and (3.5).

Judging from the equations 3.3, 3.4 and 3.5, GA_{Num} will initially be GA_{NumMax} and slowly become GA_{NumMin} as the number of iterations increases. This is because the fraction PSO_{CurrIt}/PSO_{MaxIt} is raised to γ which is a positive whole number as given by Garg[13], hence, the whole term $(PSO_{CurrIt}/PSO_{MaxIt})^{\gamma}$ will initially be very small and eventually will be equal to 1 when $PSO_{CurrIt} = PSO_{MaxIt}$.

This is also the case for both $GA_{PopSize}$ and GA_{MaxItr}. $GA_{PopSize}$ will initially start equal to $GA_{MinPopSize}$ then slowly become $GA_{MaxPopSize}$. GA_{MaxItr} will initially start equal to GA_{MinItr} then slowly become GA_{MaxItr}. Since the factors will be in fractions, there is a need to get the floor values of GA_{Num}, $GA_{PopSize}$ and GA_{MaxItr}. This is because GA_{Num}, $GA_{PopSize}$ and GA_{MaxItr} must be positive integers because they dictate

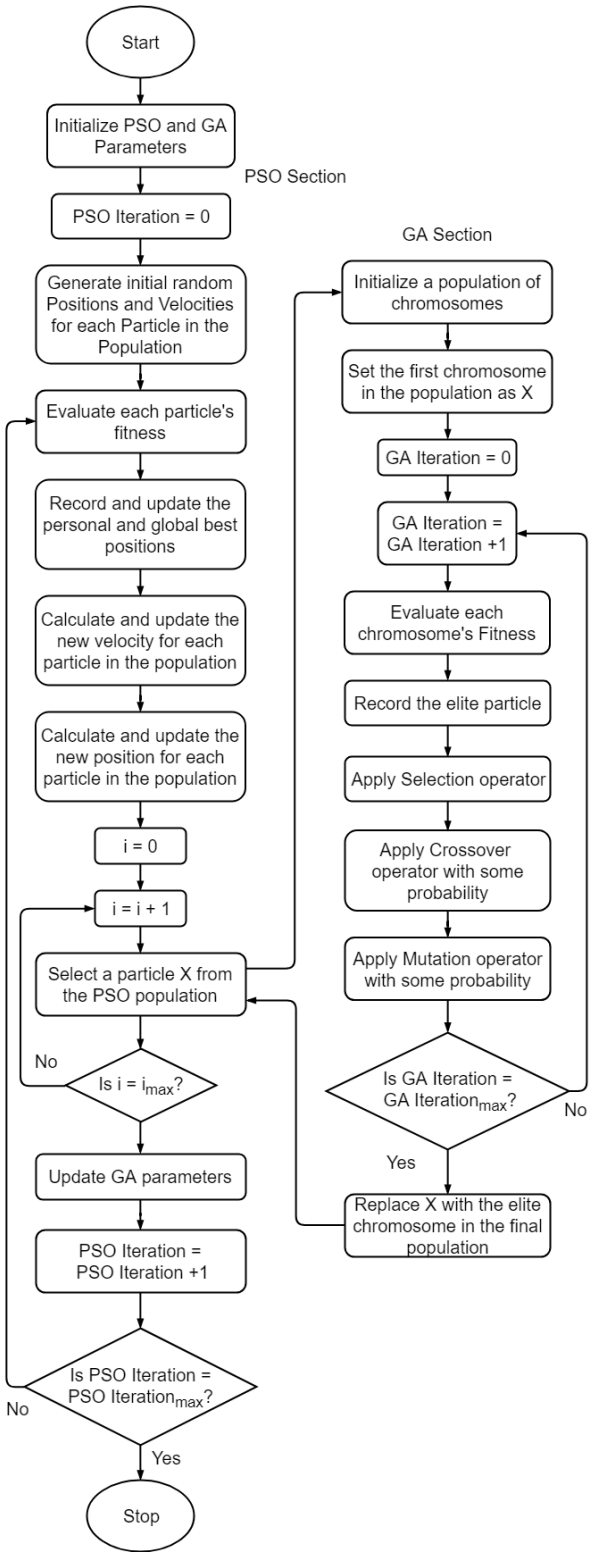

Figure 3.7: Flowchart of PSO GA Algorithm

array sizes. However, in the case of Inertial Weight w, which changes according to the equation $w = w_{max} - (w_{max} - w_{min})(PSO_{CurrIt}/PSO_{MaxIt})$, it is most of the time a fraction. Garg[13] started $w = 0.9$ initially then becoming $w = 0.4$ as the number of iterations increases.

Chapter 4

Methodology

In this chapter, we first discuss the model of the problem then show how the hybrid PSO-GA algorithm of Harish Garg[13] is implemented. Then we test the effectiveness of the algorithm in solving vehicle routing problems using the VRPTW model used by Liu et.al.[18]. We now consider developing the specific model for the Baguio City waste collection routing problem. We start by reevaluating what we know about the current waste collection system in Baguio City. The data on the vehicles and working hours was provided by the Sold Waste Management Division of Baguio City.

- Each driver works for 9 hours each working day on different shifts; morning, afternoon and night.
- Each driver is assigned a 5-day work schedule on different sets of days.
- Each vehicle is assigned to service about 2 to 5 Barangays each day.
- There a total of 14 waste collection vehicles currently. Two of which are quick response teams, these two teams are those that operate the compactor vehicles. As for the other 4 new vehicles purchased by the city last January, there was no available data given.
- There are four kinds of vehicles used for waste collection. The details are seen on the table below.

Table 4.1: Waste Collection Vehicles of Baguio City

Vehicle Brand	**Type**	**Capacity**	**Number**
Isuzu	Truck	$10.19m^3$	3
Isuzu	10-Wheeler	$30m^3$	5
Hino	Truck	$14m^3$	4
Hino	Compactor	$14m^3$	2

- Each vehicle's carrying capacity has two main partitions for biodegradable

and residual waste, however, the partitioning is not fixed.

- Each vehicle start and ends at the Irisan ERS/MRF.
- Each vehicle is empty before leaving the ERS/MRF.
- The vehicles are full when they return to the ERS/MRF but their load is deposited at the site for final segregation.
- After segregation, residual waste is brought to the Garbage Transfer Station at Barangay Dontogan where it will be gathered and loaded onto vehicles that transport it to Capas, Tarlac.
- There are 129 known Barangays (Villages) in the City.
- No time windows are alloted due to variability of traffic, road availability, weather conditions, and quantity of waste.

4.1 Waste Collection Vehicle Routing Problem Model

The objective in Waste Collection Vehicle Routing Problem is to determine a feasible set of routes that minimizes the total cost involved in waste collection with the following constraints:

1. All vehicles start at and return to the depot;
2. All vehicles are homogeneous, they have the same maximum capacity of $Q = 14$ m^3;
3. A waste collection site is visited by only one vehicle;
4. The total amount of waste collected by vehicle must not exceed its maximum;
5. Distances between the depot, collection sites and the disposal site are determined;
6. We assume that the disposal site is the same as the depot. This is because the waste collected by trucks will have to be sorted at the Eco-Waste Recovery Services-Material Recovery Facility before it is transported to the Garbage Transfer Station where it will be gathered and brought to Capas Tarlac.

7. The demand at each collection site should be less than the maximum capacity of the vehicle;

We represent our network of collection sites and ERS-MRF depot/disposal site as a complete undirected graph $G = (V, E)$ of V vertices and E edges.

The set of vertices V encapsulates the set of waste collection sites (V^c) and the single depot also considered as the single disposal site (V^d), that is $V = \{V^d \cup V^c\}$. The number of vertices is therefore $|V| = |V^d| + |V^c| = 1 + n = N$ where n is the number of waste collection sites.

$$V = \{v_i\}, i \in 0, 1, 2, \ldots, n$$

where

$$v_i = \begin{cases} v_0 & \text{is the Depot} \\ v_1, v_2, \ldots, v_n & \text{are the Collection Sites} \end{cases}$$

Each vertex $v_i \in V$ is associated with a demand q_i equivalent to the amount of garbage to be collected in cubic meters.

$$q_i = \begin{cases} q_0 = 0 \text{ m}^3 \\ q_1, q_2, \ldots, q_n \in \mathbb{R}, \text{ specifically } \in (0, Q) \text{ m}^3 \end{cases}$$

wherein,

$$\sum_{i=1}^{n} q_i = W < Q \cdot m$$

where W is the total amount of garbage generated by all collection sites and Q is the maximum carrying capacity of any vehicle defined below.

The set of edges

$$E = \{(v_i, v_j) | v_i, v_j \in V, \ i, j \in 0, 1, 2 \ldots, n\}$$

The edge $(v_i, v_j) \in E$ connects an arbitrary pair of vertices v_i, v_j in graph G.

Each edge $(v_i, v_j) \in E$ is associated to a distance $d_{i,j}$ in kilometers. In this case, we use the exact distance in kilometers given by Google Maps© Distance API. The distances obtained are seen on table B.1

Let $dist_{ij}$ be the total distance covered in traveling a path from vertex v_i to v_j.

Let $K = \{k_i\}, i \in 1, 2, 3, \ldots, m$ be the set of waste collection vehicles. There are a fixed number of m vehicles.

Let Q be the maximum carrying capacity of any vehicle $k \in K$. This is the maximum amount of garbage that can be collected and carried by a vehicle along it's path.

The decision variables of the model depend on the vehicle capacity Q and the waste quantity at the next waste collection site it visits. These are modeled as follows:

$$X_{i,j,l} = \begin{cases} 1, & \text{if vehicle } k_l \text{ can travel from vertex } v_i \text{ to } v_j \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

where $i, j \in 0, 1, 2 \ldots, n.$ and $l \in 1, 2, \ldots, m.$

$$A_{i,j,l} \in \mathbb{R} \tag{4.2}$$

where each element in A is the accumulated amount collected by vehicle $k_l \in K$ when moving between v_i and v_j where $l \in 1, 2, \ldots, m$ and $v_i, v_j \in V$, $i, j \in 0, 1, 2 \ldots, n$.

$$Y_{i,l} = \begin{cases} 1, & \text{if vertex } v_i \text{ is visited by vehicle } k_l \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

where $i \in 1, 2 \ldots, n$ and $l \in 1, 2, \ldots, m$. Note that we do not consider the depot here because it is bound to be visited more than once by any vehicle.

We can say that X is an $N \times N \times m$ matrix where each $N \times N$ is the adjacency matrix of the route of vehicle k_l, $l \in 1, 2, ..., m$. Hence, it is the adjacency matrix of a subgraph of G where either none, few, many or all of the edges may have been taken out. Moreover, if we combine all of the m $N \times N$ matrices, we come-up with a denser subgraph of G or G itself. It follows that A is also the same $N \times N \times m$ matrix where instead of taking binary variables, it takes on values that represent the accumulative amount of waste collected by a vehicle during it's run through edge (v_i, v_j). Y is a $n \times m$ matrix that acts more like a checklist that shows which waste collection sites $v_i \in V^c$ were visited by vehicle $k_l \in K$.

Our aim is to minimize the total amount of travel distance while minimizing the fleet size (number of vehicles used). Note that if we use more vehicles than necessary, there will be an abundance of unused vehicle space/capacity. Therefore, in order to minimize

the number of vehicles, we must make sure there is a minimum amount of unused vehicle space/capacity. Since our vehicle capacity is in cubic meters and our distance is in kilometers, we calculate the total cost in terms of operational cost. In order to do this, we must know how much fuel in liters is needed to travel the total distance covered by all vehicles. Then we convert the liters of fuel into operational cost. Hence, our conversion is done as follows:

$$\text{Total Distance } \cancel{\text{Km}} \times \frac{\tau \cancel{\text{Liter}}}{\cancel{\text{Km}}} \times \frac{\lambda \text{ Pesos}}{\cancel{\text{Liter}}} = \text{ Total Distance} \cdot \tau \cdot \lambda \text{ Pesos} \tag{4.4}$$

where τ is the fuel efficiency of the vehicle and λ is the cost of a liter of fuel. Fuel efficiency τ is obtained by calculating the average daily fuel consumption and travel distance of the vehicle. This data was obtained through the Monthly Report of Fuel Consumption and Official Travel produced by the Solid Waste Management Division. This report consists of the distance traveled by the vehicle and the amount of gas used for the day. Measuring distance traveled and fuel consumption is done by the odometer of the vehicle. These measurements are made by the driver before and after vehicle use. Fuel efficiency of the vehicle used in this model is approximately 0.27 Liters per Kilometers. The cost of the liter of fuel is obtained by checking the gas prices at the petrol stations for a particular span of time. Specifically, we recorded the diesel prices from June 28 to 30 of 2018 and observed that the diesel costs 46.20 Philippine Pesos (Php) per liter on all three days. Note that cost will be in Philippine Peso (Php)

As for the remaining vehicle capacities, we sum all of the unused volume and convert that to operational cost by knowing how much operation cost is alloted for driver salaries. We assume that each vehicle is driven by a driver at the same operational cost no matter if the vehicle is full or not as long as the driver has visited all the barangays it is assigned. We can compute how much operational cost is wasted on the unused volume of all vehicles and set that as the penalty for the vehicles that has not completely used up its capacity. However, we must divide the amount by the capacity a vehicle so that we know how much wasted expenses was paid to the driver(s). This is represented by the equation

$$\alpha \cdot \frac{\sum_{l=1}^{m} \sum_{i=1}^{n} (Q - A_{i,0,l})}{Q} \tag{4.5}$$

where α is the operational cost of driver salaries of any vehicle. We divide the amount by Q so that we know how much space was wasted in terms of vehicle count. The cost

conversion is done as follows:

$$\text{Total Wasted Capacity}\,\cancel{\text{m}^3}\times\frac{1\cancel{\text{Vehicle}}}{Q\,\cancel{\text{m}^3}}\times\frac{1\cancel{\text{Driver}}}{\cancel{\text{Vehicle}}}\times\alpha\frac{Pesos}{\cancel{\text{Driver}}}=\frac{\alpha\cdot\text{Total Wasted Capacity}}{Q}\text{Pesos} \tag{4.6}$$

The value of α in this case is only an approximate since there are differences in driver salaries based on their years of service. According to the data gathered, driver daily incomes range from Php 480.00 to 1200.00 and above. We use Php 500 as the value of α so that we come up with some sort of beginner's salary for every vehicle that is used.

Hence, our objective function is represented by the equation:

$$\min\ F=\tau\cdot\lambda\cdot\left(\sum_{l=1}^{m}\sum_{i=0}^{n}\sum_{j=0}^{n}X_{i,j,l}\cdot d_{i,j}\right)+\alpha\cdot\left(\sum_{l=1}^{m}\sum_{i=1}^{n}(Q-A_{i,0,l})\right) \tag{4.7}$$

In order to make satisfy our assumptions, we subject our objective function following constraints:

$$\sum_{i=1}^{n}\sum_{l=1}^{m}X_{i,j,l}=1, \qquad \forall j=1,2,\ldots,n \tag{4.8}$$

$$\sum_{i=1}^{n}Y_{i,l}=\sum_{i=1}^{n}X_{i,j,l}, \qquad \forall l=1,2,\ldots,m;j=1,2,\ldots,n \tag{4.9}$$

$$\sum_{j=0}^{n}X_{0,j,l}=1, \qquad \forall l=1,2,\ldots,m \tag{4.10}$$

$$\sum_{i=0}^{n}X_{i,0,l}=1, \qquad \forall l=1,2,\ldots,m \tag{4.11}$$

$$\sum_{j=1}^{n}A_{0,j,l}=0, \qquad \forall l=1,2,\ldots,m \tag{4.12}$$

$$\sum_{l=1}^{m}\sum_{j=1}^{n}A_{i,j,l}\leq Q, \qquad \forall i=0,1,\ldots,n \tag{4.13}$$

$$\sum_{l=1}^{m}(A_{j,h,l} - A_{i,j,l}) = \sum_{l=1}^{m} X_{i,j,l} \cdot q_j, \qquad \forall i, h = 0, 1, \ldots, n; j = 1, 2, .., n-1 \tag{4.14}$$

$$dist_{i,j} = dist_{j,i}, \qquad \forall i = 0, 1, \ldots, n; j = 0, 1, \ldots, n \tag{4.15}$$

$$X_{i,j,l} \in 1, 0 \tag{4.16}$$

$$Y_{i,l} \in 1, 0 \tag{4.17}$$

$$A_{i,j,l} \in \mathbb{R} \tag{4.18}$$

Constraint (4.8) specifies that collection site v_i is visited by not more than one vehicle k_l and (4.9) specifies that a collection site v_i is in the route of vehicle k_l. Since the values of each X_{ijl} is 1 if vehicle k_l moves from vertex v_i to v_j and 0 otherwise, then if we get the sum of the values, we will know how many times v_i is visited by all vehicles. However, we assumed that vehicles only visit each collection site once, hence, the sum must be equal to one.

Constraints (4.10) and (4.11) imposes that each vehicle $k_l \in K$ must start and end at the depot. Constraint (4.12) imposes that each vehicle $k_l \in K$ must have no accumulated waste before leaving and returning to the depot.

Constraint (4.13) imposes that the accumulated amount of any vehicle $k_l \in K$ traveling between any pair of vertices v_i and v_j must be less than the maximum capacity.

Constraint (4.14) imposes that the vehicle k_l completely collects all waste when it visits vertex v_j.

Constraint (4.15) imposes that the total distance traveled from vertex v_i to vertex v_j must be the same when the vehicle k_l travels from vertex v_j to vertex v_i.

Constraints (4.16), (4.17), and (4.18) define the domain of the decision variables.

We have now established the model for the problem however, we show the reasoning behind the added cost of wasted volume. If the problem was just about obtaining the shortest distance, then the problem becomes a Traveling Salesman Problem. The problem would be as simple as finding the shortest connections between nodes by using either the Dijsktra's or Floyd-Warshall algorithms. We give an example of the first 4 barangays as nodes. We have a symmetric and complete graph G containing the depot and the first four barangays on the alphabetical list. The distances are the same from

table B.1. Then therefore we have $\frac{4!}{2}$ possible permutations as solutions. We disregard vehicle capacities. Hence, the distance traveled for each permutation would be:

1. $v_0 \to v_1 \to v_2 \to v_3 \to v_4 \to v_0 = v_0 \to v_4 \to v_3 \to v_2 \to v_1 \to v_0 = 15.5460$ Km
2. $v_0 \to v_1 \to v_2 \to v_4 \to v_3 \to v_0 = v_0 \to v_3 \to v_2 \to v_2 \to v_1 \to v_0 = 14.9710$ Km
3. $v_0 \to v_1 \to v_3 \to v_2 \to v_4 \to v_0 = v_0 \to v_4 \to v_2 \to v_3 \to v_1 \to v_0 = 16.0830$ Km
4. $v_0 \to v_1 \to v_4 \to v_2 \to v_3 \to v_0 = v_0 \to v_3 \to v_2 \to v_4 \to v_1 \to v_0 = 15.5080$ Km
5. $v_0 \to v_1 \to v_4 \to v_3 \to v_2 \to v_0 = v_0 \to v_2 \to v_3 \to v_4 \to v_1 \to v_0 = 15.5460$ Km
6. $v_0 \to v_1 \to v_3 \to v_4 \to v_2 \to v_0 = v_0 \to v_2 \to v_4 \to v_3 \to v_1 \to v_0 = 15.5460$ Km
7. $v_0 \to v_2 \to v_1 \to v_3 \to v_4 \to v_0 = v_0 \to v_4 \to v_3 \to v_1 \to v_2 \to v_0 = 18.0410$ Km
8. $v_0 \to v_2 \to v_1 \to v_4 \to v_3 \to v_0 = v_0 \to v_3 \to v_4 \to v_1 \to v_2 \to v_0 = 17.4660$ Km
9. $v_0 \to v_3 \to v_1 \to v_2 \to v_4 \to v_0 = v_0 \to v_4 \to v_2 \to v_1 \to v_3 \to v_0 = 18.0030$ Km
10. $v_0 \to v_4 \to v_1 \to v_2 \to v_3 \to v_0 = v_0 \to v_3 \to v_2 \to v_1 \to v_4 \to v_0 = 18.0030$ Km
11. $v_0 \to v_2 \to v_3 \to v_1 \to v_4 \to v_0 = v_0 \to v_4 \to v_1 \to v_3 \to v_2 \to v_0 = 18.5780$ Km
12. $v_0 \to v_2 \to v_4 \to v_1 \to v_3 \to v_0 = v_0 \to v_3 \to v_1 \to v_4 \to v_2 \to v_0 = 18.0030$ Km

Given the above, the optimal solution to the TSP is 14.9710 Km. We now look into adding demands and vehicle constraints. Let the maximum vehicle capacity $Q = 12$ m^3. Let the demands at the vertices be as follows:

$$v_0 = 0.0000$$
$$v_1 = 1.9856$$
$$v_2 = 10.4665$$
$$v_3 = 6.6595$$
$$v_4 = 4.0003$$

We use the same model as described however, the objective is only to minimize the cost of traveling distance hence, the equation (4.7) becomes:

$$\min F = \tau \cdot \lambda \cdot \left(\sum_{l=1}^{m} \sum_{i=0}^{n} \sum_{j=0}^{n} X_{i,j,l} \cdot d_{i,j} \right) \tag{4.19}$$

$\tau = 0.27$ Liters per Kilometer and $\lambda = 46.20$ Pesos per Liter. Hence, the cost for all unique routes is given by the following: (Note that vehicles can interchange their routes and there won't be any effect on the total cost. Also note that the route for each vehicle yields the same distance and amount collected to and fro)

1. Vehicle 1: $v_0 \to v_1 \to v_0$
   Vehicle 2: $v_0 \to v_2 \to v_0$
   Vehicle 3: $v_0 \to v_4 \to v_3 \to v_0$
   The cost is 428.00789.
   The number of permutations with the same results is 8.

2. Vehicle 1: $v_0 \to v_2 \to v_0$
   Vehicle 2: $v_0 \to v_4 \to v_0$
   Vehicle 3: $v_0 \to v_1 \to v_3 \to v_0$
   The cost is 465.82906.
   The number of permutations with the same results is 8.

3. Vehicle 1: $v_0 \to v_2 \to v_0$
   Vehicle 2: $v_0 \to v_3 \to v_0$
   Vehicle 3: $v_0 \to v_1 \to v_4 \to v_0$
   The cost is 458.65651.
   The number of permutations with the same results is 8.

Given the above, the solution would be Php 428.00789 and there are 8 possible permutations that give the same solution. Now we use the cost function (4.7) of our model. The possible solutions given this is.

1. Vehicle 1: $v_0 \rightarrow v_1 \rightarrow v_0$
   Vehicle 2: $v_0 \rightarrow v_2 \rightarrow v_0$
   Vehicle 3: $v_0 \rightarrow v_4 \rightarrow v_3 \rightarrow v_0$
   The cost is 547.74539.
   The number of permutations with the same results is 8.

2. Vehicle 1: $v_0 \rightarrow v_2 \rightarrow v_0$
   Vehicle 2: $v_0 \rightarrow v_4 \rightarrow v_0$
   Vehicle 3: $v_0 \rightarrow v_1 \rightarrow v_3 \rightarrow v_0$
   The cost is 863.04572.
   The number of permutations with the same results is 8.

3. Vehicle 1: $v_0 \rightarrow v_2 \rightarrow v_0$
   Vehicle 2: $v_0 \rightarrow v_3 \rightarrow v_0$
   Vehicle 3: $v_0 \rightarrow v_1 \rightarrow v_4 \rightarrow v_0$
   The cost is 745.07317.
   The number of permutations with the same results is 8.

We discuss how the PSO-GA was implemented. We identify the method of encoding each particle or chromosome in the population. Each particle or chromosome is a vector having n dimensions equivalent to the number of customers or collection sites. In this case, $n = 129$ since we have 129 barangays. We borrow the encoding scheme of Masrom[1] wherein each particle's component or each chromosome's gene is assigned a real number, specifically we assign a random number uniformly distributed in the interval $(0, 1)$. These numbers will be used to determine the order at which nodes are visited or inserted in the route. This particular encoding scheme is used in order to simplify the methods used in computing particle positions and velocities, and chromosome crossover and mutation. Each particle/chromosome is represented as follows:

$$\begin{array}{cc} \text{Nodes} & \begin{array}{ccccc} 1 & 2 & 3 & \ldots & n \end{array} \\ \text{Particle} & \left[\begin{array}{ccccc} r_1 & r_2 & r_3 & \ldots & r_n \end{array}\right] \end{array}$$

where each r_j, $i = 1, 2, \ldots, n$ is a random number uniformly distributed in the interval $(0, 1)$.

For example, if we have 5 collection sites, each particle position or chromosome in the population will have 5 components. We let node 0 to be the depot and nodes $1 - 5$ as the collection sites. Given

$$\begin{array}{cc} \text{Nodes} & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \text{Particle} & \left[\begin{array}{ccccc} 0.3 & 0.4 & 0.1 & 0.5 & 0.2 \end{array}\right] \end{array}$$

We arrange the nodes based on their respective component values. This results in the sequence,

$$\text{Nodes} \begin{bmatrix} 3 & 5 & 1 & 2 & 4 \end{bmatrix}$$

Therefore, the sequence of collection becomes:

$$3 \rightarrow 5 \rightarrow 1 \rightarrow 2 \rightarrow 4$$

Route construction is done by sequentially adding the nodes in the order obtained to a vehicle's route until its maximum capacity is reached, this is done until all nodes are placed in a route. The algorithm for route construction is seen on figure 4.1.

An initial population of size PSO_{PopNum} is created by generating a set of random feasible particles $PSOx_i, i = 1, 2, \ldots, PSO_{PopNum}$. The population will be stored in a matrix, $PSO_Population[PSO_{PopNum}][n]$. Each row is a particle having n dimensions. The initial velocities $PSOv_i, i = 1, 2, \ldots, PSO_{PopNum}$ are also randomly produced however, they must follow the maximum and minimum values of velocities. These bounds are given by the midpoint formula

$$\frac{u_j + l_j}{2} = \frac{1 + 0}{2} = 0.5$$

where $u_j = 1$ and $l_j = 0$ are the upper and lower bounds of the values that can be taken by the j^{th} component of particle $PSOx_i$, $j = 1, 2, \ldots, n$; $i = 1, 2, \ldots, PSO_{PopNum}$. The fitness value $F(PSOx_i)$ of each particle $PSOx_i$, $i = 1, 2, \ldots, PSO_{PopNum}$ of the population is then obtained by converting the individual into a set of routes using the route construction algorithm presented above. Then we fitness by using the objective function in our model where the input is the set of routes constructed previously. The

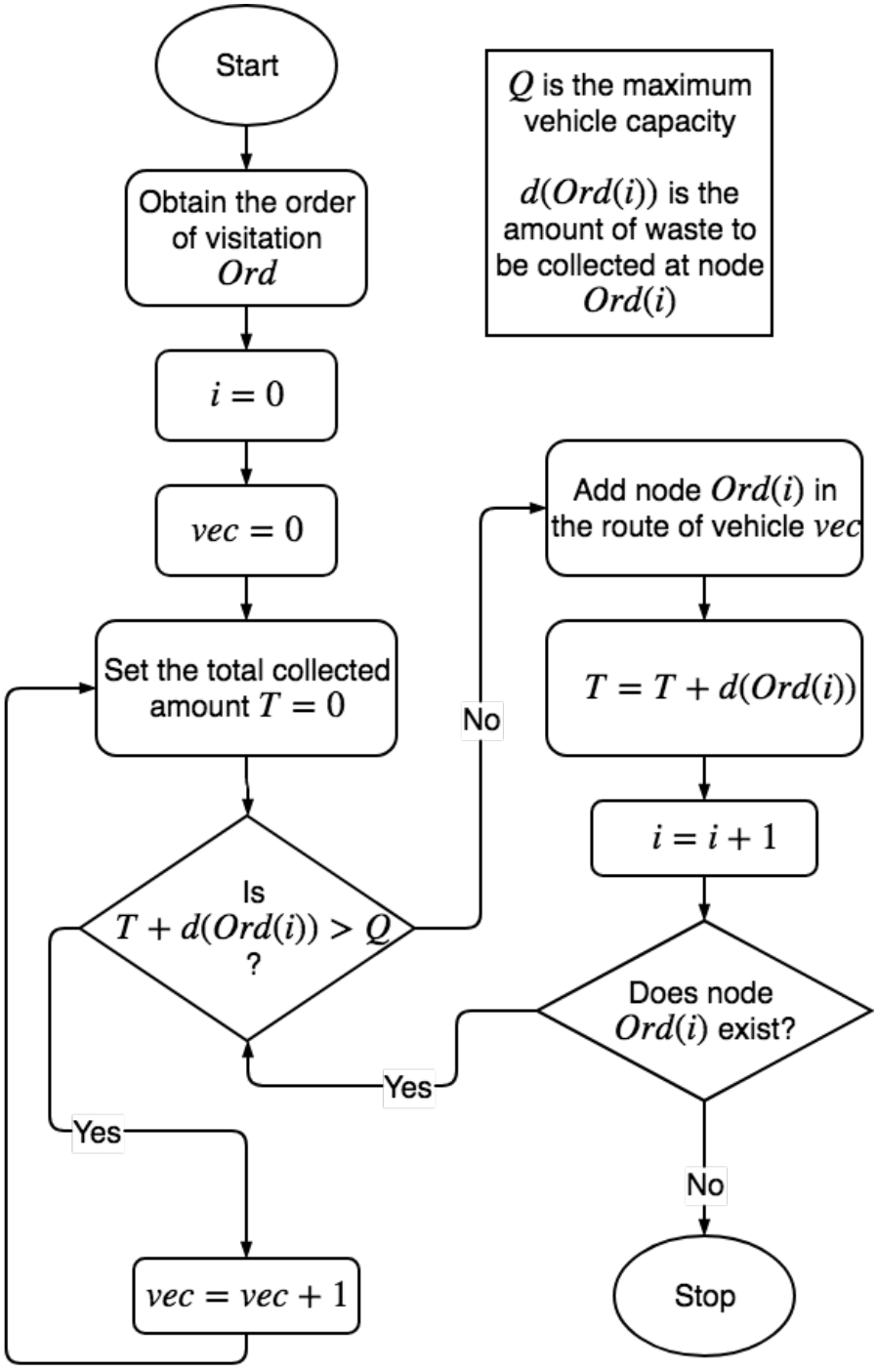

Figure 4.1: Flowchart of Route Construction

initial personal best $Pbest_i$ location of each particle and the global best location $Pbest_g$ of the population is then recorded based from the fitness values obtained.

The new inertia weight value and velocities vectors are then computed using the following equations

$$w = w_{max} - (w_{max} - w_{min}) \times (\frac{PSO_{CurrIt}}{PSO_{MaxIt}}) \tag{4.20}$$

where w_{max} and w_{min} are the upper and lower bounds of the inertia weight. Inertial weight dictates how much of the previous velocity is retained by the individual. PSO_{CurrIt} is the current PSO iteration/generation while PSO_{MaxIt} is the maximum PSO iteration. When the PSO_{CurrIt} becomes equal to the PSO_{MaxIt}, the algorithm stops. We then compute the new velocities and positions of each particle using the following equations.

$$PSOv_i = PSOv_i \times w + c_1 \times rand() \times (Pbest_i - PSOx_i) + c_2 \times rand() \times (Pbest_g - PSOx_i) \tag{4.21}$$

$$PSOx_i = PSOx_i + PSOv_i \tag{4.22}$$

where c_1 and c_2 are the cognitive and social biases which affect how each particle adapts velocity from personal and global best data. $rand()$ is a random number uniformly distributed in the interval $(0, 1)$. $PSOv_i$ at the left side of the equation (4.21) is the new velocity vector while the one of the right is the old velocity vector. $PSOx_i$ at the left side of equation (4.22) is the new position vector while the one on the right is the old position vector.

The fitness value $F(PSOx_i)$ of each of the particles $PSOx_i$, $i = 1, 2, \ldots, PSO_{PopNum}$ in the new population is obtained. Some members of the new PSO population is then selected to undergo GA where the result of GA replaces an infeasible individual in the population.

The number of particles selected at each PSO iteration is given by GA_{num} which is calculated as:

$$GA_{Num} = GA_{NumMax} - \left(\frac{PSO_{CurrIt}}{PSO_{MaxIt}}\right)^{\gamma} \times (GA_{NumMax} - GA_{NumMin}) \tag{4.23}$$

where GA_{NumMin} and GA_{NumMax} are the minimum and maximum values which GA_{Num} can take. γ is a constant factor that determines how much the ratio of the

PSO current and maximum iterations affect the number of individuals obtained. Given that we subtract from the maximum number, it is a given that the number of individuals to be selected becomes lower as the PSO iteration reaches its maximum.

The population size of GA is given by the equation:

$$GA_{PopSize} = GA_{MinPopSize} + \left(\frac{PSO_{CurrIt}}{PSO_{MaxIt}}\right)^{\gamma} \times (GA_{MaxPopSize} - GA_{MinPopSize}) \quad (4.24)$$

where $GA_{MinPopSize}$ and $GA_{MaxPopSize}$ are the minimum and maximum values which $GA_{PopSize}$ can take. γ is the same constant factor above. Given that we add from the minimum number, it is a given that the population size of GA increases at the PSO iteration reaches its maximum.

The maximum iterations of GA is given by the equation:

$$GA_{MaxItr} = GA_{MinItr} + \left(\frac{PSO_{CurrIt}}{PSO_{MaxIt}}\right)^{\beta} \times (GA_{MaxItr} - GA_{MinItr}) \quad (4.25)$$

where GA_{MinItr} and GA_{MaxItr} are the minimum and maximum values which GA_{MaxItr} can take. β is a constant factor that determines how much the ratio of the PSO current and maximum iterations affect the number of individuals obtained. Given that we add from the maximum number, it is a given that the maximum iteration increases as the PSO iteration reaches its maximum.

We set the number of currently selected individuals $GA_{CurrNum} = 0$. We select a random position $PSOx_s$ from the PSO population, s is the index of the selected individual. Then we iterate the number of currently selected individuals by 1, $GA_{CurrNum} = GA_{CurrNum}+1$. We generate a random population of size $GA_{PopSize}$, the same method of random generation is used as the one in PSO. Set the first chromosome in the population of GA as the selected position in PSO, $GAx_1 = PSOx_s$. We then set the generation/iteration number of GA to , $GA_{CurrItr} = 1$. We initialize the replacement chromosome GA_{rep} as the fitness of We then obtain the fitness values $F(GAx_i)$ of each chromosome $GAx_i, i = 1, 2, \ldots, GAx_{PopSize}$ in the GA population using the same objective function in the model. Then we obtain the best feasible solution in the population GAx_b whose fitness value is the minimum, b is the index of the best feasible solution. We then compare the values of the best feasible solution in the GA population

Table 4.2: Vertices and their Characteristics

Vertices (i)	0	1	2	3	4	5	6	7	8
Load (q_i) in tons	0	2	1.5	4.5	3	1.5	4	2.5	3
Service Time (s_i) in hours	0	1	2	1	3	2	2.5	3	0.8
Time Window $[a_i, b_i]$	[0,14]	[1,4]	[4,6]	[1,2]	[4,7]	[3,5.5]	[2,5]	[5,8]	[1.5,4]

4.2 PSO-GA Testing

We first tested the effectiveness of the hybrid PSO-GA proposed by Garg[13] in solving vehicle routing problems by using the same particle encoding and model done by Liu et.al.[18]. Liu et.al.[18] worked on a Hyrbid PSO wherein the cross-over mechanism of GA was implemented on the PSO population. Each member of the population was crossed with their respective personal and global best positions.

They tested the algorithm for a small scale example, they used an undirected graph G whose vertices are given in table 4.2. Vertex 0 is the depot while the rest are the 8 customers. Each customer location has q_i amount to be delivered to it. Each customer location also specifies a time windows $[a_i, b_i]$ where a_i and b_i are the earliest and latest possible time that vehicle k can start serving customer i. The amount of time needed to service customer i is given by the service time s_i. The distances assigned to the edges are given in table 4.3. How this works is that since it is an undirected/symmetric graph, $(i, j) = (j, i)$, if we want to know the distance from any two nodes say the depot to the first customer location, we check the values of the table at $d_{(0,1)}$ or $d_{(1,0)} = 40$. The distances are measured yards, the time is measured in hours. Travel time is dictated by distance, it is assumed that all vehicles travel at constant speed of 50 yards per hour. Hence, if a vehicle travels from the depot to the first customer location, the distance traveled is 40 yards while the travel time is given as $40 \cancel{\text{yards}} \times \frac{1 \text{ hour}}{50 \cancel{\text{yards}}} = 0.8$ hours $= 48$ minutes. Three vehicles are considered in this test case, $K = \{1, 2, 3\}$. Each of the vehicles has the maximum capacity of $Q = 8$ tons.

We keep track of when a vehicle $l \in K$ arrives at a customer location ($arrival_{i,l}$). When a vehicle arrives too early at a customer location, it will have to become idle and wait there until the customer is ready to be served. 'Early' pertains to when the vehicle arrives at customer i before the span of it's time window ($a_i > arrival_{i,l}$). We

Table 4.3: Distances from Point to Point

$d_{(i,j)}$	0	1	2	3	4	5	6	7	8
0	0	40	60	75	90	200	100	160	80
1	40	0	65	40	100	50	75	110	100
2	60	65	0	75	100	100	75	75	75
3	75	40	75	0	100	50	90	90	150
4	90	100	100	100	0	100	75	75	100
5	200	50	100	50	100	0	70	90	75
6	100	75	75	90	75	70	0	70	100
7	160	110	75	90	75	90	70	0	100
8	80	100	75	150	100	75	100	100	0

penalize the vehicle for being too early at customer location, by calculating how much time it waits. **Waiting time** (P_E) of a vehicle at any given customer location is given as $P_{E_{i,l}} = \max\{0, a_i - arrival_{i,l}\}$. When a vehicle arrives too late at a customer location ($b_i < arrival_{i,l}$), it will still service the customer but a penalty is given for every hour after the designated time window. We calculate how much time has lapsed before the vehicle services customer i through **Delay Time** $P_{L_{i,l}} = \max\{0, arrival_{i,l} - b_i\}$.

The decision variables are as follows:

$$y_{k,i} = \begin{cases} 1 & \text{if the task of delivering at customer } i \text{ is completed by vehicle } k \\ 0 & \text{otherwise} \end{cases}$$

$$x_{i,j,k} = \begin{cases} 1 & \text{if vehicle } k \text{ travels from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases}$$

The mathematical model is given as follows:

$$\min z = \sum_i \sum_j \sum_k d_{i,j} \cdot x_{i,j,k} + P_E \sum_{i=1}^{L} \max(a_i - arrival_{i,l}, 0) + P_L \sum_{i=1}^{L} \max(arrival_{i,l} - LT_i, 0)$$

subjected to the following constraints:

$$\sum_i q_i \cdot y_{i,l} \leq Q, \qquad \forall k$$

$$\sum_k y_{k,i} = 1, \qquad i = 1, 2, \ldots, L$$

$$\sum_i x_{i,j,k} = y_{k,j}, \qquad j = 1, 2, \ldots, L; \forall k$$

$$\sum_j x_{i,j,k} = y_{k,i}, \qquad i = 1, 2, \ldots, L; \forall k$$

$$x_{ijk}, y_{k,i} = 0 \text{ or } 1, \qquad i, j = 1, 2, \ldots, L; \forall k$$

We want to minimize the total amount of distance traveled, the amount of time all vehicles stay idle and the amount of time all vehicles are late. The cost of travel distance is $1:1$, that is, one yard of distance traveled is equivalent to one cost. The amount of time a vehicle is idle and late are converted into distance so that they can be added to the cost. We know that the total amount of time that all vehicles are idle is given by the summation $\sum_{l=1}^{3}\sum_{i=0}^{8} P_{E_{i,l}}$. We also know that the total amount of time that all vehicles are late is given by the summation $\sum_{l=1}^{3}\sum_{i=0}^{8} P_{L_{i,l}}$. P_E and P_L can be converted into distance given the speed of the vehicles. For example, a vehicle waits for 1 hour, then we convert that into distance by $1 \cancel{\text{hour}} \times \frac{50 \text{ yards}}{1 \cancel{\text{hour}}} = 50$ yards. To clarify what we are computing, we let our cost function be

$$\text{Total Cost } = \text{Cost}_{\text{distance}} + \text{Cost}_{\text{waiting}} + \text{Cost}_{\text{late}} \tag{4.26}$$

The known solution to this problem is given by the routes:

- Vehicle 1: $(0 \to 3 \to 1 \to 2 \to 0)$
- Vehicle 2: $(0 \to 8 \to 5 \to 7 \to 0)$
- Vehicle 3: $(0 \to 6 \to 4 \to 0)$

To evaluate these routes, we add the distances traveled by each vehicle:

- Vehicle 1:

  $d_{(0,3)} + d_{(3,1)} + d_{(1,2)} + d_{(2,0)}$

  $75 + 40 + 65 + 60 = 240$
- Vehicle 2:

  $d_{(0,8)} + d_{(8,5)} + d_{(5,7)} + d_{(7,0)}$

  $80 + 75 + 90 + 160 = 405$

- Vehicle 3:

  $d_{(0,6)} + d_{(6,4)} + d_{(4,0)}$

  $100 + 75 + 90 = 265$

Thus, the total distance traveled by all vehicles is 910 yards. Next we compute how much waiting time and delay time are experienced by each vehicle in their respective routes

- Vehicle 1:

$$arrival_{0,1} = 0$$

$$P_{E_{0,1}} = \max\{0, (a_0 - arrival_{0,1})\} = \max\{0, 0\} = 0$$

$$P_{L_{0,1}} = \max\{0, (arrival_{0,1} - b_0)\} = \max\{0, -14\} = 0$$

$$arrival_{3,1} = \max\{a_0, arrival_{0,1}\} + d_{(0,3)} \times \frac{1 \text{ hour}}{50 \text{ yards}} + s_0$$

$$= 0 + \frac{75}{50} + 0 = 1.5 \text{ hours}$$

$$P_{E_{3,1}} = \max\{0, (a_3 - arrival_{3,1})\} = \max\{0, -0.5\} = 0$$

$$P_{L_{3,1}} = \max\{0, (arrival_{3,1} - b_3)\} = \max\{0, -0.5\} = 0$$

$$arrival_{1,1} = \max\{a_3, arrival_{3,1}\} + d_{(3,1)} \times \frac{1 \text{ hour}}{50 \text{ yards}} + s_3$$

$$= 1.5 + \frac{40}{50} + 1 = 3.3 \text{ hours}$$

$$P_{E_{1,1}} = \max\{0, (a_1 - arrival_{1,1})\} = \max\{0, -2.3\} = 0$$

$$P_{L_{1,1}} = \max\{0, (arrival_{1,1} - b_1)\} = \max\{0, -0.7\} = 0$$

$$arrival_{2,1} = \max\{a_1, arrival_{1,1}\} + d_{(1,2)} \times \frac{1 \text{ hour}}{50 \text{ yards}} + s_1$$

$$= 3.3 + \frac{65}{50} + 1 = 5.6 \text{ hours}$$

$$P_{E_{2,1}} = \max\{0, (a_2 - arrival_{2,1})\} = \max\{0, -1.6\} = 0$$

$$P_{L_{2,1}} = \max\{0, (arrival_{2,1} - b_2)\} = \max\{0, -0.4\} = 0$$

$$arrival_{0,1} = \max\{a_2, arrival_{2,1}\} + d_{(2,0)} \times \frac{1 \text{ hour}}{50 \text{ yards}} + s_2$$

$$= 5.6 + \frac{60}{50} + 2 = 7.0 \text{ hours}$$

$$P_{E_{0,1}} = \max\{0, (a_0 - arrival_{0,1})\} = \max\{0, -7.0\} = 0$$

$$P_{L_{0,1}} = \max\{0, (arrival_{0,1} - b_0)\} = \max\{0, -7.0\} = 0$$

- Vehicle 2:

$$
\begin{aligned}
arrival_{0,2} &= 0 \\
P_{E_{0,2}} &= \max\{0, (a_0 - arrival_{0,2})\} = \max\{0, 0\} = 0 \\
P_{L_{0,2}} &= \max\{0, (arrival_{0,2} - b_0)\} = \max\{0, -14\} = 0 \\
arrival_{8,2} &= \max\{a_0, arrival_{0,2}\} + d_{(0,8)} \times \frac{1 \text{ hour}}{50 \text{ yards}} + s_0 \\
&= 0 + \frac{80}{50} + 0 = 1.6 \text{ hours} \\
P_{E_{8,2}} &= \max\{0, (a_8 - arrival_{8,2})\} = \max\{0, -0.1\} = 0 \\
P_{L_{8,2}} &= \max\{0, (arrival_{8,2} - b_8)\} = \max\{0, -2.4\} = 0 \\
arrival_{5,2} &= \max\{a_8, arrival_{8,2}\} + d_{(8,5)} \times \frac{1 \text{ hour}}{50 \text{ yards}} + s_8 \\
&= 1.6 + \frac{75}{50} + 0.8 = 3.9 \text{ hours} \\
P_{E_{5,2}} &= \max\{0, (a_5 - arrival_{5,2})\} = \max\{0, -0.9\} = 0 \\
P_{L_{5,2}} &= \max\{0, (arrival_{5,2} - b_5)\} = \max\{0, -1.1\} = 0 \\
arrival_{7,2} &= \max\{a_5, arrival_{5,2}\} + d_{(5,7)} \times \frac{1 \text{ hour}}{50 \text{ yards}} + s_5 \\
&= 3.9 + \frac{90}{50} + 2 = 7.7 \text{ hours} \\
P_{E_{7,2}} &= \max\{0, (a_7 - arrival_{7,2})\} = \max\{0, -2.7\} = 0 \\
P_{L_{7,2}} &= \max\{0, (arrival_{7,2} - b_7)\} = \max\{0, -0.3\} = 0 \\
arrival_{0,2} &= \max\{a_7, arrival_{7,2}\} + d_{(7,0)} \times \frac{1 \text{ hour}}{50 \text{ yards}} + s_7 \\
&= 7.7 + \frac{160}{50} + 3 = 13.9 \text{ hours} \\
P_{E_{0,2}} &= \max\{0, (a_0 - arrival_{0,2})\} = \max\{0, -13.9\} = 0 \\
P_{L_{0,2}} &= \max\{0, (arrival_{0,2} - b_0)\} = \max\{0, -0.1\} = 0
\end{aligned}
$$

- Vehicle 3:

$$
\begin{aligned}
arrival_{0,3} &= 0 \\
P_{E_{0,3}} &= \max\{0, (a_0 - arrival_{0,3})\} = \max\{0, 0\} = 0 \\
P_{L_{0,3}} &= \max\{0, (arrival_{0,3} - b_0)\} = \max\{0, -14\} = 0 \\
arrival_{6,3} &= \max\{a_0, arrival_{0,3}\} + d_{(0,6)} \times \frac{1 \text{ hour}}{50 \text{ yards}} + s_0 \\
&= 0 + \frac{100}{50} + 0 = 2 \text{ hours} \\
P_{E_{6,3}} &= \max\{0, (a_6 - arrival_{6,3})\} = \max\{0, 0\} = 0 \\
P_{L_{6,3}} &= \max\{0, (arrival_{6,3} - b_6)\} = \max\{0, -3\} = 0 \\
arrival_{4,3} &= \max\{a_6, arrival_{6,3}\} + d_{(6,4)} \times \frac{1 \text{ hour}}{50 \text{ yards}} + s_6 \\
&= 2 + \frac{75}{50} + 2.5 = 6 \text{ hours} \\
P_{E_{4,3}} &= \max\{0, (a_4 - arrival_{4,3})\} = \max\{0, -2\} = 0 \\
P_{L_{4,3}} &= \max\{0, (arrival_{4,3} - b_4)\} = \max\{0, -1\} = 0 \\
arrival_{0,3} &= \max\{a_2, arrival_{4,3}\} + d_{(4,0)} \times \frac{1 \text{ hour}}{50 \text{ yards}} + s_4 \\
&= 6 + \frac{90}{50} + 3 = 10.8 \text{ hours} \\
P_{E_{0,3}} &= \max\{0, (a_0 - arrival_{0,3})\} = \max\{0, -10.8\} = 0 \\
P_{L_{0,3}} &= \max\{0, (arrival_{0,3} - b_0)\} = \max\{0, -3.2\} = 0
\end{aligned}
$$

The total of the penalties are both 0. Thus, out total cost is 910.

The particle and chromosome encoding scheme is done as follows. Each particle/chromosome is an $N + K - 1$ dimensional vector. The order of each element value denotes the order at which customers are visited. For example, if we have 3 vehicles and 5 collection sites, we have a particle of $5 + 2$ components. We let node 0 to be the depot and nodes $1 - 5$ as the collection sites. Given

$$
\begin{array}{cc}
\text{Nodes} & \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 0 & 0 \end{array} \\
\text{Particle} & \left[\begin{array}{ccccccc} 2 & 1 & 7 & 4 & 5 & 6 & 3 \end{array}\right]
\end{array}
$$

We arrange the nodes based on the respective particle component value. This results in the sequence,

$$\text{Nodes} \begin{bmatrix} 2 & 1 & 0 & 4 & 5 & 0 & 3 \end{bmatrix}$$

Therefore, the sequence of collection becomes:

$$0 \to 2 \to 1 \to 6 \to 0 \to 4 \to 5 \to 6 \to 0 \to 3 \to 6 \to 0$$

Hence, the routes of each vehicle becomes:

$$\text{Vehicle 1: } 0 \to 2 \to 1 \to 6 \to 0$$

$$\text{Vehicle 2: } 0 \to 4 \to 5 \to 6 \to 0$$

$$\text{Vehicle 3: } 0 \to 3 \to 6 \to 0$$

The PSO-GA algorithm was run with the following parameters:

- PSO Population Size = 40
- PSO Maximum Iterations = 200, 400, 800, 1200
- Cognitive and Social factors $c_1 = 1.5$, $c_2 = 1.5$
- Initial and final inertia weight $w_i = 0.9$ $w_f = 0.4$
- Crossover Rate = 0.85
- Mutation Rate = 0.02
- $\gamma = 10$
- $\beta = 15$
- GA Initial Population Size = 10
- GA Final Population Size = 5
- GA Minimum Iterations = 10
- GA Maximum Iterations = 15
- Acceptance Threshold = 1×10^{-5}

The algorithm and problems were encoded and run using Matlab v.2015 on a computer with the following specifications:

CPU = Intel i5-6200U 2.3 GHz

RAM = 16 Gb

OS = Windows 10 Home 2017

Table 4.4: Distances from Point to Point (Yards)

$d_{(i,j)}$	**0**	**1**	**2**	**3**
0	**0**	**120**	**80**	**50**
1	**120**	**0**	**70**	**100**
2	**80**	**70**	**0**	**30**
3	**50**	**100**	**30**	**0**

Table 4.5: Node Characteristics

Vertices (i)	**0**	**1**	**2**	**3**
Load (q_i)	**0**	**4**	**2**	**2**
Service Time (s_i)	**0**	**1**	**2**	**1**
Time Window $[a_i, b_i]$	**[0,12]**	**[1,4]**	**[5,6]**	**[2,5]**

4.3 Simpler Test Cases

4.3.1 TSP with Time Windows

We try a simple example wherein we have 3 customers, 1 depot and 1 vehicle. Time is measured in hours and distance in yards. Node 0 is the depot and the rest of the nodes are the 3 customers respectively. Note that this is a TSP problem since there is only one vehicle but it involves time windows. The distances of each edge $(i, j) \in V$ are given in table **??**. The service time, waste to collect, and time windows at each node $i \in V$ are given in table **??**. Note that vertex 0 is the depot and nodes $1 - 3$ are customers.

We consider 1 vehicle. Hence, there are $3! = 6$ possible permutations from which there is one known solution whose fitness value is 400:

- Vehicle Route: $(0 \to 1 \to 3 \to 2 \to 0)$

Note that we are still using the same cost function (4.26).

We solve the fitness of this route by first calculating the total distance

$$\text{Total Distance} = d_{(0,1)} + d_{(1,3)} + d_{(3,2)} + d_{(2,0)}$$

$$\text{Total Distance} = 120 + 100 + 30 + 80$$

$$\text{Total Distance} = 330$$

Now we calculate the cost of being early and late.

- Vehicle Route:

$$
\begin{aligned}
arrival_0 &= 0 \\
P_{E_0} &= \max\{0, (a_0 - arrival_0)\} = \max\{0, 0\} = 0 \\
P_{L_0} &= \max\{0, (arrival_0 - b_0)\} = \max\{0, -12\} = 0 \\
arrival_1 &= \max\{a_0, arrival_0\} + d_{(0,1)} \times \frac{1 \text{ hour}}{50 \text{ yards}} + s_0 \\
&= 0 + \frac{120}{50} + 0 = 2.4 \text{ hours} \\
P_{E_1} &= \max\{0, (a_1 - arrival_1)\} = \max\{0, -1.4\} = 0 \\
P_{L_1} &= \max\{0, (arrival_1 - b_1)\} = \max\{0, -1.6\} = 0 \\
arrival_3 &= \max\{a_1, arrival_1\} + d_{(1,3)} \times \frac{1 \text{ hour}}{50 \text{ yards}} + s_1 \\
&= 2.4 + \frac{100}{50} + 1 = 5.4 \text{ hours} \\
P_{E_3} &= \max\{0, (a_3 - arrival_3)\} = \max\{0, -3.4\} = 0 \\
P_{L_3} &= \max\{0, (arrival_3 - b_3)\} = \max\{0, 0.4\} = 0.4 \\
arrival_2 &= \max\{a_3, arrival_3\} + d_{(3,2)} \times \frac{1 \text{ hour}}{50 \text{ yards}} + s_3 \\
&= 5.4 + \frac{30}{50} + 1 = 7 \text{ hours} \\
P_{E_2} &= \max\{0, (a_2 - arrival_2)\} = \max\{0, -2\} = 0 \\
P_{L_2} &= \max\{0, (arrival_2 - b_2)\} = \max\{0, 1\} = 1 \\
arrival_0 &= \max\{a_2, arrival_2\} + d_{(2,0)} \times \frac{1 \text{ hour}}{50 \text{ yards}} + s_2 \\
&= 7 + \frac{80}{50} + 2 = 10.6 \text{ hours} \\
P_{E_0} &= \max\{0, (a_0 - arrival_0)\} = \max\{0, -10.6\} = 0 \\
P_{L_0} &= \max\{0, (arrival_0 - b_0)\} = \max\{0, -1.4\} = 0
\end{aligned}
$$

- Total Time Early:

$$P_E = 0 + 0 + 0 + 0 + 0$$
$$= 0$$

- Total Time Late:

$$P_E = 0 + 0 + 0.4 + 1 + 0$$
$$= 1.4$$

Hence the total cost of being early and late is $1.4 \cancel{\text{hour}} \times \frac{50 \text{ yards}}{1 \cancel{\text{hour}}} = 70$ yards. Therefore the total cost in yards is $330 + 70 = 400$. The possible permutations and their respective fitness values are given as follows:

- Vehicle Route: $(0 \to 3 \to 2 \to 1 \to 0) = 610$
- Vehicle Route: $(0 \to 2 \to 3 \to 1 \to 0) = 960$
- Vehicle Route: $(0 \to 3 \to 1 \to 2 \to 0) = 470$
- Vehicle Route: $(0 \to 2 \to 1 \to 3 \to 0) = 1010$
- Vehicle Route: $(0 \to 1 \to 2 \to 3 \to 0) = 410$
- Vehicle Route: $(0 \to 1 \to 3 \to 2 \to 0) = 400$

We test the basic PSO algorithm using the following parameters:

- PSO Population Size = 3, 6, 15, 40
- PSO Maximum Iterations = PSO Population Size $\times$ 5
- Cognitive and Social factors $c_1 = 1.5$, $c_2 = 1.5$
- Initial and final inertia weight $w_i = 0.9$ $w_f = 0.4$
- Acceptance Threshold = 1×10^{-5}

The same version of Matlab as well as the same computer was used to run the code.

Now we test the PSO-GA algorithm using the same problem for comparison. The PSO-GA algorithm was used using the following PSO-GA paramters:

- PSO Population Size = 3, 6, 15, 40
- PSO Maximum Iterations = PSO Population Size $\times$ 5
- Cognitive and Social factors $c_1 = 1.5$, $c_2 = 1.5$
- Initial and final inertia weight $w_i = 0.9$ $w_f = 0.4$

- Crossover Rate = 0.85
- Mutation Rate = 0.02
- $\gamma = 10$
- $\beta = 15$
- GA Minimum Replacement = 1
- GA Maximum Replacement = 1 (for Pop. Size = 3), 3 (for the rest)
- GA Initial Population Size = 10
- GA Final Population Size = 5
- GA Minimum Iterations = 10
- GA Maximum Iterations = 15
- Acceptance Threshold = 1×10^{-5}

4.3.2 VRPTW with 2 Vehicles

Now we test the same 3 nodes but this time, we now consider 2 vehicles and reduce each vehicle's capacity $Q = 4$. Hence, there are $4! = 24$ possible permutations from which there are two known solutions whose fitness is given as 520:

- Vehicle 1: $(0 \rightarrow 1 \rightarrow 0)$
- Vehicle 2: $(0 \rightarrow 3 \rightarrow 2 \rightarrow 0)$

  and
- Vehicle 1: $(0 \rightarrow 3 \rightarrow 2 \rightarrow 0)$
- Vehicle 2: $(0 \rightarrow 1 \rightarrow 0)$

We again test the PSO algorithm first using the following parameters:

- PSO Population Size = 7, 15, 24, 40
- PSO Maximum Iterations = PSO Population Size $\times$ 5
- Cognitive and Social factors $c_1 = 1.5$, $c_2 = 1.5$
- Initial and final inertia weight $w_i = 0.9$ $w_f = 0.4$
- Acceptance Threshold = 1×10^{-5}

Now we test the PSO-GA algorithm for the same problem. The PSO-GA algorithm was implemented using the following parameters:

- PSO Population Size = 7, 15, 24, 40

- PSO Maximum Iterations = PSO Population Size $\times$ 5
- Cognitive and Social factors $c_1 = 1.5$, $c_2 = 1.5$
- Initial and final inertia weight $w_i = 0.9$ $w_f = 0.4$
- Crossover Rate = 0.85
- Mutation Rate = 0.02
- $\gamma = 10$
- $\beta = 15$
- GA Minimum Taken = 1
- GA Maximum Taken = 5
- GA Initial Population Size = 10
- GA Final Population Size = 5
- GA Minimum Iterations = 10
- GA Maximum Iterations = 15
- Acceptance Threshold = 1×10^{-5}

Chapter 5

Results and Discussion

We now present the results obtained from the tests specified in the previous chapter. The results of the 8 node test case from Liu et.al [18] are presented in table 5.1.

Table 5.1: Summary of results for the test case having 8 Customers considering 3 Vehicles using the hybrid PSO-GA algorithm

Maximum Iteration	Best Value Found	Worst Value Found	Number of Converged Runs	Number of Successful Converged Runs	Average Running Time (s)	Standard Deviation
200	910	1425	3/10	1/3	104.094591	144.076561
400	910	1370	4/10	1/4	189.477791	131.471374
800	910	1255	8/10	1/8	377.513529	87.311702
1200	910	1100	4/10	1/4	588.434658	63.674519
1600	910	1070	7/10	3/7	742.915872	68.315038

As we can see, the algorithm found the optimal solution, which has the cost of 910. However, most the tests only had 1 successful converged runs. A successful converged run is where the population converged on the optimal solution. There were other converged runs, however, the population converged at sub-optimal solutions. It is also observed that there is a large standard deviation, this means that the solutions found in all runs vastly varies but the variation decreases as maximum iteration increases. Comparing this outcome with the results obtain by Liu et.al. [18] who obtained a 97% successful search rate, the PSO-GA can be said to be inferior however, there was a discrepancy in terms of solution acceptance. The PSO method applied by Liu et.al. did not us acceptance threshold that was implemented in the PSO-GA. They practically let the algorithm run

its course and output the global best fitness recorded. In our implementation, I only accept solutions wherein the population converged into a single solution and where the maximum iteration has not been reached.

The results of the simple 3 node test using the basic PSO are presented in table 5.2.

Table 5.2: Summary of results for the test case having 3 Customers considering 1 Vehicle using the basic PSO algorithm

Population Size	Maximum Iteration	Best Value Found	Worst Value Found	Number of Converged Runs	Number of Successful Converged Runs	Average Running Time (s)	Standard Deviation
3	15	400	610	15/20	11/15	0.005620	51.121630
6	30	400	470	19/20	17/19	0.012434	15.694451
15	75	400	1255	19/20	19/19	0.072328	0.000000
40	200	400	1100	18/20	18/18	0.563054	0.000000

The results show that as the population size increases, the solutions obtained become more stable. The number of successful converged runs (runs that converged to the optimal solution) for all test are more than 50%. The standard deviation tells us that there are no deviations for the solutions obtained on higher population sizes.

The results of the simple 3 node test using the hybrid PSO-GA are presented in table **??**. The results show that as the population size increases, the solution becomes more stable. Moreover, when we compare the values from table **??** and **??**, PSO and PSO-GA have almost the same results with a few differences. The major difference is in the average running time. PSO-GA is expected to run slower because it is a hybrid where more computations are needed.

The results of the simple 3 node test with 2 vehicles using the basic PSO algorithm are presented in table **??**. The results show the same trend as when there was only 1 vehicles. As the population size increases, the solution becomes more stable however, this time if the population size exceeds some number, there are less converged runs. This is so maybe because there are more members in the population for convergence to occur

before the maximum iteration is reached. In more than 50% of the runs, the population successfully converged at the optimal solution.

The results of the simple 3 node test with 2 vehicles using the hybrid PSO-GA algorithm are presented in table **??**. The results reflect the same trend as with the test case involving only 1 vehicle. As the population size increases, the solution becomes more stable however, at after some point, the number of converged runs become less. This is so maybe because there are more members in the population that need to converge and that the maximum iteration is too small for convergence to occur. If we take a look at the results of both the basic PSO and hybrid PSO-GA algorithms, PSO-GA performed better and is more consistent with the number of iterations. All of PSO-GA's converged runs are all successful ones, this means that for all the times the population converged, it obtained the optimal solution. If we take the average running time into consideration, PSO-GA is less efficient but this is normal because the PSO-GA algorithm has more complexity and computations.

The results of the Baguio City waste collection vehicle routing of 129 collection sites using the hybrid PSO-GA is shown on table **??**. As we can see, there are no converged runs for maximum iterations lower than 600. The single converged run generated a sequence that gives a total amount of $1,284.830000$ kilometers using 74 vehicles, the surplus of distance comes from the penalty of having a large sum of available space left unused in the 74 vehicles. Given that there the total amount randomly generated for all 129 barangays is 924 cubic meters, and that the capacity of each vehicle is 14s cubic meters, then the estimated number of vehicle is given by

$$m = \frac{W}{Q} = \frac{924}{14} = 66$$

Then therefore there was an additional 112 cubic meters of unused space or capacity in the 74 vehicles. The penalty was $1500\sqrt[3]{112} = 7230.426792$ kilometers which was extremely high. Therefore, α should be set to a smaller number.

Chapter 6

Conclusion and Recommendation

A hybrid PSO-GA algorithm was used in order to solve the waste collection vehicle routing problem. The results obtained during the preliminary testing show that the hybrid PSO-GA proposed by Harish Garg[13] can indeed solve the vehicle routing problems however, in order to have population convergence, the maximum iterations needed must be very high even for small population sizes. The optimal set of routes gave the minimum distance of $1,284.830000$ kilometers using 74 vehicles that have the capacity of 14 m^3.

For the next study, I recommend that the specific collection sites per barangay be used instead of just barangay halls and landmarks. This will surely produce more accurate results.

List of References

[1] *Hybrid Particle Swarm Optimization for vehicle routing problem with Time Windows.*

[2] M. Akhtar, M. A. Hannan, and H. Basri, *Particle swarm optimization modeling for solid waste collection problem with constraints*, in 2015 IEEE 3rd International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA), Nov 2015, pp. 1–4.

[3] K. Buhrkal, A. Larsen, and S. Ropke, *The waste collection vehicle routing problem with time windows in a city logistics context*, Procedia - Social and Behavioral Sciences, 39 (2012), pp. 241 – 254. Seventh International Conference on City Logistics which was held on June 7- 9,2011, Mallorca, Spain.

[4] J. Carr, *An introduction to genetic algorithms*, (2014).

[5] M. Clerc, *The swarm and the queen: towards a deterministic and adaptive particle swarm optimization*, Evolutionary Computation, (1999).

[6] M. Clerc and J. Kennedy, *The particle swarm - explosion, stability, and convergence in a multidimensional complex space*, IEEE Transactions on Evolutionary Computation, 6 (2002), pp. 58–73.

[7] J.-F. Cordeau, G. Laporte, M. W. Savelsbergh, and D. Vigo, *Chapter 6 vehicle routing*, in Transportation, C. Barnhart and G. Laporte, eds., vol. 14 of Handbooks in Operations Research and Management Science, Elsevier, 2007, pp. 367 – 428.

[8] G. B. Dantzig and J. H. Ramser, *The truck dispatching problem*, Manage. Sci., 6 (1959), pp. 80–91.

[9] R. C. Eberhart and J. Kennedy, *A new optimizer using particle swarm theory*, (1995).

[10] ——, *Particle swarm optimization*, (1995).

[11] R. C. Eberhart and Y. Shi, *Particle swarm optimization: developments, applications and resources*, Evolutionary Computation, (2001).

[12] Full Advantage Philippines International Incorporated, *Ten-year ecological solid waste management plan 2015-2014*.

[13] H. Garg, *A hybrid pso-ga algorithm for constrained optimization problems*, Applied Mathematics and Computation, 274 (2016), pp. 292–305.

[14] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st ed., 1989.

[15] F. Heppner and G. U.

[16] A. Kaveh and S. Talatahari, *Engineering optimization with hybrid particle swarm and ant colony opitmization*, Asian Journal of Civil Engineering, 10 (2009), pp. 611–628.

[17] H. C. Lacsamana, *Baguio pays out p1 b for waste hauling in 10 yrs*, Baguio Midland Courier.

[18] J. Liu and J. Lampinen, *A fuzzy adaptive differential evolution algorithm*, 9 (2005), pp. 448–462.

[19] M. M. Solomon, *Algorithms for the vehicle routing and scheduling problems with time window constraints*, 35 (1987), pp. 254–266.

[20] T. Nuortio, J. Kytjoki, H. Niska, and O. Brysy, *Improved route planning and scheduling of waste collection and transport*, Expert Systems with Applications, 30 (2006), pp. 223 – 232.

[21] M. Obitko, *Intoduction to genetic algorithms*.

[22] F. Olowan, E. Bilog, L. Y. Jr., E. Avila, J. Alangsab, E. Datuin, P. Fianza, L. Farinas, A. Allad-iw, B. Bomogao, and M. Lawana, *Baguio city council okays plastic free ordinance*, Sun Star Baguio.

[23] M. Omran, *Codeq: an effective metaheuristic for continuous global optimisation*, International Journal of Metaheuristics, 1 (2010), pp. 108–131.

[24] C. W. Reynolds, *Flocks, herds, and schools: A distributed behavioral model*, ACM SIGGRAPH Computer Graphics, 21 (1987), pp. 25–34.

[25] D. See, *Approval of citys solid waste plan in order*, Sun Star Baguio.

[26] ——, *City to purchase 4 more trucks*, Sun Star Baguio.

[27] L. H. Son, *Optimizing municipal solid waste collection using chaotic particle swarm optimization in gis based environments: A case study at danang city, vietnam*, Expert Systems with Applications, 41 (2014), pp. 8062 – 8074.

[28] J. Sun, B. Feng, and W. Xu, *Particle swarm optimization with particles having quantum behavior*, in Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753), vol. 1, June 2004, pp. 325–331 Vol.1.

[29] J. Sun, W. Xu, and B. Feng, *A global search strategy of quantum-behaved particle swarm optimization*, in IEEE Conference on Cybernetics and Intelligent Systems, 2004., vol. 1, Dec 2004, pp. 111–116 vol.1.

[30] D. Tung and A. Pinnoi, *Vehicle routingscheduling for waste collection in hanoi*, 125 (2000), pp. 449–468.

[31] T. Xiang, X. Liao, and K.-w. Wong, *An improved particle swarm optimization algorithm combined with piecewise linear chaotic map*, 190 (2007), pp. 1637–1645.

Appendix A

Table of Node Markers

Table A.1: Nodes and their Google Maps Markers

BARANGAY	**MARKER**
A. Bonifacio-Caguioa-Rimando (ABCR)	Abanao-Zandueta-Kayong-Chugum-Otek (AZKCO), Baguio, Benguet
Abanao-Zandueta-Kayong-Chugum-Otek (AZKCO)	ABCR MULTI-PURPOSE BRGY.HALL, Rimando Road, Baguio City, Benguet, Philippines
Alfonso Tabora	Alfonso Tabora, Baguio City, Benguet, Philippines
Ambiong	AMBIONG BAGUIO BARANGAY HALL, Ambiong Road, Baguio City, Benguet, Philippines
Andres Bonifacio (Lower Bokawkan)	Andres Bonifacio (Lower Bokawkan), Baguio City, Benguet, Philippines
Apugan-Loakan	Apugan Barangay Hall, Loakan Road, Baguio, 2600 Benguet
Asin Road	Asin Road, Baguio City, Benguet, Philippines
Atok Trail	Atok Trail Barangay Hall, Baguio City, Benguet, Philippines
Aurora Hill Proper (Malvar-Sgt. Floresca)	Aurora Hill Proper Barangay Hall, Malvar Street, Baguio City, Benguet, Philippines
Aurora Hill, North Central	Aurora Hill, North Central, Baguio City, Benguet, Philippines

Table A.1 continued from previous page

Aurora Hill, South Central	Aurora Hill, South Central, Baguio City, Benguet, Philippines
Bagong Lipunan (Market Area)	Bagong Lipunan (Market Area), Baguio City, Benguet, Philippines
Bakakeng Central	Bakakeng Central, Marcos Highway, Brgy., Baguio, 2600 Benguet
Bakakeng North	Bakakeng Sur Rd, Norte, Baguio, 2600 Benguet
Bal-Marcoville (Marcoville)	Bal-Marcoville (Marcoville), Baguio City, Benguet, Philippines
Balsigan	Balsigan Road, Baguio, Benguet
Bayan Park East	East Bayan Park Aurora Hill Barangay Multi Purpose Hall
Bayan Park Village	Bayan Park Village Barangay Hall
Bayan Park West (Bayan Park)	Bayan Park West (Bayan Park), Baguio City, Benguet
BGH Compound	Baguio General Hospital Driveway, Baguio, 2600 Benguet
Brookside	35 Lower Brookside, Baguio, Benguet
Brookspoint	Brookspoint Rd, Baguio, Benguet
Cabinet Hill-Teacher's Camp	Cabinet Hill-Teacher's Camp, Baguio City, Benguet, Baguio, Benguet

Table A.1 continued from previous page

Camdas Subdivision	Camdas Subdivision, Baguio City, Benguet, Philippines
Camp 7	CAMP 7 BARANGAY HALL, Kennon Road, Baguio City, Benguet, Philippines
Camp 8	Camp 8 Health Center, Kennon Road, Baguio City, Benguet, Philippines
Camp Allen	CAMP HENRY T. ALLEN, Baguio City, Benguet, Philippines
Campo Filipino	CAMPO FILIPINO BARANGAY HALL, Quirino Highway, Baguio City, Benguet, Philippines
City Camp Central	City Camp District Health Center, City Camp Road, City Camp Central, Baguio City, Benguet, Philippines
City Camp Proper	City Camp Barangay Hall, City Camp Road, Baguio City, Benguet, Philippines
Country Club Village	Country Club Village Baguio City, Upper Country Club Road, Baguio, Benguet, Philippines
Cresencia Village	Cresencia Village Barangay, Bado Dangwa, Baguio City, Benguet, Philippines

Table A.1 continued from previous page

Dagsian, Lower	83, Lower Dagsian Barangay Hall, Baguio City, Benguet, Philippines
Dagsian, Upper	Upper Dagsian Barangay Hall, Lower Dagsian Road, Baguio City, Benguet, Philippines
Dizon Subdivision	DIZON-MANZANILLO SUBDIVISION, Kalapati Street, Baguio City, Benguet, Philippines
Dominican Hill-Mirador	DOMINICAN - MIRADOR BARANGAY, Extension Road, Baguio City, Benguet, Philippines
Dontogan	Dontogan Barangay Hall, Santo Tomas Road, Baguio City, Benguet, Philippines
DPS Compound	DPS Compound Barangay Hall, DBS Compound, Baguio City, Benguet, Philippines
Engineers' Hill	Engineer's Hill Barangay Hall, Marcoville Street, Baguio City, Benguet, Philippines
Fairview Village	FAIRVIEW Barangay Hall, Upper Fairview Ferguson Road, Baguio City, Benguet, Philippines
Ferdinand (Happy Homes-Campo Sioco)	Brgy. Ferdinand Barangay Hall, Baguio City, Benguet, Philippines
Fort del Pilar	Fort Del Pilar, Loakan Road, Baguio City, Benguet, Philippines

Table A.1 continued from previous page

Gabriela Silang	Gabriela Silang Covered Court, Gabriela Silang Road, Baguio City, Benguet, Philippines
General Emilio F. Aguinaldo (QuirinoMagsaysay, Lower)	Danes Bakeshop, Everlasting Street, Baguio
General Luna, Upper	LOWER GENERAL LUNA BARANGAY HALL, Baguio City, Benguet, Philippines
General Luna, Lower	UPPER GENERAL LUNA, Gen. Luna Road, Baguio City, Benguet, Philippines
Gibraltar	Gibraltar Barangay Hall, C. Arellano Street, Baguio City, Benguet, Philippines
Greenwater Village	Greenwater Village, Baguio City, Benguet, Philippines
Guisad Central	Central Guisad Barangay Hall, Pucay Subdivision Road, Baguio City, Benguet, Philippines
Guisad Sorong	Guisad Surong Barangay Hall, Unnamed Road, Baguio City, Benguet, Philippines
Happy Hollow	Happy Hallow Barangay Hall, Brgy. Happy Hallow, Baguio, Benguet
Happy Homes (Happy Homes-Lucban)	Happy Homes (Happy Homes-Lucban) Barangay Hall
Harrison-Claudio Carantes	Harrison-Claudio Carantes, Baguio City, Benguet, Philippines

Table A.1 continued from previous page

Hillside	Hillside Baranggay Hall, Hillside Road, Baguio, Benguet
Holy Ghost Extension	Holy Ghost Extension Barangay Hall, Holy Ghost Extension Road, Baguio City, Benguet, Philippines
Holy Ghost Proper	Holy Ghost Proper, Baguio City, Benguet, Philippines
Honeymoon (Honeymoon-Holy Ghost)	Honeymoon-Holyghost Barangay Hall, Baguio City, Benguet, Philippines
Imelda R. Marcos (La Salle)	Imelda R. Marcos (La Salle), Baguio City, Benguet, Philippines
Imelda Village	Imelda Village Barangay Multipurpose Hall, Baguio City, Benguet, Philippines
Irisan	IRISAN BARANGAY HALL, Baguio City, Benguet, Philippines
Kabayanihan	Kabayanihan Barangay Hall, Central Business District, Mabini Street, Baguio, Benguet
Kagitingan	Kagitingan Barangay Hall, Lower Bonifacio Street, Baguio City, Benguet, Philippines

Table A.1 continued from previous page

Kayang Extension	Kayang Extension, Baguio City, Benguet, Philippines
Kayang-Hilltop	Kayang Hilltop Barangay, Hilltop Street, Brgy. Kayang Hilltop, Baguio City, Benguet, Philippines
Kias	Kias, Baguio City, Benguet, Philippines
Legarda-Burnham-Kisad	Barangay Office Burnham-Legarda Brgy., Gen. Lim Street, Baguio City, Benguet, Philippines
Liwanag-Loakan	Liwanag-Loakan, Baguio City, Benguet, Philippines
Loakan Proper	Loakan Proper Barangay Hall, Purok Bubon, Baguio City, Benguet, Philippines
Lopez Jaena	Lopez Jaena, Baguio City, Benguet, Philippines
Lourdes Subdivision Extension	Lourdes Subdivision Extension, Baguio City, Benguet, Philippines
Lourdes Subdivision, Lower	Lower Lourdes Day Care and Multipurpose Hall, Baguio City, Benguet, Philippines
Lourdes Subdivision, Proper	Lourdes Barangay Hall, Baguio City, Benguet, Philippines
Lualhati	LUALHATI BARANGAY HALL, Baguio City, Benguet, Philippines
Lucnab	Lucnab, Baguio City, Beanguet, Philippines

Table A.1 continued from previous page

Magsaysay Private Road	Magsaysay Private Road, Baguio City, Benguet, Philippines
Magsaysay, Lower	Lower Magsaysay Barangay Multi-Purpose Hall, Lower Magsaysay Avenue, Baguio City, Benguet, Philippines
Magsaysay, Upper	Magsaysay, Upper, Baguio City, Benguet, Philippines
Malcolm Square-Perfecto (Jose Abad Santos)	Malcolm Square-perfecto (Jose Abad Santos), Baguio City, Benguet, Philippines
Manuel A. Roxas	Manuel Roxas Barangay, Baguio City, Benguet, Philippines
Market Subdivision, Upper	Market Subdivision, Upper, Baguio City, Benguet, Philippines
Middle Quezon Hill Subdivision (Quezon Hill Middle)	Middle Quezon Hill Subdivision (Quezon Hill M, Baguio City, Benguet, Philippines
Military Cut-off	MILITARY CUT OFF BARANGAY HALL, Military Cutoff Road, Baguio City, Benguet, Philippines
Mines View Park	Mines View Multipurpose Cooperative, Baguio City, Benguet, Philippines

Table A.1 continued from previous page

Modern Site, East	EAST MODERNSITE BRGY, P. Ledesma Street, Baguio, Benguet
Modern Site, West	Modern Site, West, Baguio City, Benguet, Philippines
MRR-Queen of Peace	MRR-Queen Of Peace, Baguio City, Benguet, Philippines
New Lucban	New Lucban Barangay Hall, New Lucban Road, Baguio City, Benguet, Philippines
Outlook Drive	Outlook Drive South, Baguio City, Benguet, Philippines
Pacdal	Pacdal Barangay Hall, Siapno Road, Baguio City, Benguet, Philippines
Padre Burgos	P. BURGOS MULTI PURPOSE HALL, Upper P. Burgos, Baguio, Benguet
Padre Zamora	Padre Zamora Barangay Hall
Palma-Urbano (Cario-Palma)	Palma-Urbano (Carino-Palma), Baguio City, Benguet, Philippines
Phil-Am	Barangay Hall Phil-am, Worcester Road, Baguio City, Benguet, Philippines
Pinget	Pinget Barangay Hall, Baguio City, Benguet, Philippines

Table A.1 continued from previous page

Pinsao Pilot Project	Barangay Hall, Pinsao Road, Baguio City, Benguet, Philippines
Pinsao Proper	Pinsao Proper Barangay Hall, Baguio City, Benguet, Philippines
Poliwes	POLIWES BARANGAY HALL, Puliwes Road, Baguio City, Benguet, Philippines
Pucsusan	Pucsusan, Baguio City, Benguet, Philippines
Quezon Hill Proper	Quezon Hill Proper Barangay Hall, Quezon Hill Road 1, Baguio City, Benguet, Philippines
Quezon Hill, Upper	Upper Quezon Hill, Tin, Brgy Upper Quezon Hill, Baguio, Benguet
Quirino Hill, East	Block 7, East Quirino Hill Barangay Hall, Quirino Hill Road, Baguio, Benguet
Quirino Hill, Lower	Lower Quirino Hill Barangay Hall, Baguio, Benguet
Quirino Hill, Middle	MIDDLE QUIRINO HILL BARANGAY HALL, Baguio, Benguet
Quirino Hill, West	West Quirino Hill Barangay Hall, Baguio, Benguet
Quirino-Magsaysay, Upper (Upper QM)	Upper Q - M Barangay Hall, Jasmin Street, Barangay Upper Q.M., Baguio City, Benguet, Philippines

Table A.1 continued from previous page

Rizal Monument Area	Rizal Monument Barangay Hall, Baguio, Benguet
Rock Quarry, Lower	MULTI PURPOSE BRGY. HALL LOWER ROCK QUARRY, Lower Rock Quarry, Baguio City, Benguet, Philippines
Rock Quarry, Middle	Barangay Middle Rock Quarry Multi - Purpose Building, Lower Rock Quarry, Brgy. Middle Rock Quarry, Baguio, Benguet
Rock Quarry, Upper	Upper Rock Quarry Barangay Hall, Lower Rock Quarry, Brgy. Upper Rock Quarry, Baguio City, Benguet, Philippines
Saint Joseph Village	St. Joseph Village Barangay Hall, Everlasting, Navy Base - Polo Field, St. Joseph Village, Baguio, Benguet
Salud Mitra	Barangay Hall, Baguio City, Benguet, Philippines
San Antonio Village	Leonila Hill Barangay Hall, Evangelista Street, Baguio City, Benguet, Philippines
San Luis Village	SAN LUIS VILLAGE BARANGAY HALL, Asin Road, Baguio City, Benguet, Philippines
San Roque Village	Church of Christ at Pines, Baguio, Benguet
San Vicente	San Vicente-Baguio City Multipurpose Cooperative, Kennon Road, Baguio, Benguet

Table A.1 continued from previous page

Sanitary Camp, North	Sanitary Camp, North, Baguio City, Benguet, Philippines
Sanitary Camp, South	Brgy. South Sanitary Camp Multi Purpose Hall, South Sanitary Camp Road, Baguio City, Benguet, Philippines
Santa Escolastica	Santa Escolastica Village Hall, Sta. Escolastica, Baguio, Benguet
Santo Rosario	Santo Rosario Barangay Hall, Sto. Rosario Village Road, Baguio City, Benguet, Philippines
Santo Tomas Proper	Sto Tomas Proper Barangay Hall, Baguio, Benguet
Santo Tomas School Area	Santo Tomas School Area, Baguio City, Benguet, Philippines
Scout Barrio	Scout Barrio Basketball Court, Baguio City, Benguet, Philippines
Session Road Area	BARANGAY HALL, Gov. Pack Road, Baguio City, Benguet, Philippines
Slaughter House Area (Santo Nio Slaughter)	BARANGAY STO. NIO SLAUGHTER BARANGAY HALL

Table A.1 continued from previous page

SLU-SVP Housing Village	SLU-SVP Housing Village Barangay, Baguio City, Benguet, Philippines
South Drive	Southdrive Barangay, South Drive, Baguio City, Benguet, Philippines
Teodora Alonzo	T Alonzo Barangay Hall, T. Alonzo Street, Baguio City, Benguet, Philippines
Trancoville	Trancoville, Baguio City, Benguet, Philippines
Victoria Village	VICTORIA VILLAGE BARANGAY HALL, Baguio City, Benguet, Philippines
DEPOT	**MARKER**
Irisan Dumpsite	Purok 18, Barangay Irisan, Baguio, 2600 Benguet

Appendix B

Table Showing the Distances Between each Barangay Marker

Table B.1: Distance between Barangay Markers in Kilometers

NODES	depot	1	2	3	4	5	6	7	8
depot	0	5.005	6.154	5.88	7.643	4.407	12.017	3.42	12.805
1	5.005	0	1.643	2.426	3.232	1.396	7.752	5.458	8.54
2	6.154	1.643	0	0.89	1.589	2.663	8.267	6.489	9.055
3	5.88	2.426	0.89	0	1.943	2.13	9.157	6.615	9.945
4	7.643	3.232	1.589	1.943	0	3.989	9.856	8.474	10.644
5	4.407	1.396	2.663	2.13	3.989	0	8.706	4.975	9.423
6	12.017	7.752	8.267	9.157	9.856	8.706	0	12.872	0.789
7	3.42	5.458	6.489	6.615	8.474	4.975	12.872	0	13.66
8	12.805	8.54	9.055	9.945	10.644	9.423	0.789	13.66	0
9	6.994	2.418	0.775	1.057	0.888	3.094	9.042	7.395	9.582
10	7.027	2.721	1.079	1.327	0.632	3.295	9.345	7.596	9.886
11	7.948	3.282	1.64	1.491	1.898	4.216	9.401	8.517	10.189
12	4.832	1.026	2.29	2.35	4.209	1.338	8.725	5.401	9.053
13	6.044	3.429	4.608	5.334	6.197	4.108	7.292	5.43	8.726
14	7.865	3.431	4.551	5.277	6.14	4.265	5.323	7.251	6.11
15	7.197	2.122	2.571	3.461	4.16	2.908	5.807	8.219	6.615
16	6.47	2.486	3.726	4.256	5.315	3.226	5.592	7.137	6.38
17	7.774	3.363	1.72	2.17	0.173	4.042	9.987	8.343	10.528
18	7.423	3.013	1.37	1.723	0.219	3.691	9.389	7.992	10.178
19	6.946	3.492	2.29	1.246	1.252	3.214	10.558	7.778	11.346
20	6.197	1.932	3.038	3.898	4.627	2.886	6.182	7.452	7.139

Table B.1 continued from previous page

21	7.048	2.129	0.486	1.136	1.619	3.074	8.765	7.665	9.307
22	7.784	3.302	1.66	2.016	0.459	4.052	9.926	8.353	10.467
23	7.172	2.895	2.416	2.955	2.884	3.678	8.026	10.35	8.867
24	5.676	2.222	2.122	0.808	2.667	1.944	9.641	6.245	9.676
25	9.024	4.759	5.865	6.672	7.454	5.713	2.993	9.88	3.781
26	6.587	2.322	3.428	4.288	5.017	3.276	5.43	7.842	6.218
27	4.383	0.577	1.841	1.901	3.76	0.889	7.818	4.952	8.604
28	4.019	0.888	2.135	1.861	3.724	0.848	8.181	4.588	8.898
29	4.852	0.786	2.05	2.752	3.639	1.74	8.323	4.657	8.52
30	4.746	0.68	1.944	2.646	3.533	1.634	7.633	5.314	8.421
31	10.157	5.892	4.38	4.919	6.235	6.162	8.383	11.335	9.225
32	4.843	1.374	2.059	1.336	3.195	0.754	8.684	5.411	9.472
33	8.233	3.968	4.796	5.686	6.385	4.922	4.746	9.488	5.535
34	8.07	3.805	4.723	5.613	6.312	4.759	4.17	9.325	4.959
35	5.414	1.96	1.894	0.546	2.405	1.682	9.583	5.983	9.448
36	3.832	1.964	3.228	3.233	5.092	1.593	9.105	4.4	9.698
37	6.616	5.917	7.097	7.823	8.686	7.653	9.587	6.002	10.375
38	7.389	2.21	2.763	3.653	4.352	3.841	5.687	8.409	6.505
39	6.188	2.406	1.982	2.872	3.571	2.694	6.214	8.451	6.9
40	4.004	1.626	2.893	2.321	4.18	0.548	8.937	4.742	9.653
41	5.774	2.202	3.442	4.168	5.133	3.156	6.823	6.678	7.611
42	15.446	11.181	11.696	12.586	13.285	12.135	3.43	16.3	3.679
43	8.148	3.883	4.711	5.601	6.3	4.837	5.049	9.335	5.821
44	4.629	0.837	2.101	2.803	3.69	1.791	7.542	4.921	8.279
45	5.47	0.884	0.978	1.868	2.567	1.976	7.355	6.039	7.776
46	5.741	1.252	1.281	2.171	2.87	2.561	6.986	6.623	7.774
47	8.588	4.122	3.941	5.017	4.409	5.431	8.88	11.832	9.807
48	7.652	2.89	3.479	4.369	5.068	4.341	4.999	8.907	5.798
49	5.148	2.56	2.406	1.626	3.485	1.058	9.853	6.996	10.163
50	5.032	2.115	2.78	1.982	3.841	0.943	9.408	5.6	10.196

Table B.1 continued from previous page

51	11.991	7.121	6.48	7.37	6.682	7.996	10.217	13.169	11.005
52	5.556	2.125	1.492	0.368	2.227	1.824	9.418	6.315	9.556
53	5.128	0.626	1.394	2.302	2.983	1.935	7.083	5.697	8.144
54	7.577	3.262	4.14	5.03	5.729	4.266	4.477	8.764	5.25
55	6.817	2.248	1.21	2.101	2.52	3.809	8.044	7.386	8.833
56	6.273	1.686	1.301	2.191	2.89	2.779	7.65	7.057	8.438
57	5.757	1.791	1.548	2.438	3.137	3.1	6.546	8.729	7.579
58	6.736	2.738	3.917	4.643	5.609	3.631	7.247	6.121	8.035
59	7.884	2.928	1.577	2.115	2.045	4.152	9.844	8.619	8.358
60	1.424	5.698	6.962	6.855	8.714	5.215	12.63	4.228	13.418
61	5.423	0.62	1.383	2.273	2.972	1.929	7.64	5.991	8.428
62	5.475	0.672	0.679	1.569	3.41	1.984	7.685	6.043	8.473
63	4.567	0.779	1.792	2.103	3.381	1.073	8.072	5.136	8.74
64	5.341	0.508	1.343	2.264	4.078	1.847	7.521	5.909	8.922
65	13.472	9.357	9.673	10.563	11.259	10.285	1.651	14.521	1.9
66	5.376	1.022	2.206	3.03	3.795	2	7.101	5.944	7.889
67	10.508	6.82	7.136	8.026	8.722	7.321	1.889	11.557	2.678
68	10.178	6.073	7.859	8.021	9.445	6.991	1.644	11.227	2.432
69	7.185	2.524	1.174	1.318	0.727	3.268	9.44	7.753	10.228
70	3.773	1.551	2.735	3.007	4.77	1.534	8.563	4.341	9.351
71	3.768	1.676	2.86	3.002	4.765	1.529	8.398	4.336	9.186
72	3.548	1.809	2.944	2.782	4.545	1.309	8.821	4.116	9.609
73	11.076	4.454	4.273	4.812	4.741	5.763	9.52	12.472	10.308
74	11.76	7.108	5.971	6.51	6.439	8.573	10.204	13.156	10.992
75	6.049	1.682	1.708	2.58	3.297	2.132	8.695	6.617	9.483
76	5.639	1.164	1.19	1.127	2.89	1.722	8.177	6.207	8.965
77	5.98	2.358	2.685	2.514	4.277	2.063	9.651	6.548	10.439
78	5.325	0.522	0.985	2.626	3.325	1.831	7.535	5.893	8.323
79	8.467	3.343	1.992	2.531	2.46	4.55	10.259	9.035	11.047
80	5.854	2.232	2.559	2.388	4.151	1.937	9.525	6.422	10.313

Table B.1 continued from previous page

81	3.76	2.796	3.809	3.769	5.532	1.91	9.982	4.986	10.77
82	5.993	1.887	3.169	3.836	4.758	2.806	5.65	7.389	6.439
83	11.192	4.878	4.697	5.236	5.165	6.187	9.636	12.588	10.424
84	7.432	2.49	1.14	1.496	0.986	3.515	9.406	8	10.194
85	6.742	1.939	0.588	1.126	1.158	3.146	8.855	7.31	9.643
86	4.037	1.815	2.999	3.271	5.034	1.798	8.827	4.605	9.615
87	5.795	1.518	0.489	1.379	2.078	2.786	8.059	6.364	8.848
88	10.757	6.492	4.407	5.297	5.996	5.374	9.036	12.012	9.825
89	7.639	5.525	3.178	4.068	4.767	4.145	8.069	11.045	8.858
90	5.015	1.561	1.38	1.656	2.969	1.283	8.145	5.584	8.934
91	5.676	1.399	1.161	2.051	2.75	2.182	7.926	6.245	8.715
92	3.689	0.626	1.89	2.592	3.479	1.58	7.645	4.945	8.433
93	5.39	2.094	3.334	4.06	4.923	3.048	6.664	7.106	7.452
94	5.808	3.042	2.976	1.628	3.487	2.764	9.741	7.065	10.53
95	5.941	3.305	3.422	2.642	4.501	2.311	10.544	7.367	11.332
96	3.734	2.953	4.032	3.252	5.111	1.708	9.961	5.16	10.749
97	6.446	2.869	3.975	4.835	5.564	3.823	5.139	8.389	5.927
98	8.665	7.239	4.892	5.782	6.481	5.859	9.783	12.759	10.572
99	3.308	2.063	3.469	2.897	4.756	1.124	9.373	4.734	10.09
100	3.274	2.564	3.969	3.397	5.256	1.624	9.995	4.999	10.783
101	6.186	3.42	3.32	2.006	3.865	3.142	10.839	7.726	11.627
102	5.35	2.584	2.484	1.17	3.029	2.306	9.249	6.89	10.791
103	5.628	2.862	2.762	1.448	3.307	2.584	9.527	6.885	10.316
104	6.482	3.716	3.616	2.302	4.161	3.438	10.381	7.739	11.17
105	4.518	1.414	2.678	3.38	4.267	2.368	7.494	5.774	8.282
106	4.145	0.354	1.395	2.32	2.984	1.308	7.658	5.401	8.447
107	3.658	1.08	2.344	3.046	3.933	2.065	7.774	4.914	8.562
108	3.307	1.431	2.695	3.397	4.284	1.714	8.125	4.563	8.913
109	3.567	1.172	2.436	3.138	4.025	1.974	7.778	4.823	8.566
110	6.84	3.251	3.067	3.957	4.656	4.034	8.712	10.706	9.501

Table B.1 continued from previous page

111	8.336	4.759	5.865	6.725	7.454	5.713	2.993	9.88	3.781
112	7.133	3.679	1.549	1.433	0.51	3.401	9.568	7.702	10.357
113	3.881	2.676	3.94	3.833	5.692	2.193	9.608	2.81	10.396
114	2.791	6.817	8.081	7.974	9.833	6.334	13.749	5.347	14.537
115	6.674	3.097	4.203	5.063	5.792	4.051	5.443	7.653	6.231
116	6.475	3.709	2.506	1.463	1.468	3.431	10.525	7.732	11.314
117	5.393	2.627	1.161	0.381	1.96	2.349	9.183	6.65	9.972
118	6.78	3.203	4.031	4.921	5.62	4.157	4.749	8.723	5.538
119	6.074	3.009	4.249	4.975	5.838	3.963	7.578	6.149	8.366
120	5.844	5.773	7.013	7.739	8.602	7.57	6.962	5.919	7.75
121	10.158	10.088	11.328	12.054	12.917	11.884	13.818	10.233	14.606
122	7.471	3.894	4.587	5.477	6.176	4.848	3.854	9.414	4.643
123	6.049	1.662	1.53	2.42	3.119	2.447	6.773	8.086	7.562
124	4.829	2.063	1.37	1.067	2.926	1.785	8.135	6.086	8.924
125	7.3	3.435	4.675	5.401	6.264	4.389	5.678	7.375	6.466
126	7.78	4.203	4.136	5.026	5.725	4.473	6.747	9.723	7.536
127	4.983	1.394	0.476	1.366	2.065	2.662	8.046	6.24	8.835
128	5.577	2.811	0.603	0.496	1.561	2.533	8.625	6.834	9.414
129	2.819	2.038	3.302	3.084	4.943	1.311	9.046	4.245	9.834
disposal	6.307	5.548	6.788	7.318	8.377	7.344	9.278	5.693	10.066

NODES	9	10	11	12	13	14	15	16	17
depot	6.994	7.027	7.948	4.832	6.044	7.865	7.197	6.47	7.774
1	2.418	2.721	3.282	1.026	3.429	3.431	2.122	2.486	3.363
2	0.775	1.079	1.64	2.29	4.608	4.551	2.571	3.726	1.72
3	1.057	1.327	1.491	2.35	5.334	5.277	3.461	4.256	2.17
4	0.888	0.632	1.898	4.209	6.197	6.14	4.16	5.315	0.173
5	3.094	3.295	4.216	1.338	4.108	4.265	2.908	3.226	4.042
6	9.042	9.345	9.401	8.725	7.292	5.323	5.807	5.592	9.987
7	7.395	7.596	8.517	5.401	5.43	7.251	8.219	7.137	8.343

Table B.1 continued from previous page

8	9.582	9.886	10.189	9.053	8.726	6.11	6.615	6.38	10.528
9	0	0.378	1.083	3.297	5.383	5.488	3.346	4.603	1.019
10	0.378	0	1.387	3.593	5.686	5.791	3.328	4.804	0.763
11	1.083	1.387	0	2.549	6.477	6.479	3.093	5.594	2.029
12	3.297	3.593	2.549	0	3.738	3.74	3.281	2.856	4.34
13	5.383	5.686	6.477	3.738	0	1.969	3.264	1.854	6.431
14	5.488	5.791	6.479	3.74	1.969	0	2.88	1.797	6.271
15	3.346	3.328	3.093	3.281	3.264	2.88	0	2.381	4.291
16	4.603	4.804	5.594	2.856	1.854	1.797	2.381	0	5.548
17	1.019	0.763	2.029	4.34	6.431	6.271	4.291	5.548	0
18	0.669	0.413	1.679	3.989	6.081	5.921	3.941	5.198	0.35
19	1.59	1.301	2.6	3.513	6.204	6.44	4.862	5.322	1.383
20	3.858	4.116	4.849	2.349	2.169	2.171	1.636	1.139	4.758
21	0.939	1.243	1.048	3.422	5.234	5.236	3.255	4.41	1.899
22	0.959	0.701	1.968	4.255	6.37	6.21	4.23	5.487	0.59
23	3.52	2.373	3.605	3.655	5.068	4.542	2.002	5.044	3.015
24	1.54	2.051	2.662	2.263	4.934	5.19	3.945	4.052	2.798
25	6.781	7.084	7.772	5.272	4.299	2.329	4.559	2.599	7.585
26	4.344	4.647	4.562	2.836	2.56	2.451	1.757	0.924	5.148
27	2.313	2.616	3.177	0.449	3.289	3.377	2.02	2.407	3.891
28	2.807	3.008	3.269	0.813	3.583	3.74	2.383	2.701	3.855
29	3.27	3.573	4.134	1.665	2.823	3.719	3.503	2.103	3.77
30	2.646	3.022	3.51	1.04	2.717	2.66	2.813	1.997	3.664
31	5.421	4.337	4.102	5.104	6.13	5.997	3.407	5.171	6.366
32	2.3	2.501	3.422	1.316	4.086	4.243	2.886	3.418	3.248
33	4.88	5.874	5.894	4.467	4.206	4.082	2.858	3.478	5.825
34	5.249	5.801	5.318	4.328	4.043	3.943	2.282	3.315	6.443
35	1.482	1.683	2.604	2.205	4.672	5.132	3.887	3.79	2.43
36	4.012	4.213	5.134	2.018	4.001	4.018	4.285	3.281	5.223
37	7.872	8.175	8.965	6.227	2.637	4.458	5.752	4.343	8.919

Table B.1 continued from previous page

38	3.176	3.48	3.245	3.471	3.456	3.07	0.152	2.573	4.122
39	2.736	3.04	2.805	3.513	3.548	3.112	0.485	2.287	3.702
40	3.325	3.526	4.447	1.569	4.338	4.496	3.139	3.456	4.311
41	4.319	4.622	5.31	2.342	1.408	1.351	2.097	0.688	13.169
42	12.223	12.527	12.292	11.694	10.721	8.751	9.256	9.021	13.4
43	4.795	5.098	5.845	4.382	4.121	3.997	2.227	3.171	6.431
44	2.803	3.106	3.667	1.197	2.542	2.503	2.49	1.84	3.821
45	1.753	2.056	2.617	1.823	3.737	3.577	1.659	2.752	2.698
46	2.056	2.359	2.84	2.191	3.85	3.852	1.29	2.967	3.001
47	3.594	3.898	3.663	5.061	6.712	6.494	3.904	5.668	5.847
48	4.254	4.281	4.046	3.901	3.625	3.516	1.019	2.897	5.199
49	2.844	3.045	3.966	2.485	4.848	5.412	4.055	4.587	3.616
50	2.946	3.147	4.068	2.04	4.733	4.967	3.61	4.142	3.972
51	5.867	6.171	7.614	8.216	7.964	7.831	5.241	7.005	8.2
52	1.658	1.611	1.749	2.05	4.814	4.977	3.276	4.152	2.358
53	2.132	2.435	2.996	1.565	3.078	3.318	1.387	2.493	3.114
54	4.224	4.527	5.274	3.811	3.55	3.551	1.656	2.6	5.169
55	1.705	2.009	1.774	3.187	5.659	5.448	3.468	4.623	2.651
56	2.076	2.379	2.94	2.625	4.416	4.494	1.954	3.669	3.021
57	2.868	3.172	2.937	2.73	3.54	3.39	0.85	2.565	3.268
58	4.795	4.995	5.786	3.047	0.839	1.278	2.573	1.163	5.637
59	1.23	1.534	1.299	4.354	6.185	6.128	1.8	5.303	2.176
60	7.634	8.098	8.235	6.893	6.851	8.64	7.651	7.001	8.845
61	2.158	2.461	3.022	1.559	3.919	3.862	2.004	3.037	3.103
62	2.593	2.794	3.35	1.611	3.976	3.919	1.989	3.094	3.541
63	2.567	2.87	3.431	0.703	3.688	3.631	2.274	2.806	3.512
64	2.118	3.462	2.982	0.201	3.812	3.755	1.825	2.725	3.063
65	10.444	10.748	10.513	9.915	9.588	6.972	7.477	7.242	11.39
66	2.981	3.284	3.845	1.63	2.321	2.264	2.281	1.439	3.926
67	7.907	8.211	7.976	6.951	6.624	4.008	4.94	4.278	8.853

Table B.1 continued from previous page

68	8.63	8.934	8.699	6.621	6.294	3.678	5.663	3.948	9.576
69	0.473	0.095	1.482	3.488	5.781	5.724	3.744	4.899	0.858
70	3.51	4.154	4.374	1.905	3.846	3.959	3.743	3.134	4.901
71	3.948	4.149	5.07	2.03	3.398	3.359	3.673	2.697	4.896
72	3.728	3.929	4.85	1.734	3.748	3.709	4.023	3.047	4.676
73	3.926	4.23	3.995	5.393	7.191	7.134	4.544	6.308	4.872
74	5.624	5.928	5.693	8.203	7.875	7.818	5.228	6.992	6.57
75	2.483	2.786	3.347	2.352	4.986	4.929	2.999	4.104	3.428
76	2.073	2.274	2.829	1.942	4.468	4.411	2.481	3.586	3.021
77	3.46	3.661	4.324	2.283	5.267	5.21	3.853	4.385	4.408
78	2.511	2.814	3.375	1.461	3.826	3.769	1.839	2.944	3.456
79	1.645	1.949	1.714	4.77	6.6	6.543	4.563	5.718	2.591
80	3.334	3.535	4.198	2.157	5.141	5.084	3.727	4.259	4.282
81	4.715	4.916	5.837	2.721	5.136	5.097	4.291	4.435	5.663
82	3.944	4.247	4.396	2.436	2.108	2.051	1.36	1.225	4.889
83	4.35	4.654	4.419	5.817	7.307	7.25	4.66	6.424	5.296
84	0.439	0.475	0.477	3.735	5.747	5.69	3.71	4.865	1.117
85	0.344	0.648	1.209	2.878	5.196	5.139	3.159	4.314	1.289
86	3.774	4.418	4.638	2.169	4.11	4.223	4.007	3.398	5.165
87	1.254	1.567	2.118	0.784	4.272	4.274	2.29	3.389	2.209
88	5.182	5.485	5.541	4.865	6.73	6.731	3.965	6.002	6.127
89	3.953	4.256	4.312	3.636	5.763	5.764	2.151	5.035	4.898
90	2.634	2.458	2.514	0.885	4.273	4.275	2.459	3.391	3.1
91	1.936	2.239	2.295	0.666	4.153	4.155	2.24	3.27	2.881
92	2.665	2.968	3.024	2.63	3.098	3.1	2.661	2.216	3.61
93	4.109	4.412	4.468	3.792	1.824	1.826	1.663	0.942	5.054
94	2.575	2.871	3.008	2.481	5.754	5.756	4.055	4.872	3.618
95	3.589	3.885	4.022	3.397	6.017	6.019	5.56	5.135	4.632
96	4.199	4.495	4.632	3.053	5.052	5.054	4.982	4.332	5.242
97	4.75	5.053	5.109	4.873	3.107	2.82	2.304	0.709	5.695

Table B.1 continued from previous page

98	5.667	5.97	6.026	5.35	7.477	7.478	3.865	6.749	6.612
99	3.844	4.14	4.277	2.974	4.596	4.598	4.318	3.893	4.887
100	4.728	4.929	5.85	2.734	5.149	5.11	4.304	4.448	5.676
101	2.738	2.939	3.86	3.461	6.445	6.388	5.143	5.563	3.686
102	1.902	2.103	3.024	2.625	5.609	5.552	4.307	4.727	2.85
103	2.395	2.691	2.828	2.267	5.574	5.576	3.841	4.692	3.438
104	3.249	3.545	3.682	3.121	6.428	6.43	4.695	5.546	4.292
105	3.453	3.756	3.812	3.418	2.131	2.133	2.494	1.411	4.398
106	2.17	2.473	2.529	2.358	3.381	3.383	1.972	2.498	3.115
107	3.119	3.422	3.478	3.084	2.843	2.845	2.773	2.123	4.064
108	3.47	3.773	3.829	3.435	3.194	3.196	3.124	2.474	4.415
109	3.592	3.895	3.57	3.176	2.935	2.937	2.778	2.215	4.156
110	3.842	4.145	4.201	3.525	5.424	5.426	2.04	4.541	4.787
111	6.64	6.943	6.999	6.763	4.299	2.33	4.194	2.599	7.585
112	0.848	0.592	0.941	2.459	6.26	6.262	3.799	5.377	0.641
113	4.78	5.076	5.213	3.871	4.915	4.701	4.629	3.979	5.823
114	8.921	9.217	9.354	8.012	7.97	9.759	8.77	8.12	9.964
115	4.978	5.281	5.337	5.101	2.37	2.372	2.532	0.517	5.923
116	1.805	1.518	1.898	3.114	6.421	6.423	4.756	5.539	1.599
117	1.328	1.344	1.761	2.032	5.339	5.341	3.414	4.457	2.091
118	4.806	5.109	5.165	4.489	3.441	3.442	1.571	1.604	5.751
119	5.024	5.327	5.383	4.707	0.867	1.808	2.578	1.495	5.969
120	7.788	8.091	8.147	7.471	2.553	3.671	5.342	4.259	8.733
121	12.103	12.406	12.462	11.786	6.868	8.656	9.657	8.574	13.048
122	5.362	5.665	5.721	5.045	4.132	4.133	2.127	3.404	6.307
123	2.305	2.608	2.664	1.988	2.804	2.806	1.087	1.921	3.25
124	2.014	2.31	2.504	0.875	4.362	4.364	2.449	3.479	3.057
125	5.45	5.753	5.809	5.133	2.093	0.356	3.004	1.921	6.395
126	4.911	5.214	5.27	4.594	4.441	4.442	1.676	3.713	5.856
127	1.251	1.554	1.61	0.66	4.148	4.15	2.277	3.265	2.196

Table B.1 continued from previous page

128	0.674	1.051	1.107	1.513	5.314	5.316	2.856	4.431	1.692
129	4.031	4.327	4.464	3.161	4.137	4.139	4.067	3.417	5.074
disposal	7.563	7.866	8.656	7.246	2.328	4.116	5.117	4.034	8.508

NODES	18	19	20	21	22	23	24	25	26
depot	7.423	6.946	6.197	7.048	7.784	7.172	5.676	9.024	6.587
1	3.013	3.492	1.932	2.129	3.302	2.895	2.222	4.759	2.322
2	1.37	2.29	3.038	0.486	1.66	2.416	2.122	5.865	3.428
3	1.723	1.246	3.898	1.136	2.016	2.955	0.808	6.672	4.288
4	0.219	1.252	4.627	1.619	0.459	2.884	2.667	7.454	5.017
5	3.691	3.214	2.886	3.074	4.052	3.678	1.944	5.713	3.276
6	9.389	10.558	6.182	8.765	9.926	8.026	9.641	2.993	5.43
7	7.992	7.778	7.452	7.665	8.353	10.35	6.245	9.88	7.842
8	10.178	11.346	7.139	9.307	10.467	8.867	9.676	3.781	6.218
9	0.669	1.59	3.858	0.939	0.959	3.52	1.54	6.781	4.344
10	0.413	1.301	4.116	1.243	0.701	2.373	2.051	7.084	4.647
11	1.679	2.6	4.849	1.048	1.968	3.605	2.662	7.772	4.562
12	3.989	3.513	2.349	3.422	4.255	3.655	2.263	5.272	2.836
13	6.081	6.204	2.169	5.234	6.37	5.068	4.934	4.299	2.56
14	5.921	6.44	2.171	5.236	6.21	4.542	5.19	2.329	2.451
15	3.941	4.862	1.636	3.255	4.23	2.002	3.945	4.559	1.757
16	5.198	5.322	1.139	4.41	5.487	5.044	4.052	2.599	0.924
17	0.35	1.383	4.758	1.899	0.59	3.015	2.798	7.585	5.148
18	0	1.032	4.408	1.549	0.528	4.115	2.137	7.235	4.798
19	1.032	0	5.061	2.39	1.561	3.586	1.971	7.888	5.451
20	4.408	5.061	0	3.833	4.697	3.965	3.545	3.189	0.753
21	1.549	2.39	3.833	0	1.714	1.843	1.809	6.438	4.001
22	0.528	1.561	4.697	1.714	0	4.404	2.498	7.665	5.228
23	4.115	3.586	3.965	1.843	4.404	0	4.415	5.856	3.419
24	2.137	1.971	3.545	1.809	2.498	4.415	0	6.468	4.032

Table B.1 continued from previous page

25	7.235	7.888	3.189	6.438	7.665	5.856	6.468	0	2.437
26	4.798	5.451	0.753	4.001	5.228	3.419	4.032	2.437	0
27	3.54	3.064	1.998	2.902	3.197	2.79	1.971	4.823	2.388
28	3.505	3.651	2.194	2.708	3.794	3.153	1.657	5.117	2.751
29	3.42	3.915	1.816	2.623	3.709	4.415	2.861	4.739	2.893
30	3.314	3.809	1.71	2.517	3.53	4.309	2.236	4.633	2.203
31	6.016	6.937	4.323	5.219	4.918	2.344	5.616	6.891	4.454
32	2.975	2.499	2.697	2.337	3.258	4.352	1.15	5.62	3.254
33	6.166	7.087	2.399	5.369	5.764	5.058	5.663	5.933	2.539
34	6.093	7.014	2.236	5.296	6.134	4.534	5.343	5.357	2.4
35	2.079	1.709	3.283	1.547	2.44	4.187	0.332	6.206	4.143
36	4.872	4.132	2.994	3.801	4.97	5.593	2.862	5.917	3.675
37	8.467	8.986	4.511	7.67	8.858	7.088	7.423	6.595	5.048
38	4.133	5.054	1.828	2.95	4.422	2.194	3.391	4.196	1.758
39	3.332	4.273	1.92	2.555	3.621	1.413	3.635	4.426	1.989
40	3.96	3.484	2.949	3.322	4.283	4.255	2.175	5.872	3.507
41	4.812	5.331	1.003	4.015	5.203	3.433	4.081	3.83	1.393
42	13.066	13.739	9.611	11.997	13.355	11.455	12.317	6.422	8.859
43	6.081	7.002	2.314	5.284	6.37	4.47	5.578	4.906	2.454
44	3.471	3.966	1.722	2.601	3.687	4.152	2.393	4.498	2.112
45	2.348	3.268	2.109	1.551	2.637	1.8	2.648	5.032	2.454
46	2.651	3.572	2.317	1.854	2.94	1.431	2.655	5.144	2.707
47	5.497	6.418	4.905	4.7	4.479	2.264	5.177	7.497	4.951
48	4.849	5.77	1.818	4.052	4.862	3.262	4.853	4.41	1.973
49	3.265	2.789	4.033	2.627	3.802	5.169	1.539	6.382	4.423
50	3.621	3.066	3.344	3.001	3.904	5.076	1.796	6.267	3.978
51	7.85	8.771	6.301	7.053	6.752	4.178	7.854	8.725	6.288
52	2.007	1.531	3.598	1.713	2.616	3.556	0.27	6.348	3.988
53	2.764	3.465	1.45	1.967	3.053	3.331	2.215	4.373	2.195
54	5.51	6.431	1.743	4.713	5.108	3.899	5.007	4.335	1.883

Table B.1 continued from previous page

55	2.301	3.222	4.031	1.479	2.59	2.787	2.585	6.954	4.325
56	2.671	3.591	2.788	1.874	2.96	2.128	2.675	5.711	3.371
57	3.464	3.839	1.877	2.642	3.753	1.978	2.922	4.835	2.267
58	5.287	5.806	1.331	4.593	5.679	4.377	4.243	3.608	1.868
59	1.826	2.746	3.556	1.004	2.115	2.312	2.812	6.479	5.005
60	8.494	8.018	6.714	7.856	8.76	8.327	6.768	9.637	7.201
61	2.673	3.673	2.349	1.876	3.042	2.085	3.053	5.176	2.739
62	3.19	2.713	2.406	2.197	3.37	2.413	1.443	5.233	2.796
63	3.162	3.169	2.036	2.278	3.451	3.044	2.276	5.079	2.523
64	2.713	3.33	2.242	1.941	3.002	2.595	2.06	5.069	2.705
65	11.04	11.96	8.001	10.194	11.576	9.729	10.538	4.643	7.08
66	3.576	4.096	1.281	2.704	3.865	4.085	2.826	4.108	1.671
67	8.503	9.423	5.037	7.756	8.792	7.192	8.001	1.679	4.116
68	9.226	9.087	4.707	8.355	9.515	7.915	7.817	1.349	3.786
69	0.508	1.196	4.211	1.179	0.796	2.468	1.946	7.038	4.601
70	4.55	4.073	2.743	3.251	4.394	5.547	2.803	5.57	3.133
71	4.545	4.068	2.578	3.946	4.906	5.477	2.798	5.405	2.968
72	4.325	3.848	3.001	3.854	4.686	4.074	2.578	5.828	3.391
73	4.522	5.442	5.604	3.629	4.811	2.586	5.509	8.028	5.591
74	6.22	7.14	6.288	5.448	6.509	4.165	7.207	8.712	6.275
75	3.078	3.646	3.416	2.231	3.367	3.423	2.376	6.727	4.29
76	2.67	2.193	2.898	1.874	2.849	2.905	0.923	5.725	3.288
77	4.057	3.58	3.831	3.369	4.344	4.4	2.31	6.658	4.221
78	3.106	4.026	2.256	2.267	3.395	2.34	2.657	5.083	2.646
79	2.241	3.161	5.03	1.461	2.53	0.516	3.228	7.857	5.42
80	3.931	3.454	3.705	3.309	4.218	4.274	2.184	6.532	4.095
81	5.312	4.835	4.162	4.946	5.673	5.061	3.565	6.989	4.552
82	4.539	4.902	0.521	3.7	4.828	3.612	3.632	2.945	0.508
83	4.946	5.866	5.72	4.166	5.235	3.01	5.933	8.144	5.707
84	0.767	1.687	4.177	1.224	1.056	2.434	2.193	7.004	4.567

Table B.1 continued from previous page

85	0.939	1.859	3.626	0.544	1.229	2.196	1.824	6.453	4.016
86	4.814	4.337	3.007	3.537	4.658	5.811	3.067	5.834	3.397
87	1.859	2.78	2.644	1.017	2.148	2.782	1.863	5.892	3.13
88	5.777	6.698	4.923	3.97	6.066	2.544	5.781	7.515	5.078
89	4.548	5.469	3.956	3.957	4.837	1.315	4.552	6.548	4.111
90	2.75	2.819	2.884	1.925	3.039	3.673	1.569	5.807	3.371
91	2.531	3.452	2.525	1.706	2.82	3.454	2.535	5.448	3.011
92	3.26	3.755	1.729	2.463	3.549	3.255	2.505	4.652	2.216
93	4.704	5.223	0.748	3.907	4.993	3.325	3.973	3.671	1.234
94	3.267	2.791	4.365	2.629	3.533	5.269	1.299	7.288	4.852
95	4.281	3.805	4.628	3.643	4.547	6.185	2.555	7.551	5.115
96	4.891	4.415	4.045	4.253	5.157	5.582	3.165	6.968	4.532
97	5.345	5.998	1.299	4.548	5.634	3.966	4.748	2.146	0.547
98	6.262	7.183	5.67	5.465	6.551	3.029	6.266	8.262	5.825
99	4.536	4.06	3.386	3.898	4.802	4.692	2.81	6.309	3.873
100	5.325	4.848	4.175	4.783	5.686	5.074	3.578	7.002	4.565
101	3.335	2.858	5.009	2.793	3.696	4.636	1.284	7.836	5.399
102	2.499	2.022	4.173	1.957	2.86	3.8	0.448	7	4.563
103	3.087	2.611	4.451	2.235	3.138	4.078	0.726	7.278	4.841
104	3.941	3.465	5.303	3.087	3.99	4.93	1.578	8.13	5.693
105	4.048	4.543	1.726	3.178	4.264	4.624	2.97	4.553	2.116
106	2.765	3.483	1.842	2.178	3.264	2.857	2.131	4.669	2.232
107	3.714	4.209	1.858	2.917	4.003	4.435	2.959	4.781	2.344
108	4.065	4.56	2.209	3.268	4.354	4.786	3.31	5.132	2.695
109	3.806	4.301	1.95	3.009	4.095	4.527	3.051	4.873	2.436
110	4.437	5.358	3.796	3.64	4.726	1.204	4.441	7.191	4.754
111	7.235	7.888	3.189	6.438	7.524	5.856	6.638	0	2.437
112	0.29	0.742	4.632	1.604	0.819	4.294	1.847	7.555	5.118
113	5.472	4.996	3.692	4.834	5.738	5.305	3.746	6.615	4.179
114	9.613	9.137	7.833	8.975	9.879	9.446	7.887	10.756	8.32

Table B.1 continued from previous page

115	5.573	6.226	1.527	4.776	5.862	4.194	4.976	2.45	0.775
116	1.248	0.217	5.032	2.561	1.777	5.251	1.877	7.955	5.519
117	1.74	1.264	3.95	1.382	2.269	3.906	0.795	6.873	4.437
118	5.401	6.322	1.634	4.604	5.69	3.79	5.082	3.117	0.764
119	5.619	6.138	1.662	4.822	5.908	4.24	4.888	4.585	2.148
120	8.383	8.902	4.427	7.586	8.672	7.004	7.652	3.97	5.989
121	12.698	13.217	8.742	11.901	12.987	11.319	11.967	10.826	9.228
122	5.957	6.878	2.325	5.16	6.246	4.346	5.961	5.041	2.48
123	2.9	3.821	1.176	2.103	3.189	2.277	2.904	4.099	1.662
124	2.706	2.23	2.734	1.943	3.029	3.663	0.98	5.657	3.22
125	6.045	6.564	2.089	5.248	6.334	4.666	5.314	2.685	2.575
126	5.506	6.427	2.634	4.709	5.795	1.331	5.51	5.226	2.789
127	1.846	2.767	2.52	1.049	2.135	2.769	1.85	5.443	3.006
128	1.342	1.728	3.686	0.825	1.632	3.348	0.979	6.609	4.172
129	4.723	4.247	3.13	4.085	4.989	4.667	2.997	6.053	3.617
disposal	8.158	8.384	4.201	7.361	8.447	6.779	7.427	6.286	4.687

NODES	27	28	29	30	31	32	33	34	35
depot	4.383	4.019	4.852	4.746	10.157	4.843	8.233	8.07	5.414
1	0.577	0.888	0.786	0.68	5.892	1.374	3.968	3.805	1.96
2	1.841	2.135	2.05	1.944	4.38	2.059	4.796	4.723	1.894
3	1.901	1.861	2.752	2.646	4.919	1.336	5.686	5.613	0.546
4	3.76	3.724	3.639	3.533	6.235	3.195	6.385	6.312	2.405
5	0.889	0.848	1.74	1.634	6.162	0.754	4.922	4.759	1.682
6	7.818	8.181	8.323	7.633	8.383	8.684	4.746	4.17	9.583
7	4.952	4.588	4.657	5.314	11.335	5.411	9.488	9.325	5.983
8	8.604	8.898	8.52	8.421	9.225	9.472	5.535	4.959	9.448
9	2.313	2.807	3.27	2.646	5.421	2.3	4.88	5.249	1.482
10	2.616	3.008	3.573	3.022	4.337	2.501	5.874	5.801	1.683
11	3.177	3.269	4.134	3.51	4.102	3.422	5.894	5.318	2.604

Table B.1 continued from previous page

12	0.449	0.813	1.665	1.04	5.104	1.316	4.467	4.328	2.205
13	3.289	3.583	2.823	2.717	6.13	4.086	4.206	4.043	4.672
14	3.377	3.74	3.719	2.66	5.997	4.243	4.082	3.943	5.132
15	2.02	2.383	3.503	2.813	3.407	2.886	2.858	2.282	3.887
16	2.407	2.701	2.103	1.997	5.171	3.418	3.478	3.315	3.79
17	3.891	3.855	3.77	3.664	6.366	3.248	5.825	6.443	2.43
18	3.54	3.505	3.42	3.314	6.016	2.975	6.166	6.093	2.079
19	3.064	3.651	3.915	3.809	6.937	2.499	7.087	7.014	1.709
20	1.998	2.194	1.816	1.71	4.323	2.697	2.399	2.236	3.283
21	2.902	2.708	2.623	2.517	5.219	2.337	5.369	5.296	1.547
22	3.197	3.794	3.709	3.53	4.918	3.258	5.764	6.134	2.44
23	2.79	3.153	4.415	4.309	2.344	4.352	5.058	4.534	4.187
24	1.971	1.657	2.861	2.236	5.616	1.15	5.663	5.343	0.332
25	4.823	5.117	4.739	4.633	6.891	5.62	5.933	5.357	6.206
26	2.388	2.751	2.893	2.203	4.454	3.254	2.539	2.4	4.143
27	0	0.364	1.216	0.591	5.933	0.867	4.018	3.879	1.756
28	0.364	0	1.462	0.885	6.227	1.284	4.312	4.173	2.343
29	1.216	1.462	0	0.106	5.849	1.718	3.934	3.795	2.607
30	0.591	0.885	0.106	0	5.743	1.612	3.828	3.689	2.501
31	5.933	6.227	5.849	5.743	0	6.14	5.468	4.892	5.962
32	0.867	1.284	1.718	1.612	6.14	0	4.815	4.676	1.191
33	4.018	4.312	3.934	3.828	5.468	4.815	0	0.576	5.789
34	3.879	4.173	3.795	3.689	4.892	4.676	0.576	0	5.626
35	1.756	2.343	2.607	2.501	5.962	1.191	5.789	5.626	0
36	1.569	1.205	1.738	1.632	7.245	2.029	5.321	5.158	2.6
37	5.778	6.072	5.312	5.206	8.618	6.575	6.694	6.531	7.161
38	3.022	3.316	2.943	2.837	3.245	3.819	2.216	2.143	3.163
39	3.064	3.358	2.985	2.879	3.772	3.861	2.743	2.67	3.407
40	1.12	0.913	1.971	1.865	6.393	0.985	5.153	4.99	1.913
41	1.893	2.187	1.408	1.302	4.963	2.69	3.039	2.876	3.276

Table B.1 continued from previous page

42	11.245	11.539	11.161	11.055	11.866	12.042	8.176	7.6	12.089
43	3.933	4.227	3.849	3.743	4.837	4.73	1.241	1.281	5.316
44	0.748	1.042	0.263	0.157	5.682	1.545	3.758	3.595	2.131
45	1.374	1.668	1.583	1.477	3.735	2.585	3.884	3.811	2.42
46	1.742	2.036	1.951	1.845	3.366	2.888	3.515	3.442	2.723
47	4.612	4.906	6.346	6.24	2.345	5.705	5.912	5.336	4.915
48	3.452	3.746	3.368	3.262	3.62	4.249	1.394	1.231	3.843
49	2.036	2.079	2.887	2.781	7.309	1.329	6.069	5.906	1.432
50	1.591	1.473	2.442	2.336	6.864	0.723	5.624	5.461	1.534
51	7.767	8.061	7.683	7.577	1.977	8.564	7.249	6.673	7.188
52	1.601	2.188	2.452	2.346	5.635	1.036	5.634	5.471	0.246
53	1.116	1.41	1.621	1.515	4.641	1.913	3.612	3.539	1.846
54	3.362	3.656	3.278	3.172	4.266	4.159	0.656	0.709	4.745
55	2.738	3.032	2.947	2.841	5.543	2.956	5.693	5.62	2.791
56	2.176	2.47	2.385	2.279	4.029	2.908	4.179	4.106	2.743
57	2.281	2.575	3.263	3.157	3.463	3.078	3.075	3.002	3.011
58	2.598	2.892	2.132	2.026	5.438	3.395	3.514	3.351	3.981
59	3.905	3.712	3.627	3.521	6.223	3.34	6.373	6.3	2.55
60	5.447	4.827	4.897	5.554	10.747	5.651	8.832	8.693	6.71
61	0.804	1.404	1.319	1.213	4.02	1.907	4.169	4.096	2.825
62	1.162	1.456	1.371	1.265	4.348	1.38	4.214	4.705	1.215
63	0.254	0.548	1.106	0.727	5.528	1.051	4.288	4.125	2.218
64	0.318	0.682	1.534	0.909	5.079	1.185	4.336	4.197	2.074
65	9.466	9.831	9.382	9.283	10.033	10.334	6.397	5.821	10.31
66	1.181	1.475	1.286	1.18	5.215	1.978	3.317	3.154	2.564
67	6.502	6.796	6.418	6.312	7.55	7.299	3.861	3.285	7.773
68	6.172	6.466	6.088	5.982	8.273	6.969	4.583	4.007	7.555
69	2.711	3.626	3.223	3.117	5.819	2.474	5.969	5.896	1.684
70	1.456	1.146	1.423	1.317	6.703	1.97	4.779	4.43	2.541
71	1.581	1.141	1.079	0.973	6.538	1.965	4.614	4.451	2.536

Table B.1 continued from previous page

72	1.285	0.921	1.429	1.323	6.961	1.745	5.037	4.874	2.316
73	4.944	5.238	6.986	6.88	2.985	6.037	6.552	5.976	5.247
74	7.754	8.048	7.67	7.564	3.669	8.551	7.236	6.66	6.945
75	1.903	2.49	2.754	2.648	5.358	1.528	5.224	5.151	2.114
76	1.493	2.08	2.344	2.238	4.84	1.118	4.706	4.633	0.695
77	1.834	2.421	2.685	2.579	6.335	1.459	5.867	5.704	2.082
78	1.012	1.306	1.221	1.115	4.275	1.809	4.064	3.991	2.395
79	2.577	4.127	4.042	3.936	6.638	3.756	6.788	6.715	2.966
80	1.708	2.295	2.559	2.453	6.209	1.333	5.741	5.578	1.956
81	2.272	1.908	2.817	2.711	8.122	2.346	6.198	6.035	3.303
82	1.987	2.281	1.903	1.797	3.97	2.784	2.045	1.882	3.37
83	5.368	5.662	7.102	6.996	3.101	6.461	6.668	6.092	5.671
84	3.286	3.274	3.189	3.083	5.785	2.721	5.935	5.862	1.931
85	2.429	2.723	2.638	2.532	5.234	2.352	5.384	5.311	1.562
86	1.72	1.41	1.687	1.581	6.967	2.234	5.043	4.88	2.805
87	1.742	1.776	2.37	1.746	4.673	2.182	4.142	4.515	1.805
88	4.486	4.849	7.063	6.373	2.448	5.352	6.068	5.492	5.723
89	3.257	3.62	6.096	5.406	1.481	4.123	5.101	4.525	4.494
90	1.31	0.996	2.2	1.575	5.518	0.679	5.002	4.863	1.511
91	1.294	1.657	2.251	1.627	5.299	2.16	4.07	4.382	2.477
92	0.692	1.055	0.682	0.399	5.762	1.558	3.847	3.708	2.447
93	2.16	2.523	2.502	1.66	4.78	3.026	2.865	2.726	3.915
94	2.791	2.477	3.681	3.056	6.436	1.97	6.483	6.344	1.082
95	3.054	2.74	4.435	3.319	7.45	2.091	6.746	6.607	2.497
96	2.702	2.082	2.228	2.335	8.078	1.488	6.163	6.024	3.107
97	2.935	3.298	3.44	2.75	5.001	3.801	3.086	2.947	4.69
98	4.971	5.334	7.81	7.12	3.195	5.837	6.815	6.239	6.208
99	1.812	1.192	1.772	1.879	7.419	1.56	5.504	5.365	2.752
100	2.313	1.693	1.768	1.875	7.618	2.06	5.703	5.564	3.252
101	3.012	3.599	3.863	3.757	7.218	2.447	7.045	6.882	1.765

Table B.1 continued from previous page

102	2.176	2.763	3.027	2.921	6.382	1.611	6.209	6.046	0.896
103	2.454	3.041	3.305	3.199	6.66	1.889	6.487	6.324	1.207
104	3.306	3.893	4.157	4.051	7.512	2.741	7.339	7.176	2.059
105	1.325	1.619	0.84	0.734	5.686	2.122	3.762	3.599	2.708
106	0.486	0.78	0.611	0.505	5.802	1.283	3.878	3.715	1.869
107	1.146	1.677	0.506	0.4	5.965	1.788	4.041	3.878	2.374
108	1.497	1.326	0.857	0.751	6.316	2.192	4.392	4.229	2.725
109	1.619	1.586	1.95	0.492	6.057	1.88	4.133	3.97	2.466
110	3.146	3.509	5.663	4.973	2.124	4.012	5.744	5.168	4.383
111	4.825	5.188	5.33	4.64	6.891	5.691	5.933	5.357	6.58
112	3.087	3.114	4.318	3.693	4.808	2.607	5.654	6.024	1.789
113	2.426	1.806	1.875	1.982	7.725	2.629	5.81	5.671	3.688
114	6.566	5.946	6.016	6.123	11.866	6.77	9.951	9.812	7.829
115	3.163	3.526	3.668	2.978	5.229	4.029	3.314	3.175	4.918
116	3.458	3.144	4.348	3.723	5.765	2.637	6.611	6.981	1.819
117	2.376	2.062	3.266	2.641	5.189	1.555	5.266	5.639	0.737
118	3.269	3.632	3.774	3.084	4.147	4.135	1.12	0.981	5.024
119	3.075	3.438	3.179	2.102	5.694	3.941	3.779	3.64	4.83
120	5.839	7.183	7.252	5.104	8.459	6.705	6.544	6.405	7.594
121	10.154	11.497	11.566	9.419	12.774	11.02	10.859	10.72	11.909
122	3.96	4.323	4.465	3.775	4.703	4.826	1.067	0.491	5.903
123	1.559	1.922	3.043	2.353	4.354	2.425	3.293	3.229	2.846
124	1.503	1.498	2.702	2.077	5.508	1.181	4.279	4.591	0.922
125	3.501	3.864	3.843	2.784	6.121	4.367	4.206	4.067	5.256
126	3.585	3.948	4.774	4.084	1.689	4.451	3.779	3.203	5.452
127	1.289	1.652	2.246	1.622	4.614	2.058	4.129	4.502	1.792
128	2.56	2.246	3.45	2.825	4.632	1.739	4.708	5.081	0.921
129	1.787	1.167	1.313	1.42	7.163	1.747	5.248	5.109	2.939
disposal	5.614	6.957	7.026	4.879	8.309	6.48	6.318	6.179	7.369

Table B.1 continued from previous page

NODES	36	37	38	39	40	41	42	43	44
depot	3.832	6.616	7.389	6.188	4.004	5.774	15.446	8.148	4.629
1	1.964	5.917	2.21	2.406	1.626	2.202	11.181	3.883	0.837
2	3.228	7.097	2.763	1.982	2.893	3.442	11.696	4.711	2.101
3	3.233	7.823	3.653	2.872	2.321	4.168	12.586	5.601	2.803
4	5.092	8.686	4.352	3.571	4.18	5.133	13.285	6.3	3.69
5	1.593	7.653	3.841	2.694	0.548	3.156	12.135	4.837	1.791
6	9.105	9.587	5.687	6.214	8.937	6.823	3.43	5.049	7.542
7	4.4	6.002	8.409	8.451	4.742	6.678	16.3	9.335	4.921
8	9.698	10.375	6.505	6.9	9.653	7.611	3.679	5.821	8.279
9	4.012	7.872	3.176	2.736	3.325	4.319	12.223	4.795	2.803
10	4.213	8.175	3.48	3.04	3.526	4.622	12.527	5.098	3.106
11	5.134	8.965	3.245	2.805	4.447	5.31	12.292	5.845	3.667
12	2.018	6.227	3.471	3.513	1.569	2.342	11.694	4.382	1.197
13	4.001	2.637	3.456	3.548	4.338	1.408	10.721	4.121	2.542
14	4.018	4.458	3.07	3.112	4.496	1.351	8.751	3.997	2.503
15	4.285	5.752	0.152	0.485	3.139	2.097	9.256	2.227	2.49
16	3.281	4.343	2.573	2.287	3.456	0.688	9.021	3.171	1.84
17	5.223	8.919	4.122	3.702	4.311	13.169	13.4	6.431	3.821
18	4.872	8.467	4.133	3.332	3.96	4.812	13.066	6.081	3.471
19	4.132	8.986	5.054	4.273	3.484	5.331	13.739	7.002	3.966
20	2.994	4.511	1.828	1.92	2.949	1.003	9.611	2.314	1.722
21	3.801	7.67	2.95	2.555	3.322	4.015	11.997	5.284	2.601
22	4.97	8.858	4.422	3.621	4.283	5.203	13.355	6.37	3.687
23	5.593	7.088	2.194	1.413	4.255	3.433	11.455	4.47	4.152
24	2.862	7.423	3.391	3.635	2.175	4.081	12.317	5.578	2.393
25	5.917	6.595	4.196	4.426	5.872	3.83	6.422	4.906	4.498
26	3.675	5.048	1.758	1.989	3.507	1.393	8.859	2.454	2.112
27	1.569	5.778	3.022	3.064	1.12	1.893	11.245	3.933	0.748
28	1.205	6.072	3.316	3.358	0.913	2.187	11.539	4.227	1.042

Table B.1 continued from previous page

29	1.738	5.312	2.943	2.985	1.971	1.408	11.161	3.849	0.263
30	1.632	5.206	2.837	2.879	1.865	1.302	11.055	3.743	0.157
31	7.245	8.618	3.245	3.772	6.393	4.963	11.866	4.837	5.682
32	2.029	6.575	3.819	3.861	0.985	2.69	12.042	4.73	1.545
33	5.321	6.694	2.216	2.743	5.153	3.039	8.176	1.241	3.758
34	5.158	6.531	2.143	2.67	4.99	2.876	7.6	1.281	3.595
35	2.6	7.161	3.163	3.407	1.913	3.276	12.089	5.316	2.131
36	0	7.077	4.121	4.163	1.553	2.586	12.339	5.027	1.441
37	7.077	0	5.616	5.658	7.42	3.886	13.016	6.543	5.031
38	4.121	5.616	0	0.637	3.331	2.289	9.146	2.117	3.009
39	4.163	5.658	0.637	0	2.925	2.381	9.541	2.512	3.101
40	1.553	7.42	3.331	2.925	0	2.942	12.294	4.982	1.797
41	2.586	3.886	2.289	2.381	2.942	0	10.2	2.887	1.145
42	12.339	13.016	9.146	9.541	12.294	10.2	0	8.479	10.971
43	5.027	6.543	2.117	2.512	4.982	2.887	8.479	0	3.673
44	1.441	5.031	3.009	3.101	1.797	1.145	10.971	3.673	0
45	2.761	6.123	1.851	1.07	2.423	2.468	10.784	3.799	1.634
46	3.129	6.376	1.482	0.701	2.791	2.721	10.415	3.43	2.002
47	7.524	9.04	3.794	2.819	5.661	5.384	12.309	5.324	6.104
48	4.546	6.062	0.9	1.295	4.501	2.406	8.429	1.309	3.126
49	2.824	9.674	4.247	3.841	1.779	4.303	13.282	5.984	2.938
50	2.218	8.278	3.802	3.396	1.173	3.858	12.837	5.539	2.493
51	8.861	10.377	5.131	5.092	8.816	6.721	13.646	6.661	7.441
52	2.933	7.523	3.468	2.97	2.021	3.868	12.847	5.549	2.503
53	2.799	5.864	1.579	1.671	2.165	2.209	10.512	3.527	1.672
54	4.456	5.972	1.546	1.941	4.411	2.316	7.907	0.584	3.036
55	4.125	7.994	3.66	2.879	3.79	4.339	12.593	5.608	2.998
56	3.563	7.04	2.146	1.365	3.742	3.385	11.079	4.094	2.436
57	4.441	5.936	1.042	0.431	3.33	2.281	9.975	2.99	3
58	3.31	3.328	2.765	2.857	3.647	0.717	10.03	3.429	1.869

Table B.1 continued from previous page

59	4.805	8.674	4.34	3.559	4.325	5.019	13.273	6.288	3.678
60	4.64	7.423	8.197	7.883	4.812	6.306	16.059	8.747	5.437
61	2.497	6.408	2.136	1.355	2.159	2.753	11.069	4.084	1.37
62	2.549	6.465	2.181	1.683	2.214	2.81	11.114	4.129	1.422
63	1.753	6.177	2.466	1.303	2.522	2.522	11.501	4.203	1.157
64	1.887	6.096	3.34	3.382	1.438	2.211	10.95	4.251	1.066
65	10.56	11.237	7.367	7.762	10.515	8.421	3.967	6.683	9.141
66	2.464	4.81	2.473	2.565	2.23	1.155	10.53	3.232	1.206
67	7.596	8.273	4.83	5.225	7.551	5.457	5.319	4.147	6.177
68	7.266	7.943	5.553	5.948	7.221	5.127	5.073	4.869	5.847
69	4.371	8.27	3.936	3.155	3.459	4.615	12.869	5.884	3.274
70	0.861	7.019	3.935	4.027	1.494	2.449	11.992	4.694	1.304
71	0.838	7.014	3.865	3.957	1.489	2.001	11.827	4.529	0.856
72	0.349	6.794	4.215	4.307	1.269	2.351	12.25	4.952	1.206
73	8.164	9.68	4.434	3.151	5.993	6.024	12.949	5.964	6.744
74	8.848	10.364	5.118	4.849	8.803	6.708	13.633	6.648	7.428
75	3.235	7.475	3.191	2.693	2.362	3.82	12.124	5.139	2.805
76	2.825	6.957	2.673	2.175	1.952	3.302	11.606	4.621	2.395
77	3.166	7.756	4.045	3.67	2.293	4.101	13.08	3.65	2.736
78	2.399	6.315	2.031	1.356	2.061	2.66	10.964	3.979	1.272
79	5.22	9.089	4.755	3.974	4.741	5.434	13.688	6.703	4.093
80	3.04	7.63	3.919	3.544	2.167	3.975	12.954	5.656	2.61
81	1.797	7.664	4.483	4.077	1.362	3.739	13.411	6.113	2.594
82	3.081	4.597	1.25	1.645	3.036	0.941	9.08	1.96	1.661
83	8.28	9.796	4.55	3.575	6.417	6.14	13.065	6.08	6.86
84	4.618	8.236	3.902	3.121	3.706	4.581	12.835	5.85	3.24
85	3.816	7.685	3.351	2.57	3.337	4.03	12.284	5.299	2.689
86	1.125	7.283	4.199	4.291	1.758	2.713	12.256	4.958	1.568
87	2.981	6.76	2.442	2.002	3.017	3.105	11.489	4.057	1.903
88	7.845	9.218	3.845	3.092	5.605	5.563	12.466	5.437	6.282

Table B.1 continued from previous page

89	6.878	8.251	2.878	1.863	4.376	4.596	11.499	4.47	5.315
90	2.201	6.762	2.649	2.893	1.514	2.877	11.575	4.917	1.732
91	2.862	6.641	2.43	2.674	2.413	2.986	11.356	3.985	1.784
92	1.465	5.587	2.851	2.893	1.975	1.701	11.074	3.762	0.556
93	2.964	4.313	1.853	1.895	3.279	0.658	10.093	2.78	1.449
94	3.682	8.243	4.245	4.489	2.995	4.358	13.171	6.398	3.213
95	4.178	10.044	5.75	5.792	2.542	4.621	13.973	6.661	3.476
96	1.971	7.837	5.172	5.214	1.536	3.637	13.39	6.078	2.492
97	4.222	5.595	2.305	2.536	4.054	1.94	8.568	3.001	2.659
98	8.592	9.965	4.592	3.577	6.09	6.31	13.213	6.184	7.029
99	1.515	7.411	4.508	4.55	0.576	3.181	12.731	5.419	2.036
100	1.511	7.377	4.712	4.754	1.076	3.177	12.93	5.618	2.032
101	4.344	8.934	5.335	4.554	3.432	5.279	14.268	6.96	3.914
102	3.508	8.098	4.499	3.718	2.596	4.443	13.432	6.124	3.078
103	3.786	8.376	4.777	3.996	2.874	4.721	13.71	6.402	3.356
104	4.638	9.228	5.629	4.848	3.726	5.573	14.562	7.254	4.208
105	2.018	4.602	3.012	3.104	2.374	0.716	10.975	3.677	0.577
106	1.789	5.767	3.034	3.126	1.535	2.112	11.091	3.793	0.662
107	1.223	5.314	3.292	3.384	2.067	1.428	11.254	3.956	0.283
108	0.872	5.665	3.643	3.735	1.716	1.779	11.605	4.307	0.634
109	1.132	5.071	3.384	3.476	2.513	1.52	11.367	4.048	0.375
110	4.714	7.912	2.192	1.752	4.265	4.257	12.142	5.113	4.977
111	6.112	6.595	4.195	4.426	5.944	3.83	6.422	4.891	4.549
112	4.319	8.748	3.951	3.511	3.632	5.093	12.998	5.569	3.85
113	1.618	5.487	4.819	4.861	1.959	3.284	13.037	5.725	2.139
114	5.759	8.542	8.96	9.002	5.931	7.425	17.178	9.866	6.28
115	4.45	4.859	2.533	2.764	4.282	1.204	8.872	3.229	2.357
116	4.349	8.91	4.908	4.468	3.662	5.025	13.955	6.526	3.88
117	3.267	7.828	3.566	3.126	2.58	3.943	12.613	5.181	2.798
118	4.556	5.929	1.451	1.978	4.388	2.274	8.179	1.035	2.993

Table B.1 continued from previous page

119	3.46	3.356	2.768	2.81	4.295	0.8	11.007	3.694	1.945
120	6.995	2.634	5.532	5.574	7.337	3.802	10.391	6.459	4.947
121	11.309	4.231	9.847	9.889	11.651	8.117	17.247	10.774	9.262
122	5.247	6.62	2.007	2.534	5.079	2.965	7.284	1.37	3.684
123	3.825	5.292	1.277	1.163	2.678	1.637	10.203	3.208	2.357
124	2.703	6.85	2.639	2.883	2.016	3.195	11.565	4.194	2.234
125	4.142	4.582	3.194	3.236	4.62	1.475	9.107	4.121	2.627
126	5.556	6.929	1.556	2.083	4.704	3.274	10.177	3.148	3.993
127	2.857	6.636	2.429	1.989	2.893	2.981	11.476	4.044	1.779
128	3.451	7.802	3.008	2.568	2.764	4.147	12.055	4.623	2.982
129	1.056	6.922	4.257	4.299	0.763	2.722	12.475	5.163	1.577
disposal	6.769	2.387	5.307	5.349	7.111	3.577	12.707	6.233	4.74

NODES	45	46	47	48	49	50	51	52	53
depot	5.47	5.741	8.588	7.652	5.148	5.032	11.991	5.556	5.128
1	0.884	1.252	4.122	2.89	2.56	2.115	7.121	2.125	0.626
2	0.978	1.281	3.941	3.479	2.406	2.78	6.48	1.492	1.394
3	1.868	2.171	5.017	4.369	1.626	1.982	7.37	0.368	2.302
4	2.567	2.87	4.409	5.068	3.485	3.841	6.682	2.227	2.983
5	1.976	2.561	5.431	4.341	1.058	0.943	7.996	1.824	1.935
6	7.355	6.986	8.88	4.999	9.853	9.408	10.217	9.418	7.083
7	6.039	6.623	11.832	8.907	6.996	5.6	13.169	6.315	5.697
8	7.776	7.774	9.807	5.798	10.163	10.196	11.005	9.556	8.144
9	1.753	2.056	3.594	4.254	2.844	2.946	5.867	1.658	2.132
10	2.056	2.359	3.898	4.281	3.045	3.147	6.171	1.611	2.435
11	2.617	2.84	3.663	4.046	3.966	4.068	7.614	1.749	2.996
12	1.823	2.191	5.061	3.901	2.485	2.04	8.216	2.05	1.565
13	3.737	3.85	6.712	3.625	4.848	4.733	7.964	4.814	3.078
14	3.577	3.852	6.494	3.516	5.412	4.967	7.831	4.977	3.318
15	1.659	1.29	3.904	1.019	4.055	3.61	5.241	3.276	1.387

Table B.1 continued from previous page

16	2.752	2.967	5.668	2.897	4.587	4.142	7.005	4.152	2.493
17	2.698	3.001	5.847	5.199	3.616	3.972	8.2	2.358	3.114
18	2.348	2.651	5.497	4.849	3.265	3.621	7.85	2.007	2.764
19	3.268	3.572	6.418	5.77	2.789	3.066	8.771	1.531	3.465
20	2.109	2.317	4.905	1.818	4.033	3.344	6.301	3.598	1.45
21	1.551	1.854	4.7	4.052	2.627	3.001	7.053	1.713	1.967
22	2.637	2.94	4.479	4.862	3.802	3.904	6.752	2.616	3.053
23	1.8	1.431	2.264	3.262	5.169	5.076	4.178	3.556	3.331
24	2.648	2.655	5.177	4.853	1.539	1.796	7.854	0.27	2.215
25	5.032	5.144	7.497	4.41	6.382	6.267	8.725	6.348	4.373
26	2.454	2.707	4.951	1.973	4.423	3.978	6.288	3.988	2.195
27	1.374	1.742	4.612	3.452	2.036	1.591	7.767	1.601	1.116
28	1.668	2.036	4.906	3.746	2.079	1.473	8.061	2.188	1.41
29	1.583	1.951	6.346	3.368	2.887	2.442	7.683	2.452	1.621
30	1.477	1.845	6.24	3.262	2.781	2.336	7.577	2.346	1.515
31	3.735	3.366	2.345	3.62	7.309	6.864	1.977	5.635	4.641
32	2.585	2.888	5.705	4.249	1.329	0.723	8.564	1.036	1.913
33	3.884	3.515	5.912	1.394	6.069	5.624	7.249	5.634	3.612
34	3.811	3.442	5.336	1.231	5.906	5.461	6.673	5.471	3.539
35	2.42	2.723	4.915	3.843	1.432	1.534	7.188	0.246	1.846
36	2.761	3.129	7.524	4.546	2.824	2.218	8.861	2.933	2.799
37	6.123	6.376	9.04	6.062	9.674	8.278	10.377	7.523	5.864
38	1.851	1.482	3.794	0.9	4.247	3.802	5.131	3.468	1.579
39	1.07	0.701	2.819	1.295	3.841	3.396	5.092	2.97	1.671
40	2.423	2.791	5.661	4.501	1.779	1.173	8.816	2.021	2.165
41	2.468	2.721	5.384	2.406	4.303	3.858	6.721	3.868	2.209
42	10.784	10.415	12.309	8.429	13.282	12.837	13.646	12.847	10.512
43	3.799	3.43	5.324	1.309	5.984	5.539	6.661	5.549	3.527
44	1.634	2.002	6.104	3.126	2.938	2.493	7.441	2.503	1.672
45	0	0.369	3.239	2.171	3.053	2.948	5.512	1.901	0.642

Table B.1 continued from previous page

46	0.369	0	2.87	1.925	3.421	2.949	5.143	2.269	1.011
47	3.239	2.87	0	4.202	6.268	5.796	4.264	5.116	3.858
48	2.171	1.925	4.202	0	5.488	5.043	5.429	4.184	2.295
49	3.053	3.421	6.268	5.488	0	0.518	8.268	1.326	2.675
50	2.948	2.949	5.796	5.043	0.518	0	9.211	1.682	2.56
51	5.512	5.143	4.264	5.429	8.268	9.211	0	7.469	6.475
52	1.901	2.269	5.116	4.184	1.326	1.682	7.469	0	1.954
53	0.642	1.011	3.858	2.295	2.675	2.56	6.475	1.954	0
54	2.475	2.571	4.848	0.724	4.921	4.806	6.1	4.887	2.911
55	1.875	2.178	5.024	4.376	3.773	3.68	7.377	2.359	2.291
56	0.683	0.766	3.51	2.862	3.725	3.632	5.863	2.449	1.734
57	1.185	0.816	2.944	1.758	3.84	3.725	5.297	3.245	2.179
58	3.046	3.159	6.02	2.933	4.157	4.042	7.272	4.123	2.387
59	2.555	2.858	5.704	5.056	3.63	3.986	8.057	2.373	2.971
60	6.495	6.863	11.244	8.46	7.066	5.84	12.799	6.555	6.237
61	0.285	0.654	3.501	2.852	2.669	2.554	5.854	2.531	0.663
62	0.614	0.982	3.829	2.897	2.197	2.104	6.182	1.323	0.715
63	1.342	1.613	4.46	3.588	2.498	1.698	7.362	1.779	1
64	0.893	2.06	4.93	2.733	2.354	1.909	8.085	1.919	1.434
65	8.638	8.392	10.669	6.66	11.025	10.91	11.921	10.418	9.006
66	1.675	2.867	5.823	2.736	2.74	2.625	7.075	2.706	1.057
67	6.101	5.855	8.132	4.124	8.061	7.946	9.384	7.881	6.469
68	6.824	6.578	8.855	4.846	7.731	7.616	10.107	7.697	5.722
69	2.151	2.454	5.3	4.652	2.764	3.12	7.653	1.506	2.567
70	2.285	2.556	7.285	4.198	2.275	2.159	8.537	2.683	1.943
71	2.41	2.681	7.12	4.033	2.27	2.154	8.372	2.678	2.068
72	2.372	2.643	7.543	4.456	2.05	1.934	8.795	2.458	2.03
73	3.571	3.202	0.712	4.732	6.327	6.683	4.819	5.07	5.069
74	5.269	4.9	3.135	5.416	9.131	8.381	5.503	6.768	6.757
75	1.624	1.992	4.839	3.907	2.444	2.252	7.192	2.256	1.725

Table B.1 continued from previous page

76	1.106	1.474	4.321	3.389	1.677	2.033	6.674	0.803	1.207
77	2.601	2.969	5.816	5.286	2.375	2.183	8.169	2.19	2.579
78	0.638	0.909	3.756	2.747	2.571	2.456	6.109	2.537	0.611
79	2.97	3.273	6.119	5.471	4.046	4.402	8.472	2.789	3.386
80	2.475	2.843	5.69	5.16	2.249	2.057	8.043	2.064	2.453
81	3.359	3.63	6.477	5.617	2.526	2.1	9.956	3.445	3.017
82	2.195	2.275	4.552	1.464	3.546	3.431	5.804	3.512	1.536
83	3.995	3.626	0.822	4.848	7.389	7.107	4.935	5.494	5.493
84	2.117	2.42	5.266	4.618	3.011	3.367	7.619	1.754	2.533
85	1.566	1.869	4.715	4.067	2.642	2.998	7.068	1.384	1.982
86	2.549	2.82	7.549	4.462	2.539	2.423	8.801	2.947	2.207
87	1.015	1.318	4.188	3.243	3.011	2.684	6.461	1.859	1.231
88	3.496	3.127	0.735	4.22	6.548	6.076	4.282	5.396	5.241
89	2.267	1.898	1.043	3.253	5.166	4.847	3.315	4.167	2.909
90	1.906	2.209	5.079	3.329	1.849	1.403	7.352	1.356	1.332
91	1.687	1.99	4.86	3.11	3.329	2.884	7.133	2.568	1.113
92	1.423	1.791	6.259	3.281	2.727	2.282	7.596	2.292	1.165
93	2.36	2.613	5.277	2.299	4.195	3.75	6.614	3.76	2.101
94	3.502	3.805	5.997	4.925	2.514	2.616	8.27	1.328	2.928
95	4.418	4.721	7.011	6.18	1.226	1.77	9.284	2.342	3.844
96	3.75	4.118	7.621	5.597	1.836	0.857	9.912	2.952	3.492
97	3.001	3.254	5.498	2.52	4.97	4.525	6.835	4.535	2.742
98	3.981	3.612	0.92	4.967	7.033	6.561	5.029	5.881	4.623
99	2.86	3.228	6.098	4.938	2.738	1.749	9.253	2.597	2.602
100	3.361	3.729	6.599	5.137	2.792	1.815	9.452	3.097	3.103
101	3.846	4.149	6.375	5.269	2.892	2.994	8.648	1.706	3.272
102	2.714	3.017	5.863	5.215	1.901	2.257	7.812	0.87	2.436
103	2.992	3.295	6.141	5.493	2.179	2.535	8.494	0.91	2.855
104	3.844	4.147	6.993	6.345	3.031	3.387	9.346	1.762	3.707
105	3.293	3.406	6.268	3.181	2.884	2.769	7.52	2.85	2.634

Table B.1 continued from previous page

106	1.155	1.426	6.384	3.297	2.045	1.93	7.636	2.011	0.813
107	1.821	2.092	6.547	3.46	2.848	2.732	7.799	2.516	1.479
108	2.172	2.443	6.898	3.811	2.497	2.381	8.15	2.867	1.83
109	1.913	3.799	6.66	3.552	3.023	2.908	7.912	2.989	3.027
110	2.156	1.787	1.685	3.896	5.208	4.736	3.958	4.056	2.798
111	4.891	5.144	7.388	4.41	6.86	6.415	8.725	6.425	4.632
112	2.527	2.83	4.369	4.752	3.151	3.253	6.642	1.965	2.906
113	3.473	3.841	8.222	5.244	4.213	2.818	9.559	3.533	3.215
114	7.614	7.982	12.363	9.385	8.185	6.959	13.7	7.674	7.356
115	3.229	3.482	5.726	2.748	5.198	4.753	7.063	4.763	2.97
116	3.484	3.787	5.326	5.709	3.181	3.283	7.599	1.995	3.561
117	2.139	2.442	4.75	4.367	2.099	2.201	7.023	0.913	2.479
118	3.119	2.75	4.644	0.63	5.304	4.859	5.981	4.869	2.847
119	3.275	3.528	6.191	3.213	5.11	4.665	7.528	4.675	3.016
120	6.039	6.292	8.956	5.978	9.591	8.195	10.293	7.439	5.78
121	10.354	10.607	13.271	10.293	13.905	12.509	14.608	11.754	10.095
122	3.675	3.306	5.2	1.32	5.995	5.55	6.537	5.292	3.403
123	0.556	0.669	3.683	1.957	3.594	3.149	6.188	2.456	0.927
124	1.896	2.199	5.069	3.319	1.919	2.021	7.342	0.767	1.322
125	3.701	3.954	6.618	3.64	5.536	5.091	7.955	5.101	3.442
126	3.224	2.855	2.186	1.931	5.62	5.175	3.523	4.841	2.952
127	1.002	1.305	4.175	3.23	2.887	2.782	6.448	1.735	1.107
128	1.581	1.884	4.193	3.809	2.283	2.385	6.466	1.097	1.96
129	2.835	3.203	6.073	4.682	2.751	1.774	8.997	2.784	2.577
disposal	5.916	6.067	8.73	5.752	9.365	7.969	10.067	7.214	5.555

NODES	54	55	56	57	58	59	60	61	62
depot	7.577	6.817	6.273	5.757	6.736	7.884	1.424	5.423	5.475
1	3.262	2.248	1.686	1.791	2.738	2.928	5.698	0.62	0.672
2	4.14	1.21	1.301	1.548	3.917	1.577	6.962	1.383	0.679

Table B.1 continued from previous page

3	5.03	2.101	2.191	2.438	4.643	2.115	6.855	2.273	1.569
4	5.729	2.52	2.89	3.137	5.609	2.045	8.714	2.972	3.41
5	4.266	3.809	2.779	3.1	3.631	4.152	5.215	1.929	1.984
6	4.477	8.044	7.65	6.546	7.247	9.844	12.63	7.64	7.685
7	8.764	7.386	7.057	8.729	6.121	8.619	4.228	5.991	6.043
8	5.25	8.833	8.438	7.579	8.035	8.358	13.418	8.428	8.473
9	4.224	1.705	2.076	2.868	4.795	1.23	7.634	2.158	2.593
10	4.527	2.009	2.379	3.172	4.995	1.534	8.098	2.461	2.794
11	5.274	1.774	2.94	2.937	5.786	1.299	8.235	3.022	3.35
12	3.811	3.187	2.625	2.73	3.047	4.354	6.893	1.559	1.611
13	3.55	5.659	4.416	3.54	0.839	6.185	6.851	3.919	3.976
14	3.551	5.448	4.494	3.39	1.278	6.128	8.64	3.862	3.919
15	1.656	3.468	1.954	0.85	2.573	1.8	7.651	2.004	1.989
16	2.6	4.623	3.669	2.565	1.163	5.303	7.001	3.037	3.094
17	5.169	2.651	3.021	3.268	5.637	2.176	8.845	3.103	3.541
18	5.51	2.301	2.671	3.464	5.287	1.826	8.494	2.673	3.19
19	6.431	3.222	3.591	3.839	5.806	2.746	8.018	3.673	2.713
20	1.743	4.031	2.788	1.877	1.331	3.556	6.714	2.349	2.406
21	4.713	1.479	1.874	2.642	4.593	1.004	7.856	1.876	2.197
22	5.108	2.59	2.96	3.753	5.679	2.115	8.76	3.042	3.37
23	3.899	2.787	2.128	1.978	4.377	2.312	8.327	2.085	2.413
24	5.007	2.585	2.675	2.922	4.243	2.812	6.768	3.053	1.443
25	4.335	6.954	5.711	4.835	3.608	6.479	9.637	5.176	5.233
26	1.883	4.325	3.371	2.267	1.868	5.005	7.201	2.739	2.796
27	3.362	2.738	2.176	2.281	2.598	3.905	5.447	0.804	1.162
28	3.656	3.032	2.47	2.575	2.892	3.712	4.827	1.404	1.456
29	3.278	2.947	2.385	3.263	2.132	3.627	4.897	1.319	1.371
30	3.172	2.841	2.279	3.157	2.026	3.521	5.554	1.213	1.265
31	4.266	5.543	4.029	3.463	5.438	6.223	10.747	4.02	4.348
32	4.159	2.956	2.908	3.078	3.395	3.34	5.651	1.907	1.38

Table B.1 continued from previous page

33	0.656	5.693	4.179	3.075	3.514	6.373	8.832	4.169	4.214
34	0.709	5.62	4.106	3.002	3.351	6.3	8.693	4.096	4.705
35	4.745	2.791	2.743	3.011	3.981	2.55	6.71	2.825	1.215
36	4.456	4.125	3.563	4.441	3.31	4.805	4.64	2.497	2.549
37	5.972	7.994	7.04	5.936	3.328	8.674	7.423	6.408	6.465
38	1.546	3.66	2.146	1.042	2.765	4.34	8.197	2.136	2.181
39	1.941	2.879	1.365	0.431	2.857	3.559	7.883	1.355	1.683
40	4.411	3.79	3.742	3.33	3.647	4.325	4.812	2.159	2.214
41	2.316	4.339	3.385	2.281	0.717	5.019	6.306	2.753	2.81
42	7.907	12.593	11.079	9.975	10.03	13.273	16.059	11.069	11.114
43	0.584	5.608	4.094	2.99	3.429	6.288	8.747	4.084	4.129
44	3.036	2.998	2.436	3	1.869	3.678	5.437	1.37	1.422
45	2.475	1.875	0.683	1.185	3.046	2.555	6.495	0.285	0.614
46	2.571	2.178	0.766	0.816	3.159	2.858	6.863	0.654	0.982
47	4.848	5.024	3.51	2.944	6.02	5.704	11.244	3.501	3.829
48	0.724	4.376	2.862	1.758	2.933	5.056	8.46	2.852	2.897
49	4.921	3.773	3.725	3.84	4.157	3.63	7.066	2.669	2.197
50	4.806	3.68	3.632	3.725	4.042	3.986	5.84	2.554	2.104
51	6.1	7.377	5.863	5.297	7.272	8.057	12.799	5.854	6.182
52	4.887	2.359	2.449	3.245	4.123	2.373	6.555	2.531	1.323
53	2.911	2.291	1.734	2.179	2.387	2.971	6.237	0.663	0.715
54	0	5.037	3.523	2.419	2.858	5.717	8.176	3.513	3.558
55	5.037	0	2.198	2.119	4.968	0.475	7.859	2.28	2.608
56	3.523	2.198	0	1.48	3.725	2.878	7.297	1.088	1.416
57	2.419	2.119	1.48	0	2.849	3.125	8.161	0.859	1.187
58	2.858	4.968	3.725	2.849	0	5.494	7.543	3.228	3.285
59	5.717	0.475	2.878	3.125	5.494	0	8.859	2.88	2.529
60	8.176	7.859	7.297	8.161	7.543	8.859	0	6.231	6.283
61	3.513	2.28	1.088	0.859	3.228	2.88	6.231	0	0.624
62	3.558	2.608	1.416	1.187	3.285	2.529	6.283	0.624	0

Table B.1 continued from previous page

63	3.498	2.689	2.145	2.417	2.997	3.369	5.375	1.058	1.893
64	3.68	3.056	1.696	2.599	3.121	4.223	5.509	1.428	1.48
65	6.112	9.695	9.056	8.441	8.897	9.22	14.28	9.341	9.9
66	2.661	3.103	2.478	2.557	1.63	4.201	6.184	1.391	2.307
67	3.576	7.158	6.519	5.904	5.933	6.683	11.316	6.804	7.363
68	4.298	7.881	7.242	6.627	5.603	7.406	10.986	6.057	8.086
69	5.313	2.104	2.474	2.721	5.09	1.629	7.993	2.476	1.852
70	4.123	3.632	3.088	4.019	3.325	4.312	4.581	2.001	2.836
71	3.958	3.757	3.213	3.949	2.725	5.593	4.576	2.126	2.961
72	4.381	3.719	3.279	2.659	3.075	4.786	4.356	2.088	2.923
73	5.393	3.177	3.866	3.716	6.5	2.702	11.884	4.406	4.71
74	6.077	4.875	5.564	5.414	7.184	4.4	12.568	6.104	6.408
75	4.568	2.605	2.557	2.197	4.295	3.285	6.857	1.634	1.809
76	4.05	2.087	2.039	1.679	3.777	2.767	6.447	1.116	1.291
77	5.211	3.582	3.534	3.174	4.576	4.262	6.788	2.611	2.786
78	3.408	2.633	1.441	0.925	3.135	2.456	6.133	0.354	1.963
79	6.132	0.896	3.293	3.54	5.909	0.421	9.275	3.295	2.671
80	5.085	3.456	3.408	3.048	4.45	4.136	6.662	2.485	2.66
81	5.542	4.706	4.162	3.646	4.463	5.773	4.568	3.075	3.91
82	1.389	3.578	2.939	1.998	1.417	3.103	6.801	1.871	3.396
83	5.509	3.601	4.29	4.14	6.616	3.126	12	4.83	5.134
84	5.279	2.07	2.44	2.687	5.056	1.595	8.24	2.442	1.818
85	4.728	1.831	1.889	2.136	4.505	1.356	7.55	1.891	1.267
86	4.387	3.896	3.352	4.283	3.589	4.576	4.845	2.265	3.1
87	3.486	1.386	1.338	2.134	3.581	2.066	6.603	1.42	1.748
88	4.866	5.304	3.79	3.224	6.038	5.984	11.565	3.781	4.109
89	3.899	4.075	2.561	1.995	5.071	4.755	8.447	2.552	2.88
90	4.346	2.277	2.229	2.497	3.582	2.957	5.823	2.311	0.701
91	3.414	2.058	2.01	2.278	3.462	2.738	6.484	2.092	0.482
92	3.191	2.787	2.225	3.171	2.407	3.467	5.185	1.159	1.211

Table B.1 continued from previous page

93	2.209	4.231	3.277	2.173	1.133	4.911	6.886	2.645	2.702
94	5.827	3.873	3.825	4.093	5.063	3.632	7.304	3.907	2.297
95	6.09	4.789	4.741	5.009	5.326	4.646	7.437	3.838	3.213
96	5.507	4.445	4.397	4.657	4.361	5.256	5.23	3.486	2.869
97	2.43	4.872	3.918	2.814	2.415	5.552	7.942	3.286	3.343
98	5.613	5.789	4.275	3.709	6.785	6.469	10.161	4.266	4.594
99	4.848	4.366	3.662	3.767	3.905	4.901	4.804	2.596	2.79
100	5.047	4.866	4.163	4.268	3.901	5.401	4.77	3.097	3.29
101	6.205	4.217	4.169	4.437	5.441	4.01	7.682	4.251	2.641
102	5.369	3.381	3.333	3.601	4.605	3.174	6.846	3.415	1.805
103	5.831	3.225	3.611	3.879	4.883	3.452	7.124	3.693	2.083
104	6.683	4.077	4.465	4.733	5.737	4.306	7.978	4.547	2.937
105	3.106	3.502	4.108	3.004	1.44	4.255	6.014	1.947	1.999
106	3.222	2.502	1.73	2.482	2.69	2.972	5.641	0.664	0.716
107	3.385	3.168	2.679	3.283	2.152	3.921	5.154	1.613	1.665
108	3.736	3.519	3.03	3.634	2.503	4.272	4.803	1.964	2.016
109	3.477	3.641	2.771	3.375	2.244	4.013	5.063	1.705	1.757
110	4.542	3.964	2.45	1.884	4.733	4.644	8.336	2.441	2.769
111	4.32	6.762	5.808	4.704	3.608	7.442	9.832	5.176	5.233
112	4.998	2.48	2.85	3.643	5.569	2.005	7.941	2.932	2.9
113	5.154	4.837	4.275	5.139	5.607	5.837	4.689	3.209	3.261
114	9.295	8.978	8.416	9.28	8.662	9.978	1.119	7.35	7.402
115	2.658	5.1	4.146	3.042	1.679	5.78	8.17	3.514	3.571
116	5.955	3.437	3.807	4.6	5.73	2.962	7.971	3.889	2.93
117	4.61	2.372	2.462	3.258	4.648	2.385	6.889	2.544	1.848
118	0.464	4.928	3.414	2.31	2.749	5.608	8.276	3.404	3.449
119	3.123	5.146	4.192	3.088	0.71	5.826	7.571	3.56	3.617
120	5.888	7.91	6.956	5.852	3.244	8.59	7.341	6.324	6.381
121	10.203	12.225	11.271	10.167	7.559	12.905	11.655	10.639	10.696
122	0.798	5.484	3.97	2.866	3.44	6.164	8.967	3.96	4.005

Table B.1 continued from previous page

123	2.637	2.427	1.235	1.125	2.113	3.107	7.545	0.841	1.169
124	3.623	2.267	2.219	2.487	3.671	2.947	6.325	2.301	0.691
125	3.55	5.572	4.618	3.514	1.402	6.252	8.797	3.986	4.043
126	2.577	5.033	3.519	2.415	3.749	5.713	9.276	3.509	3.554
127	3.473	1.373	1.325	2.121	3.457	2.053	6.479	1.407	1.735
128	4.052	1.814	1.904	2.7	4.623	1.828	7.073	1.986	2.032
129	4.592	4.199	3.637	3.742	3.446	5.088	4.315	2.571	2.623
disposal	5.662	7.685	6.731	5.627	3.019	8.365	7.114	6.099	6.156

NODES	63	64	65	66	67	68	69	70	71
depot	4.567	5.341	13.472	5.376	10.508	10.178	7.185	3.773	3.768
1	0.779	0.508	9.357	1.022	6.82	6.073	2.524	1.551	1.676
2	1.792	1.343	9.673	2.206	7.136	7.859	1.174	2.735	2.86
3	2.103	2.264	10.563	3.03	8.026	8.021	1.318	3.007	3.002
4	3.381	4.078	11.259	3.795	8.722	9.445	0.727	4.77	4.765
5	1.073	1.847	10.285	2	7.321	6.991	3.268	1.534	1.529
6	8.072	7.521	1.651	7.101	1.889	1.644	9.44	8.563	8.398
7	5.136	5.909	14.521	5.944	11.557	11.227	7.753	4.341	4.336
8	8.74	8.922	1.9	7.889	2.678	2.432	10.228	9.351	9.186
9	2.567	2.118	10.444	2.981	7.907	8.63	0.473	3.51	3.948
10	2.87	3.462	10.748	3.284	8.211	8.934	0.095	4.154	4.149
11	3.431	2.982	10.513	3.845	7.976	8.699	1.482	4.374	5.07
12	0.703	0.201	9.915	1.63	6.951	6.621	3.488	1.905	2.03
13	3.688	3.812	9.588	2.321	6.624	6.294	5.781	3.846	3.398
14	3.631	3.755	6.972	2.264	4.008	3.678	5.724	3.959	3.359
15	2.274	1.825	7.477	2.281	4.94	5.663	3.744	3.743	3.673
16	2.806	2.725	7.242	1.439	4.278	3.948	4.899	3.134	2.697
17	3.512	3.063	11.39	3.926	8.853	9.576	0.858	4.901	4.896
18	3.162	2.713	11.04	3.576	8.503	9.226	0.508	4.55	4.545
19	3.169	3.33	11.96	4.096	9.423	9.087	1.196	4.073	4.068

Table B.1 continued from previous page

20	2.036	2.242	8.001	1.281	5.037	4.707	4.211	2.743	2.578
21	2.278	1.941	10.194	2.704	7.756	8.355	1.179	3.251	3.946
22	3.451	3.002	11.576	3.865	8.792	9.515	0.796	4.394	4.906
23	3.044	2.595	9.729	4.085	7.192	7.915	2.468	5.547	5.477
24	2.276	2.06	10.538	2.826	8.001	7.817	1.946	2.803	2.798
25	5.079	5.069	4.643	4.108	1.679	1.349	7.038	5.57	5.405
26	2.523	2.705	7.08	1.671	4.116	3.786	4.601	3.133	2.968
27	0.254	0.318	9.466	1.181	6.502	6.172	2.711	1.456	1.581
28	0.548	0.682	9.831	1.475	6.796	6.466	3.626	1.146	1.141
29	1.106	1.534	9.382	1.286	6.418	6.088	3.223	1.423	1.079
30	0.727	0.909	9.283	1.18	6.312	5.982	3.117	1.317	0.973
31	5.528	5.079	10.033	5.215	7.55	8.273	5.819	6.703	6.538
32	1.051	1.185	10.334	1.978	7.299	6.969	2.474	1.97	1.965
33	4.288	4.336	6.397	3.317	3.861	4.583	5.969	4.779	4.614
34	4.125	4.197	5.821	3.154	3.285	4.007	5.896	4.43	4.451
35	2.218	2.074	10.31	2.564	7.773	7.555	1.684	2.541	2.536
36	1.753	1.887	10.56	2.464	7.596	7.266	4.371	0.861	0.838
37	6.177	6.096	11.237	4.81	8.273	7.943	8.27	7.019	7.014
38	2.466	3.34	7.367	2.473	4.83	5.553	3.936	3.935	3.865
39	1.303	3.382	7.762	2.565	5.225	5.948	3.155	4.027	3.957
40	2.522	1.438	10.515	2.23	7.551	7.221	3.459	1.494	1.489
41	2.522	2.211	8.421	1.155	5.457	5.127	4.615	2.449	2.001
42	11.501	10.95	3.967	10.53	5.319	5.073	12.869	11.992	11.827
43	4.203	4.251	6.683	3.232	4.147	4.869	5.884	4.694	4.529
44	1.157	1.066	9.141	1.206	6.177	5.847	3.274	1.304	0.856
45	1.342	0.893	8.638	1.675	6.101	6.824	2.151	2.285	2.41
46	1.613	2.06	8.392	2.867	5.855	6.578	2.454	2.556	2.681
47	4.46	4.93	10.669	5.823	8.132	8.855	5.3	7.285	7.12
48	3.588	2.733	6.66	2.736	4.124	4.846	4.652	4.198	4.033
49	2.498	2.354	11.025	2.74	8.061	7.731	2.764	2.275	2.27

Table B.1 continued from previous page

50	1.698	1.909	10.91	2.625	7.946	7.616	3.12	2.159	2.154
51	7.362	8.085	11.921	7.075	9.384	10.107	7.653	8.537	8.372
52	1.779	1.919	10.418	2.706	7.881	7.697	1.506	2.683	2.678
53	1	1.434	9.006	1.057	6.469	5.722	2.567	1.943	2.068
54	3.498	3.68	6.112	2.661	3.576	4.298	5.313	4.123	3.958
55	2.689	3.056	9.695	3.103	7.158	7.881	2.104	3.632	3.757
56	2.145	1.696	9.056	2.478	6.519	7.242	2.474	3.088	3.213
57	2.417	2.599	8.441	2.557	5.904	6.627	2.721	4.019	3.949
58	2.997	3.121	8.897	1.63	5.933	5.603	5.09	3.325	2.725
59	3.369	4.223	9.22	4.201	6.683	7.406	1.629	4.312	5.593
60	5.375	5.509	14.28	6.184	11.316	10.986	7.993	4.581	4.576
61	1.058	1.428	9.341	1.391	6.804	6.057	2.476	2.001	2.126
62	1.893	1.48	9.9	2.307	7.363	8.086	1.852	2.836	2.961
63	0	0.572	9.602	1.317	6.638	6.308	3.15	1.592	1.717
64	0.572	0	9.784	1.499	6.635	6.49	3.357	1.774	1.899
65	9.602	9.784	0	8.751	3.54	3.294	11.09	10.213	10.048
66	1.317	1.499	8.751	0	5.784	5.454	3.552	1.926	2.062
67	6.638	6.635	3.54	5.784	0	1.238	8.554	7.249	7.084
68	6.308	6.49	3.294	5.454	1.238	0	9.277	6.919	6.754

Appendix C

Table of Demands per Barangay

Table C.1: Demands per Barangay

Nodes	Demand	Node	Demand	Node	Demand	Node	Demand
Depot	0	**42**	7	**84**	2	**126**	4
1	5	**43**	2	**85**	4	**127**	11
2	7	**44**	9	**86**	7	**128**	11
3	4	**45**	13	**87**	14	**129**	5
4	10	**46**	13	**88**	5		
5	2	**47**	6	**89**	11		
6	5	**48**	5	**90**	13		
7	7	**49**	6	**91**	8		
8	2	**50**	11	**92**	10		
9	9	**51**	2	**93**	6		
10	8	**52**	1	**94**	12		
11	3	**53**	1	**95**	6		
12	5	**54**	14	**96**	8		
13	12	**55**	10	**97**	13		
14	11	**56**	10	**98**	4		
15	1	**57**	6	**99**	10		
16	8	**58**	12	**100**	12		
17	12	**59**	11	**101**	11		
18	9	**60**	3	**102**	5		
19	14	**61**	9	**103**	2		
20	10	**62**	10	**104**	3		

21	13	**63**	5	**105**	1	
22	12	**64**	7	**106**	4	
23	2	**65**	11	**107**	9	
24	14	**66**	4	**108**	2	
25	12	**67**	4	**109**	2	
26	3	**68**	8	**110**	12	
27	6	**69**	11	**111**	1	
28	3	**70**	8	**112**	2	
29	1	**71**	3	**113**	7	
30	1	**72**	6	**114**	4	
31	4	**73**	9	**115**	2	
32	13	**74**	3	**116**	5	
33	2	**75**	9	**117**	10	
34	5	**76**	3	**118**	7	
35	4	**77**	8	**119**	6	
36	1	**78**	4	**120**	11	
37	8	**79**	14	**121**	13	
38	9	**80**	1	**122**	6	
39	5	**81**	7	**123**	14	
40	12	**82**	12	**124**	3	
41	12	**83**	11	**125**	11	