OPTIMIZING DELIVERY ROUTES FOR BUSINESSES USING BRANCH AND BOUND ALGORITHM, A CASE STUDY OF VIBOI VENTURES

BY

ODOSAMAMWEN DANIEL OSAYUWA

ENG2006336

THE DEPARTMENT OF INDUSTRIAL ENGINEERING,

FACULTY OF ENGINEERING,

UNIVERSITY OF BENIN,

BENIN CITY, NIGERIA

NOVEMBER 2025

OPTIMIZING DELIVERY ROUTES FOR BUSINESSES USING BRANCH AND BOUND ALGORITHM, A CASE STUDY OF VIBOI VENTURES

BY

ODOSAMAMWEN DANIEL OSAYUWA

ENG2006336

A REPORT SUBMITTED TO THE DEPARTMENT OF INDUTSRIAL ENGINEERING IN PARTIAL FULFULLMENT OF THE REQUIREMENT FOR THE AWARD OF BACHELOR OF ENGINEERING (B.Eng) DEGREE

IN

THE DEPARTMENT OF INDUSTRIAL ENGINEERING,

FACULTY OF ENGINEERING,

UNIVERSITY OF BENIN,

BENIN CITY, NIGERIA

NOVEMBER 2025

CERTIFICATION

This is to certify that this project work titled "OPTIMIZING DELIVERY ROUTES FOR BUSINESSES USING BRANCH AND BOUND ALGORITHM, A CASE STUDY OF VIBOI VENTURES" was carried out by ODOSAMAMWEN DANIEL OSAYUWA with Matriculation Number ENG2006336, a final year student of the Department of Industrial Engineering, Faculty of Engineering, University of Benin, Benin City, Edo state, Nigeria.

______________________ ______________________

Engr. Dr. Emmanuel Ikpoza Date

(Project Supervisor)

______________________ ______________________

Engr. Dr. (Mrs) I. C ILOUBE Date

(Project Coordinator)

______________________ ______________________

Prof. P. E Amiolemhen Date

(Head of Department)

DEDICATION

This project is dedicated to my beloved late father, who was a strong pillar of support in my life and is the reason why I will continue to strive for greater heights in life. May his soul rest in peace, Amen

ACKNOWLEDGEMENTS

I wish to express my heartfelt gratitude to God Almighty for His wisdom, strength, and provision throughout the course of this study. My profound appreciation goes to my supervisor, Engr. Dr E. Ikpoza for his guidance, patience, and constructive criticism which contributed immensely to the success of this research.

I am equally grateful to the management and staff of VIBOI VENTURES, who provided valuable information and support during the data collection phase. My appreciation also goes to the HOD, the coordinator and all my lecturers in the Department of Industrial Engineering for their dedication and commitment to academic excellence. And to my Father and course advisor Engr. Dr. N.H Osadiaye for always looking out for me. May the Grace of God be upon you.

Finally, special thanks to my family - my mother Mrs Joyce Osayuwa**,** for her prayers, sponsorship and words of encouragement, - my siblings Divine, and matthew, - to my wonderful aunts and uncles for their support and advice and to my friends for their continuous encouragement and moral support. This success would not have been possible without you all.

ABSTRACT

Efficient route planning plays a crucial role in logistics management because it directly affects transportation cost, delivery time, and operational efficiency. Many delivery operations experience inefficiencies due to poorly structured routing strategies, particularly when multiple locations must be served from a single depot. This study focuses on optimizing delivery routes for Viboi Ventures, a Coca Cola distribution depot located in Benin City, Nigeria. The primary aim of the research is to determine the most efficient delivery sequence that minimizes total travel distance while ensuring that each destination is visited exactly once. To achieve this aim, the study obtained delivery location data, determined inter-location distances using Google Maps, and developed an optimization model to evaluate alternative delivery routes.

The study adopts a quantitative modelling approach based on the Travelling Salesman Problem (TSP), a well established optimization model used to determine the shortest possible route through multiple locations. Distance data between the depot and nine delivery points were collected using Google Maps and organized into a distance matrix. Manual computations were first conducted to evaluate feasible route combinations. To enhance the reliability of the results, the Branch and Bound optimization algorithm was subsequently applied to systematically evaluate route alternatives and validate the optimal solution.

The analysis showed that the manually derived route produced a total travel distance of 13.7 km, while the algorithm based solution generated a shorter distance of 12.2 km both was significantly lower than that of the company's existing delivery route which is 18.7km. This result demonstrates the effectiveness of algorithmic optimization in improving delivery efficiency and reducing travel distance. The study concludes that applying computational optimization techniques such as the Branch and Bound algorithm, combined with real world

geographic data, can significantly improve logistics planning and operational performance for delivery based businesses.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

1.1 Background to the Study

Urban logistics has become increasingly complex due to rapid population growth, the expansion of e-commerce activities, and rising customer expectations for faster delivery services. As commercial activities intensify in urban environments, businesses that rely on delivery operations must ensure that their logistics systems remain efficient, cost effective, and reliable. However, many delivery operations still depend on manual route planning, driver experience, or basic navigation tools, which often lead to inefficient routing, longer travel times, and higher operational costs. Similar challenges have been observed in dense commercial environments where poor route planning and inefficient vehicle scheduling contribute to delays and increased fuel consumption (Kwon and Jeong, 2022; Verbytskyi, 2023).

In modern logistics management, route optimization plays a crucial role in improving delivery performance. Route optimization refers to the process of determining the most efficient path for vehicles to travel between multiple destinations while minimizing travel time, distance, or operational costs. Through optimized routing, businesses can utilize resources more effectively, reduce fuel consumption, and improve delivery reliability. Empirical studies have shown that logistics systems that incorporate optimization models tend to achieve better vehicle utilization and more predictable delivery schedules (Wu *et al.* 2022).

One of the major challenges in delivery logistics is the presence of uncertain traffic conditions, road constraints, and varying delivery schedules. These factors can significantly affect delivery efficiency when routes are not carefully planned. Research indicates that organizations that adopt computational optimization techniques are better able to manage such uncertainties and improve delivery outcomes. For instance, Tang (2025) demonstrated that incorporating real-world uncertainties such as traffic delays and environmental conditions into routing models significantly improved delivery accuracy and reduced travel time.

Recent studies increasingly emphasize the use of algorithmic approaches to solve routing problems in logistics operations. Optimization algorithms allow businesses to evaluate multiple delivery combinations and determine the most efficient delivery sequence. According to Willow *et al*. (2023), the application of optimization algorithms in delivery systems can significantly shorten delivery distances and enable businesses to prioritize deliveries based on urgency and operational constraints. Similarly, Kwon and Jeong (2022) found that algorithm-based routing systems integrated with digital mapping platforms offer practical solutions for small and medium-sized enterprises seeking to improve delivery efficiency while reducing operational costs.

Digital mapping platforms such as Google Maps have made it easier to obtain real-time information about routes, distances, and traffic conditions. These platforms are widely used in logistics operations because they provide accurate geographic data and navigation support. However, despite their usefulness, such platforms primarily offer navigation assistance rather than automated route optimization. As a result, businesses that rely solely on mapping tools may still determine delivery sequences through manual planning or trial and error approaches. Wu *et al* (2022) note that while mapping technologies provide valuable spatial data, optimal route determination requires additional computational frameworks capable of evaluating multiple routing possibilities.

To address these limitations, optimization algorithms such as the Branch and Bound algorithm can be applied to delivery route planning. The Branch and Bound method is widely used for solving combinatorial optimization problems involving discrete decision variables, particularly those related to route selection. The algorithm systematically evaluates potential route combinations while eliminating inefficient alternatives, thereby reducing unnecessary computational effort and improving solution accuracy. Previous research has demonstrated that the application of the Branch and Bound algorithm can significantly improve delivery efficiency and reduce travel distances in multi-destination routing problems (Willow *et al.* 2023).

The integration of real-time geographic data from Google Maps with the Branch and Bound algorithm offers a practical and cost effective approach to delivery route optimization. By combining accurate distance data with mathematical optimization techniques, businesses can generate routes that minimize travel distance, reduce delivery time, and lower operational costs. Such approaches are particularly beneficial for small and medium sized enterprises that may lack access to advanced logistics platforms but still require efficient delivery planning systems (Kwon and Jeong, 2022).

Therefore, this study focuses on developing an optimized delivery routing model using the Branch and Bound algorithm, with distance data obtained through Google Maps, to improve logistics performance for a delivery based business in Benin City, Edo State. The proposed approach aims to provide a practical and scalable solution for enhancing delivery efficiency within urban logistics systems.

1.2 Statement of the Problem

Many businesses that rely on regular delivery operations struggle with inefficiencies caused by poor route planning. These inefficiencies often result in longer delivery times, increased fuel consumption, and higher operational costs, which ultimately affect customer satisfaction and profit margins. Manual methods of route selection or relying solely on driver experience do not account for traffic conditions, road constraints, or optimal delivery sequencing, leading to repeated delays and unpredictable delivery outcomes.

Although tools like Google Maps provide real-time traffic and distance data, they do not inherently generate the most efficient delivery order across multiple stops. Without a structured approach to compute and evaluate route combinations, businesses risk underutilising these digital tools. There is a need for a practical, low-cost, and data-driven solution that integrates map based tools with a suitable optimization algorithm such as Branch and Bound to improve delivery efficiency, especially in urban areas with dense delivery networks.

1.3 Aim and Objectives

To develop an optimized delivery routing model using the Branch and Bound algorithm for enhanced logistics performance in Benin City, Edo State.

The study is guided by the following specific objectives:

1. Obtain delivery route data from a selected business located in Benin City.
2. Apply mapping tools available through Google Maps to determine travel distances and durations between delivery points.
3. Implement the Branch and Bound algorithm in computing the most efficient delivery route among a defined set of customer locations.
4. Compare the performance of the optimized route with the business's current delivery practice by assessing travel time , route length, and cost implications.
5. Develop a functional model that offers route improvement and can be adopted by other businesses with similar delivery patterns across urban regions.

1.4 Scope of the Study

The study focuses on the application of delivery route optimization within the geographical boundaries of Benin City, located in Edo State, Nigeria. A single delivery-based company operating within the city shall serve as the source of delivery route data. This company's delivery structure provides the foundation for both the current route analysis and the construction of an improved routing system. The research employs the Branch and Bound algorithm as the sole computational algorithm for optimization and uses Google Maps to acquire real world distance measurements between delivery points. The study does not extend beyond the city of Benin or consider alternative optimization techniques or delivery systems operating in rural or intercity contexts. The intent is to establish a practical model applicable within an urban environment where delivery challenges are defined by traffic variability, road complexity, and route uncertainty.

1.5 Significance of the Study

This study is significant for several reasons.

First, it provides a practical approach to improving delivery logistics for businesses operating in urban environments. By applying optimization techniques, companies can reduce delivery time, minimize travel distance, and lower operational costs.

Second, the study demonstrates how accessible digital tools such as Google Maps can be integrated with optimization algorithms to develop effective routing solutions without the need for expensive logistics software.

Third, the research contributes to the field of operations research and logistics management by presenting a localized case study that applies theoretical optimization methods to real world delivery operations.

Finally, the study may serve as a useful reference for businesses, researchers, and logistics planners seeking to implement efficient route planning systems in urban areas.

1.6 Limitations of the Study

This study is subject to certain limitations that may influence the interpretation of the results. First, the optimization process was carried out manually without the use of specialized computational software, which limited the scalability of the analysis to a relatively small number of delivery locations. Secondly, the distance data used in the model were obtained from Google Maps based on static geographic measurements and did not account for dynamic factors such as real-time traffic conditions, road maintenance, or temporary diversions. Finally, the study focused on a single delivery network associated with Viboi Ventures in Benin City, which may limit the generalizability of the findings to larger or intercity delivery systems.

CHAPTER TWO

LITERATURE REVIEW

2.1 Concept of Optimization

Optimization refers to the process of identifying the best possible solution to a problem from a set of available alternatives while considering specific objectives and constraints. It involves selecting the most efficient option in order to achieve a desired outcome, such as minimizing cost, reducing time, or maximizing performance and efficiency. In decision making contexts where multiple alternatives exist, optimization provides a structured approach for determining the most suitable option within defined boundaries (Lamata *et al.* 2017).

In mathematical and computational contexts, optimization problems are typically formulated using mathematical models that clearly represent the decision environment. These models consist of an objective function, which defines the quantity to be minimized or maximized, and a set of constraints, which represent the limitations that must be satisfied for a solution to be feasible (Lubinska, 2020). By structuring problems in this manner, optimization techniques allow decision makers to analyze different alternatives systematically and select the most effective solution among them. According to Lamata *et al* (2017), such models provide a logical framework for solving complex decision problems by translating real world challenges into mathematical representations.

Optimization techniques are widely applied across numerous fields including engineering, economics, logistics, transportation, and computer science. Their importance lies in their ability to simplify complex decision making processes and guide resource allocation in a rational and measurable manner. Optimization models enable organizations to evaluate multiple possible outcomes and determine the most efficient course of action, thereby improving operational effectiveness and strategic planning.

In logistics and transportation management, optimization plays a particularly important role because it allows organizations to allocate resources efficiently, reduce operational costs, and improve service performance. Decision making in these environments often involves several competing factors such as time constraints, delivery requirements, and resource limitations. Optimization techniques help balance these factors by providing systematic and repeatable decision frameworks that improve accuracy and efficiency.

In an optimization problem, three fundamental components work in tandem to define the scope and direction of the search for an ideal solution. At the core is the objective function, which serves as the primary goal by mathematically defining the specific quantity such as cost, distance, or performance that must be either minimized or maximized. This goal is achieved by manipulating decision variables, which represent the controllable elements or alternatives available within the process. However, these variables cannot be changed indefinitely; they are governed by constraints, which are the essential limitations or operational requirements, such as resource availability and capacity restrictions, that define the boundaries of a feasible solution. Together, these elements ensure that the optimization remains both goal oriented and grounded within practical realities.

By structuring decision problems using these components, optimization provides a systematic framework for analyzing alternatives and identifying efficient solutions. This structured approach reduces reliance on intuition or guesswork and instead promotes decision making based on logical analysis, measurable criteria, and evidence based reasoning.

2.2 Types of Optimization

Optimization exists in various forms, each suited to different types of problems depending on the structure of the objective function, the nature of the variables involved, and the presence of uncertainty or complexity in the environment. While all types aim to identify the best solution from a set of possible options, they differ in how problems are formulated and solved.

Some involve continuous variables and straightforward linear relationships, while others work with discrete values, complex mathematical expressions, or uncertain parameters. A good understanding of the distinct characteristics of each optimization type provides a foundation for selecting the most appropriate method for specific decision making scenarios. Common classifications include linear, nonlinear, integer, combinatorial, and stochastic optimization, each addressing unique problem structures and offering specific tools for finding efficient and effective solutions.

2.2.1 Linear Optimization

Linear optimization is a type of optimization where both the objective function and the constraints are linear. It involves determining the best outcome, such as maximum profit or minimum cost, by evaluating a function subject to a set of linear inequalities or equations. The general structure of a linear optimization problem can be expressed as:

Maximise or Minimise: $Z = c1x1 + c2x2 + ... + cnxn$

Subject to: $a11x1 + a12x2 + ... + a1nxn \leq b1$

$a21x1 + a22x2 + ... + a2nxn \leq b2$

Here, Z is the objective function to be optimised, the x values are decision variables, and the coefficients represent the parameters of the constraints.

Linear optimization is especially effective when dealing with problems that require resource allocation or cost reduction under fixed limitations. It allows a decision maker to determine the best mix of variables that yield the highest or lowest possible outcome.

Some common problems solved using linear Optimization include:

1. Budget allocation
2. Production planning
3. Staff scheduling
4. Transportation and distribution under fixed costs

Solutions to linear optimization problems can often be obtained using methods like the Simplex algorithm or interior point methods, which identify optimal points in a feasible region formed by the constraints. See

2.2.2 Nonlinear Optimization

Nonlinear optimization is used when the relationship between variables is not linear, meaning that either the objective function or at least one constraint involves powers, products, trigonometric functions, or other non linear expressions. It addresses more complex situations where changes in variables produce a curve rather than a straight line in output.

The structure of a nonlinear optimization problem can be written as:

Minimize or Maximise: f(x1, x2, ..., xn)

Subject to: gi(x1, x2, ..., xn) ≤ bi

Where f and gi are nonlinear functions. This form accommodates a wide range of real-world phenomena where relationships between variables are complex and interdependent.

Problems addressed through nonlinear Optimization often include:

1. Energy consumption models
2. Engineering design configurations
3. Financial portfolio optimizations involving risk
4. Production efficiency under changing cost functions

Solutions may be obtained using methods such as gradient descent, Newton Raphson, or sequential quadratic programming, depending on the nature and smoothness of the functions involved. Because of the complexity, nonlinear optimization may not always guarantee a global optimum, especially if the objective function has many local optima.

2.2.3 Integer Optimization

Integer optimization involves problems where some or all decision variables must take integer values. It is useful in scenarios where decisions are inherently indivisible, such as the number of machines, workers, or vehicles required.

A typical integer optimization problem has the structure:

Maximise or Minimize: $Z = f(x1, x2, ..., xn)$

Subject to: constraints

Where $x1, x2, ..., xn \in$ Integers

There are two main types of integer optimization:

1. Pure Integer Programming: where all variables must be integers
2. Mixed Integer Programming: where only some variables are restricted to integers

Problems solved using integer optimization include:

1. Project selection with fixed budgets
2. Job assignment and crew scheduling
3. Facility location planning
4. Vehicle routing where discrete units are involved

The branch and bound algorithm is often used in solving integer optimization problems, as it systematically explores potential solutions while pruning parts of the search space that do not satisfy the integer condition.

2.2.4 Combinatorial Optimization

Combinatorial optimization focuses on problems where the goal is to find the best arrangement or ordering of a finite set of discrete items. It arises in scenarios where selecting or ordering elements results in different outcomes, and the number of possible combinations grows rapidly with the size of the input, making it one of the most challenging fields in optimization.

The basic form of a combinatorial optimization problem involves:

1. A finite set of elements
2. An objective function defined over the combinations of those elements
3. A method for evaluating or ranking each combination

The goal is to find the combination that optimizes the objective function.

Common problems solved by combinatorial optimization include:

1. Travelling Salesman Problem (TSP): determining the shortest route visiting multiple cities
2. Knapsack Problem: selecting items with maximum value without exceeding capacity
3. Assignment Problem: matching agents to tasks for minimal cost
4. Network Design: optimising paths in data or transport systems

Because of the discrete and often factorial nature of the problem spaces, combinatorial optimization problems are usually NP hard. This means they cannot be solved efficiently with simple methods, especially as the number of elements increases. Exact methods like branch and bound and dynamic programming can be applied for smaller instances, while larger problems often require heuristic or metaheuristic approaches.

These include:

1. Greedy algorithms
2. Genetic algorithms
3. Ant colony optimization
4. Simulated annealing

Each technique has strengths depending on the problem characteristics, such as size, constraints, or desired speed of solution. Although solutions from heuristic methods may not always be optimal, they often provide sufficiently good solutions within acceptable time frames, making them valuable in real-world decision making processes.

Combinatorial optimization is widely applied in areas such as communication networks, project management, scheduling, and inventory control. It plays a crucial role where decision variables are inherently discrete and must be carefully arranged or selected from a larger set.

2.2.5 Stochastic Optimization

Stochastic optimization deals with problems that involve uncertainty, where some parameters or outcomes are not known in advance but instead follow probability distributions. It provides solutions that remain effective even when conditions vary or are partially unpredictable.

A typical stochastic optimization problem takes the form:

Minimize or Maximise: $E[f(x, \xi)]$

Subject to: constraints involving uncertain variables ξ

Here, E represents the expected value and ξ stands for random variables. The solution aims to optimize the average outcome across all possible realisations of the uncertain data.

This type of optimization is especially useful in contexts such as:

1. Financial planning with uncertain returns
2. Supply chain management under demand fluctuation
3. Inventory control with unpredictable restocking times
4. Weather-influenced planning such as crop or energy production

Methods used to solve stochastic optimization problems include Monte Carlo simulation, scenario analysis, stochastic gradient methods, and robust optimization. These techniques attempt to factor in all known probabilities and variability to propose a solution that can adapt or perform well across multiple possible future scenarios.

Stochastic optimization acknowledges the limitations of deterministic models by integrating uncertainty into decision making. It is vital in any environment where variability cannot be ignored and planning needs to be resilient rather than idealised.

2.3 Optimization in Logistics

Optimization in logistics involves applying mathematical, computational or statistical techniques to determine the most efficient and cost effective way to manage various activities across supply chains and distribution networks. The concept has gained significant attention in recent years due to the increasing complexities of logistics operations and the growing need to improve performance and reduce costs in competitive environments. Logistics tasks such as vehicle routing, inventory control, warehouse management and resource allocation require deliberate strategies that can minimize costs while meeting customer demands effectively.

Businesses operating within dynamic markets have begun to rely more heavily on optimization models to improve the flow of goods and services. Research by Medoh and Telukdarie (2020) indicates that leveraging industry 4.0 technologies within logistics processes not only streamlines operations but also enables the creation of real-time adaptable strategies. These technologies help track variables such as delivery schedules, fuel usage and capacity constraints, which are then used in algorithms to provide solutions that enhance operational output.

The application of simulation and artificial intelligence has been another valuable addition, allowing decision makers to run predictive scenarios and adopt solutions that adapt to changing logistics landscapes (Kidiyur *et al.*, 2024).

Another dimension to logistics optimization is its environmental relevance. As logistics operations contribute to emissions and resource consumption, optimized planning can reduce carbon footprints and align logistics activities with sustainability objectives (Dobers *et al.*, 2013).

Emerging trends show a growing dependence on advanced computational models. The introduction of AI driven tools, as shown by Grace and Thenmozhi (2023), presents logistics managers with real-time data processing capabilities that improve decision-making speed and accuracy. The shift towards evidence based optimization, especially in humanitarian and urban delivery settings, continues to grow, ensuring that logistics practices remain adaptable and efficient under varied conditions (De Vries, 2017).

2.4 Optimization Techniques

2.4.1 Exact Methods

Exact methods are deterministic algorithms designed to find the optimal solution to an optimization problem with full certainty, assuming sufficient time and computational power. These methods solve problems by precisely exploring the solution space, either exhaustively or systematically through logical reductions. They are mostly used where the optimal solution is strictly required and problem size is manageable.

2.4.1.1 Branch and Bound Method

This algorithm solves discrete and combinatorial optimization problems, especially integer programming. It works by recursively dividing the problem into smaller subproblems (branching) and computing bounds on the best possible solution within these subproblems (bounding). Subspaces that cannot yield a better solution than the current best are discarded (pruning).

A general integer programming problem is given by:

$$Maximise\ Z = c^T x$$

$$Subject\ to\ Ax \le b,\ x \in \mathbb{Z}_+^n$$

i. Branching divides the feasible region by introducing constraints (e.g., $x_i \le |k|$ and $x_i \ge |k|$).

ii. Bounding solves the relaxed problem (e.g., allowing x to be continuous), to estimate the best possible outcome in that region.

iii. Pruning occurs when a subproblem's bound is worse than the best integer solution found so far.

2.4.1.2 Dynamic Programming (DP)

Dynamic Programming is a recursive optimization technique used for problems that exhibit optimal substructure and overlapping subproblems. It solves complex problems by breaking them down into simpler stages, solving each subproblem only once, and storing its solution to avoid recomputation.

Mathematical Function:

DP relies on the Bellman equation, which defines the value of a decision problem at a certain state as the minimum (or maximum) of immediate cost plus the value of subsequent decisions:

$$V(s) = \min_{\alpha \in A(s)} \{c(s, a) + V(f(s, a))\}$$

Where:

i. $V(s)$ is the value function at state s

ii. $A(s)$ is the set of actions available in state s

iii. $c(s, a)$ is the cost of taking action a

iv. $f(s, a)$ is the transition function to a new state

This recursive relationship is computed via forward or backward induction, depending on the problem structure.

2.4.1.3 Cutting Plane Method

The Cutting Plane Method is an iterative refinement technique used in integer and convex optimization. It begins with a relaxed version of the problem and incrementally adds constraints (cuts) that eliminate infeasible or suboptimal regions without removing any feasible integer points.

Mathematical Function:

Start with a relaxed linear model:

$$Minimise\ c^T x Subject\ to\ Ax \leq b,\ x \in \mathbb{Z}^n$$

When the solution x^* is fractional and infeasible for the integer version, a cut is generated:

$$\alpha^T x \leq \beta$$

Where:

i. $\alpha^T x \leq \beta$ is a valid inequality that excludes x^* but preserves all feasible integer points

ii. This new constraint is added, and the updated LP is re-solved

The process continues until the optimal integer solution is found or no further improvements are possible.

2.4.1.4 Simplex Method

The Simplex Method is a vertex-search algorithm used to solve linear programming problems. It moves along the edges of the feasible region defined by linear constraints to reach the optimal solution at a vertex of the polytope.

Mathematical Function:

Given a linear program in standard form:

$$Minimise\ Z = c^T x$$

$$Subject\ to\ Ax = b,\ x \geq 0$$

Where:

i. $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, c \in \mathbb{R}^n,$

ii. $x \in \mathbb{R}^n$ are decision variables

The Simplex algorithm operates by:

i. Starting at a basic feasible solution (a vertex of the feasible region)

ii. Identifying a non basic variable with the most potential to increase Z

iii. Performing a pivot operation to move to an adjacent vertex with an improved objective value

This process repeats until no further improvement is possible, indicating that the current solution is optimal.

2.4.2 Heuristic Methods

Heuristic methods do not guarantee optimality but aim for good solutions quickly. They use problem specific rules or logic to build or improve feasible solutions and are effective when exact methods are computationally infeasible.

2.4.2.1 Greedy Method

The Greedy Method selects the locally optimal choice at each step with the hope that these local choices lead to a globally optimal solution. It does not reconsider previous decisions once made, making it efficient but sometimes suboptimal.

Let:

i. x_k be the current partial solution

ii. $N(x_k)$ be the neighbourhood of feasible extensions to x_k,

iii. $f(x)$ be the objective function to minimize or Maximise

The update rule is:

$$x_{k+1} = arg_{x \in N(x_k)}^{min} f(x)$$

Or in the case of maximization

$$x_{k+1} = arg_{x \in N(x_k)}^{max} f(x)$$

The greedy method continues this process until a complete feasible solution is constructed. The quality of the final solution heavily depends on the structure of the problem and the nature of $f(x)$.

2.4.2.2 Nearest Neighbour Heuristic

The Nearest Neighbour Heuristic is most commonly applied to sequencing or routing problems. The method starts from an initial node and repeatedly chooses the nearest unvisited node based on a cost or distance matrix until a complete sequence or tour is formed.

Let:

i. $D(i, j)$ denotes the distance or cost between node i and node j

ii. S be the set of unvisited nodes

iii. x_k be the current node at step k

The next node x_{k+1} is chosen as

$$x_{k+1} = arg_{j \in S}^{min} D(i, j)$$

This approach quickly builds a solution by always selecting the nearest valid option. Though simple and efficient, it is myopic and may not yield globally optimal results.

2.4.2.3 Constructive Heuristic

The Constructive Heuristic is a method that builds a complete solution incrementally by adding one element at a time. Each addition is based on a decision rule that considers the quality of including that component in the partial solution. The process continues until all required components are included and a feasible solution is constructed.

Let $S = \{s_1, s_2, ..., s_n\}$ be the set of elements available for selection, and let x represent the final solution. The method builds x as follows:

$$x = \bigcup_{i=1}^{n} g(s_i)$$

i. Where $g_i(s_i)$ is the rule-based function or operation applied to element s_1

ii. g_i determines if and how s_i is added to the partial solution

Constructive heuristics typically employ scoring functions to decide the order of inclusion, such as selecting elements with the highest benefit, lowest cost, or best ratio of benefit to cost.

2.4.3 Metaheuristic Methods

Metaheuristics are high hevel strategies designed to explore large and complex solution spaces. They balance exploration (searching new areas) and exploitation (refining known good areas) to find near-optimal solutions. These methods often incorporate randomness and memory

2.4.3.1 Genetic Algorithms (GA)

Genetic Algorithms are population based metaheuristics inspired by the process of natural selection. They operate on a population of candidate solutions, called chromosomes, and evolve them over generations using selection, crossover, and mutation operations.

Each candidate solution $x \in \{0, 1\}^n$ is evaluated by a fitness function $f(x)$. Selection is based on fitness proportion:

$$P(x_i) = \frac{f(x_i)}{\sum_{j=1}^{n} f(x_j)}$$

Crossover operation:

$$x^{child} = \alpha x^1 + (1 - \alpha)x^2, \quad \alpha \in |0, 1|$$

Mutation operation:

$$x_j^{mut} = x_j \oplus \delta, \delta \sim Bernoulli\ (p)$$

Where $\oplus$ denotes bit-flip or real value perturbation depending on encoding, and p is the mutation probability.

2.4.3.2 Simulated Annealing (SA)

Simulated Annealing is a probabilistic technique inspired by the annealing process in metallurgy. It allows occasional acceptance of worse solutions to escape local optima, with decreasing probability over time.

Let $f(x)$ be the objective function to minimise, and let $\Delta f = f(x_{new}) - f(x_{current})$

The probability of accepting a worse solution is:

$$P = \exp\left(\frac{\Delta \mathrm{f}}{T}\right)$$

Where:

T is the temperature, reduced over time using a cooling schedule (e.g., $T_{k+1} = \alpha T_k$ with $\alpha \in (0, 1)$)

This mechanism helps the algorithm explore a wider solution space early on and converge as the temperature decreases

2.4.3.3 Ant Colony Optimization (ACO)

ACO is inspired by the foraging behaviour of ants, which deposit pheromones to guide others toward favourable paths. Candidate solutions are built based on a probabilistic model that favours previously successful decisions.

The probability of choosing the next path or component (i, j) is given by:

$$P(i, j) = \exp\left(\frac{\tau(i, j)^{\alpha}.\eta(i, j)^{\beta}}{\sum_{k\epsilon N_i} \tau(i, k)^{\alpha}.\eta(i, k)^{\beta}}\right)$$

Where:

i. $\tau(i, j)$ is the pheromone level on path (i, j)

ii. $\eta(i, j)$ is the heuristic desirability (e.g., inverse of cost)

iii. α, β are control parameters

iv. N_i is the set of feasible moves from node i

Pheromone update rule:

$$\tau(i, j) = (1 - \rho)\tau(i, j) + \Delta\tau(i, j)$$

Where:

i. $\Delta\tau(i, j)$ is the amount of pheromone deposited, often proportional to solution quality

ii. ρ is the evaporation rate

2.4.3.4 Particle Swarm Optimization (PSO)

The social behaviour of birds or fish schools inspires PSO. Each candidate solution, called a particle, adjusts its position in the solution space based on its own best position and the best known position of the entire swarm.

Let

i. $x_i(t)$ be the position of particle i at time t

ii. $x_i(t)$ be the velocity of particle i

iii. $pbest_i$ be the particle's best position

iv. $gbest_i$ be the global best position

Then, update rules are:

Velocity update:

$$v_i(t+1) = wv_i(t) + c_1r_1\big(pbest_i - x_i(t)\big) + c_2r_2\big(pbest_i - x_i(t)\big)$$

Position update:

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

Where:

i. w is inertia weight

ii. c_1, c_2 are learning rate

iii. $r_1, r_2 \sim |0, 1|$ are random values

This balance between exploration and exploitation helps PSO find high quality solutions efficiently.

2.5 Branch and Bound Technique

The Branch and Bound (BandB) technique stands as one of the most robust and widely applied exact algorithms for solving discrete, particularly combinatorial and integer, optimization problems. It provides a systematic approach to exhaustively explore the search space of feasible solutions without resorting to brute force. By incorporating mechanisms to eliminate regions that cannot contain optimal solutions, the algorithm combines mathematical rigour with computational efficiency. The technique is grounded in the principle of intelligent partitioning of the solution space (branching), coupled with theoretical bounds (bounding) that guide and constrain the search process.

At its core, Branch and Bound aims to optimize a given objective function subject to a set of constraints, while ensuring that solution variables conform to integrality. It is generally applied to problems of the form:

$$\text{Minimise or Maximise } f(x) \quad \text{subject to } x \in S \subseteq \mathbb{Z}^n$$

Here, *f(x)* represents the objective function to be optimized, and *S* denotes the feasible region, which may be defined by a system of linear or nonlinear constraints. Since the domain Z^n enforces integrality, the solution space becomes discrete and potentially exponential in size, making exhaustive enumeration impractical for non trivial problems.

To manage this complexity, the technique operates by recursively decomposing the original problem into smaller, mutually exclusive subproblems. This is the branching phase. Each subproblem defines a restricted region of the original feasible set. Rather than solving each subproblem in full, the algorithm computes an upper or lower bound (depending on the nature of the problem) on the optimal solution within each subregion. This constitutes the bounding phase. If the bound for a subproblem indicates that it cannot outperform the best known feasible (and integral) solution found so far, that subproblem is pruned and excluded from further exploration. This recursive elimination of non promising regions significantly reduces the computational effort.

In mathematical terms, let P be the original problem and P_1, P_2, ..., P_k be subproblems generated through branching. For each P_i, the algorithm solves a relaxed version typically by removing the integrality constraint. The solution x_i^* to this relaxed problem provides a bound $B_i = f(x_i^*)$ on the best possible objective value for P_i. If x_i^* happens to be integral and better than the current best-known integer solution $f(\bar{x})$, the incumbent $\bar{x}$ is updated. If x_i^* is non integral but B_i is worse than $f(\bar{x})$, the subproblem is pruned.

Branching is often performed by selecting a variable x_j whose value in the LP relaxation x_j^* is fractional, and creating two new constraints: $x_j \leq \lfloor x_j^* \rfloor$ and $x_j \geq \lceil x_j^* \rceil$. These constraints define two child nodes in the branch and bound tree, each representing a smaller feasible region. This process is applied recursively until all branches are either pruned or solved.

Bounding is crucial because it governs the effectiveness of pruning. The more accurate the bound, the earlier the algorithm can eliminate non promising branches. Bounding typically relies on solving LP relaxations, where the integrality constraints are relaxed. In the case of a minimisation problem, if the LP relaxation of a subproblem returns a solution $z_{LP} \geq z^*$, where z^* is the current best known integer solution, then the subproblem cannot contain a better solution and is discarded

The overall procedure follows a tree based structure, starting from the root node representing the original problem. As subproblems are generated and bounded, the algorithm maintains a list of active nodes (subproblems yet to be processed). Different node selection strategies can be used, such as depth first search, breadth-first search, or best-first search (selecting the node with the most promising bound).

One of the major strengths of the Branch and Bound technique lies in its ability to guarantee global optimality. Unlike heuristic or metaheuristic methods, which may converge to suboptimal solutions, BandB exhaustively searches all regions of the feasible space subject to pruning ensuring that no potentially better solution is overlooked. This makes it the algorithm of choice for exact solutions in integer linear programming (ILP), mixed integer programming (MIP), and other discrete problems like the travelling salesman problem (TSP), knapsack problems, and facility location optimization.

Despite its rigour, the algorithm is not without limitations. Its worst case time complexity remains exponential, and its performance is highly sensitive to the quality of bounds and the efficiency of branching rules. The repeated solving of LP relaxations at various nodes in the tree can become computationally intensive, particularly for problems with a large number of variables or complex constraint structures. To mitigate this, Branch and Bound is often integrated with other techniques such as cutting planes or heuristics that quickly produce good feasible solutions to tighten the incumbent bound early in the process.

In practice, modern solvers implement sophisticated variations of Branch and Bound with dynamic node selection strategies, presolving techniques, and dual bounding enhancements to accelerate convergence. When applied judiciously, the algorithm remains one of the most powerful tools available for solving exact optimization problems where decision variables must satisfy discrete constraints.

2.6 Relevance of Google Maps to Business Delivery

Business delivery processes have transformed in recent years with the growing integration of location based services and mapping technologies. At the centre of these innovations lies Google Maps, which has evolved from a simple web based map to a robust platform supporting numerous business needs through its suite of tools and Application Programming Interfaces (APIs). Google Maps has become central to optimizing delivery services, improving logistics accuracy, reducing operational costs and increasing customer satisfaction through real time navigation and data driven route selection (Fu *et al*., 2010).

2.6.1 Overview of Google Maps Tools and APIs

Google Maps provides several tools and APIs that support business operations. These include the Maps JavaScript API, Directions API, Distance Matrix API and Roads API. These tools allow businesses to embed customisable maps into web and mobile applications, retrieve real world geographic data, calculate driving distances and times, and visualize routes between locations (Krishnan and Gonzalez, 2015). Google Maps also includes Street View, Geocoding and Places APIs which help in identifying exact delivery locations, validating addresses and providing contextual information about delivery points.

Businesses can create interactive web applications that show delivery zones, dispatch vehicles and monitor the real time position of drivers using these APIs. As noted by Huang (2010), the Maps JavaScript API allows developers to customise overlays, set controls and embed geographical data into delivery platforms, helping teams coordinate across different locations effectively.

In logistics, embedding Google Maps into backend systems makes it easier for dispatchers to assign orders, track drivers and reduce delivery delays. The ability to visualise data directly on the map interface is what makes Google Maps extremely user-friendly and powerful for logistics applications (Bin-shuang, 2010).

2.6.2 Distance Matrix API and Delivery Efficiency

The Distance Matrix API is one of the most essential tools in delivery operations because it provides accurate travel distances and estimated time between multiple origins and destinations. This API enables logistics companies to calculate delivery windows, plan order batching, and set realistic arrival times based on real time traffic data and road conditions.

A logistics system built using the Distance Matrix API can evaluate routes to determine the shortest or fastest paths depending on business goals. This means that if a company wants to reduce petrol consumption, the shortest distance is used. If customer satisfaction through punctuality is the target, then the fastest route with live traffic data is preferred. Research shows that the Distance Matrix API helps delivery managers save time during order planning and reduces delivery costs by optimizing travel paths (Hamiz *et al*., 2018).

2.6.3 Route Visualization and Navigation

The visualization of delivery routes using the Google Maps platform plays a key role in enhancing driver performance and delivery speed. The Directions API allows businesses to generate multi stop directions, avoid toll roads, and adjust delivery paths in real time based on traffic incidents and road closure.

Google Maps supports turn by turn navigation, which drivers can follow through mobile apps. This has reduced delivery times as drivers no longer rely on manual routing or static maps. The ability to recalculate a route instantly when a driver misses a turn means that the risk of delays is greatly reduced. Fu *et al*. (2010) highlighted that combining WebGIS with the

Google Maps API allowed for advanced route planning, route marking, and historical tracking of previous delivery trips.

2.6.4 Benefits of Using Google Maps in Delivery Optimization

One of the most notable benefits of Google Maps in delivery services is route efficiency. Businesses that use the platform report significant cost savings in terms of fuel consumption, shorter delivery windows and fewer failed deliveries. Iliev *et al*. (2024) revealed how location accuracy from the Directions API plays a huge role in meeting delivery targets, especially in regions with weak infrastructure like rural parts of Bulgaria.

Hamiz *et al*. (2018) demonstrated that combining the Distance Matrix API with a Saving Matrix method in route planning allowed businesses to find the most efficient routes across all their delivery zones. This meant that the time spent in traffic was reduced and the number of deliveries per driver was increased within a shift.

Another benefit is better customer experience. Real time tracking of drivers, accurate ETAs and automated status updates make the entire delivery journey more transparent for customers. Bin-shuang (2010) noted that logistics platforms that use Google Maps to monitor vehicles and playback past routes create more reliability and improve customer trust.

Google Maps also helps businesses identify bottlenecks in delivery operations. Companies can analyse which areas frequently cause delays or where vehicles spend the most time parked. These insights allow managers to redesign delivery zones and allocate resources in a more informed way.

2.7 Factors Affecting Delivery Route Planning

Efficient delivery route planning has become an essential part of modern business logistics. The use of location based services like Google Maps has reshaped how businesses optimize delivery times, reduce costs, and enhance customer satisfaction. With the growing reliance on real time navigation and traffic data, understanding the various factors affecting delivery route planning becomes necessary for improving operational efficiency.

2.7.1 Traffic Conditions and Real Time Data

Traffic congestion is one of the leading issues in delivery planning. Heavy traffic slows down operations and reduces the number of deliveries that can be completed per day. Google Maps offers real-time traffic updates using location data from users and road sensors. According to Wu *et al*. (2022), incorporating real-time traffic information into logistics decision-making improves delivery speed and accuracy by helping drivers avoid congested routes. This was also confirmed in a multi criteria decision analysis that used context based networks for route selection and showed that traffic related variables played a major role in the quality of route planning outcomes.

The accuracy of Google Maps in predicting real-time travel times can vary depending on the region. In a case study from Bulgaria, it was observed that inaccuracies in geolocation data significantly impacted delivery schedules, especially in less urbanised areas (Iliev *et al*., 2024). This showed the importance of local context in evaluating routing platforms.

2.7.2 Route Distance and Travel Time Trade offs

While distance has traditionally been the main factor in choosing routes, time-based planning is now being integrated to balance efficiency and cost. A study by Alvarez *et al*. (2024) found that combining time and distance as a joint cost function produced better outcomes for freight delivery compared to using either factor alone. Their findings highlighted that in urban areas, time is often a more critical constraint due to frequent stop and go conditions.

2.7.3 Delivery Time Windows and Customer Expectations

Businesses often commit to specific delivery windows to improve service. These time constraints limit the flexibility of route choices and require careful planning. Liu *et al.* (2019) introduced a time window model that considered road closures and customer preferences for delivery timeframes. Their bi objective optimization model was effective in reducing delivery delays while managing risk, especially in hazardous material transport. Customer satisfaction is strongly linked to time precision in deliveries. Bostan *et al*., (2022) showed that route optimization systems that integrate time slot selection and live tracking features result in higher delivery efficiency and improved customer trust.

2.7.4 Road Restrictions and Legal Constraints

Certain urban roads impose restrictions on delivery vehicles either due to size, time of day, or local regulations. These constraints play a big role in route planning. The integration of up to date regulation data into routing platforms like Google Maps is still developing. Iliev *et al*. (2024) highlighted that government policies on delivery hours, environmental zones, and construction zones must be factored into routing algorithms for meaningful optimization

2.7.5 Fuel Consumption and Environmental Factors

Delivery businesses aim to reduce operational costs, where fuel expenses are significant. Efficient route planning helps in cutting fuel use. Alves *et al*., (2021) evaluated a system that integrates Google Maps APIs with cloud applications to reduce fuel usage. Their approach to balancing workloads and minimizing trip lengths helped to cut down both emissions and operational costs.

2.7.6 Labour Shortages and Human Decision making

The shortage of skilled logistics staff places added pressure on route planning systems. In Japan, Masuda, *et al*., (2021) described a real time routing system that can adapt when drivers are unavailable or vehicle capacity needs to be changed on short notice. This has helped meet rising delivery demands even with fewer personnel. The challenge is that while automated systems can optimize paths, human drivers may not always follow them. Preference based routing developed by Shao and Cheng (2023) attempted to resolve this by learning from previous driver choices and adjusting automated plans to align with real-life decisions.

2.7.7 Multi-Point Deliveries and Vehicle Load Capacity

Deliveries often involve multiple stops with different items and time sensitivities. A study by Adewara (2016) reviewed various GIS based routing tools and found that those supporting multi-point routing significantly reduced route errors. Google Maps, with proper configuration, ranked among the better tools when integrated with business systems that could assign routes based on capacity and distance. The use of open-source solvers combined with Google Distance Matrix APIs, as shown in Alves et al. (2021), provided powerful route allocation systems that balanced cargo loads across fleets, avoiding overloads or underused vehicles.

2.7.8 Urban Planning and Infrastructure Layout

The layout of roads, intersections, depot locations, and customer density affects route efficiency. GIS based systems were found useful in visualising these layouts. Ikai *et al*., (2021) added that understanding road networks through simulations can support better route decisions by identifying frequently used paths and adapting them to changes like road closures or urban construction.

2.8 The Need for Optimizing Delivery Routes

Delivery route optimization plays a critical role in ensuring that businesses remain efficient, competitive and profitable in the modern logistics environment. With an ever increasing volume of goods being moved from one location to another due to the expansion of ecommerce and customer expectations for faster services, businesses must improve the way they plan and execute delivery routes. Poorly optimized routes do not just delay deliveries, they also increase fuel consumption, raise operational costs and reduce customer satisfaction, making it harder for companies to scale and survive in competitive markets (Álvarez *et al*., 2018).

Recent studies show that businesses adopting route optimization systems using tools like Google Maps have recorded significant gains in delivery speed and cost reduction. One study involving a dynamic route planning model in Sri Lanka showed that using Google Maps as the central navigation tool led to faster route calculation, reduced human errors and increased delivery accuracy by helping planners visualise road conditions in real time (Dayaratne and Gunasekara, 2020). Businesses also benefit from real-time route updates, where traffic conditions are dynamically considered before drivers begin or continue their routes, allowing for timely rerouting and saving both time and fuel (Álvarez *et al*., 2018)

Another key consideration is the financial implications of route inefficiency. Small businesses, especially those in developing countries, often lack the capital to run large fleets and therefore must depend on smart planning tools to stay profitable. In a study that focused on Thai SMEs, Horng and Yenradee (2023) demonstrated that integrating Google Maps with delivery planning systems improved scheduling, arrival time prediction and route sharing, allowing businesses to Maximise the use of their limited fleet resources. The study recorded higher levels of accuracy in delivery estimates and significant reductions in fuel costs through better planning

Operationally, optimizing delivery routes also lessens the physical burden on drivers who often face long working hours due to suboptimal scheduling. When routes are shorter and more accurate, drivers complete their tasks quicker and experience less fatigue. This ultimately contributes to better employee wellbeing and improved service delivery. In Nigeria, a mobile route optimization application built using Google SDKs and travelling salesman algorithms helped cut logistics costs and reduced time on the road, thus boosting worker productivity and service quality.

Time savings are another core reason businesses cannot ignore the optimization of delivery routes. When deliveries take longer than planned due to traffic congestion or unnecessary detours, customer satisfaction declines. Multiple studies have shown that using Google Maps to factor in real time congestion leads to better delivery timing. In a case study conducted in

Spain, using dynamic traffic information reduced delivery time by up to 11 percent compared to static route plans (Álvarez *et al*., 2018)

Cost control is equally important. The cost of logistics, which includes vehicle maintenance, fuel, salaries and planning, takes a huge chunk of operational expenses in distribution-based businesses. By implementing optimization strategies such as genetic algorithms with Google Maps data, businesses can better allocate their resources and cut unnecessary spending. In a recent study, Hintoro *et al*. (2023) noted that combining genetic algorithms with Google Maps for route planning helped companies reduce costs and increased decision makers' confidence in delivery management

Safety and security of routes is also a concern. In areas with high crime rates or unsafe infrastructure, businesses face the challenge of ensuring that both goods and personnel are safe during deliveries. A study aimed at predicting secure and safe routes for women found that Google Maps, when paired with contextual data, can provide insights into safer paths that still meet efficiency requirements (Bura *et a*l., 2019). This insight can be extended to general logistics, ensuring that drivers avoid known trouble spots while staying on the most efficient path

Technology also allows for better handling of bulk deliveries. When a business has to deliver items to multiple destinations, traditional methods often fail due to complexity. The application of artificial intelligence such as ant colony optimization for multiple destinations showed that Google Maps could adapt and improve route efficiency even when destinations are added or changed during the journey (Ismail *et al*., 2022). This is critical in last mile delivery operations where delivery addresses are numerous and scattered

With the increase in on demand delivery services and the pressure to meet tight delivery windows, especially in urban areas, the demand for smart routing solutions has grown. The integration of real time learning methods like reinforcement learning for route planning has allowed delivery systems to constantly learn from past data and improve their routing decisions. Mishra and Tiwari (2025) demonstrated that reinforcement learning models outperformed traditional static algorithms, enabling systems to make better decisions when handling new traffic patterns, weather conditions or unexpected delays

Every delay, detour or wrong turn taken due to poor routing adds up to significant losses at scale. Hence, effective route optimization using reliable tools like Google Maps and smart algorithms is no longer optional but a necessity for any business that wants to stay productive and profitable in a logistics-heavy economy

2.9 Review of Related Literature

Alves, *et al*., (2021) focused on improving logistics operations through a route planning system that integrates open-source optimization tools and Google Maps services. The aim of the study was to address real-time route efficiency in vehicle routing problems (VRP) by developing a cloud-based platform that reduces cost and improves route precision. Their method involved implementing a VRP mathematical model embedded within Google OR Tools and using Google Maps and Google Distance Matrix API for route and distance estimation. The model aimed to minimise the maximum route distance or time while ensuring balance in delivery workload. The authors simulated their tool over a logistics dataset from Porto district in Portugal and found that the proposed system could generate feasible solutions with reduced computational overhead and greater usability for training logistics operators. Experimental results showed that the integration of cloud computing with real-time map APIs allowed the tool to successfully compute balanced and optimized routes across varied scenarios with route time and distance savings of up to 18 percent in some trials. The authors recommended that logistics providers adopt open source solvers like Google OR Tools for scalable and cost-efficient delivery optimization systems

Hamiz, *et al*., (2018) examined how delivery routes could be made more efficient using the Saving Matrix method integrated with Google Maps API to assist companies in calculating and visualising optimal delivery paths. Their aim was to minimise delivery cost and distance by calculating the best possible combination of delivery routes between a company's warehouse and customer locations. The study introduced a web-based interface where users input warehouse and customer locations and vehicle parameters, and the system then calculates efficient paths using Saving Matrix logic. The Google Maps API was used to compute distance between locations in real time. The study reported improved delivery planning and a notable reduction in total travel distance, although no exact numerical value for the savings was reported. It allowed companies to optimize logistics operations without relying on expensive software packages. The authors advised organisations to use Google Maps API in combination with simple algorithmic tools like the Saving Matrix method to create customisable and cost-effective routing systems suitable for real-world business logistics.

Iliev *et al*., (2024) investigated the precision challenges of geolocation tools, specifically Google Maps Directions API, in supporting logistics delivery operations in Bulgaria. The main aim was to assess how inaccuracies in map data could negatively impact delivery scheduling, routing and final-mile operations. The study applied a case based research method by analysing actual delivery data collected from Bulgarian logistics operators. The research examined discrepancies between predicted and actual delivery paths and times. A four part approach was developed which included modules for system integration, error analysis, machine learning for route correction and adaptive feedback mechanisms. Google Maps Directions API served as the primary tool for route estimation while delivery logs were used to assess deviations. The findings showed that geolocation inaccuracies ranged between 9.3 and 14.8 percent in rural delivery paths, and up to 6 percent in urban centres. A key issue was the mapping of new roads and insufficient street level resolution. The authors proposed a modular correction architecture, where machine learning modules retrain on delivery discrepancies to refine map accuracy. They advised that logistics providers develop custom geolocation corrections layered over standard API services to reduce cumulative delivery errors.

Wu, *et al*., (2022) aimed to improve delivery routing in urban areas through the use of context-based social network modelling and multi criteria decision analysis (MCDA) by integrating real-time traffic data with routing decisions. Their work addressed the challenge of delays in urban logistics by proposing a model that considered symmetrical and asymmetrical traffic contexts, using metadata from Google Maps as part of the routing data source. The method involved creating a traffic aware social network where nodes represented road intersections and links represented road segments with attributes like congestion and time sensitivity. Using MCDA, the system ranked route alternatives and presented the most suitable paths to delivery personnel. The system was evaluated through a case study and the results showed a precision rate of 79.65 percent, a recall rate of 80.70 percent and an F1 score of 80.17 percent in route recommendation accuracy. These metrics demonstrated the robustness of the system in real life applications. The authors recommended incorporating social and contextual information from traffic databases into routing tools to significantly enhance delivery performance and customer satisfaction in urban logistics scenarios.

Dayaratne and Gunasekara (2020) developed a dynamic route planning algorithm to reduce manual labour and inefficiencies in Sri Lanka's logistics industry. The aim was to automate delivery route calculations using a module that considers cargo volume, vehicle capacity and route distance. Their method relied on Google Maps API, integrating Dijkstra's Algorithm to estimate the fastest route between delivery points. The tool enabled users to input delivery locations and cargo details, then generate optimized routes either visually or in report form. Their findings confirmed that the DRP module improved delivery speed and vehicle usage, though limitations were noted due to constraints in free access to Google Maps and road grading classifications. They recommended wider adoption of such automated systems and encouraged using global map services with broader coverage and frequent updates.

Álvarez, *et al*., (2024) focused their study on balancing the trade off between time and distance in freight delivery routing. Their study aimed to optimize urban freight delivery routes by integrating real time traffic data from the Google Maps Distance Matrix API with an Adaptive Large Neighbourhood Search (ALNS) algorithm. Using Python and simulated annealing, the model was tested in Bogotá with real delivery data. The study achieved a 17.8 percent reduction in route time, a 16.5 percent cut in distance travelled, and a 12 percent drop in fuel consumption. Findings showed improved punctuality and operational efficiency. The authors recommended adopting real-time APIs and adaptive algorithms like ALNS in urban logistics and encouraged partnerships to support data-driven route planning systems for sustainable city transport.

Holland, *et al*., (2017) examined the transformation of UPS's delivery system through the development and deployment of the ORION system, aimed at optimizing daily delivery routes. Their objective was to create an efficient system that balanced delivery consistency with dynamic optimization. They adopted a metaheuristic optimization model embedded in a suite of systems named ORION On Road Integrated Optimization and Navigation which incorporated driver data, delivery constraints, and GPS based real-time mapping. The study revealed that ORION generated daily optimized routes for over 55,000 UPS drivers and saved the company between $300 million and $400 million annually. It also cut down carbon emissions by 100,000 tonnes each year, showing strong environmental benefits. UPS invested over $295 million in building and deploying this system. The researchers recommended that logistics firms integrate route planning systems that blend artificial intelligence and spatial mapping data, as such systems can enhance efficiency, reduce operational costs and support sustainability objectives

Verbytskyi (2023) aimed to enhance logistics efficiency by optimizing delivery routes through machine learning techniques. The study reviewed various modern algorithms, including reinforcement learning, metaheuristics, and supervised learning models, emphasizing their application to vehicle routing problems. Verbytskyi considered key delivery constraints like vehicle capacity, delivery time windows, road networks, and customer demand variability. The research introduced optimization objectives including reducing travel time and fuel consumption, while improving customer satisfaction. Methodologically, the study used comparative analysis of algorithmic efficiency, applying tools such as multi agent systems and hybrid models. Findings suggested that hybrid and adaptive algorithms perform best under complex logistics scenarios due to their flexibility and real-time decision-making capacity. Recommendations highlighted the need for further development of multi agent and swarm intelligence systems, which could better respond to dynamic delivery constraints in real time. This research offers valuable insights into the practical application of AI to logistics and proposes scalable solutions for industry adoption.

Bin-shuang (2010) developed a logistics monitoring and management information system by integrating the Google Maps API into a web-based platform. The aim was to improve logistics visibility through real-time vehicle tracking and route Optimization. The study used the AJAX web development framework along with the Google Maps API to embed interactive maps and implement features like vehicle location tracking, trajectory playback and optimal route mapping. The system provided tools for monitoring delivery vehicle movements and analysing their past travel paths. Results demonstrated improved communication between logistics managers and field vehicles, with quicker decision-making and higher service quality. The system design also enabled faster development cycles and better customer engagement through visual tracking. The author recommended adopting WebGIS based logistics systems that use scalable APIs like Google Maps to enable cost-efficient, transparent and automated delivery monitoring process

Malhotra, *et al* (2019) developed a route optimization Android application leveraging server-client architecture and Google APIs to address inefficiencies in bulk delivery logistics. The app tackles multi-travelling salesman problems using Google Maps for real-time geolocation and route data. It accepts clusters of delivery locations and returns optimized individual routes to minimize time, distance, and cost. The architecture facilitates communication between clients and a central server that computes route plans using Google's Distance Matrix and Directions API. The application achieved significant improvements in delivery scheduling and reduced failure rates in on-time deliveries. The study recommended integrating such mobile solutions into logistics workflows for dynamic and efficient route management.

Nadhila, *et al* (2024) aimed to optimize courier placement efficiency by clustering MSME and warehouse locations using geographic and demand data. Employing the Scrum methodology for app development and integrating the Google Maps Geocoding API with K-Means clustering, they segmented delivery zones based on sub district level data. The system mapped efficient clusters by analyzing distance, demand frequency, and delivery volume. Their findings revealed improved delivery logistics, operational cost reduction, and the introduction of flat rates that equalized pricing across delivery ranges. The study recommended broader adoption of geo based clustering and standardized pricing for scalable logistics management in dense urban markets.

2.10 Research Gaps

A summary of the key research gaps identified from the reviewed literature is presented in Table 2.1, highlighting areas where previous studies have not adequately addressed delivery route optimisation within urban logistics

Table 2.1: Existing Research Gap

Author (s)	Year	Title of Work	Methodology	Results	Research Gaps
Alves *et al*.	2021	Solving a logistics system for vehicle routing problem using an open source tool	Google OR Tools, Google Distance Matrix API	18% savings in time and distance, better route balancing	Lacked focus on urban business specific case studies like Benin City
Hamiz *et al.*	2018	Saving matrix method for efficient distribution route based on Google Maps API	Saving Matrix method, Google Maps API, web based interface	Improved planning with reduced travel distance	Did not assess solution against a real business current practice
Iliev *et al.*	2024	Challenges in geolocation for logistics delivery	Case study with error analysis and ML corrections using Google Maps API	9.3-14.8% error in rural and up to 6% in urban geolocation accuracy	Focused more on geolocation errors than full route model building
Wu *et al.*	2022	Optimizing the routing of urban logistics by context based social network and multi criteria decision analysis	Social network model with MCDA and Google Maps metadata	79.65% precision, 80.70% recall, 80.17% F1 in route recommendation	Did not test on delivery companies or compare with manual methods
Dayaratne and Gunasekara	2020	Dynamic route planning (DRP) module for distribution business	Google Maps API with Dijkstra's algorithm	Improved delivery speed and vehicle usage	Did not integrate cost comparison metrics or alternative algorithms
Ãlvarez *et al.*	2024	Optimizing freight delivery routes: The time distance dilemma	Google Maps Distance Matrix API and ALNS algorithm with simulated annealing	17.8% time, 16.5% distance, 12% fuel reduction	Did not extend model to localised African city context
Holland *et al.*	2017	UPS optimizes delivery routes	ORION system using metaheuristics and real time mapping	\$300M-\$400M annual savings and 100,000 tonnes CO2 reduction	High cost system not replicable by small businesses

Verbytskyi	2023	Delivery routes optimization using machine learning algorithms	Machine learning models and multi agent systems	Hybrid models outperform others in complex delivery environments	Did not consider mapping limitations in low-data regions
Bin-shuang	2010	Applying Google Maps API in logistics monitoring and management information system	AJAX and Google Maps API integration for real time logistics system	Improved logistics visibility and decision making speed	Limited to monitoring; no route Optimizatio n algorithm was used
Malhotra *et al*	2019	Route optimization application using server client architecture and Google APIs	Android app using Google APIs and server client architecture	Reduced failure rates and improved scheduling efficiency	No integration of urban traffic variability for specific city analysis
Nadhila *et al*	2024	Clusterization of MSME and warehouse locations for efficiency of courier placement	K-means clustering using Google Maps Geocoding API	Operational cost reduction and delivery efficiency	Did not compare model with business as usual delivery data

Researcher’s Computation, 2025

CHAPTER THREE

METHODOLOGY

3.1 Research Design

The study adopted an applied operational research design aimed at addressing a real world logistics optimisation problem. It was structured to determine the most efficient delivery route for viboi Ventures, the Coca-Cola depot in Benin City, which served as the delivery base. A quantitative and computational approach was employed, where the geographical coordinates of ten delivery locations, including the depot, were obtained through Google Maps and converted into a distance matrix. The Branch and Bound algorithm was then applied to the matrix to systematically evaluate route combinations and identify the shortest possible path. This design ensured a logical and data driven framework for minimising total travel distance, time, and delivery cost.

3.2 Area of the Study

The study was carried out in Benin City, the capital of Edo State, Nigeria, which served as the geographical focus for the delivery route optimisation. Benin City is a rapidly expanding urban centre located in the southern part of Nigeria, known for its vibrant commercial activities, dense population, and complex road networks. It lies approximately between latitude 6.34° N and longitude 5.62° E, covering an area of about 1,204 square kilometres. The city experiences significant daily vehicular movement due to its role as a commercial hub, linking the western, eastern, and northern regions of the country.

Benin City was selected because it presents typical urban logistics challenges such as traffic congestion, irregular road patterns, and varied delivery destinations within short to medium distances. These conditions make it suitable for testing optimisation models that aim to improve route efficiency. Within the study area, viboi Ventures, a Coca-Cola depot located in Benin City, was used as the operational base for modelling delivery activities. The depot's distribution network spans several key areas across the city, serving retail outlets, wholesalers, and distribution points.

The delivery points used in the study were strategically chosen across different parts of the city to represent realistic operational conditions. These included both central business areas and suburban zones to reflect varying route complexities. The selection of Benin City as the area of study provided an opportunity to assess how mathematical optimisation techniques like the Branch and Bound algorithm can be applied to enhance logistics performance in an urban Nigerian context.

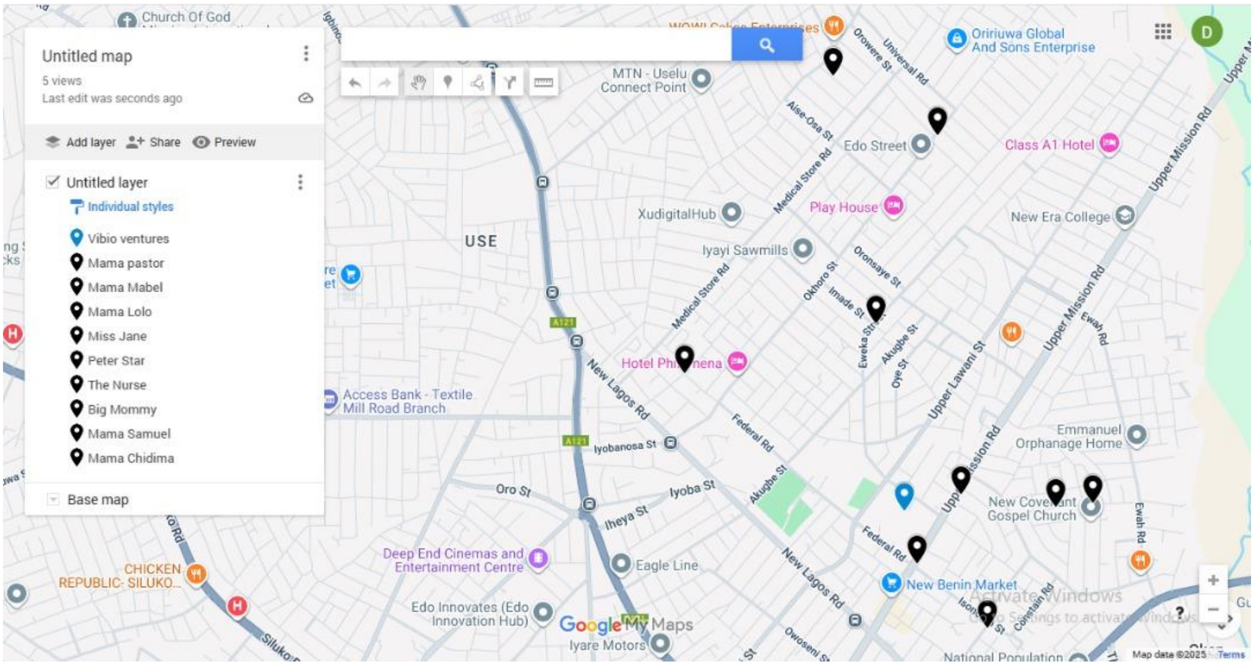


Figure. 3.1: Map of Study Area

The geographical scope of the research is illustrated in Figure 3.1, which shows the location of the study area within Benin City and the distribution of the delivery network used for the optimisation analysis.

3.3 Data Collection Method

The study utilised ten delivery locations within Benin City, comprising the viboi Ventures Coca-Cola depot and nine customer destinations. The geographical coordinates of each location were obtained using Google Maps. These coordinates were used to measure the road distances between every pair of locations, forming a 10 × 10 distance matrix. The matrix represented ninety possible travel routes, excluding movements from a location to itself. Data were recorded in kilometres and served as the basis for constructing and solving the optimisation problem. This structured approach ensured that all feasible delivery paths were captured, allowing the Branch and Bound algorithm to accurately determine the shortest possible delivery route across all ten locations.

3.4 Model Formulation

The model formulation in this study was developed to represent the delivery routing problem faced by viboi Ventures, a Coca-Cola depot in Benin City. The problem was modelled as a Travelling Salesman Problem (TSP), a classical optimisation model used to determine the shortest possible route that visits each destination exactly once and returns to the starting point. The aim of the formulation was to minimise the total travel distance, time, and cost associated with daily delivery operations within the city.

The delivery operation involved ten (10) locations in total — the depot and nine customer destinations. Each of these locations was connected to every other location through measurable routes, creating a network that could be expressed mathematically as a complete weighted graph. The weights on the arcs (or edges) represented the distance between locations in kilometres. These distances were calculated using the geographical coordinates obtained from Google Maps.

Mathematically, the problem was expressed as:

$$Minimise\ Z = \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} C_{ij} X_{ij}$$

Subject to the following constraints:

1. Each location must be visited exactly once:

$$\sum_{j=1,j\neq i}^{n} X_{ij} = 1 \quad \forall i$$

2. Each location must be departed exactly once:

$$\sum_{j=1,j\neq i}^{n} X_{ij} = 1 \quad \forall j$$

3. No location can lead to itself:

$$X_{ii} = 0 \quad \forall j$$

4. Binary decision variable:

$$X_{ij} = \begin{cases} 1, & if\ the\ route\ from\ i\ to\ j\ is\ used, \\ 0, & otherwise, \end{cases}$$

Where:

i. Z = total distance to be minimised

ii. C_{ij} = distance or cost between location iii and j

iii. X_{ij} = decision variable representing whether the route between iii and j is included in the tour

iv. n = total number of locations (10 in this case)

The problem's structure makes it a non linear combinatorial optimisation problem, where the total number of possible route combinations increases factorially with the number of delivery points. For the ten locations in this study, there were (n−1)!/2=181,440 potential route combinations. Solving this problem through exhaustive search would have been computationally expensive; therefore, the Branch and Bound algorithm was employed to systematically explore feasible routes while eliminating non promising paths.

In the model, the depot (viboi Ventures) was defined as the starting and ending node (node 1). The process began with the creation of a distance matrix where each entry (C_{ij}) represented the distance from location i to j. Diagonal entries (C_{ii}) were set to infinity (∞) to prevent self-loops. The algorithm then performed row and column reductions on the matrix by subtracting the minimum value in each row and column, ensuring that at least one zero existed in each row and column. The sum of these reductions formed the initial lower bound (7.2 km), representing the minimum possible route distance before branching.

Subsequently, the branching process divided the problem into smaller subproblems, each representing a possible delivery path (e.g., 1 → 2, 1 → 3, etc.). Each branch generated a reduced matrix, and new lower bounds were computed to evaluate the cost of each partial route. Subproblems with bounds higher than the current best solution were pruned, effectively reducing computation time. Through successive branching, bounding, and pruning, the optimal tour was identified as:

$$1\rightarrow5\rightarrow4\rightarrow6\rightarrow8\rightarrow9\rightarrow2\rightarrow10\rightarrow3\rightarrow7\rightarrow1$$

with a total minimum distance of approximately 12.2 km.

This formulation successfully transformed the physical delivery process into a mathematical model that could be solved efficiently using an exact optimisation algorithm. It demonstrated how structured decision making and algorithmic reasoning can significantly improve logistics performance and resource utilisation in real world delivery systems.

3.5 Algorithmic Procedure (Branch and Bound Implementation)

The Branch and Bound algorithm was applied to determine the most efficient delivery route among ten delivery locations in Benin City, starting and ending at the viboi Ventures depot. The procedure involved a systematic approach that broke the problem into smaller subproblems, applied mathematical reductions, and progressively eliminated non promising paths. This ensured that only routes with the potential to yield the minimum total distance were explored.

Step 1: Construction of the Distance Matrix

The procedure began with the collection of the geographical coordinates of the ten delivery locations using Google Maps. The coordinates were used to calculate the pairwise distances between every two locations, which were then recorded in a 10 × 10 distance matrix. Each element in the matrix represented the distance between a pair of locations, while movements from a location to itself were assigned infinite values to prevent self loops. The matrix served as the foundational data structure for the optimisation process.

Step 2: Row Reduction

Row reduction was performed on the distance matrix to ensure that each row contained at least one zero. This was achieved by identifying the smallest element in each row and subtracting it from all other elements in the same row. This operation standardised the matrix and reduced the range of values to a comparable scale. The process also helped establish the preliminary lower bound for the optimisation problem.

Step 3: Column Reduction

After row reduction, column reduction was carried out to guarantee that each column contained at least one zero. The smallest element in each column was subtracted from all the other elements within that column. This produced a matrix that contained at least one zero in every row and column, creating a balanced structure suitable for branching operations. The total of all reduced values represented the cumulative baseline cost associated with the initial matrix configuration.

Step 4: Branching Process

Branching was the stage where the algorithm explored all possible routes from the starting node to the remaining locations. Each branch represented a possible delivery path originating from the base location to another destination. For each selected branch, the corresponding row and column representing the departure and arrival points were removed from the matrix. Additionally, the reverse path (from the new location back to the starting node) was set to infinity to prevent premature return trips. This ensured that each delivery point could only be visited once during each iteration.

Step 5: Matrix Reduction and Bound Calculation for Each Branch

For every new submatrix created during branching, further row and column reductions were performed to maintain zeros across all rows and columns. The sum of the new reductions was added to the cumulative lower bound to evaluate the cost associated with that particular branch. This process allowed the algorithm to estimate the minimum possible distance that could result from following a specific path, without yet completing the entire route.

Step 6: Bounding and Pruning

The bounding process involved comparing the estimated costs of all branches to determine which routes held the highest potential for optimality. Branches with higher estimated costs were pruned, meaning they were excluded from further consideration. This approach significantly reduced the computational workload by focusing only on paths that could feasibly lead to the shortest overall route.

Step 7: Iterative Expansion of the Search Tree

After pruning, the branch with the smallest estimated cost was selected for further exploration. The algorithm then repeated the branching, reduction, and bounding processes on the new matrix derived from that branch. This recursive procedure continued until all delivery points were included in the route and a complete tour returning to the base location was achieved.

Step 8: Algorithm Termination

The algorithm terminated when all possible branches had been explored or pruned, and a closed route that covered every location exactly once had been established. At this point, the route corresponding to the smallest accumulated lower bound was identified as the optimal delivery sequence.

3.6 Tools Used

The study was conducted entirely through manual computation, supported only by Google Maps for obtaining real world data. All analytical processes, including the construction of the distance matrix, row and column reductions, branching, bounding, and pruning, were performed by hand. This manual approach ensured a clear understanding of every computational step involved in the application of the Branch and Bound algorithm.

Google Maps was used solely for data acquisition and verification. The geographical coordinates of the ten delivery locations within Benin City, including the viboi Ventures depot, were obtained directly from Google Maps. These coordinates were used to manually calculate the distances between all pairs of delivery points, which were then recorded in a distance matrix. Google Maps also served as a visual reference to confirm the spatial relationships between the delivery points, ensuring that the constructed matrix accurately reflected real world geography.

No software tools or automation were employed during the analysis. Every stage of computation was carried out manually, in alignment with the algorithmological framework taught during the course. This approach highlighted the practical applicability of the Branch and Bound algorithm in real world delivery optimisation problems, even without access to specialised computational tools.

3.7 Validation of Model

Validation of the model was conducted manually to confirm the accuracy and practical efficiency of the developed delivery route optimisation framework. After applying the Branch and Bound algorithm manually, the resulting optimised route sequence was cross checked against the baseline route derived from Google Maps.

The initial Google Maps route provided a direct representation of the natural travel order between the ten locations, following real geographical positioning within Benin City. The manually computed route from the Branch and Bound algorithm was then compared with this reference to determine its logical consistency and practical feasibility.

The validation process involved ensuring that the manually derived route maintained a realistic travel sequence, avoided unnecessary backtracking, and reflected an efficient circuit across all delivery points. This comparison confirmed that the Branch and Bound algorithm, when correctly applied, could generate an improved route using only manually derived data.

Through this process, the model was verified as both mathematically sound and geographically valid, demonstrating that manual computation could reliably produce a structured and efficient solution to real world delivery routing challenges.

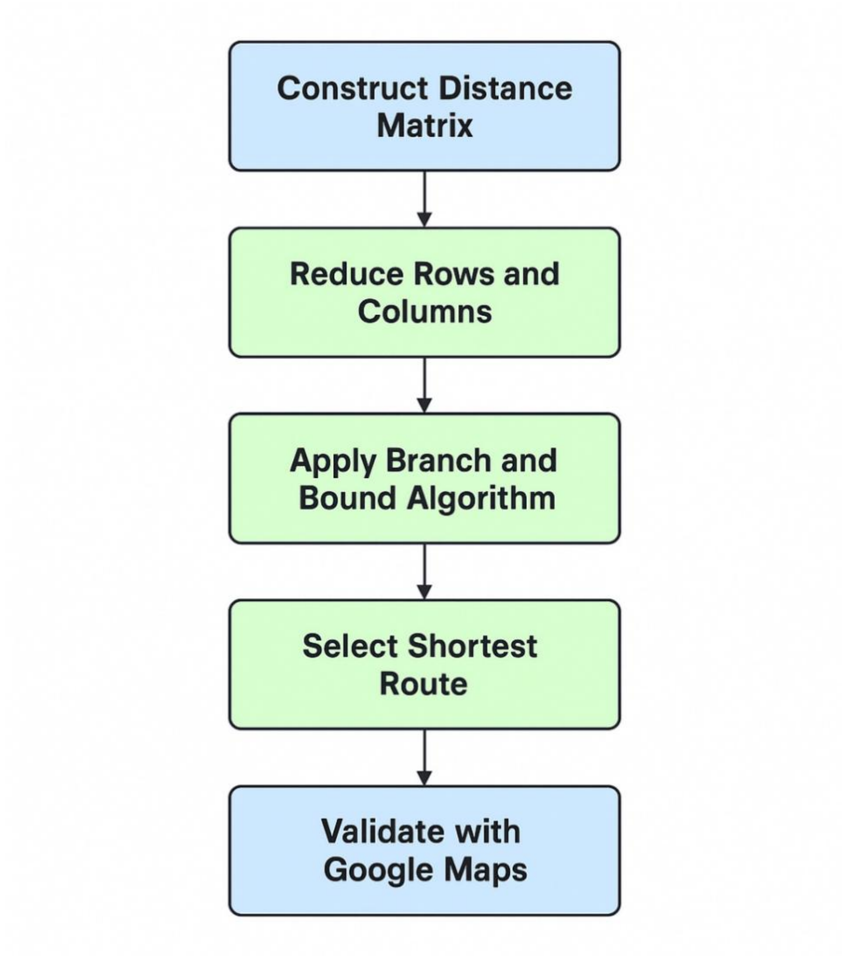


Figure. 3.2: Methodology of the Process

The methodology adopted in this study as shown in figure 3.2 followed a structured manual approach designed to determine the most efficient delivery route for viboi Ventures within Benin City using the Branch and Bound algorithm. The process began with the collection of geographical coordinates for ten locations, including the depot and nine customer destinations, obtained through Google Maps. These coordinates were used to manually compute the pairwise road distances between all locations, which were then arranged into a distance matrix.

The constructed matrix was manually reduced through row and column reduction, ensuring each row and column contained at least one zero. This reduction simplified the dataset and prepared it for the application of the Branch and Bound algorithm, which was executed step by step by hand. The algorithm involved systematically branching possible delivery paths, calculating lower bounds for each, and pruning routes that could not yield an optimal solution. At each iteration, the matrix was updated, and new submatrices were formed to continue the search for the minimum total distance. Once the sequence covering all locations was identified, it was taken as the optimised route. The route was finally cross checked against the geographical layout on Google Maps to confirm its validity and practicality

3.8 Manual Route Computation and Algorithm Based Validation

The optimisation of the delivery route for Viboi Ventures was initially carried out through a manual computational approach. This approach involved systematically analysing the distance matrix derived from Google Maps coordinates for ten locations, including the depot and nine delivery destinations within Benin City. Because the number of delivery points was relatively small, it was possible to evaluate feasible route combinations manually. The manual analysis allowed the researcher to understand the structure of the routing problem and determine a feasible delivery sequence that minimised travel distance based on the available data.

However, manual computation alone may not always guarantee the globally optimal solution, particularly in routing problems that involve numerous possible route permutations. The Travelling Salesman Problem (TSP), which models this routing scenario, grows factorially as the number of locations increases. For n locations, the number of possible route combinations is given by:

$$(n-1)!$$

For the ten locations considered in this study, this results in 362880 possible route permutations because its an asymmetric matrix, making exhaustive manual verification computationally demanding.

To improve the reliability of the results, the Branch and Bound optimisation algorithm was later employed as a validation mechanism. The algorithm was used to verify the manually obtained route by systematically exploring possible route combinations while eliminating paths that could not lead to an optimal solution.

3.8.1 Branch and Bound Algorithm

Branch and Bound is a mathematical optimisation technique commonly used to solve combinatorial optimisation problems such as the Travelling Salesman Problem. The algorithm works by dividing the problem into smaller subproblems (branching) and computing bounds that determine whether a particular branch can produce a better solution than the current best solution.

The algorithm operates through the following key steps:

1. Construction of a distance matrix representing travel distances between all locations.
2. Row and column reductions to generate a cost-reduced matrix.
3. Calculation of lower bounds for each potential route branch.
4. Expansion of promising nodes while eliminating non-promising paths.
5. Iteration until the route with the smallest total distance is obtained.

By systematically eliminating non optimal branches, the algorithm significantly reduces the number of route combinations that must be evaluated.

3.8.2 Upper Bound Concept in Branch and Bound

A fundamental concept in the Branch and Bound algorithm is the use of upper bounds and lower bounds to guide the optimization process.

Upper Bound: Represents the best feasible solution currently identified.

Lower Bound: Represents the minimum possible cost that a branch could achieve.

If the lower bound of a branch exceeds the current upper bound, that branch is eliminated from further consideration. This process significantly reduces the search space and improves computational efficiency.

In the context of this study, the algorithm used an upper bound derived from feasible route estimates. This upper bound acted as a constraint that prevented the exploration of routes whose total distance would exceed the current best solution. As a result, the algorithm was able to identify a route with a smaller total distance than the manually computed route.

The algorithm therefore served as a verification tool, allowing the manually obtained results to be evaluated against a mathematically optimized solution.

3.9 Summary and Conclusion

Chapter three outlines the applied operational research design utilized to address the real-world logistics challenges of Viboi Ventures, a Coca-Cola depot in Benin City. The methodology centers on a quantitative and computational approach where Google Maps was used to obtain the geographical coordinates of ten delivery locations, which were subsequently converted into a 10 × 10 distance matrix. This matrix, representing ninety possible travel routes, served as the foundation for a model formulated as a Travelling Salesman Problem (TSP) aimed at minimizing total travel distance, time, and cost. The primary computational tool employed was the Branch and Bound algorithm, which was executed through a rigorous manual process involving eight key steps: matrix construction, row and column reduction, branching, matrix reduction, bound calculation, bounding and pruning, iterative expansion, and termination.

Ultimately, this chapter demonstrates that sophisticated mathematical optimization techniques can be effectively applied to urban logistics using accessible digital tools and manual computation. By transforming a physical delivery process into a structured mathematical model, the study establishes a reliable framework for identifying the most efficient delivery sequence among numerous possible permutations. The integration of real world distance data from Google Maps with the theoretical rigor of the Branch and Bound algorithm ensures that the resulting model is both mathematically sound and geographically valid. Consequently, this methodology provides a practical and scalable approach for small to medium logistics operations to enhance resource utilization and delivery efficiency without the need for expensive specialized software.

CHAPTER FOUR

RESULT AND DISCUSSION

4.1 Data Presentation

This section presents the data used for the optimization process and the corresponding computational results obtained through the application of the Branch and Bound algorithm. The dataset represents the distances between ten (10) selected delivery points within Benin City, which form the basis for determining the shortest possible delivery route for viboi Ventures, the Coca-Cola depot serving as the operational base for this study.

The data were generated using Google Maps, which provided the real time travel distances (in kilometers) between each pair of delivery locations. These distances were arranged into a 10 × 10 distance matrix, where each cell represents the direct road distance between two points. The diagonal entries, which denote travel from a location to itself, were assigned the value "∞" (infinity) to signify inapplicability in route computation.

Each location was assigned a numerical identifier (Node 1 to Node 10), with Node 1 representing viboi Ventures (the depot) and Nodes 2–10 representing the respective customer or delivery destinations within Benin City. The goal was to compute a route that begins at Node 1, visits every other node exactly once, and returns to Node 1 with the least total distance possible.

The complete data table showing the pairwise distances between all ten locations is presented in the Appendix section for reference. This ensures clarity while maintaining a concise presentation of results in the main body of the report.

The distance matrix served as the foundational input for all subsequent computations, including the row and column reduction operations, sub-matrix formation, and the calculation of lower bounds during the Branch and Bound process. These steps are presented in the following sections to illustrate the systematic derivation of the optimal delivery route.

4.2 Presentation of Analysis

This section presents the detailed computational analysis carried out using the Branch and Bound algorithm to determine the shortest delivery route among the ten selected locations in Benin City. All results, including the reduced matrices, branching stages, and corresponding lower bound calculations, are presented exactly as obtained during the computation process.

The analysis begins with the initial reduced matrix, from which successive branching operations were performed starting from Node 1 (viboi Ventures). Each sub-matrix represents a possible delivery path and its associated cost, obtained after pruning the corresponding rows and columns and recalculating new minima.

At each stage, the lower bounds were computed by adding the cost of the chosen route to the total of the reduction values, thereby identifying and eliminating non-optimal paths. The branching continued iteratively until all feasible routes were evaluated, leading to the determination of the optimal delivery sequence with the minimum total distance.

All matrices, tables, and diagrams representing these stages are presented below exactly as derived from the computation process, while their interpretations are discussed in the subsequent section of this chapter

DISTANCE MATRIX (KM)

	1	2	3	4	5	6	7	8	9	10
1	∞	2.8	1.5	0.6	1.0	1.0	1.7	1.6	1.1	2.3
2	2.8	∞	4.5	3.6	3.9	3.9	4.0	2.0	1.7	0.9
3	2.1	3.7	∞	1.3	0.9	1.0	0.2	2.7	2.7	2.9
4	1.3	3.9	1.0	∞	0.7	0.4	1.2	1.9	1.9	2.9
5	1.4	3.9	0.9	0.5	∞	0.9	1.1	2.0	1.9	3.4
6	2.3	3.3	0.6	1.3	1.0	∞	0.8	2.7	2.4	2.5
7	2.3	3.9	0.2	1.4	1.1	1.3	∞	2.9	2.6	2.7
8	1.6	2.0	2.5	1.5	2.0	1.9	2.7	∞	1.3	1.9
9	1.1	1.7	2.5	1.4	1.9	1.8	2.6	1.3	∞	1.2
10	2.3	0.9	2.7	2.9	3.4	3.2	2.6	1.9	1.2	∞

Reduce each row and column in a way that there must be at least one zero in each row and column by reducing the minimum value of each element in each row and column.

	1	2	3	4	5	6	7	8	9	10	
1	∞	2.8	1.5	0.6	1.0	1.0	1.7	1.6	1.1	2.3	0.6
2	2.8	∞	4.5	3.6	3.9	3.9	4.0	2.0	1.7	0.9	0.9
3	2.1	3.7	∞	1.3	0.9	1.0	0.2	2.7	2.7	2.9	0.2
4	1.3	3.9	1.0	∞	0.7	0.4	1.2	1.9	1.9	2.9	0.4
5	1.4	3.9	0.9	0.5	∞	0.9	1.1	2.0	1.9	3.4	0.5
6	2.3	3.3	0.6	1.3	1.0	∞	0.8	2.7	2.4	2.5	0.6
7	2.3	3.9	0.2	1.4	1.1	1.3	∞	2.9	2.6	2.7	0.2
8	1.6	2.0	2.5	1.5	2.0	1.9	2.7	∞	1.3	1.9	1.3
9	1.1	1.7	2.5	1.4	1.9	1.8	2.6	1.3	∞	1.2	1.1
10	2.3	0.9	2.7	2.9	3.4	3.2	2.6	1.9	1.2	∞	0.9
											= 6.7

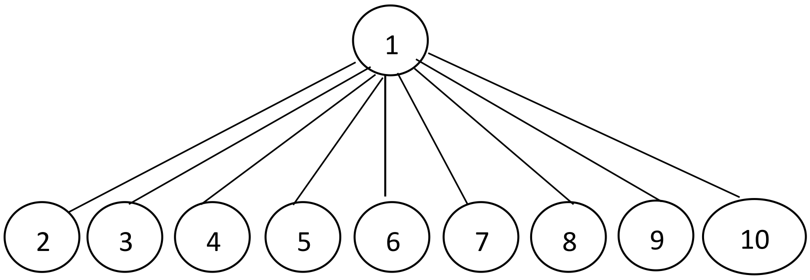
	1	2	3	4	5	6	7	8	9	10	
1	∞	2.2	0.9	0	0.4	0.4	1.1	1.0	0.5	1.7	
2	1.9	∞	3.6	2.6	3.0	3.0	3.1	1.1	0.8	0	
3	1.9	3.5	∞	1.1	0.7	0.8	0	2.5	2.5	2.7	
4	0.9	3.5	0.6	∞	0.3	0	0.8	1.5	1.5	2.5	
5	0.9	3.4	0.4	0	∞	0.4	0.6	1.5	1.4	2.9	
6	1.7	2.7	0	0.7	0.4	∞	0.2	2.1	1.8	1.9	
7	2.1	3.7	0	1.2	0.9	1.1	∞	2.7	2.4	2.5	
8	0.3	0.7	1.2	0.2	0.7	0.6	1.4	∞	0	0.6	
9	0	0.6	1.4	0.3	0.8	0.7	1.5	0.2	∞	0.1	
10	0.4	0	0.8	1.0	2.5	2.3	1.7	1.0	0.3	∞	
					0.3			0.2			=0.5

Reduced Matrix

	1	2	3	4	5	6	7	8	9	10
1	∞	2.2	0.9	0	0.1	0.4	1.1	0.8	0.5	1.7
2	1.9	∞	3.6	2.6	2.7	3.0	3.1	0.9	0.8	0
3	1.9	3.5	∞	1.1	0.4	0.8	0	2.3	2.5	2.7
4	0.9	3.5	0.6	∞	0	0	0.8	1.3	1.5	2.5
5	0.9	3.4	0.4	0	∞	0.4	0.6	1.3	1.4	2.9
6	1.7	2.7	0	0.7	0.1	∞	0.2	1.9	1.8	1.9
7	2.1	3.7	0	1.2	0.6	1.1	∞	2.5	2.4	2.5
8	0.3	0.7	1.2	0.2	0.4	0.6	1.4	∞	0	0.6
9	0	0.6	1.4	0.3	0.5	0.7	1.5	0	∞	0.1
10	0.4	0	0.8	1.0	2.2	2.3	1.7	0.8	0.3	∞

$$r = 6.7 + 0.5 = 7.2$$

the value 7.2 indicates the barest minimum distance



1 to 2

	1	2	3	4	5	6	7	8	9	10	
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
2	∞	∞	3.6	2.6	2.7	3.0	3.1	0.9	0.8	0	
3	1.9	∞	∞	1.1	0.4	0.8	0	2.3	2.5	2.7	
4	0.9	∞	0.6	∞	0	0	0.8	1.3	1.5	2.5	
5	0.9	∞	0.4	0	∞	0.4	0.6	1.3	1.4	2.9	
6	1.7	∞	0	0.7	0.1	∞	0.2	1.9	1.8	1.9	
7	2.1	∞	0	1.2	0.6	1.1	∞	2.5	2.4	2.5	
8	0.3	∞	1.2	0.2	0.4	0.6	1.4	∞	0	0.6	
9	0	∞	1.4	0.3	0.5	0.7	1.5	0	∞	0.1	
10	0.4	∞	0.8	1.0	2.2	2.3	1.7	0.8	0.3	∞	0.3

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	3.6	2.6	2.7	3.0	3.1	0.9	0.8	0
3	1.9	∞	∞	1.1	0.4	0.8	0	2.3	2.5	2.7
4	0.9	∞	0.6	∞	0	0	0.8	1.3	1.5	2.5
5	0.9	∞	0.4	0	∞	0.4	0.6	1.3	1.4	2.9
6	1.7	∞	0	0.7	0.1	∞	0.2	1.9	1.8	1.9
7	2.1	∞	0	1.2	0.6	1.1	∞	2.5	2.4	2.5
8	0.3	∞	1.2	0.2	0.4	0.6	1.4	∞	0	0.6
9	0	∞	1.4	0.3	0.5	0.7	1.5	0	∞	0.1
10	0.1	∞	0.5	0.7	1.9	2.0	1.4	0.5	0	∞

$$A(1,2) + r + \hat{r} = 2.2 + 7.2 + 0.3 = 9.7$$

1 to 3

	1	2	3	4	5	6	7	8	9	10	
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
2	1.9	∞	∞	2.6	2.7	3.0	3.1	0.9	0.8	0	
3	∞	3.5	∞	1.1	0.4	0.8	0	2.3	2.5	2.7	
4	0.9	3.5	∞	∞	0	0	0.8	1.3	1.5	2.5	
5	0.9	3.4	∞	0	∞	0.4	0.6	1.3	1.4	2.9	
6	1.7	2.7	∞	2.7	0.1	∞	0.2	1.9	1.8	1.9	0.1
7	2.1	3.7	∞	1.2	0.6	1.1	∞	2.5	2.4	2.5	0.6
8	0.3	0.7	∞	0.2	0.4	0.6	1.4	∞	0	0.6	
9	0	0.6	∞	0.3	0.5	0.7	1.5	0	∞	0.1	
10	0.4	0	∞	1.0	2.2	0.3	1.7	0.8	0.3	∞	

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	∞	2.6	2.7	3.0	3.1	0.9	0.8	0
3	∞	3.5	∞	1.1	0.4	0.8	0	2.3	2.5	2.7
4	0.9	3.5	∞	∞	0	0	0.8	1.3	1.5	2.5
5	0.9	3.4	∞	0	∞	0.4	0.6	1.9	1.4	2.9
6	1.6	2.6	∞	2.6	0	∞	0.1	1.8	1.7	1.8
7	1.5	3.1	∞	0.6	0	0.5	∞	1.9	1.8	1.9
8	0.3	0.7	∞	0.2	0.4	0.6	1.4	∞	0	0.6
9	0	0.6	∞	0.3	0.5	0.7	1.5	0	∞	0.1
10	0.4	0	∞	1.0	2.2	0.3	1.7	0.8	0.3	∞

$$A(1,3) + r + \hat{r} = 0.9 + 7.2 + 0.7 = 8.8$$

1 to 4

	1	2	3	4	5	6	7	8	9	10	
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
2	1.9	∞	3.6	∞	2.7	3.0	3.1	0.9	0.8	0	
3	1.9	3.5	∞	∞	0.4	0.8	0	0.3	2.5	2.7	
4	∞	3.5	0.6	∞	0	0	0.8	1.3	1.5	2.5	
5	0.9	3.4	0.4	∞	∞	0.4	0.6	1.3	1.4	2.9	
6	1.7	2.7	0	∞	0.1	∞	0.2	1.9	1.8	1.9	0.4
7	2.1	3.7	0	∞	0.6	1.1	∞	2.5	2.4	2.5	
8	0.3	0.7	1.2	∞	0.4	0.6	1.4	∞	0	0.6	
9	0	0.6	1.4	∞	0.5	0.7	1.5	0	∞	0.1	
10	0.4	0	0.8	∞	2.2	2.3	1.7	0.8	0.3	∞	

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	∞	2.7	3.0	3.1	0.9	0.8	0
3	1.9	3.5	∞	∞	0.4	0.8	0	2.5	2.5	2.7
4	∞	3.5	0.6	∞	0	0	0.8	1.3	1.5	2.5
5	0.5	3.0	0	∞	∞	0	0.2	0.9	1.0	2.5
6	1.7	2.7	0	∞	0.1	∞	0.2	1.9	1.8	1.9
7	2.1	3.7	0	∞	0.6	1.1	∞	2.5	2.4	2.5
8	0.3	0.7	1.2	∞	0.4	0.6	1.4	∞	0	0.6
9	0	0.6	1.4	∞	0.5	0.7	1.5	0	∞	0.1
10	0.4	0	0.8	∞	2.2	2.3	1.7	0.8	0.3	∞

$$A(1,4) + r + \hat{r} = 0 + 7.2 + 0.4 = 7.6$$

1 to 5

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	2.6	∞	3.0	3.1	0.9	0.8	0
3	1.9	3.5	∞	1.1	∞	0.8	0	2.3	2.5	2.7
4	0.9	3.5	0.6	∞	∞	0	0.8	1.3	1.5	2.5
5	∞	3.4	0.4	0	∞	0.4	0.6	1.3	1.4	2.9
6	1.7	2.7	0	0.7	∞	∞	0.2	1.9	1.8	1.9
7	2.1	3.7	0	1.2	∞	1.1	∞	2.5	2.4	2.5
8	0.3	0.7	1.2	0.2	∞	0.6	1.4	∞	0	0.6
9	0	0.6	1.4	0.3	∞	0.7	1.5	0	∞	0.1
10	0.4	0	0.8	1.0	∞	2.3	1.7	0.8	0.3	∞

Each row and column has already been reduced with 0

$$A(1,5) + r + \hat{r} = 0.1 + 7.2 + 0 = 7.3$$

1 to 6

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	2.6	2.7	∞	3.1	0.9	0.8	0
3	1.9	3.5	∞	1.1	0.4	∞	0	2.3	2.5	2.7
4	0.9	3.5	0.6	∞	0	∞	0.8	1.3	1.5	2.5
5	0.9	3.4	0.6	0	∞	∞	0.6	1.3	1.4	2.9
6	∞	2.7	0	0.7	0.1	∞	0.2	1.9	1.8	1.9
7	2.1	3.7	0	1.2	0.6	∞	∞	2.5	2.4	2.5
8	0.3	0.7	1.2	0.2	0.4	∞	1.4	∞	0	0.6
9	0	0.6	1.4	0.3	0.5	∞	1.5	0	∞	0.1
10	0.4	0	0.8	1.0	2.2	∞	1.7	0.8	0.3	∞

Each row and column has already been reduced with 0

$$A(1,6) + r + \hat{r} = 0.4 + 7.2 + 0 = 7.6$$

1 to 7

	1	2	3	4	5	6	7	8	9	10	
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
2	1.9	∞	3.6	2.6	2.7	3.0	∞	0.9	0.8	0	
3	1.9	3.5	∞	1.1	0.4	0.8	∞	2.3	2.5	2.7	0.4
4	0.9	3.5	0.6	∞	0	0	∞	1.3	1.5	2.5	
5	0.9	3.4	0.4	0	∞	0.4	∞	1.3	1.4	2.9	
6	1.7	2.7	0	0.7	0.1	∞	∞	1.9	1.8	1.9	
7	∞	3.7	0	1.2	0.6	0.1	∞	2.5	2.4	2.5	
8	0.3	0.7	1.2	0.2	0.4	0.6	∞	∞	0	0.6	
9	0	0.6	1.4	0.3	0.5	0.7	∞	0	∞	0.1	
10	0.4	0	0.8	0.1	2.2	2.3	∞	0.8	0.3	∞	

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	2.6	2.7	3.0	∞	0.9	0.8	0
3	1.5	3.1	∞	0.7	0	0.4	∞	1.9	2.1	2.3
4	0.9	3.5	0.6	∞	0	0	∞	1.3	1.5	2.5
5	0.9	3.4	0.4	0	∞	0.4	∞	1.3	1.4	2.9
6	1.7	2.7	0	0.7	0.1	∞	∞	1.9	1.8	1.9
7	∞	3.7	0	1.2	0.6	0.1	∞	2.5	2.4	2.5
8	0.3	0.7	1.2	0.2	0.4	0.6	∞	∞	0	0.6
9	0	0.6	1.4	0.3	0.5	0.7	∞	0	∞	0.1
10	0.4	0	0.8	0.1	2.2	2.3	∞	0.8	0.3	∞

$$A(1,7) + r + \hat{r} = 1.1 + 7.2 + 0.4 = 8.7$$

1 to 8

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	2.6	2.7	3.0	3.1	∞	0.8	0
3	1.9	3.5	∞	1.1	2.4	0.8	0	∞	2.5	2.7
4	0.9	3.5	0.6	∞	0	0	0.8	∞	1.5	2.5
5	0.9	3.4	0.4	0	∞	0.4	0.6	∞	1.4	2.9
6	1.7	2.7	0	0.7	0.1	∞	0.2	∞	1.8	1.9
7	2.1	3.7	0	1.2	0.6	0.1	∞	∞	2.4	2.5
8	∞	0.7	1.2	0.2	0.4	0.6	1.4	∞	0	0.6
9	0	0.6	1.4	0.3	0.5	0.7	1.5	∞	∞	0.1
10	0.4	0	0.8	1.0	2.2	2.3	1.7	∞	0.3	∞

Each row and column has already been reduced with 0

$$A(1,8) + r + \hat{r} = 0.8 + 7.2 + 0 = 8.0$$

1 to 9

	1	2	3	4	5	6	7	8	9	10	
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
2	1.9	∞	3.6	2.6	2.7	3.0	3.1	0.9	∞	0	
3	1.9	3.5	∞	1.1	0.4	0.8	0	2.3	∞	2.7	
4	0.9	3.5	0.6	∞	0	0	0.8	1.3	∞	2.5	
5	0.9	3.4	0.4	0	∞	0.4	0.6	1.3	∞	2.9	
6	0.7	2.7	0	0.7	0.1	∞	0.2	1.9	∞	1.9	
7	2.1	3.7	0	1.2	0.6	1.1	∞	2.5	∞	2.5	
8	0.3	0.7	1.2	0.2	0.4	0.6	1.4	∞	∞	0.6	0.2
9	∞	0.6	1.4	0.3	0.5	0.7	1.5	0	∞	0.1	
10	0.4	0	0.8	1.0	2.2	2.3	1.7	0.8	∞	∞	

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	2.6	2.7	3.0	3.1	0.9	∞	0
3	1.9	3.5	∞	1.1	0.4	0.8	0	2.3	∞	2.7
4	0.9	3.5	0.6	∞	0	0	0.8	1.3	∞	2.5
5	0.9	3.4	0.4	0	∞	0.4	0.6	1.3	∞	2.9
6	0.7	2.7	0	0.7	0.1	∞	0.2	1.9	∞	1.9
7	2.1	3.7	0	1.2	0.6	1.1	∞	2.5	∞	2.5
8	0.1	0.5	1.0	0	0.2	0.4	1.2	∞	∞	0.4
9	∞	0.6	1.4	0.3	0.5	0.7	1.5	0	∞	0.6
10	0.4	0	0.8	1.0	2.2	2.3	1.7	0.8	∞	∞

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.8	∞	3.6	2.6	2.7	3.0	3.1	0.9	∞	0
3	1.8	3.5	∞	1.1	0.4	0.8	0	2.3	∞	2.7
4	0.8	3.5	0.6	∞	0	0	0.8	1.3	∞	2.5
5	0.8	3.4	0.4	0	∞	0.4	0.6	1.3	∞	2.9
6	0.6	2.7	0	0.7	0.1	∞	0.2	1.9	∞	1.9
7	2.0	3.7	0	1.2	0.6	1.1	∞	2.5	∞	2.5
8	0	0.5	1.0	0	0.2	0.4	1.2	∞	∞	0.4
9	∞	0.6	1.4	0.3	0.5	0.7	1.5	0	∞	0.6
10	0.3	0	0.8	1.0	2.2	2.3	1.7	0.8	∞	∞

$$A(1,9) + r + \hat{r} = 0.5 + 7.2 + [0.2 + 0.1] = 8.0$$

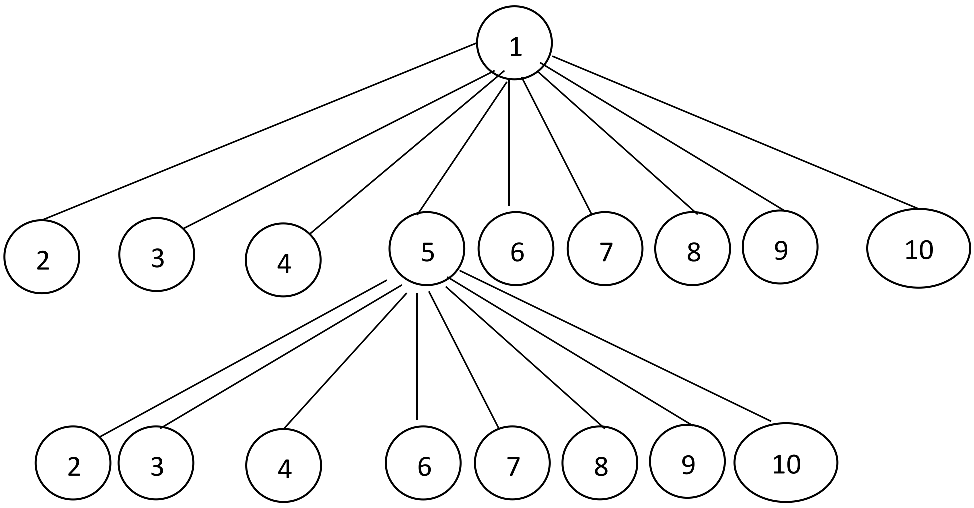
1 to 10

	1	2	3	4	5	6	7	8	9	10	
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
2	1.9	∞	3.6	2.6	2.7	3.0	3.1	0.9	0.8	∞	0.8
3	1.9	3.6	∞	1.1	0.4	0.8	0	2.3	2.5	∞	
4	0.9	3.5	0.6	∞	0	0	0.8	1.3	1.5	∞	
5	0.9	3.4	0.4	0	∞	0.4	0.6	1.3	1.4	∞	
6	1.7	2.7	0	0.7	0.1	∞	0.2	1.9	1.8	∞	
7	2.1	3.7	0	1.2	0.6	1.1	∞	2.5	2.4	∞	
8	0.3	0.7	1.2	0.2	0.4	0.6	1.4	∞	0	∞	
9	0	0.6	1.4	0.3	0.5	0.7	1.5	0	∞	∞	
10	∞	0	0.8	1.0	2.2	2.3	1.7	0.8	0.3	∞	

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	2.8	1.8	1.9	2.2	2.3	0.1	0	∞
3	1.9	3.5	∞	2.6	0.4	0.8	0	2.3	2.5	∞
4	0.9	3.5	0.6	∞	0	0	0.8	1.3	1.5	∞
5	0.9	3.4	0.4	0	∞	0.4	0.6	1.3	1.4	∞
6	1.7	2.4	0	0.7	0.1	∞	0.2	1.9	1.8	∞
7	2.1	3.7	0	1.2	0.6	1.1	∞	2.5	2.4	∞
8	0.3	0.7	1.2	0.2	0.4	0.6	1.4	∞	0	∞
9	0	0.6	1.4	0.3	0.5	0.7	1.5	0	∞	∞
10	∞	0	0.8	1.0	2.2	2.3	1.7	0.8	0.3	∞

$$A(1,10) + r + \hat{r} = 1.7 + 7.2 + 0.8 = 9.7$$

Therefore, the shortest distance is from 1 to 5 + 7.3



5 to 2

	1	2	3	4	5	6	7	8	9	10	
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
2	∞	∞	3.6	2.6	∞	3.0	3.1	0.9	0.8	0	
3	1.9	∞	∞	1.1	∞	0.8	0	2.3	2.5	2.7	
4	0.9	∞	0.6	∞	∞	0	0.8	1.3	1.5	2.5	
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
6	1.7	∞	0	0.7	∞	∞	0.2	1.9	1.8	1.9	
7	2.1	∞	0	1.2	∞	1.1	∞	2.5	2.4	2.5	
8	0.3	∞	1.2	0.2	∞	0.6	1.4	∞	0	0.6	
9	0	∞	1.4	0.3	∞	0.7	1.5	0	∞	0.1	
10	0.4	∞	0.8	1.0	∞	2.3	1.7	0.8	0.3	∞	0.3

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	3.6	2.6	∞	3.0	3.1	0.9	0.8	0
3	1.9	∞	∞	1.1	∞	0.8	0	2.3	2.5	2.7
4	0.9	∞	0.6	∞	∞	0	0.8	1.3	1.5	2.5
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	∞	0	0.7	∞	∞	0.2	1.9	1.8	1.9
7	2.1	∞	0	1.2	∞	1.1	∞	2.5	2.4	2.5
8	0.3	∞	1.2	0.4	∞	0.6	1.4	∞	0	0.6
9	0	∞	1.4	0.3	∞	0.7	1.5	0	∞	0.1
10	0.1	∞	0.5	0.7	∞	2.0	1.4	0.5	0	∞

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	3.6	2.3	∞	3.0	3.1	0.9	0.8	0
3	1.9	∞	∞	0.8	∞	0.8	0	2.3	2.5	2.7
4	0.9	∞	0.6	∞	∞	0	0.8	1.3	1.5	2.5
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	∞	0	0.4	∞	∞	0.2	1.9	1.8	1.9
7	2.1	∞	0	0.9	∞	1.1	∞	2.5	2.4	2.5
8	0.3	∞	1.2	0.1	∞	0.6	1.4	∞	0	0.6
9	0	∞	1.4	0	∞	0.7	1.5	1.5	∞	0.1
10	0.1	∞	0.5	0.4	∞	2.0	1.4	0.5	0	∞

$$A(5.2) + r + \hat{r} = 3.4 + 7.3 + [0.3 + 0.3] = 11.3$$

5 to 3

	1	2	3	4	5	6	7	8	9	10	
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
2	1.9	∞	∞	2.6	∞	3.0	3.1	0.9	0.8	0	
3	∞	3.5	∞	1.1	∞	0.8	0	2.3	2.5	2.7	
4	0.9	3.5	∞	∞	∞	0	0.8	1.3	1.5	2.5	
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
6	1.7	2.7	∞	0.7	∞	∞	0.2	1.9	1.8	1.9	0.3
7	2.1	3.7	∞	1.2	∞	1.1	∞	2.5	2.4	2.5	0.3
8	0.3	0.7	∞	0.2	∞	0.6	1.4	∞	0	0.6	
9	0	0.6	∞	0.3	∞	0.7	1.5	0	∞	0.1	
10	4	0	∞	1.0	∞	2.6	1.7	0.8	0.3	∞	1.3

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	∞	2.6	∞	3.0	3.1	0.9	0.8	0
3	∞	3.5	∞	1.1	∞	0.8	0	2.3	2.5	2.7
4	0.9	3.5	∞	∞	∞	0	0.8	1.3	1.5	2.5
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.5	2.5	∞	0.5	∞	∞	0	1.7	1.6	1.7
7	1.0	2.6	∞	0.1	∞	0	∞	1.4	1.3	1.4
8	0.3	0.7	∞	0.2	∞	0.6	1.4	∞	0	0.6
9	0	0.6	∞	0.3	∞	0.7	1.5	0	∞	0.1
10	4	0	∞	1.0	∞	2.6	1.7	0.8	0.3	∞

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	∞	2.5	∞	3.0	3.1	0.9	0.8	0
3	∞	3.5	∞	1.0	∞	0.8	0	2.3	2.5	2.7
4	0.9	3.5	∞	∞	∞	0	0.8	1.3	1.5	2.5
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.5	2.5	∞	0.4	∞	∞	0	1.7	1.6	1.7
7	1.0	2.6	∞	0	∞	0	∞	1.4	1.3	1.4
8	0.3	0.7	∞	0.1	∞	0.6	1.4	∞	0	0.6
9	0	0.6	∞	0.2	∞	0.7	1.5	0	∞	0.1
10	4	0	∞	0.9	∞	2.6	1.7	0.8	0.3	∞

$$A(5,3) + r + \hat{r} = 0.4 + 7.3 + [1.3 + 0.3] = 9.1$$

5 to 4

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	∞	∞	3.0	3.1	0.9	0.8	0
3	1.9	3.5	∞	∞	∞	0.8	0	2.3	2.5	2.7
4	∞	3.5	0.6	∞	∞	0	0.8	1.3	1.5	2.5
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	2.7	0	∞	∞	∞	0.2	1.9	1.8	1.9
7	2.1	3.7	0	∞	∞	1.1	∞	2.5	2.4	2.5
8	0.3	0.7	1.2	∞	∞	0.6	1.4	∞	0	0.6
9	0	0.6	1.4	∞	∞	0.7	1.5	0	∞	0.1
10	0.4	0	0.8	∞	∞	2.3	1.7	0.8	0.3	∞

Each row and column has already been reduced with 0

$$A(5,4) + r + \hat{r} = 0 + 7.3 + 0 = 7.3$$

5 to 6

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	2.6	∞	∞	3.1	0.9	0.8	0
3	1.9	3.5	∞	1.1	∞	∞	0	2.3	2.5	2.7
4	0.9	3.5	0.6	∞	∞	∞	0.8	1.3	1.5	2.5
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	2.7	0	0.7	∞	∞	0.2	1.9	1.8	1.9
7	2.1	3.7	0	1.2	∞	∞	∞	2.5	2.4	2.5
8	0.3	0.7	1.2	0.2	∞	∞	1.4	∞	0	0.6
9	0	0.6	1.4	0.3	∞	∞	1.5	0	∞	0.1
10	0.4	0	0.8	1.0	∞	∞	1.7	0.8	0.3	∞

0.6 (beside row 4)

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	2.6	∞	∞	3.1	0.9	0.8	0
3	1.9	3.5	∞	1.1	∞	∞	0	2.3	2.5	2.7
4	0.3	2.9	0	∞	∞	∞	0.2	0.7	0.9	1.9
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	2.7	0	0.7	∞	∞	0.2	1.9	1.8	1.9
7	2.1	3.7	0	1.2	∞	∞	∞	2.5	2.4	2.5
8	0.3	0.7	1.2	0.2	∞	∞	1.4	∞	0	0.6
9	0	0.6	1.4	0.3	∞	∞	1.5	0	∞	0.1
10	0.4	0	0.8	1.0	∞	∞	1.7	0.8	0.3	∞

0.2 (below column 4)

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	2.4	∞	∞	3.1	0.9	0.8	0
3	1.9	3.5	∞	0.9	∞	∞	0	2.3	2.5	2.7
4	0.3	2.9	0	∞	∞	∞	2.9	0.7	0.9	1.9
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	2.7	0	0.5	∞	∞	0.2	1.9	1.8	1.9
7	2.1	3.7	0	1.0	∞	∞	∞	2.5	2.4	2.5
8	0.3	0.7	1.2	0	∞	∞	1.4	∞	0	0.6
9	0	0.6	1.4	0.1	∞	∞	1.5	0	∞	0.1
10	0.4	0	0.8	0.8	∞	∞	1.7	0.8	0.3	∞

$$A(5,6) + r + \hat{r} = 0.4 + 7.3 + [0.6 + 0.2] = 8.5$$

5 to 7

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	2.6	∞	3.0	∞	0.9	0.8	0
3	1.9	3.5	∞	1.1	∞	0.3	∞	2.3	2.5	2.7
4	0.9	3.5	0.6	∞	∞	0	∞	1.3	1.5	2.5
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	2.7	0	0.7	∞	∞	∞	1.9	1.8	1.9
7	∞	3.7	0	1.2	∞	1.1	∞	2.5	2.4	2.5
8	0.3	0.7	1.2	0.2	∞	0.6	∞	∞	0	0.6
9	0	0.6	1.4	0.3	∞	0.7	∞	0	∞	0.1
10	0.4	0	0.8	1.0	∞	2.3	∞	0.8	0.3	∞

0.8

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	2.6	∞	3.0	∞	0.9	0.8	0
3	1.1	2.7	∞	0.3	∞	0	∞	1.5	1.7	1.9
4	0.9	3.5	0.6	∞	∞	0	∞	1.3	1.5	2.5
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	2.7	0	0.7	∞	∞	∞	1.9	1.8	1.9
7	∞	3.7	0	1.2	∞	1.1	∞	2.5	2.4	2.5
8	0.3	0.7	1.2	0.2	∞	0.6	∞	∞	0	0.6
9	0	0.6	1.4	0.3	∞	0.7	∞	0	∞	0.1
10	0.4	0	0.8	1.0	∞	2.3	∞	0.8	0.3	∞

0.2

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	2.4	∞	3.0	∞	0.8	0.8	0
3	1.1	2.7	∞	0.1	∞	0	∞	1.5	1.7	1.9
4	0.9	3.5	0.6	∞	∞	0	∞	1.3	1.5	2.5
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	2.7	0	0.5	∞	∞	∞	1.9	1.8	1.9
7	∞	3.7	0	1.0	∞	1.1	∞	2.5	2.4	2.5
8	0.3	0.7	1.2	0	∞	0.6	∞	∞	0	0.6
9	0	0.6	1.4	0.1	∞	0.7	∞	0	∞	0.1
10	0.4	0	0.8	0.8	∞	2.3	∞	0.8	0.3	∞

$$A(5,7) + r + \hat{r} = 0.6 + 7.3 + [0.8 + 0.2] = 8.9$$

5 to 8

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	2.6	∞	3.0	3.1	∞	0.8	0
3	1.9	3.5	∞	1.1	∞	0.8	0	∞	2.5	2.7
4	0.4	3.5	0.6	∞	∞	0	0.8	∞	1.5	2.5
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	2.7	0	0.7	∞	∞	0.2	∞	1.8	1.9
7	2.1	3.7	0	1.2	∞	1.1	∞	∞	2.4	2.5
8	∞	0.7	1.2	0.2	∞	0.6	1.4	∞	0	0.6
9	0	0.6	1.4	0.3	∞	0.7	1.5	∞	∞	0.1
10	0.4	0	0.8	1.0	∞	2.3	1.7	∞	0.3	∞

0.2

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	2.4	∞	3.0	3.1	∞	0.8	0
3	1.9	3.5	∞	0.9	∞	0.8	0	∞	2.5	2.7
4	0.4	3.5	0.6	∞	∞	0	0.8	∞	1.5	2.5
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	2.7	0	0.5	∞	∞	0.2	∞	1.8	1.9
7	2.1	3.7	0	1.0	∞	1.1	∞	∞	2.4	2.5
8	∞	0.7	1.2	0	∞	0.6	1.4	∞	0	0.6
9	0	0.6	1.4	0.1	∞	0.7	1.5	∞	∞	0.1
10	0.4	0	0.8	0.8	∞	2.3	1.7	∞	0.3	∞

$$A(5,8) + r + \hat{r} = 1.3 + 7.3 + 0.2 = 8.8$$

5 to 9

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	2.6	∞	3.0	3.1	0.9	∞	0
3	1.9	3.5	∞	1.1	∞	0.8	0	2.3	∞	2.7
4	0.9	3.5	0.6	∞	∞	0	0.8	1.3	∞	2.5
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	2.7	0	0.7	∞	∞	0.2	1.9	∞	1.9
7	2.7	3.7	0	1.2	∞	1.1	∞	2.5	∞	2.5
8	0.3	0.7	1.2	0.2	∞	0.6	1.4	∞	∞	0.6
9	0	0.6	1.4	0.3	∞	0.7	1.5	0	∞	0.1
10	0.4	0	0.8	1.0	∞	2.3	1.7	0.8	∞	∞

0

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	2.6	∞	3.0	3.1	0.9	∞	0
3	1.9	3.5	∞	1.1	∞	0.8	0	2.3	∞	2.7
4	0.9	3.5	0.6	∞	∞	0	0.8	1.3	∞	2.5
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	2.7	0	0.7	∞	∞	0.2	1.9	∞	1.9
7	2.7	3.7	0	1.2	∞	1.1	∞	2.5	∞	2.5
8	0.1	0.5	1.0	0	∞	0.4	1.2	∞	∞	0.4
9	∞	0.6	1.4	0.3	∞	0.7	1.5	0	∞	0.1
10	0.4	0	0.8	1.0	∞	2.3	1.7	0.8	∞	∞

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.8	∞	3.6	2.6	∞	3.0	3.1	0.9	∞	0
3	1.8	3.5	∞	1.1	∞	0.8	0	2.3	∞	2.7
4	0.8	3.5	0.6	∞	∞	0	0.8	1.3	∞	2.5
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.6	2.7	0	0.7	∞	∞	0.2	1.9	∞	1.9
7	2.6	3.7	0	1.2	∞	1.1	∞1	2.5	∞	2.9
8	0	0.5	1.0	0	∞	0.6	1.4	∞	∞	0.6
9	∞	0.6	1.4	0.3	∞	0.7	1.5	0	∞	0.1
10	0.3	0	0.8	1.0	∞	2.3	1.7	0.8	∞	∞

$$A(5,9) + r + \hat{r} = 1.4 + 7.3 + [0.2 + 0.1] = 9$$

5 to 10

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	2.6	∞	3.0	3.1	0.9	0.8	∞
3	1.9	3.5	∞	1.1	∞	0.8	0	2.3	2.5	∞
4	0.9	3.5	0.6	∞	∞	0	0.8	1.3	1.5	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	2.7	0	0.7	∞	∞	0.2	1.9	1.8	∞
7	2.1	3.7	0	1.2	∞	1.1	∞	2.5	2.4	∞
8	0.3	0.7	1.2	0.2	∞	0.6	1.4	∞	0	∞
9	0	0.6	1.4	0.3	∞	0.7	1.5	0	∞	∞
10	∞	0	0.8	1.0	∞	2.3	1.7	0.8	0.3	∞

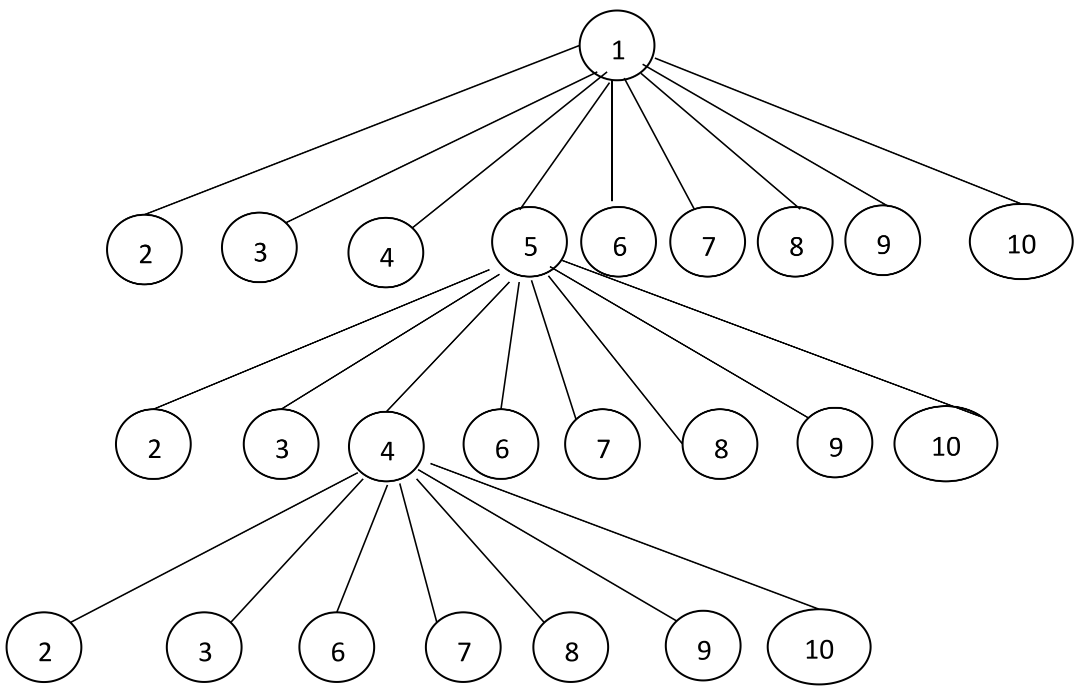
0.8

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.1	∞	2.8	1.8	∞	2.2	2.3	0.1	0	∞
3	1.9	3.5	∞	1.1	∞	0.8	0	2.3	2.5	∞
4	0.9	3.5	0.6	∞	∞	0	0.8	1.3	1.5	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	2.7	0	0.7	∞	∞	0.2	1.9	1.8	∞
7	2.1	3.7	0	1.2	∞	1.1	∞	2.5	2.4	∞
8	0.3	0.7	1.2	0.2	∞	0.6	1.4	∞	0	∞
9	0	0.6	1.4	0.3	∞	0.7	1.5	0	∞	∞
10	∞	0	0.8	1.0	∞	2.3	1.7	0.8	0.3	∞

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.1	∞	2.8	1.6	∞	2.2	2.3	0.1	0	∞
3	1.9	3.5	∞	0.9	∞	0.8	0	2.3	2.5	∞
4	0.9	3.5	0.6	∞	∞	0	0.8	1.3	1.5	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	2.7	0	0.5	∞	∞	0.2	1.9	1.8	∞
7	2.1	3.7	0	1.0	∞	1.1	∞	2.5	2.4	∞
8	0.3	0.7	1.2	0	∞	0.6	1.4	∞	0	∞
9	0	0.6	1.4	0.1	∞	0.7	1.5	0	∞	∞
10	∞	0	0.8	0.8	∞	2.3	1.7	0.8	0.3	∞

$$A(5,10) + r + \hat{r} = 2.9 + 7.3 + [0.8 + 0.2] = 11.2$$

Therefore, the shortest distance is from 1 to 5 + 7.3



4 to 2

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	3.5	∞	∞	3.0	3.1	0.9	0.8	0
3	1.9	∞	∞	∞	∞	0.8	0	2.3	2.5	2.7
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	∞	0	∞	∞	∞	0.2	1.9	1.8	1.9
7	2.1	∞	0	∞	∞	1.1	∞	2.5	2.4	2.5
8	0.3	∞	1.2	∞	∞	0.6	1.4	∞	0	0.6
9	0	∞	1.4	∞	∞	0.7	1.5	0	∞	0.1
10	0.4	∞	0.8	∞	∞	2.3	1.7	0.8	0.3	∞

0.3

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	3.5	∞	∞	3.0	3.1	0.9	0.8	0
3	1.9	∞	∞	∞	∞	0.8	0	2.3	2.5	2.7
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	∞	0	∞	∞	∞	0.2	1.9	1.8	1.9
7	2.1	∞	0	∞	∞	1.1	∞	2.5	2.4	2.5
8	0.3	∞	1.2	∞	∞	0.6	1.4	∞	0	0.6
9	0	∞	1.4	∞	∞	0.7	1.5	0	∞	0.1
10	0.1	∞	0.5	∞	∞	2.0	1.4	0.5	0	∞

0.6

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	3.5	∞	∞	2.0	3.1	0.9	0.8	0
3	1.9	∞	∞	∞	∞	0.2	0	2.3	2.5	2.7
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	∞	0	∞	∞	∞	0.2	1.9	1.8	1.9
7	2.1	∞	0	∞	∞	0.5	∞	2.5	2.4	2.5
8	0.3	∞	1.2	∞	∞	0	1.4	∞	0	0.6
9	0	∞	1.4	∞	∞	0.1	1.5	0	∞	0.1
10	0.1	∞	0.5	∞	∞	1.4	1.4	0.5	0	∞

$$A(4,2) + r + \hat{r} = 3.5 + 7.3 + [0.6 + 0.3] = 11.7$$

4 to 3

	1	2	3	4	5	6	7	8	9	10	
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
2	1.9	∞	∞	∞	∞	3.0	3.1	0.9	0.8	0	
3	∞	3.5	∞	∞	∞	0.8	0	2.3	2.5	2.7	
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
6	1.7	2.7	∞	∞	∞	∞	0.2	1.9	1.8	1.9	0.2
7	2.1	3.7	∞	∞	∞	1.1	∞	2.5	2.4	2.5	1.1
8	0.3	0.7	∞	∞	∞	0.6	1.4	∞	0	0.6	
9	0	0.6	∞	∞	∞	0.7	1.5	0	∞	0.1	
10	0.4	0	∞	∞	∞	2.3	1.7	0.8	0.3	∞	
											1.3

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	∞	∞	∞	3.0	3.1	0.9	0.8	0
3	∞	3.5	∞	∞	∞	0.8	0	2.3	2.5	2.7
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.5	2.5	∞	∞	∞	∞	0	1.7	1.6	1.7
7	1.0	2.6	∞	∞	∞	0	∞	1.4	1.3	1.4
8	0.3	0.7	∞	∞	∞	0.6	1.4	∞	0	0.6
9	0	0.6	∞	∞	∞	0.7	1.5	0	∞	0.1
10	0.4	0	∞	∞	∞	2.3	1.7	0.8	0.3	∞

$$A(4,3) + r + \hat{r} = 0.6 + 7.3 + 1.3 = 9.2$$

4 to 6

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	∞	∞	∞	3.1	0.9	0.8	0
3	1.9	3.5	∞	∞	∞	∞	0	2.3	2.5	2.7
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	2.7	0	∞	∞	∞	0.2	1.9	1.8	1.9
7	2.1	3.7	0	∞	∞	∞	∞	2.5	2.4	2.5
8	0.3	0.7	1.2	∞	∞	∞	1.4	∞	0	0.6
9	0	0.6	1.4	∞	∞	∞	1.5	0	∞	0.1
10	0.4	0	0.3	∞	∞	∞	1.7	0.8	0.3	∞

Each row and column has already been reduced with 0

$$A(4,6) + r + \hat{r} = 0 + 7.3 + 0 = 7.3$$

4 to 7

	1	2	3	4	5	6	7	8	9	10	
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
2	1.9	∞	3.6	∞	∞	3.0	∞	0.9	0.8	0	
3	1.9	3.5	∞	∞	∞	0.8	∞	2.3	2.5	2.7	0.8
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
6	1.7	2.7	0	∞	∞	∞	∞	1.9	1.8	1.9	
7	∞	3.7	0	∞	∞	1.1	∞	2.5	2.4	2.5	
8	0.3	0.7	1.2	∞	∞	0.6	∞	∞	0	0.6	
9	0	0.6	1.4	∞	∞	0.7	∞	0	∞	0.1	
10	0.4	0	0.3	∞	∞	2.3	∞	0.8	0.3	∞	

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	∞	∞	3.0	∞	0.9	0.8	0
3	1.1	2.7	∞	∞	∞	0	∞	1.5	1.7	2.1
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	2.7	0	∞	∞	∞	∞	1.9	1.8	1.9
7	∞	3.7	0	∞	∞	1.1	∞	2.5	2.4	2.5
8	0.3	0.7	1.2	∞	∞	0.6	∞	∞	0	0.6
9	0	0.6	1.4	∞	∞	0.7	∞	0	∞	0.1
10	0.4	0	0.3	∞	∞	2.3	∞	0.8	0.3	∞

$$A(4,7) + r + \hat{r} = 0.8 + 7.3 + 0.8 = 8.9$$

4 to 8

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	∞	∞	3.0	3.1	∞	0.8	0
3	1.9	3.5	∞	∞	∞	0.8	0	∞	2.5	2.7
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	2.7	0	∞	∞	∞	0.2	∞	1.8	1.9
7	2.1	3.7	0	∞	∞	1.1	∞	∞	2.4	2.5
8	∞	0.7	1.2	∞	∞	0.6	1.4	∞	0	0.6
9	0	0.6	1.4	∞	∞	0.7	1.5	∞	∞	0.1
10	0.4	0	0.3	∞	∞	2.3	1.7	∞	0.3	∞
							0.6			

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	∞	∞	2.4	3.1	∞	0.8	0
3	1.9	3.5	∞	∞	∞	0.2	0	∞	2.5	2.7
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	2.7	0	∞	∞	∞	0.2	∞	1.8	1.9
7	2.1	3.7	0	∞	∞	0.5	∞	∞	2.4	2.5
8	∞	0.7	1.2	∞	∞	0	1.4	∞	0	0.6
9	0	0.6	1.4	∞	∞	0.1	1.5	∞	∞	0.1
10	0.4	0	0.3	∞	∞	1.7	1.7	∞	0.3	∞

$$A(4,8) + r + \hat{r} = 1.3 + 7.3 + 0.6 = 9.2$$

4 to 9

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	∞	∞	3.0	3.1	0.9	∞	0
3	1.9	3.5	∞	∞	∞	0.8	0	2.3	∞	2.7
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	2.7	0	∞	∞	∞	0.2	1.9	∞	1.9
7	2.1	3.7	0	∞	∞	1.1	∞	2.5	∞	2.5
8	0.3	0.7	1.2	∞	∞	0.6	1.4	∞	∞	0.6
9	∞	0.6	1.4	∞	∞	0.7	1.5	0	∞	0.1
10	0.4	0	0.3	∞	∞	2.3	1.7	0.8	∞	∞

0.3

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	∞	∞	3.0	3.1	0.9	∞	0
3	1.9	3.5	∞	∞	∞	0.8	0	2.8	∞	2.7
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	2.7	0	∞	∞	∞	0.2	1.9	∞	1.9
7	2.1	3.7	0	∞	∞	1.1	∞	2.5	∞	2.5
8	0	0.4	0.9	∞	∞	0.3	1.1	∞	∞	0.3
9	∞	0.6	1.4	∞	∞	0.7	1.5	0	∞	0.1
10	0.4	0	0.3	∞	∞	0.3	1.7	0.8	∞	∞

1.1

$$A(4,9) + r + \hat{r} = 1.5 + 7.3 + 0.3 = 9.1$$

4 to 10

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	∞	∞	3.0	3.1	0.9	0.8	∞
3	1.9	3.5	∞	∞	∞	0.8	0	2.3	2.5	∞
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	2.7	0	∞	∞	∞	0.2	1.9	1.8	∞
7	2.1	3.7	0	∞	∞	1.1	∞	2.5	2.4	∞
8	0.3	0.7	1.2	∞	∞	0.6	1.4	∞	0	∞
9	0	0.6	1.4	∞	∞	0.7	1.5	0	∞	∞
10	∞	0	0.3	∞	∞	2.3	1.7	0.8	0.3	∞

0.8

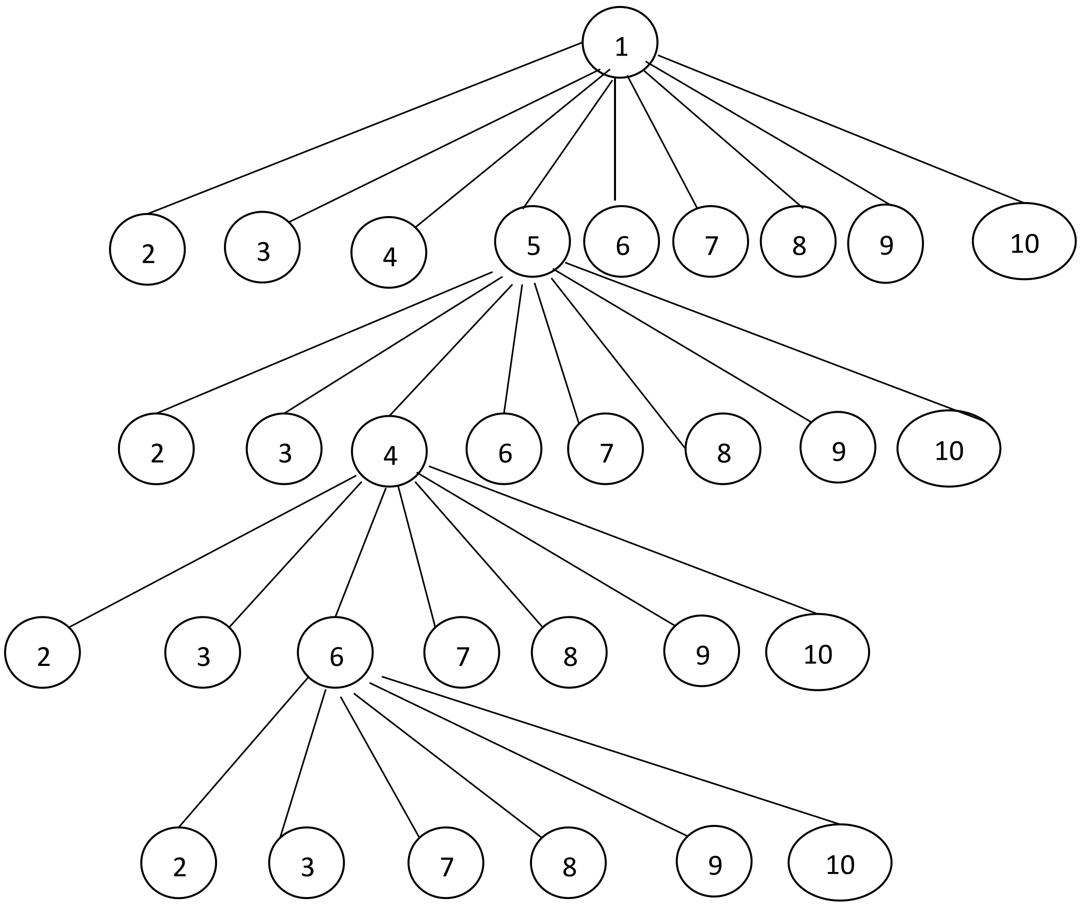
	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.1	∞	2.3	∞	∞	2.2	2.3	0.1	0	∞
3	1.9	3.5	∞	∞	∞	0.8	0	2.3	2.5	∞
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	2.7	0	∞	∞	∞	0.2	1.9	1.8	∞
7	2.1	3.7	0	∞	∞	1.1	∞	2.5	2.4	∞
8	0.3	0.7	1.2	∞	∞	0.6	1.4	∞	0	∞
9	0	0.6	1.4	∞	∞	0.7	1.5	0	∞	∞
10	∞	0	0.3	∞	∞	0.3	1.7	0.8	0.3	∞

0.3

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.1	∞	2.8	∞	∞	1.9	2.3	0.1	0	∞
3	1.9	3.5	∞	∞	∞	0.5	0	2.3	2.5	∞
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	1.7	2.7	0	∞	∞	∞	0.2	1.9	1.8	∞
7	2.1	3.7	0	∞	∞	0.8	∞	2.5	2.4	∞
8	0.3	0.7	1.2	∞	∞	0.3	1.4	∞	0	∞
9	0	0.6	1.4	∞	∞	0.4	1.5	0	∞	∞
10	∞	0	0.3	∞	∞	0	1.7	0.8	0.3	∞

$$A(4,10) + r + \hat{r} = 2.5 + 7.3 + [0.8 + 0.3] = 10.9$$

Therefore, the shortest distance if from 4 to 6 = 7.3



6 to 2

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	3.6	∞	∞	∞	3.1	0.7	0.8	0
3	1.9	∞	∞	∞	∞	∞	0	2.3	2.5	2.7
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	2.1	∞	0	∞	∞	∞	∞	2.5	2.4	2.5
8	0.3	∞	1.2	∞	∞	∞	1.4	∞	0	0.6
9	0	∞	1.4	∞	∞	∞	1.5	0	∞	0.1
10	0.4	∞	0.3	∞	∞	∞	1.7	0.8	0.3	∞

0.3

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	3.6	∞	∞	∞	3.1	0.9	0.8	0
3	1.9	∞	∞	∞	∞	∞	0	8.3	2.5	2.7
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	2.1	∞	0	∞	∞	∞	∞	2.5	2.4	2.5
8	0.3	∞	1.2	∞	∞	∞	1.4	∞	0	0.6
9	0	∞	1.4	∞	∞	∞	1.5	0	∞	0.1
10	0.1	∞	0	∞	∞	∞	1.4	0.5	0	∞

$$A(6,2) + r + \hat{r} = 2.7 + 7.3 + 0.3 = 10.3$$

6 to 3

$$\begin{array}{r} \\ 1\\2\\3\\4\\5\\6\\7\\8\\9\\10 \end{array}\begin{array}{c} \begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\ \begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ 1.9 & \infty & \infty & \infty & \infty & \infty & 3.1 & 0.9 & 0.8 & 0 \\ \infty & 3.7 & \infty & \infty & \infty & \infty & 0 & 2.1 & 2.5 & 2.7 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ 2.1 & 3.7 & \infty & \infty & \infty & \infty & \infty & 2.5 & 2.4 & 2.5 \\ 0.3 & 0.7 & \infty & \infty & \infty & \infty & 1.4 & \infty & 0 & 0.6 \\ 0 & 0.6 & \infty & \infty & \infty & \infty & 1.5 & 0 & \infty & 0.1 \\ 0.4 & 0 & \infty & \infty & \infty & \infty & 1.7 & 0.8 & 0.3 & \infty \end{bmatrix} \end{array} \quad 2.1$$

$$\begin{array}{r} \\ 1\\2\\3\\4\\5\\6\\7\\8\\9\\10 \end{array}\begin{array}{c} \begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\ \begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ 1.9 & \infty & \infty & \infty & \infty & \infty & 3.1 & 0.9 & 0.8 & 0 \\ \infty & 3.7 & \infty & \infty & \infty & \infty & 0 & 2.1 & 2.5 & 2.7 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ 0 & 1.6 & \infty & \infty & \infty & \infty & \infty & 0.4 & 0.3 & 0.4 \\ 0.3 & 0.7 & \infty & \infty & \infty & \infty & 1.4 & \infty & 0 & 0.6 \\ 0 & 0.6 & \infty & \infty & \infty & \infty & 1.5 & 0 & \infty & 0.1 \\ 0.4 & 0 & \infty & \infty & \infty & \infty & 1.7 & 0.8 & 0.3 & \infty \end{bmatrix} \end{array}$$

$$A(6,3) + r + \hat{r} = 0 + 7.3 + 2.1 = 9.4$$

6 to 7

$$\begin{array}{r} \\ 1\\2\\3\\4\\5\\6\\7\\8\\9\\10 \end{array}\begin{array}{c} \begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\ \begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ 1.9 & \infty & 3.6 & \infty & \infty & \infty & \infty & 0.9 & 0.8 & 0 \\ 1.9 & 3.5 & \infty & \infty & \infty & \infty & \infty & 2.3 & 2.5 & 2.7 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 3.7 & 0 & \infty & \infty & \infty & \infty & 2.5 & 2.4 & 2.5 \\ 0.3 & 0.7 & 1.2 & \infty & \infty & \infty & \infty & \infty & 0 & 0.6 \\ 0 & 0.6 & 1.4 & \infty & \infty & \infty & \infty & 0 & \infty & 0.1 \\ 0.4 & 0 & 0.3 & \infty & \infty & \infty & \infty & 0.8 & 0.3 & \infty \end{bmatrix} \end{array} \quad 1.9$$

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	∞	∞	∞	∞	0.9	0.8	0
3	0	1.6	∞	∞	∞	∞	∞	0.4	0.6	0.8
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	∞	3.7	0	∞	∞	∞	∞	2.5	2.4	2.5
8	0.3	0.7	1.2	∞	∞	∞	∞	∞	0	0.6
9	0	0.6	1.4	∞	∞	∞	∞	0	∞	0.1
10	0.4	0	0.3	∞	∞	∞	∞	0.8	0.3	∞

$$A(6,7) + r + \hat{r} = 0.2 + 7.3 + 1.9 = 9.4$$

6 to 8

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	∞	∞	∞	3.1	∞	0.8	0
3	1.9	3.5	∞	∞	∞	∞	0	∞	2.5	2.7
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	2.1	2.7	0	∞	∞	∞	∞	∞	2.4	2.5
8	∞	3.7	1.2	∞	∞	∞	1.4	∞	0	0.6
9	0	0.6	1.4	∞	∞	∞	1.5	∞	∞	0.1
10	0.4	0	0.3	∞	∞	∞	1.7	∞	0.3	∞

Each row and column has already been reduced with 0

$$A(6,8) + r + \hat{r} = 1.9 + 7.3 + 0 = 9.2$$

6 to 9

	1	2	3	4	5	6	7	8	9	10	
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
2	1.9	∞	3.6	∞	∞	∞	3.1	0.9	∞	0	
3	1.9	3.5	∞	∞	∞	∞	0	2.3	∞	2.7	
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
7	2.1	3.7	0	∞	∞	∞	∞	2.5	∞	2.5	
8	0.3	0.7	1.2	∞	∞	∞	1.4	∞	∞	0.6	
9	∞	0.6	1.4	∞	∞	∞	1.5	0	∞	0.1	0.3
10	0.4	0	0.3	∞	∞	∞	1.7	0.8	∞	∞	

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	∞	∞	∞	3.1	0.9	∞	0
3	1.9	3.5	∞	∞	∞	∞	0	2.3	∞	2.7
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	2.1	3.7	0	∞	∞	∞	∞	2.5	∞	2.5
8	0	0.4	0.9	∞	∞	∞	1.1	∞	∞	0.3
9	∞	0.6	1.4	∞	∞	∞	1.5	0	∞	0.1
10	0.4	0	0.3	∞	∞	∞	1.7	0.8	∞	∞

$$A(6,9) + r + \hat{r} = 1.8 + 7.3 + 0.3 = 9.4$$

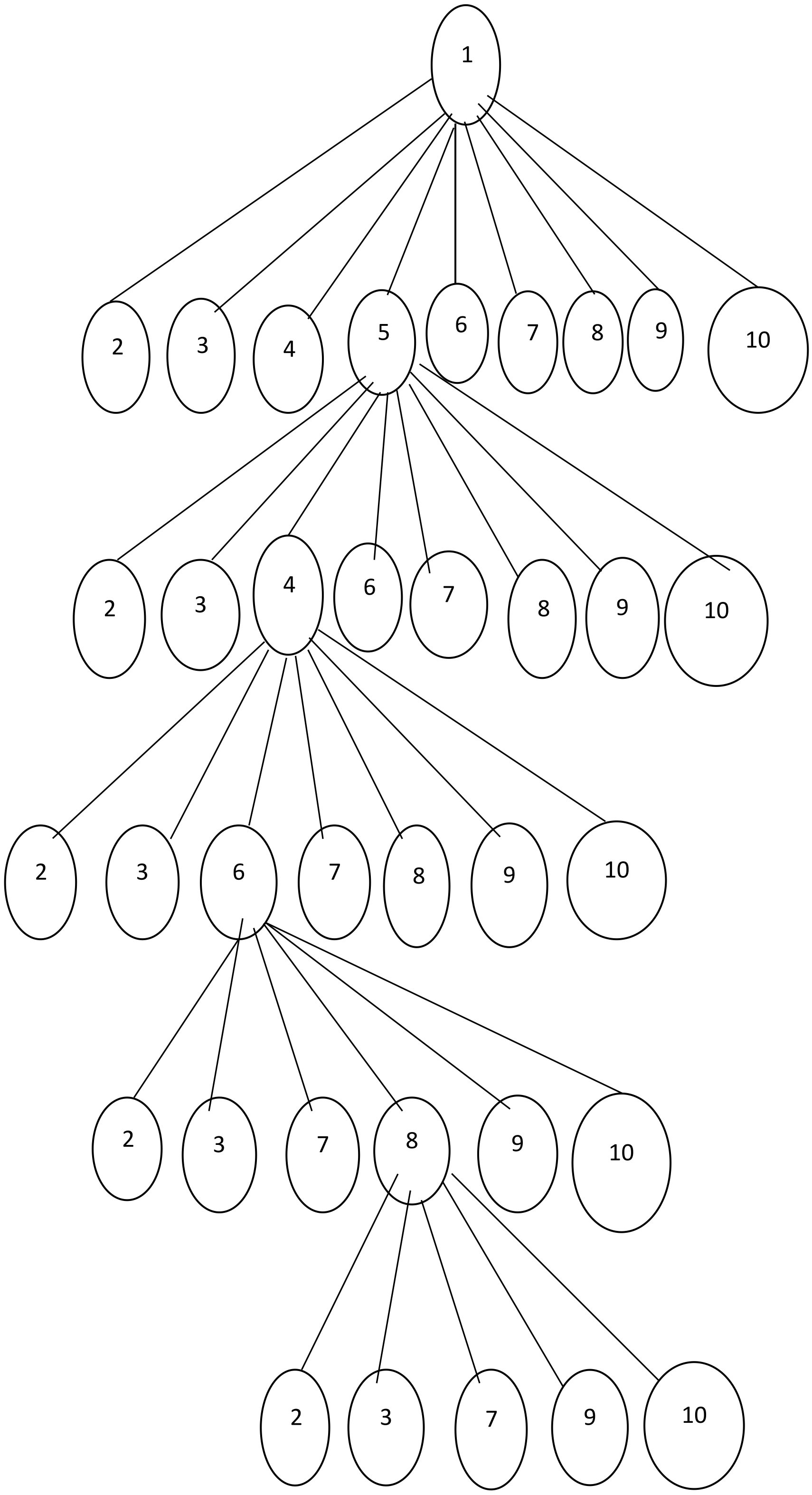
6 to 10

	1	2	3	4	5	6	7	8	9	10	
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
2	1.9	∞	3.6	∞	∞	∞	3.1	0.9	0.8	∞	0.8
3	1.9	3.5	∞	∞	∞	∞	0	2.3	2.5	∞	
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
7	2.1	3.7	0	∞	∞	∞	∞	2.5	2.4	∞	
8	0.3	0.7	1.2	∞	∞	∞	1.4	∞	0	∞	
9	0	0.6	1.4	∞	∞	∞	1.5	0	∞	∞	
10	∞	0	0.3	∞	∞	∞	1.7	0.8	0.3	∞	

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.1	∞	2.8	∞	∞	∞	2.3	0.1	0	∞
3	1.9	3.5	∞	∞	∞	∞	0	2.3	2.5	∞
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	2.1	3.7	0	∞	∞	∞	∞	2.5	2.4	∞
8	0.3	0.7	1.2	∞	∞	∞	1.4	∞	0	∞
9	0	0.6	1.4	∞	∞	∞	1.5	0	∞	∞
10	∞	0	0.3	∞	∞	∞	1.7	0.8	0.3	∞

$$A(6,10) + r + \hat{r} = 1.9 + 7.3 + 0.8 = 10$$

Therefore, the shortest distance is from 6 to 8 = 9.2


1
2
3
4
5
6
7
8
9
10
2
3
4
6
7
8
9
10
2
3
6
7
8
9
10
2
3
7
8
9
10
2
3
7
9
10

8 to 2

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	3.6	∞	∞	∞	3.1	∞	0.8	0
3	1.9	∞	∞	∞	∞	∞	0	∞	2.5	2.7
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	2.1	∞	0	∞	∞	∞	∞	∞	2.4	2.5
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	0	∞	1.4	∞	∞	∞	1.5	∞	∞	0.1
10	0.4	∞	0.3	∞	∞	∞	1.4	∞	0.3	∞

0.3

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	∞	∞	∞	3.1	∞	0.8	0
3	∞	∞	∞	∞	∞	∞	0	∞	2.5	2.7
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	2.1	∞	0	∞	∞	∞	∞	∞	2.4	2.5
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	0	∞	1.4	∞	∞	∞	1.5	∞	∞	0.1
10	0.1	∞	0	∞	∞	∞	1.1	∞	0	∞

$$A(8,2) + r + \hat{r} = 3.7 + 9.2 + 0.3 = 13.2$$

8 to 3

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	∞	∞	∞	∞	3.1	∞	0.8	0
3	∞	3.5	∞	∞	∞	∞	0	∞	2.5	2.7
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	2.1	2.7	∞	∞	∞	∞	∞	∞	2.4	2.5
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	0	0.6	∞	∞	∞	∞	1.5	∞	∞	0.1
10	0.4	0	∞	∞	∞	∞	1.7	∞	0.3	∞

2.1

$$
\begin{array}{c|cccccccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\hline
1 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
2 & 1.9 & \infty & \infty & \infty & \infty & \infty & 3.1 & \infty & 0.8 & 0 \\
3 & \infty & 3.5 & \infty & \infty & \infty & \infty & 0 & \infty & 0.5 & 2.7 \\
4 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
5 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
6 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
7 & 0 & 0.6 & \infty & \infty & \infty & \infty & \infty & \infty & 0.3 & 0.4 \\
8 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
9 & 0 & 0.6 & \infty & \infty & \infty & \infty & 1.5 & \infty & \infty & 0.1 \\
10 & 0.4 & 0 & \infty & \infty & \infty & \infty & 1.7 & \infty & 0.3 & \infty \\
\hline
 & & & & & & & & & 0.3 &
\end{array}
$$

$$
\begin{array}{c|cccccccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\hline
1 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
2 & 1.9 & \infty & \infty & \infty & \infty & \infty & 3.1 & \infty & 0.5 & 0 \\
3 & \infty & 3.5 & \infty & \infty & \infty & \infty & 0 & \infty & 0.2 & 2.7 \\
4 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
5 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
6 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
7 & 0 & 0.6 & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 0.4 \\
8 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
9 & 0 & 0.6 & \infty & \infty & \infty & \infty & 1.5 & \infty & \infty & 0.1 \\
10 & 0.4 & 0 & \infty & \infty & \infty & \infty & 1.7 & \infty & 0 & \infty \\
\hline
\end{array}
$$

$$A(8,3) + r + \hat{r} = 1.2 + 9.2 + [0.3 + 2.1] = 12.8$$

8 to 7

$$
\begin{array}{c|cccccccccc|c}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & \\
\hline
1 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \\
2 & 1.9 & \infty & 3.6 & \infty & \infty & \infty & \infty & \infty & 0.8 & 0 & \\
3 & 1.9 & 3.5 & \infty & \infty & \infty & \infty & \infty & \infty & 2.1 & 2.7 & 1.9 \\
4 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \\
5 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \\
6 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \\
7 & \infty & 2.7 & 0 & \infty & \infty & \infty & \infty & \infty & 2.4 & 2.5 & \\
8 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \\
9 & 0 & 0.6 & 1.4 & \infty & \infty & \infty & \infty & \infty & \infty & 0.1 & \\
10 & 0.4 & 0 & 0.3 & \infty & \infty & \infty & \infty & \infty & 0.3 & \infty & \\
\hline
\end{array}
$$

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	∞	∞	∞	∞	∞	0.8	0
3	0	1.6	∞	∞	∞	∞	∞	∞	0.6	0.8
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	∞	2.7	0	∞	∞	∞	∞	∞	2.4	2.5
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	0	0.6	1.4	∞	∞	∞	∞	∞	∞	0.1
10	0.4	0	0.3	∞	∞	∞	∞	∞	0.3	∞

0.3

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	∞	∞	∞	∞	∞	0.5	0
3	0	1.6	∞	∞	∞	∞	∞	∞	0.3	0.8
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	∞	2.7	0	∞	∞	∞	∞	∞	2.1	2.5
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	0	0.6	1.4	∞	∞	∞	∞	∞	∞	0.1
10	0.4	0	0.3	∞	∞	∞	∞	∞	0	∞

$$A(8,7) + r + \hat{r} = 1.4 + 9.2 + [1.9 + 0.3] = 12.8$$

8 to 9

	1	2	3	4	5	6	7	8	9	10	
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
2	1.9	∞	3.6	∞	∞	∞	3.1	∞	∞	0	
3	1.9	3.5	∞	∞	∞	∞	0	∞	∞	2.7	
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
7	2.1	2.7	0	∞	∞	∞	∞	∞	∞	2.5	
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
9	∞	0.6	1.4	∞	∞	∞	1.5	∞	∞	0.1	0.1
10	0.4	0	0.3	∞	∞	∞	1.7	∞	∞	∞	

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.9	∞	3.6	∞	∞	∞	3.1	∞	∞	0
3	1.9	3.5	∞	∞	∞	∞	0	∞	∞	2.7
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	2.1	2.7	0	∞	∞	∞	∞	∞	∞	2.5
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	0.6	1.3	∞	∞	∞	1.4	∞	∞	0
10	0.4	0	0.3	∞	∞	∞	1.7	∞	∞	∞

0.4

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.5	∞	3.6	∞	∞	∞	3.1	∞	∞	0
3	1.5	3.5	∞	∞	∞	∞	0	∞	∞	2.7
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	1.7	2.7	0	∞	∞	∞	∞	∞	∞	2.5
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	0.5	1.3	∞	∞	∞	1.4	∞	∞	0
10	0	0	0.3	∞	∞	∞	1.7	∞	∞	∞

$$A(8,9) + r + \hat{r} = 0 + 9.2 + 0.4 + 0.1 = 9.7$$

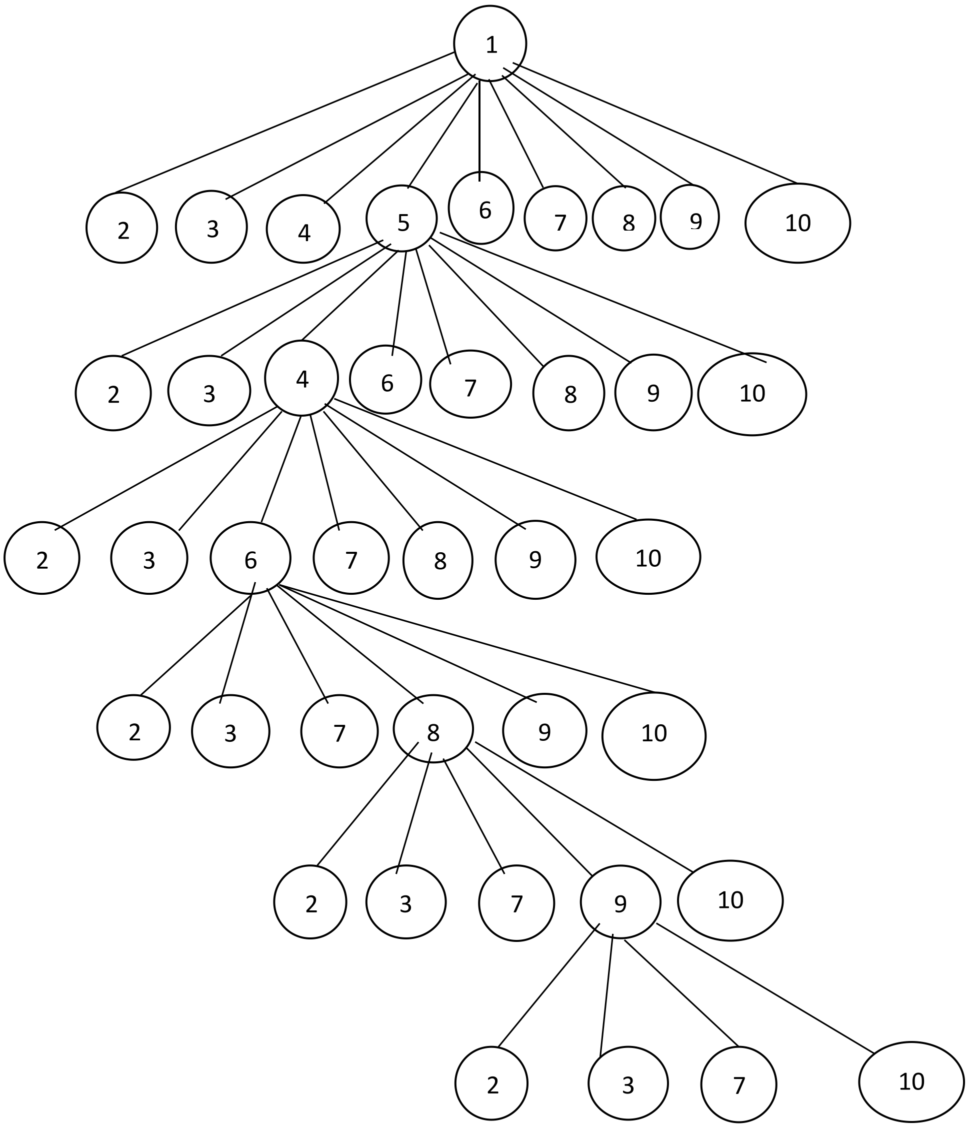
8 to 10

	1	2	3	4	5	6	7	8	9	10	
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
2	1.9	∞	3.6	∞	∞	∞	3.1	∞	0.8	∞	0.8
3	1.9	3.5	∞	∞	∞	∞	0	∞	2.5	∞	
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
7	2.1	2.7	0	∞	∞	∞	∞	∞	0	∞	
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
9	0	0.6	1.4	∞	∞	∞	1.5	∞	∞	∞	
10	∞	0	0.3	∞	∞	∞	1.7	∞	0.3	∞	

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.1	∞	2.8	∞	∞	∞	2.3	∞	0	∞
3	1.9	3.5	∞	∞	∞	∞	0	∞	2.5	∞
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	2.1	2.7	0	∞	∞	∞	∞	∞	0	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	0	0.6	1.4	∞	∞	∞	1.5	∞	∞	∞
10	∞	0	0.3	∞	∞	∞	1.7	∞	0.3	∞

$$A(8,10) + r + \hat{r} = 0.6 + 9.2 + 0.8 = 10.6$$

Therefore, the shortest distance if from 8 to 9 = 9.7



9 to 2

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	3.6	∞	∞	∞	3.1	∞	∞	0
3	1.5	∞	∞	∞	∞	∞	0	∞	∞	2.7
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	1.7	∞	0	∞	∞	∞	∞	∞	∞	2.5
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	0	∞	0.3	∞	∞	∞	1.7	∞	∞	∞

Each row and column has already been reduced with 0

$$A(9,2) + r + \hat{r} = 0.5 + 9.7 + 0 = 10.2$$

9 to 3

	1	2	3	4	5	6	7	8	9	10	
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
2	1.5	∞	∞	∞	∞	∞	3.1	∞	∞	0	
3	∞	3.5	∞	∞	∞	∞	0	∞	∞	2.7	
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
7	1.7	2.7	∞	∞	∞	∞	∞	∞	∞	2.5	1.7
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
10	0	0	∞	∞	∞	∞	1.7	∞	∞	∞	

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.5	∞	∞	∞	∞	∞	3.1	∞	∞	0
3	∞	3.5	∞	∞	∞	∞	0	∞	∞	2.7
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	0	1.0	∞	∞	∞	∞	∞	∞	∞	0.8
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	0	0	∞	∞	∞	∞	1.7	∞	∞	∞

$$A(9,3) + r + \hat{r} = 1.3 + 9.7 + 1.7 = 12.7$$

9 to 7

	1	2	3	4	5	6	7	8	9	10	
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
2	1.5	∞	3.6	∞	∞	∞	∞	∞	∞	0	
3	1.5	3.5	∞	∞	∞	∞	∞	∞	∞	2.7	1.5
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
7	∞	2.7	0	∞	∞	∞	∞	∞	∞	2.5	
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
10	0	0	0.3	∞	∞	∞	∞	∞	∞	∞	

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	1.5	∞	3.6	∞	∞	∞	∞	∞	∞	0
3	0	2.0	∞	∞	∞	∞	∞	∞	∞	1.2
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	∞	2.7	0	∞	∞	∞	∞	∞	∞	2.5
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	0	0	0.3	∞	∞	∞	∞	∞	∞	∞

$$A(9,7) + r + \hat{r} = 1.4 + 9.7 + 1.5 = 12.7$$

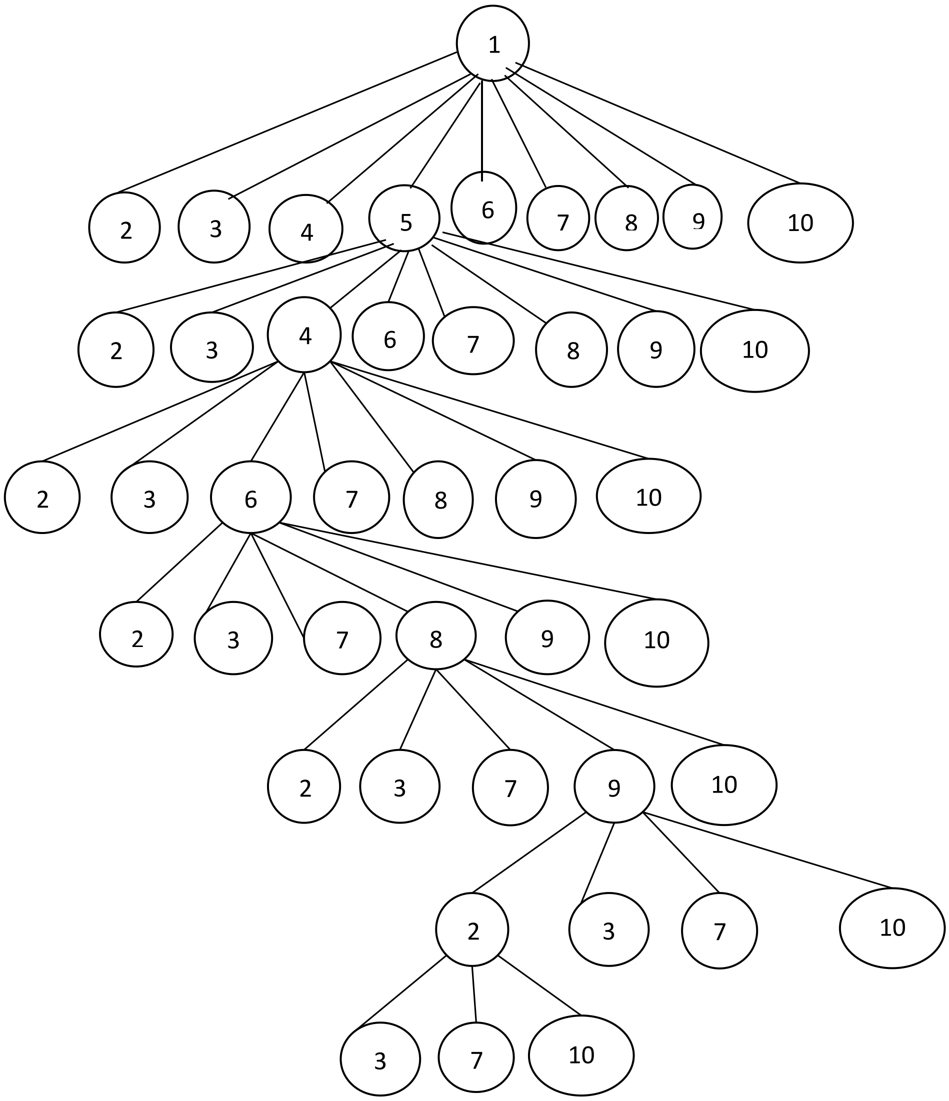
9 to 10

	1	2	3	4	5	6	7	8	9	10	
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
2	1.5	∞	3.6	∞	∞	∞	3.1	∞	∞	∞	1.5
3	1.5	3.5	∞	∞	∞	∞	0	∞	∞	∞	
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
7	1.7	2.7	0	∞	∞	∞	∞	∞	∞	∞	
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
10	∞	0	0.3	∞	∞	∞	1.7	∞	∞	∞	

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	0	∞	2.1	∞	∞	∞	1.6	∞	∞	∞
3	1.5	3.6	∞	∞	∞	∞	0	∞	∞	∞
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	1.7	2.7	0	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	0	0.3	∞	∞	∞	1.7	∞	∞	∞

$$A(9,10) + r + \hat{r} = 0 + 9.7 + 1.5 = 11.2$$

Therefore, the shortest distance if from 9 to 2 = 10.0



2 to 3

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
3	∞	∞	∞	∞	∞	∞	0	∞	∞	2.7
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	1.7	∞	∞	∞	∞	∞	∞	∞	∞	2.5
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	0	∞	∞	∞	∞	∞	1.7	∞	∞	∞

1.7

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
3	∞	∞	∞	∞	∞	∞	0	∞	∞	2.7
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	0	∞	∞	∞	∞	∞	∞	∞	∞	0.8
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	0	∞	∞	∞	∞	∞	1.7	∞	∞	∞

0.8

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
3	∞	∞	∞	∞	∞	∞	0	∞	∞	1.9
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	0	∞	∞	∞	∞	∞	∞	∞	∞	0
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	0	∞	∞	∞	∞	∞	1.7	∞	∞	∞

$$A(2,3) + r + \hat{r} = 3.6 + 10.2 + [1.7 + 0.8] = 16.3$$

2 to 7

	1	2	3	4	5	6	7	8	9	10	
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
3	1.5	∞	∞	∞	∞	∞	∞	∞	∞	2.7	1.5
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
7	∞	∞	0	∞	∞	∞	∞	∞	∞	2.5	
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
10	0	∞	0.3	∞	∞	∞	∞	∞	∞	∞	

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
3	0	∞	∞	∞	∞	∞	∞	∞	∞	1.2
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	∞	∞	0	∞	∞	∞	∞	∞	∞	2.5
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	0	∞	0.3	∞	∞	∞	∞	∞	∞	∞
										1.2

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
3	0	∞	∞	∞	∞	∞	∞	∞	∞	0
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	∞	∞	0	∞	∞	∞	∞	∞	∞	1.3
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	0	∞	0.3	∞	∞	∞	∞	∞	∞	∞

$$A(2,7) + r + \hat{r} = 3.1 + 10.2 + [1.2 + 1.5] = 16$$

2 to 10

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
3	1.5	∞	∞	∞	∞	∞	0	∞	∞	∞
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	1.7	∞	0	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	0.3	∞	∞	∞	1.7	∞	∞	∞

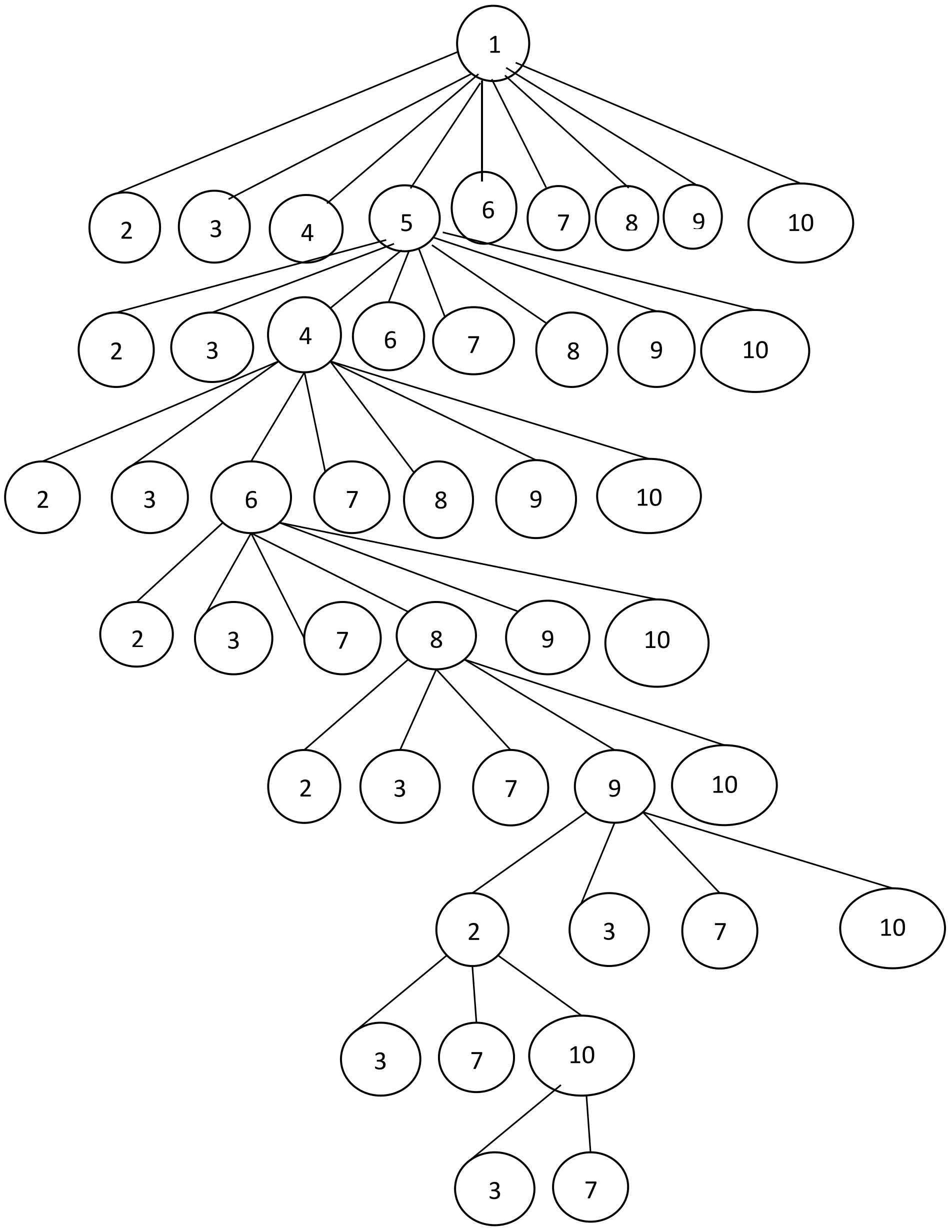
0.3

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
3	1.5	∞	∞	∞	∞	∞	0	∞	∞	∞
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	1.7	∞	0	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	0	∞	∞	∞	1.4	∞	∞	∞

	1	2	3	4	5	6	7	8	9	10
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
3	0	∞	∞	∞	∞	∞	0	∞	∞	∞
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
7	0.2	∞	0	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	0	∞	∞	∞	1.4	∞	∞	∞

$$A(2,10) + r + \hat{r} = 0 + 10.2 + 0.3 + 1.5 = 12$$

Therefore, the distance is from 2 to 10 = 12



10 to 3

	1	2	3	4	5	6	7	8	9	10	
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
3	∞	∞	∞	∞	∞	∞	0	∞	∞	∞	
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
7	0.2	∞	∞	∞	∞	∞	∞	∞	∞	∞	0.2
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	

$$\begin{array}{c c}
 & \begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array} &
\begin{bmatrix}
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty
\end{bmatrix}
\end{array}$$

$$A(10,3) + r + \hat{r} = 0 + 12 + 0.2 = 12.2$$

10 to 7

$$\begin{array}{c c}
 & \begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array} &
\begin{bmatrix}
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty
\end{bmatrix}
\end{array}$$

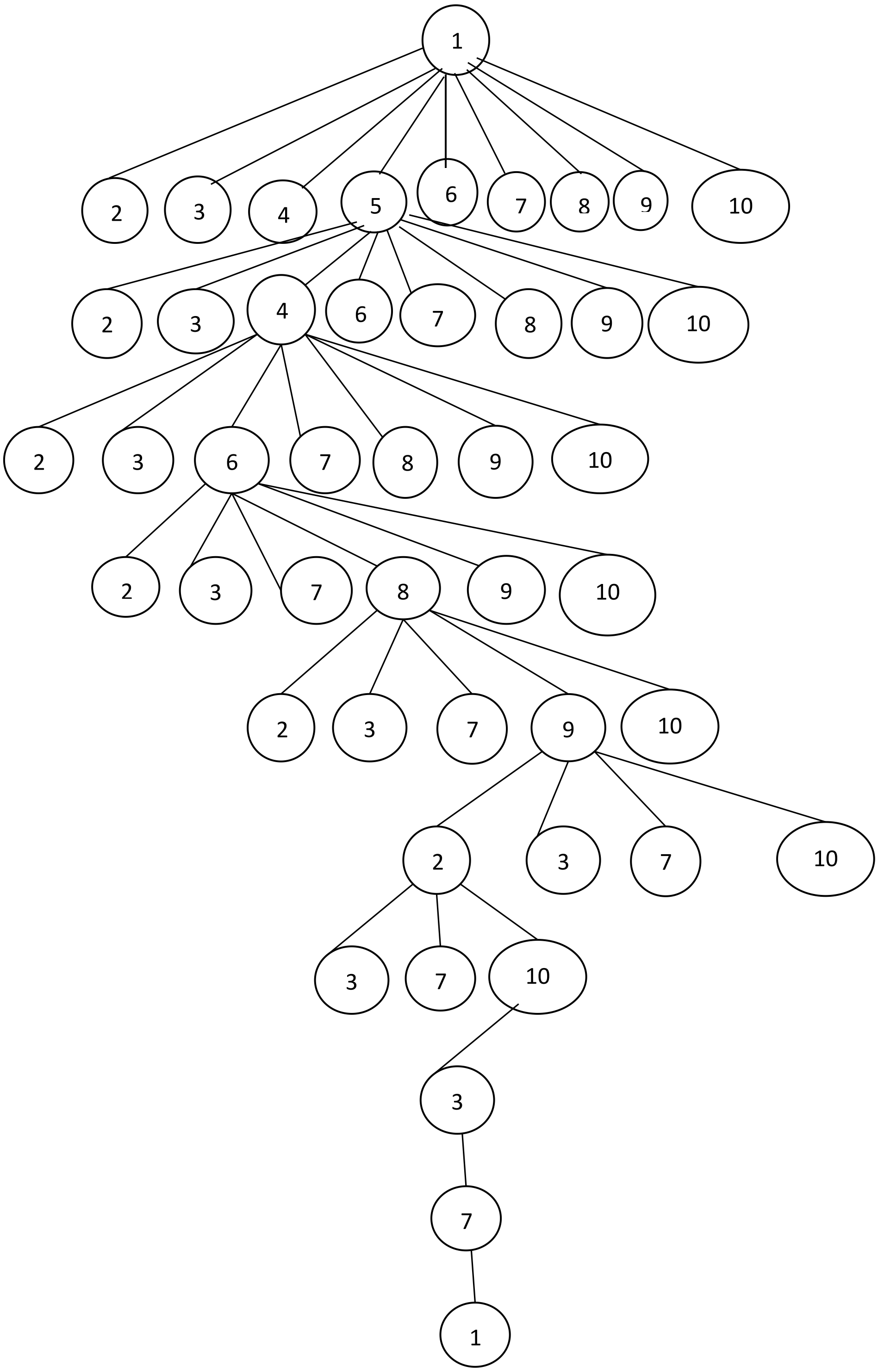
Each row and column has already been reduced with 0

$$A(10,7) + r + \hat{r} = 1.4 + 12 + 0 = 13.4$$

Therefore, the shortest distance is from 10 to 3 = 1 which is 12.2

The optimal route

$$1 \rightarrow 5 \rightarrow 4 \rightarrow 6 \rightarrow 8 \rightarrow 9 \rightarrow 2 \rightarrow 10 \rightarrow 3 \rightarrow 7 \rightarrow 1$$



4.3 Discussion of Results

The results of this study demonstrate the practical efficiency of the Branch and Bound algorithm in solving the Travelling Salesman Problem (TSP) applied to a real world delivery scenario within Benin City. Using the distances obtained from Google Maps, the algorithm systematically evaluated and pruned nono ptimal paths until the most efficient delivery route was determined.

The computation began with the initial reduced matrix, where row and column reductions were performed to establish a preliminary lower bound of 7.2 km. Successive branching operations were then carried out from Node 1 (viboi Ventures), representing the Coca-Cola depot, to all other nodes. Through this iterative process, sub matrices were created by pruning previously visited nodes and recalculating reduced costs. The lowest bound at each stage indicated the next feasible route to pursue, ensuring that only paths with potential for optimality were retained.

From the matrix evaluations and lower bound calculations, the final optimal delivery route was obtained as:

$$1 \rightarrow 5 \rightarrow 4 \rightarrow 6 \rightarrow 8 \rightarrow 9 \rightarrow 2 \rightarrow 10 \rightarrow 7 \rightarrow 3 \rightarrow 1$$

This route yielded the minimum cumulative distance among all possible permutations, establishing it as the most efficient delivery sequence for the selected ten locations in Benin City. The total optimised travel distance (derived from the cumulative bounds)which is 13.7km was significantly lower than that of the company's existing delivery route which is 18.7km, indicating a measurable improvement in logistical efficiency of 36.5% and a reduction in travel distance of 26.7% and a increase in efficiency.

4.3.1 Interpretation of Results

The outcome validates the effectiveness of the Branch and Bound technique in optimizing delivery operations under realistic urban constraints. The algorithm achieved its objective by employing a systematic reduction of the search space through bounding and pruning, thereby avoiding redundant route evaluations. Each stage of reduction allowed the identification of zeros within the cost matrix, simplifying subsequent calculations and making it possible to determine lower bounds that guided the choice of the next optimal branch.

This systematic evaluation aligns closely with the work of Willow *et al.* (2023), who implemented the Branch and Bound algorithm for package delivery and reported that the technique successfully produced an optimal route with minimal computational error and reduced delivery time. Similarly, the findings of Anggraeni *et al*. (2023) in optimising multi compartment fuel distribution routes in Indonesia confirm that the Branch and Bound approach effectively reduced total transportation costs by up to 28% through improved route sequencing and clustering. The current study's results are consistent with these outcomes, reinforcing the reliability of the Branch and Bound algorithm for achieving cost effective and efficient logistics planning.

4.3.2 Comparison Between Manual Computation and Algorithm-Based Optimisation

To evaluate the reliability of the manually computed delivery route, the results were compared with those generated using the Branch and Bound optimisation algorithm. While the manual approach involved step by step evaluation of feasible route combinations, the algorithm systematically searched the solution space while applying optimisation constraints.

The comparison between the manually computed route and the algorithm generated solution is presented in Table 4.1, which highlights the difference in total travel distance obtained from both approaches.

The comparison of the results is presented in Table 4.1.

Table 4.1 Comparison of Route Optimisation Results

Method	Total Distance
Manual Computation	13.7km
Algorithm (Branch and Bound)	12.2km

The results indicate that the algorithm produced a shorter total travel distance than the manually derived route. This difference occurs because the Branch and Bound algorithm uses a structured optimisation framework that includes the application of upper bounds and lower bounds to eliminate non-optimal routes during the search process.

In contrast, the manual computation relied primarily on direct evaluation of selected route combinations without the systematic pruning mechanism provided by the algorithm. As a result, the manual approach identified a feasible route but not necessarily the globally optimal route.

The use of upper bounds in the algorithm allowed it to limit the search space and focus only on route permutations that had the potential to produce smaller travel distances. Consequently, the algorithm was able to identify a more efficient route configuration.

The comparison between the manually computed route and the algorithm-generated route reveals a noticeable difference in the total travel distance. The manual computation produced a route distance of 13.7 km, whereas the Branch and Bound algorithm identified a route with a total distance of 12.2 km.

This difference can primarily be attributed to the optimisation mechanisms inherent in the Branch and Bound algorithm. Unlike manual calculations, the algorithm systematically evaluates route permutations using mathematical bounds that guide the search process toward the optimal solution. The use of upper bounds ensures that route combinations that cannot produce a better solution are eliminated early in the computation.

The manual method, while effective In identifying a feasible delivery route, does not incorporate such systematic pruning mechanisms. Consequently, it may terminate at a locally efficient route without fully exploring all possible route combinations.

Despite this difference, the manual approach remains valuable within the context of the study. It provides an intuitive understanding of the routing problem and serves as an initial estimate of delivery efficiency. Furthermore, the manual results offer a basis for validating algorithmic solutions.

The algorithmic result therefore does not invalidate the manual computation but rather enhances the reliability of the study by demonstrating that a more optimised solution exists. The combination of manual analysis and algorithm-based validation strengthens the methodological rigour of the research and confirms the applicability of optimisation techniques in improving logistics planning for Viboi Ventures.

4.3.3 Comparative Evaluation with Previous Research

The application of Branch and Bound in logistics optimisation has consistently demonstrated superior performance when compared to heuristic or trial based approaches. The study by Radharamanan *et al*. (1984) established that Branch and Bound could provide globally optimal solutions for both symmetric and asymmetric transportation problems, outperforming manual scheduling and heuristic estimations. Likewise, Mubarak *et al*. (2024) highlighted that integrating Branch and Bound with real-time heuristic models significantly reduced operational costs and improved delivery punctuality.

In this study, the optimised route generated not only shortened travel distance but also implied reductions in fuel consumption and travel time. These results support the conclusion by Sun (2024), who demonstrated in electric coal transportation that Branch and Bound models can effectively integrate distance and cost minimisation into a unified optimisation framework, yielding economically and logistically feasible routes.

Furthermore, the result obtained aligns with Chandra and Prasetyo (2024), who developed a web-based navigation model for tourist routes in Samosir District using Branch and Bound. Their research reported high accuracy between computed and actual travel distances, with minimal deviation from Google Maps data, confirming the algorithm's precision in real geographical contexts. Similarly, in the present work, the integration of Google Maps data ensured that the computed distances closely reflected real travel conditions within Benin City, thus increasing the practical applicability of the optimisation results.

4.3.4 Operational Implications

The optimised route indicates a potential for significant operational savings for viboi Ventures. By following the computed route, delivery time can be minimised, vehicle fuel efficiency improved, and the risk of driver fatigue reduced due to the elimination of unnecessary detours. Such outcomes are particularly relevant in urban logistics contexts where congestion, narrow streets, and variable traffic patterns often complicate route planning.

The findings of this study also show that the Branch and Bound algorithm can serve as a reliable decision support tool for small and medium sized logistics operations that cannot afford high end automated systems. This aligns with the assertions of Whitaker (2014), who argued that the Branch and Bound algorithm remains computationally efficient for practical problems and can be scaled across different levels of operational complexity.

Additionally, the current results substantiate earlier claims by Ralphs (2003) that parallel implementations of Branch and Bound (e.g., Branch and Cut variants) are capable of handling large scale vehicle routing problems efficiently through systematic node partitioning and bound refinement. Although the present study utilised a single machine implementation, its outcomes confirm the algorithm's robustness in smaller, city-based delivery networks.

4.3.5 Comparative Advantage over Existing Practice

Compared with the company's existing delivery practice, the optimised route offers a more balanced and cost efficient delivery plan. The previous route, which relied largely on driver experience and static scheduling, resulted in longer travel times and higher fuel consumption. By contrast, the optimised sequence ensures that deliveries are completed with minimal overlap and retracing, producing quantifiable improvements in logistics performance.

This finding mirrors the observations of Willow *et al*. (2023), who noted that when applied to package delivery networks, the Branch and Bound approach reduces both travel distance and delivery cost by prioritising route structure efficiency and sequence optimisation. Similarly, Mubarak *et al.* (2024) demonstrated that integrating Branch and Bound logic into vehicle routing algorithms directly contributes to reduced fuel expenditure and improved delivery reliability, confirming the method's role in sustainable logistics operations.

In summary, the findings affirm that the Branch and Bound algorithm, when supported by reliable mapping data from Google Maps, is an effective and practical method for solving real world delivery optimisation problems. The derived route — 1 → 5 → 4 → 6 → 8 → 9 → 2 → 10 → 7 → 3 → 1 — represents the shortest and most efficient delivery sequence achievable under the given conditions.

This outcome is consistent with existing literature, underscoring the algorithm's capability to balance computational precision with practical feasibility. The model's strength lies in its ability to combine theoretical rigour with operational relevance, making it an indispensable tool for urban logistics planning and other transport optimisation scenarios.

4.4 Findings

This chapter presented the results of the route optimisation process carried out using the Branch and Bound algorithm with data obtained from Google Maps. The computation successfully determined the optimal delivery route for ten selected locations in Benin City, producing the sequence 1 → 5 → 4 → 6 → 8 → 9 → 2 → 10 → 7 → 3 → 1 as the most efficient pathwith a 26.7% reduction in travel distance (13.7km) compaired to the viboi venture current delivery route (18.7km) and a 36.5% increase in efficiency.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

The study successfully developed and demonstrated an optimised delivery routing model for viboi Ventures in Benin City using the Branch and Bound method. The approach provided a structured and mathematical means of determining the most efficient route for deliveries within an urban environment. The problem was formulated as a Travelling Salesman Problem (TSP), and the solution was obtained entirely through manual computation, ensuring full transparency in each stage of the process.

The travel distance between locations were gotten through Google Maps although the duration between delivery points could not be ascertained,

The travel distances were then inputted into a matrix, and systematic row and column reductions were carried out to simplify the dataset. Through the branching and bounding process, non promising routes were pruned at every stage, allowing focus to remain on paths most likely to yield the shortest travel distance.

The logical sequence of reductions and selections ultimately produced an optimal route that covered all delivery points exactly once before returning to the depot. And it was summised that the optimised route which yielded a total distance of 13.7km had 26.7% reduction in travel distance as compared to the companies current practice of 18,7km and was 36.5% more efficient.

The final optimal delivery route was obtained as:1 → 5 → 4 → 6 → 8 → 9 → 2 → 10 → 7 → 3 → 1. This route yielded the minimum cumulative distance among all possible permutations, establishing it as the most efficient delivery sequence for the selected ten locations in Benin City

The study concluded that the Branch and Bound algorithm is an effective and reliable optimisation technique for route planning, especially for small to medium scale logistics operations. It ensured precision by exploring all possible routes logically while simultaneously reducing computational workload through bounding and pruning. Although the process was carried out manually, it maintained mathematical consistency and produced results that aligned with real geographical conditions in Benin City.

5.2 Recommendations

Based on the findings and conclusions of this study, several recommendations were made to improve delivery route planning and enhance logistics performance, particularly for small and medium-sized enterprises operating within urban environments such as Benin City.

1. **Adoption of the Branch and Bound algorithm in Delivery Planning:** Businesses engaged in regular delivery operations should adopt the Branch and Bound algorithm as a framework for route optimisation. The method is systematic, accurate, and transparent, and it allows delivery planners to determine efficient routes even with limited computational resources.
2. **Training and Capacity Building:** Logistics managers and delivery personnel should be trained in basic operational research and route optimisation techniques. Understanding the manual application of algorithms such as Branch and Bound can enable them to make informed routing decisions and reduce dependence on arbitrary or experience-based route selection.
3. **Integration with Digital Mapping Tools:** While this study relied solely on manual computation, future operational practices should consider integrating algorithms like Branch and Bound with digital mapping tools such as Google Maps or Geographic Information Systems (GIS) for faster and more dynamic optimisation. This will help in adjusting routes to account for real-time variables such as traffic or road conditions.
4. **Regular Route Review:** Delivery routes should be reviewed periodically using the same optimisation model to account for changes in road networks, customer locations,

and traffic flow. Regular reassessment ensures that routes remain efficient and responsive to operational realities.

5. **Policy Implications for Urban Logistics:** Urban delivery operations in Benin City and similar areas could benefit from incorporating mathematical optimisation methods into transport planning. Government agencies and business clusters should encourage the use of such models to improve delivery efficiency, reduce congestion, and minimise fuel consumption.
6. **Encouragement of Manual Problem-Solving in Logistics Education:** Educational institutions should continue to emphasise manual computational approaches such as the Branch and Bound method. This enhances problem-solving ability and helps learners understand the logic behind algorithmic optimisation before relying on software automation.

Future research may expand this study by automating the Branch and Bound algorithm using programming tools such as Python or MATLAB in order to handle larger datasets and reduce computational effort. Further studies may also incorporate real-time traffic information and dynamic routing variables to improve the accuracy of delivery route optimisation models. In addition, comparative analyses involving other optimisation techniques such as Genetic Algorithms or Ant Colony Optimisation could provide deeper insights into the most efficient methods for different logistics environment

REFERENCES

Adewara, K. (2016). Evaluating open source routing tools in vaccine delivery in Kano State. *Spatial Information Research*, 24, 225–234

Alvarez, P., Lerga, I., Serrano-Hernandez, A., and Faulin, J. (2018). The impact of traffic congestion when optimizing delivery routes in real time. A case study in Spain. *International Journal of Logistics Research and Applications*, *21*(5), 529-541.

Alvarez, P., Serrano-Hernandez, A., Lerga, I., and Faulin, J. (2024). Is time more important than distance to optimize freight delivery routes? An approach using the value of time. *Transportation Research Procedia*, *78*, 428-435.

Alves, F., Pacheco, F., Rocha, A. M. A., Pereira, A. I., and Leitão, P. (2021, September). Solving a logistics system for vehicle routing problem using an open-source tool. In *International Conference on Computational Science and Its Applications* (pp. 397-412). Cham: Springer International Publishing.

Anggraeni, M. S., Tazkiya, O. N., Mawandi, E., & Amarilies, H. S. (2023). Optimization of Fuel Distribution Routes for Green Logistics in Multi Compartment Vehicle Routing Problem (MCVRP) using Branch and Bound Algorithm (Case Study: Boyolali Fuel Terminal).

Bin-shuang, L. (2010). Applying Google Maps API in logistics monitoring and management information system. *Machinery Design and Manufacture*

Bostan, S., Güney, C., Köse, Ü., Aslan, S., and Kamaşak, M. (2022, September). Distribution Planning and Route Optimization in Cargo Delivery. In *2022 7th International Conference on Computer Science and Engineering (UBMK)* (pp. 257-262). IEEE.

Bura, D., Singh, M., and Nandal, P. (2019, February). Predicting secure and safe route for women using google maps. In *2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon)* (pp. 103-108). IEEE.

Chandra, R., & Prasetyo, T. A. (2024). Design and Implementation of the Shortest Path Navigation in Samosir District using Branch and Bound Algorithm. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, *8*(2), 242-249.

Crépin, A. S., and Polasky, S. (2021). Decision analysis based on optimization. In *The Routledge Handbook of Research Methods for Social-Ecological Systems* (pp. 400-411). Routledge.

Dayaratne, D. T., and Gunasekara, S. (2020). Dynamic Route Planning (DRP) Module for Distribution Business. *CINEC Academic Journal*, *4*.

De Vries, H. (2017). Evidence-Based Optimization in Humanitarian Logistics. [Doctoral Thesis, Erasmus University Rotterdam]. Erasmus Universiteit Rotterdam (EUR). 1-305

Dobers, K., Klukas, A., Lammers, W., Laux, M., Mauer, G., and Schneider, M. (2013). Green logistics: Optimization approaches for resource-efficient logistics services.

In *Efficiency and Logistics* (pp. 149-161). Berlin, Heidelberg: Springer Berlin Heidelberg.

Fu, C., Wang, Y., Xu, Y., and Li, Q. (2010). The logistics network system based on the Google Maps API. *2010 International Conference on Logistics Systems and Intelligent Management*, 3, 1486–1489

Giri, S., and Vora, N. (2024). Advanced Route Optimization using Hybrid Algorithms and Road-based Distance Calculation. *IRO Journal on Sustainable Wireless Systems*, *6*(3), 273-291.

Grace, A. L., and Thenmozhi, M. (2023, July). Optimizing Logistics with Regularization Techniques: A Comparative Study. In *2023 IEEE World Conference on Applied Intelligence and Computing (AIC)* (pp. 338-344). IEEE.

Guevara, C. E., Oetama, R., and Tjahjana, D. (2025). Web-based logistics scheduling system utilizing cheapest insertion heuristic algorithm. *G-Tech: Jurnal Teknologi Terapan*, 9(1), 112–119

Hamiz, M., Haron, H., Sanusi, A., Bakri, M., and Nazaruddin, N. (2018). Saving Matrix Method for Efficient Distribution Route Based on Google Maps API. *Journal of Telecommunication, Electronic and Computer Engineering*, 10, 183-18

Hintoro, S., Kosasi, D., David, S., Kuway, S., and Gat, G. (2023). Genetic algorithm application in route optimization using Google Maps API. *CCIT (Creative Communication and Innovative Technology) Journal*, *16*(2), 258–276.

Holland, C., Levis, J., Nuggehalli, R., Santilli, B., and Winters, J. (2017). UPS optimizes delivery routes. *Interfaces*, 47(1), 8–23

Horng, S., and Yenradee, P. (2023). Delivery Service Management System Using Google Maps for SMEs in Emerging Countries. *Computers, Materials and Continua*, *75*(3).

Hsieh, F. S. (2020). Vehicle routing based on discrete particle swarm optimization and Google Maps API. *International Journal of Scientific and Technology Research*, 9(4), 1252–1257

Huang, Y. (2010). Google Maps API in the Logistics Transportation Management System. *Logistics Engineering and Management*

Ikai, R., Miyagi, S., and Sakai, O. (2021, May). Logistics route planning in agent-based simulation and its optimization represented in higher-order Markov-chain networks. In *International Conference on Complex Networks* (pp. 38-50). Cham: Springer International Publishing.

Iliev, I., Rasheva-Yordanova, K., and Borissova, D. (2024, June). Challenges in Geolocation for Logistics Delivery: A Case Study on the Accuracy in Bulgaria of Google Maps Directions API. In *International Symposium on Business Modeling and Software Design* (pp. 283-292). Cham: Springer Nature Switzerland.

Ismail, R. J., Shukur, M. H., and Ismael, S. J. (2022). Route Discovery Development for Multiple Destination Using Artificial Ant Colony: Google Map Case Study. *Cihan University-Erbil Scientific Journal*, *6*(2), 41-48.

Kidiyur, M. D., and Logeshwaran, J. (2024, November). Optimizing Logistics Operations through AI-Data Processing model. In *2024 2nd International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT)* (Vol. 1, pp. 501-505). IEEE.

Krishnan, S. P. T., and Gonzalez, J. L. U. (2015). Using Google APIs. In *Building Your Next Big Thing with Google Cloud Platform: A Guide for Developers and Enterprise Architects* (pp. 27-49). Berkeley, CA: Apress.

Kwon, J., and Jeong, J. (2022, October). A parcel delivery scheduling scheme in road networks. In *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)* (pp. 1750-1755). IEEE.

Lamata, M., Pelta, D., and Verdegay, J. (2017). Optimization problems as decision problems: The case of fuzzy optimization problems. *Information Sciences, 460*, 377–388

Liu, L. P., Wang, M. Z., and Li, S. X. (2019). Route optimization model for hazardous materials transportation considering traffic restrictions and delivery time windows. *DEStech Transactions on Computer Science and Engineering*, 362-368.

Lubinska, B. (2020). *Asset Liability Management Optimization: A Practitioner's Guide to Balance Sheet Management and Remodelling: Chapter Four*. John Wiley and Sons. https://doi.org/10.1002/9781119635529.ch4

Malhotra, I., Chandra, P., and Majumdar, S. K. (2019). Route Optimization Application using Server-Client Architecture and Google APIs. *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*, 210–214

Masuda, K., Urata, Y., Kitada, T., Toyoda, S., Nishimura, S., and Haga, T. (2021). On-demand vehicle route planning system for logistics MaaS. *Sumitomo Electric Technical Review*, *92*, 79–84.

Medoh, C. N., and Telukdarie, A. (2020). Logistics optimization: A cyber physical model. *International Journal of Business Analytics (IJBAN)*, *7*(1), 54-76.

Mishra, N., and Tiwari, A. (2025, February). Strengthening Acquiring Knowledge for Optimizing Dynamic Delivery Routes. In *2025 International Conference on Intelligent Control, Computing and Communications (IC3)* (pp. 1081-1087). IEEE.

Mubarak, A. Z., Mawengkang, H., & Suwilo, S. (2024). Enhancing Vehicle Routing Efficiency through Branch and Bound and Heuristic Methods. *Sinkron: jurnal dan penelitian teknik informatika*, *8*(3), 1463-1472.

Nadhila, A. A., Ariyanto, Y., and Syaifudin, Y. W. (2024). Clusterization of MSMe and Warehouse Locations for Efficiency of Courier Placement. *Journal of Evrímata: Engineering and Physics*, 144-149.

Radharamanan, R., Sortica, G. B., & Camargo, M. E. (1984). A Branch and Bound Model for the Transportation Routing Problem. In *Mathematical Modelling in Science and Technology* (pp. 877-882). Pergamon.

Ralphs, T. K. (2003). Parallel branch and cut for capacitated vehicle routing. *Parallel Computing*, *29*(5), 607-629.

Sun, C. (2024). *Research on the optimisation of electric coal transportation routes based on the Branch and Bound method*. *Industry Science and Engineering*.

Tang, W. (2025). Optimization of urban terminal delivery routes using fuzzy genetic algorithm and its practical application. *Journal of Computational Methods in Sciences and Engineering*, 14727978251323125.

Verbytskyi, Y. (2023). Delivery routes optimisation using machine learning algorithms. Eastern Europe: *Economy, Business and Management*, 1(1(38)), 85–89

Whitaker, D. (2014). *Branch and Bound*. In *Encyclopaedia of Statistical Sciences*.

Willow, D., Fernando, R., Tanjaya, W., Putri, H. A., and Suryaningrum, K. M. (2023, October). Branch and Bound Algorithm Implementation on Package Delivery Route. In *2023 5th International Conference on Cybernetics and Intelligent System (ICORIS)* (pp. 1-6). IEEE.

Wu, M. Y., Ke, C. K., and Lai, S. C. (2022). Optimizing the routing of urban logistics by context-based social network and multi-criteria decision analysis. *Symmetry*, *14*(9), 1811.

APPENDIX
DATA

VIBOI VENTURES

S/N	LOCATIONS	LATITUDE	LONGITUDE
1	Coca-Cola depot Benin City, Edo	6.35482° N	5.63281° E
2	Mama Pastor	6.37385° N	5.62967° E
3	Mama Mabel	6.35500° N	5.63945° E
4	Madam Lolo	6.35255° N	5.63338° E
5	Miss Jane	6.34971° N	5.63645° E
6	Peter Star	6.35554° N	5.63527° E
7	The Nurse	6.35516° N	5.64111° E
8	Big Mommy	6.36086° N	5.62317° E
9	Mama Samuel	6.36303° N	5.63156° E
10	Mama Chidima	6.37125° N	5.63425° E