Hybrid Ant Colony Optimization Algorithm for Dynamic Routing in Software Defined Networking

100054, Eddah Babu

Submitted in partial fulfilment of the requirements for the degree of Master of Science in Data Science and Analytics of Strathmore University

Strathmore Institute of Mathematical Sciences and iLab Africa

Strathmore University

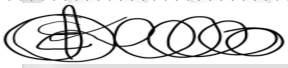
Nairobi, Kenya

June 2025

Declaration

I declare that this work has not been previously submitted and approved for award of a degree by this or any other University. To the best of my knowledge and belief, the dissertation contains no material previously published or written by another person except where due reference is made in the dissertation itself.

Name: **Babu Eddah Waruguru**

Signature: 

Date: May 20, 2025

Approval

The dissertation of Babu, Eddah Waruguru was reviewed and approved by:

Name: **Dr. Allan Omondi**

Senior Lecturer,

School of Computing and Engineering Sciences, Strathmore University.

Name:**Dr. Godfrey Achono Madigu**

Dean,

Strathmore Institute of Mathematical Sciences, Strathmore University.

Name:**Prof. Bernard Shibwabo**

Director of Graduate Studies,

Strathmore University.

Abstract

In dynamic network environments where traffic patterns were expected to fluctuate rapidly and unforeseen events like link failures could occur, routing protocols needed to swiftly adapt to maintain optimal performance. To optimize network routing, this research focused on key input parameters such as packet loss ratio, average packet delay, bandwidth utilization, and router processor load, which were essential for understanding network performance. The approach involved developing a mathematical algorithm designed to process these parameters to generate optimization variables, including the required number of routers, necessary link bandwidth capacity, and router processing capacity. This algorithm enabled Internet Service Providers to enhance performance quality through real-time adaptivity, ensuring efficient data transmission, reduced latency, and minimal packet loss. The methodology included conducting interviews with Internet Service Providers in Kenya to gather data and insights on their adoption of technologies like Software Defined Networking and Network Function Virtualization. These interviews provided valuable information on the state of network routing and the challenges faced by Internet Service Providers. By integrating this data with the algorithm, we aimed to create a dynamic routing algorithm that would swiftly adapt to changing network conditions and optimize routing in real-time, ultimately enhancing network reliability and performance.

KEY WORDS: Dynamic, Network, Algorithm

Table of contents

List of figures

List of tables

List of abbreviations

ACO	Ant Colony Optimization	API	Application Programming Interface
BFGS	Broyden-Fletcher-Goldfarb-Shanno	BCR	Box-covering based Routing
DE	Differential Evolution	HACO	Hybrid Ant Colony Optimization
ISP	Internet Service Provider	NFV	Network Function Virtualization
OSPF	Open Shortest Path First	SA	Simulated Annealing
SDN	Software Defined Networking	TOC	Theory of Change

Acknowledgements

I would like to express my heartfelt gratitude to everyone who supported and guided me throughout the journey of completing this dissertation. Above all, I give thanks to the Almighty God for His boundless grace, strength, and guidance that have sustained me through every stage of this academic pursuit.

I am deeply appreciative of my supervisor, Dr. Allan Omondi, for his steadfast support, insightful feedback, and invaluable mentorship. His guidance has played a pivotal role in shaping both the direction and the quality of this research, greatly contributing to my academic development.

My sincere thanks also go to my fellow classmates for their continuous encouragement, collaboration, and friendship throughout the course. I am equally grateful to the administrative and academic staff of Strathmore University, as well as the broader university community, for providing essential resources and fostering an environment that supports learning and research.

Finally, I acknowledge the contributions of the scholars and researchers whose work laid the groundwork for this study. Their commitment to advancing knowledge has been a constant source of inspiration and insight during this research journey.

Chapter 1

Introduction

1.1 Background

Network routing optimization has become increasingly critical in modern communication systems, driven by the need for improved performance quality, the introduction of new services and the continuous evolution of technology. Efficient routing of data in communication networks was essential to meet the demands of modern digital services. As networks grew in complexity and scale, ensuring low latency, high availability, and optimal throughput became increasingly difficult. These challenges required innovative solutions that could adapt to real-time changes in traffic conditions, system loads, and service demands (Raouf and Askr, 2019).

Several technological shifts contributed to this evolution. The growth of cloud computing, big data analytics, and software-defined systems placed higher requirements on network infrastructures, requiring flexible and dynamic routing solutions (El-Hefnawy et al., 2022). Software-defined networking (SDN) and network function virtualization (NFV) emerged as key enablers of such flexibility by allowing centralized control and on-demand reconfiguration of routing paths.

Data centers also experienced changes in traffic patterns, from traditional north-south client-server interactions to more frequent east-west machine-to-machine communications, creating the need for efficient lateral traffic handling (Montazerolghaem et al., 2017). This, along with the increased use of personal devices in enterprise environments, added layers of complexity to routing processes.

The explosion of big data further compounded these challenges. As noted in Vicentini et al. (2019), the increase in traffic volumes demanded higher bandwidth and smarter routing strategies to maintain performance and reduce congestion. In addition, data centers were under pressure to improve energy efficiency, with routing playing a critical role in reducing the energy footprint of large-scale systems.

Key network performance indicators such as packet loss ratio, average delay, bandwidth utilization, and processor load were essential in evaluating and optimizing routing decisions. Mathematical models and algorithms that account for these parameters allowed network engineers to enhance routing efficiency by adaptively allocating resources and avoiding congestion hotspots.

The adoption of SDN-based dynamic routing frameworks provided a foundation for developing context-sensitive scalable routing strategies. Although many Internet Service Providers (ISPs) in developing regions had not fully transitioned to these technologies, they were expected to benefit significantly from such optimizations. Addressing the unique challenges in these environments required tailored frameworks grounded in both global research and local infrastructure realities.

1.2 Problem Statement

Traditional static routing protocols were limited by their inability to respond to real-time network dynamics. These protocols predefined fixed paths for data transmission, failing to adapt to changes such as congestion, link failures, or varying traffic demands (Castro et al., 2012). This often led to inefficient utilization of network resources, higher latency, and decreased reliability in data delivery.

In modern heterogeneous network environments, traffic loads fluctuated rapidly due to factors such as high-bandwidth applications, virtualization, and the growing number of connected devices. Static routing approaches struggled to maintain performance under these conditions,

especially in contexts where infrastructure constraints limited the adoption of more dynamic systems (Karim and Khan, 2011).

Despite the emergence of SDN and NFV technologies, their uptake remained limited in many developing regions, primarily due to costs, complexity, and infrastructure limitations. The result was persistent performance degradation, particularly in networks supporting critical services such as banking, cloud access, and enterprise communication.

To address these issues, there was a need for a hybrid routing optimization algorithm capable of real-time adaptation. Such an algorithm would integrate traffic metrics, predict congestion, and dynamically reroute data to maintain quality of service. Its adoption by key stakeholders including ISPs, data centers, and large enterprises was essential for improving the overall performance, reliability, and scalability of network systems in dynamic environments.

1.3 Aim

The main aim of this research was to design and implement a dynamic network routing optimization algorithm that improved traffic flow, reduced congestion, and enhanced overall performance in modern networks through real-time adaptability.

1.4 Research Objectives

i. To identify the performance metrics relevant to dynamic network routing optimization, including packet loss, latency, bandwidth utilization, and router processor load.

ii. To review and evaluate existing technologies such as SDN and NFV, and related optimization algorithms.

iii. To develop a hybrid algorithm incorporating real-time performance data for traffic management and routing optimization.

iv. To test the algorithm in a simulated network environment and assess its effectiveness under various traffic conditions.

1.5 Research Questions

i. What were the key challenges in optimizing routing within dynamic environments?

ii. Which existing technologies and algorithms were best suited to address these challenges?

iii. How could a hybrid routing algorithm be designed to improve real-time adaptability and performance?

iv. How effective was the proposed algorithm in reducing latency and congestion in a simulated environment?

1.6 Justification

This research was essential for key stakeholders, including Internet Service Providers (ISPs), large enterprises, data centers, and governmental organizations, as it addressed the critical challenge of managing dynamic and rapidly fluctuating network traffic. By developing a dynamic network routing optimization algorithm, the research aimed to enhance network efficiency, reduce congestion, and improve service quality. This enabled stakeholders to better manage their networks, leading to more reliable and efficient data transmission. Ultimately, the benefits extended to end users, who experienced improved internet connectivity and access to digital services.

1.7 Assumptions

It was assumed that the network environments in which the dynamic routing optimization algorithm would be implemented were equipped with sufficient computational resources

and infrastructure to support real-time data processing and adaptive routing. Additionally, it was assumed that stakeholders, including Internet Service Providers and other organizations, would be willing to integrate and adopt the new algorithm into their existing network management systems. The research also presumed that the algorithm would be compatible with current technologies, such as Software Defined Networking (SDN) and Network Function Virtualization (NFV), and that it would effectively address the dynamic nature of network traffic as anticipated. Furthermore, it was assumed that stakeholders would provide accurate and comprehensive data for testing and validation purposes.

1.8 Scope and Limitations

1.8.1 Scope

The scope of this project centered on developing a dynamic network routing optimization algorithm specifically designed to manage and adapt to real-time network traffic conditions within Kingdom Bank's infrastructure. The system requirements within the project's boundaries included real-time data processing capabilities, integration with Kingdom Bank's existing network management systems, and addressing key performance metrics such as congestion, load balancing, and latency. The project also focused on creating algorithms compatible with current technologies like Software Defined Networking (SDN) and Network Function Virtualization (NFV). However, the scope excluded requirements related to physical network infrastructure, such as hardware upgrades or changes to network topology, and did not extend to non-networking systems or applications outside the domain of network traffic management.

1.8.2 Limitations

During the project, several challenges arose, including obtaining permission to access and use real-time network data from Kingdom Bank, ensuring the accuracy of test results, and

addressing potential gaps in expertise related to advanced technologies. Specifically, gaining access to Kingdom Bank's network data required navigating institutional approvals and adhering to data privacy regulations. To address this, the project sought formal agreements and ensured compliance with all legal and ethical standards. Ensuring accurate test results involved rigorous validation procedures and the use of reliable testing environments. Additionally, to bridge any knowledge gaps, the project leveraged expertise from field specialists and consulted existing literature and case studies related to dynamic network routing and associated technologies. Collaboration with Kingdom Bank's technical team and industry experts was essential in overcoming these challenges and ensuring successful project outcomes.

Chapter 2

Literature Review

2.1 Introduction

The Literature Review for this research was structured to provide a comprehensive foundation for the study of network routing optimization. It began with the Theoretical Framework, where two essential theories, Theory of Change and Optimization Theory, were explored to establish the conceptual basis for the research. This was followed by a review of Existing Algorithms and Architectures, examining the current methodologies and technical solutions in network routing. The Empirical Framework then highlighted similar solutions developed by other researchers, focusing on major breakthroughs, current trends, and relevant applications and tools. Finally, the review identified the Research Gap that this study aimed to address, leading to the development of a Conceptual Framework that guided the research design and methodology. This structure ensured a logical flow from foundational theories to practical applications, positioning the research within the broader context of existing knowledge and innovation.

2.2 Theoritical Framework

2.2.1 Optimization Theory

Optimization Theory was a mathematical discipline that identified the best possible solution to a problem within a given set of constraints. In network systems, this theory was crucial for developing efficient algorithms that determined optimal paths, resource allocation, and system placements to ensure smooth and efficient data flow. This section delved into the

intricacies of Optimization Theory, its relevance to Software-Defined Networking (SDN), and its application in traffic engineering, resource allocation, Network Function Virtualization (NFV) placement, energy efficiency, and load balancing.

Objective Functions

At the heart of any optimization problem lay the objective function, a mathematical representation of the goal to be achieved. In the context of SDN, objective functions were designed to either minimize or maximize specific network parameters depending on the operational requirements.

i. Traffic Engineering: In traffic engineering, the objective function typically aimed to minimize network congestion. This involved routing traffic in such a way that it avoided overloaded network segments, thereby distributing the data flow evenly across the network. By minimizing congestion, network performance improved, leading to reduced latency and increased data throughput.

ii. Resource Allocation: Resource allocation within a network involved the optimal distribution of processing power, memory, and bandwidth among various services. Here, the objective function might have been designed to maximize resource utilization, ensuring that no processing power or bandwidth went to waste. This was particularly critical in environments where resources were limited, and efficient allocation was necessary to maintain service quality.

iii. Network Function Virtualization (NFV) Placement: NFV placement dealt with the strategic positioning of network functions like firewalls, load balancers, and intrusion detection systems within the network. The objective function in this case might have aimed to maximize bandwidth utilization by placing these functions in locations that optimized data flow paths, thus reducing unnecessary data transmission and enhancing overall network efficiency.

iv. Energy Efficiency: In energy-efficient networking, the objective function often sought to minimize energy consumption while maintaining network performance. This might have

involved dynamically adjusting the power states of network components or optimizing the routing paths to reduce the energy required for data transmission, thereby striking a balance between performance and energy usage.

v. Load Balancing: Load balancing was essential for preventing network bottlenecks. The objective function here was designed to distribute traffic or computational load evenly across multiple servers or network paths. By avoiding overloading any single network component, load balancing ensured smoother operation and better performance across the network.

These objective functions were central to SDN optimization, guiding the design and implementation of algorithms that managed network resources effectively (Ruder, 2016).

Continuous Function Optimization versus Combinatorial Function Optimization

Optimization problems could be broadly categorized into continuous and combinatorial function optimization, depending on the nature of the decision variables involved.

i. Continuous Function Optimization: This type of optimization dealt with problems where the decision variables could take any value within a given range. Continuous optimization was common in scenarios where network parameters, such as bandwidth allocation or latency, could be adjusted incrementally. Techniques like gradient descent were often used to find the optimal solution by iteratively improving the decision variables.

ii. Combinatorial Function Optimization: In contrast, combinatorial optimization involved problems where the decision variables were discrete, often representing choices between different network configurations or routing paths. Combinatorial optimization was particularly relevant in network routing, where the objective was to select the best path from a finite set of possible routes. Techniques such as branch-and-bound or genetic algorithms were typically employed to solve these problems.

The distinction between continuous and combinatorial optimization was crucial for selecting the appropriate optimization techniques and algorithms for a given network problem (Kennedy and Eberhart, 1995).

Differentiable versus Non-Differentiable Objective Functions

Another important classification in optimization theory was based on whether the objective function was differentiable or non-differentiable.

i. Differentiable Objective Functions: Differentiable functions were those for which the derivative existed, allowing for the use of calculus-based methods to find the optimal solution. These functions were common in continuous optimization problems where small changes in the decision variables led to predictable changes in the objective function. For example, minimizing latency or maximizing bandwidth utilization could often be modeled with differentiable functions, where gradient-based methods such as gradient descent or Newton's method were effective.

ii. Non-Differentiable Objective Functions: Non-differentiable functions, on the other hand, did not have a well-defined derivative. These functions often arose in combinatorial optimization problems, where the objective function might involve discrete decisions, such as selecting a specific routing path or network configuration. In these cases, traditional calculus-based methods were not applicable, and alternative optimization techniques, such as direct search methods or stochastic algorithms, were required.

Understanding the nature of the objective function was essential for selecting the most appropriate optimization technique, as the differentiability of the function determined the set of algorithms that could be effectively applied (Koutsoupias and Papadimitriou, 2011).

Categories of Algorithms for Differentiable Objective Functions

For differentiable objective functions, several categories of optimization algorithms could be employed, each suited to different types of problems:

i. Bracketing Algorithms: Bracketing algorithms were used to identify the interval within which the optimal solution lay. These algorithms were particularly useful in one-dimensional optimization problems. Examples included the Golden-section search and Brent's method, which iteratively narrowed down the search interval until the optimal solution was found.

ii. Local Descent Algorithms: Local descent algorithms focused on moving from the current solution to a better one by following the gradient of the objective function. Gradient descent was the most widely used local descent algorithm, especially in machine learning and neural networks, where it helped in minimizing loss functions.

iii. First-Order Algorithms: First-order algorithms, such as steepest descent and gradient descent, relied on the first derivative of the objective function to guide the optimization process. These algorithms were effective for smooth, well-behaved functions where the gradient provided a reliable direction towards the optimum.

iv. Second-Order Algorithms: Second-order algorithms, such as Newton's method and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, used both the first and second derivatives of the objective function. These algorithms converged faster than first-order methods, especially in problems where the objective function had a quadratic form. However, they were computationally more expensive due to the need to calculate second derivatives or approximate the Hessian matrix.

The choice of algorithm depended on the specific characteristics of the optimization problem, including the complexity of the objective function and the computational resources available (Michalewicz and Fogel, 2004).

Categories of Algorithms for Non-Differentiable Objective Functions

According to (Kirkpatrick et al., 1983a), for non-differentiable objective functions, alternative optimization algorithms were required, which did not rely on gradient information:

i. Direct Algorithms: Direct algorithms, such as the Nelder-Mead method, did not require derivative information and instead relied on evaluating the objective function at various points to find the optimal solution. These algorithms were useful for problems where the objective function was noisy or discontinuous.

ii. Stochastic Algorithms: Stochastic algorithms incorporated randomness into the search process, allowing them to escape local optima and explore a wider solution space. Examples

included Simulated Annealing and Genetic Algorithms, both of which were widely used in combinatorial optimization problems. These algorithms were particularly effective in large, complex networks where deterministic methods might struggle to find the global optimum.

iii. Population Algorithms: Population-based algorithms, such as Particle Swarm Optimization (PSO) and Differential Evolution (DE), operated on a population of potential solutions. These algorithms were inspired by natural processes, such as the social behavior of birds or the evolutionary process, and were well-suited for optimizing non-differentiable, multimodal functions.

Each category of algorithm offered unique strengths, making them suitable for different types of non-differentiable optimization problems. The selection of the appropriate algorithm was critical to the success of the optimization process, especially in complex network environments where the objective function might not be smooth or easily defined.

2.3 Existing Algorithms and Architectures

With the expansion of network infrastructures, effective resource allocation became critical to prevent congestion and maximize network utilization. In traditional network architectures, the control plane, responsible for decision-making, was tightly coupled with the data plane, which forwarded traffic. This rigid structure limited the ability of routing algorithms to collect real-time, comprehensive information about network conditions. As a result, routing decisions made without a global view led to inefficiencies, such as suboptimal utilization of network paths and increased congestion, especially in large-scale, dynamic environments.

Software-Defined Networking (SDN) introduced a paradigm shift by decoupling the control plane from the data plane, thereby centralizing network management and control. In SDN, the control logic, which dictated how data flowed through the network, was moved from individual network devices (like routers and switches) to a centralized software-based controller. This controller communicated with the network's physical devices via southbound APIs such as OpenFlow. By doing so, SDN enabled a global view of the network, allowing for more

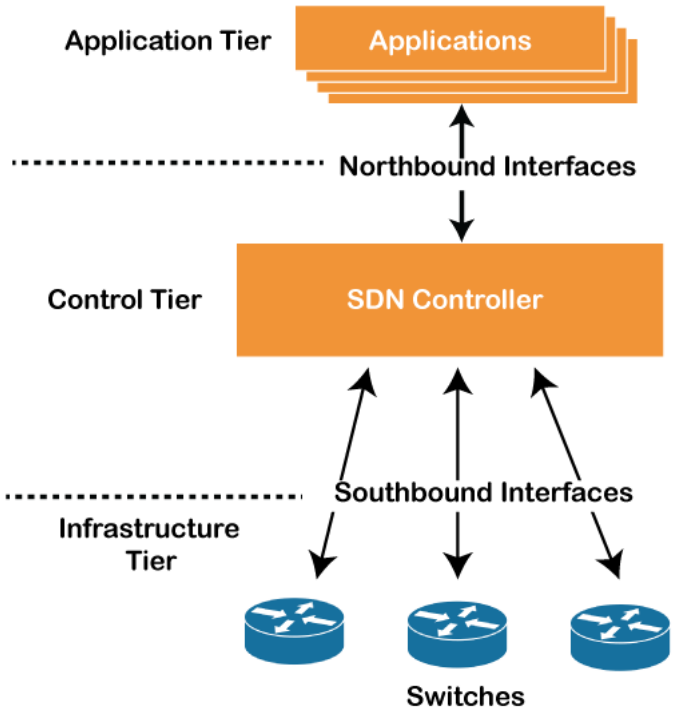


Figure 2.1: SDN Architecture

intelligent decision-making regarding data forwarding, traffic management, and resource allocation (McKeown et al., 2008).

At the heart of the SDN architecture was the interaction between the control and data planes, facilitated through southbound APIs. These APIs enabled the controller to monitor the state of the network and dynamically update forwarding paths, optimizing resource allocation and traffic flow. On the other hand, northbound APIs, such as RESTful APIs, allowed applications like routers, firewalls, and load balancers to communicate with the controller. This separation not only simplified network management but also enhanced flexibility by enabling dynamic adjustments to routing paths based on real-time network conditions (Foundation, 2014). Consequently, traditional routing algorithms, such as Open Shortest Path First (OSPF), were no longer efficient in SDN environments because they were not equipped to adapt quickly to dynamic traffic patterns.

To address the challenges posed by dynamic and stochastic traffic, algorithms like Hybrid Ant Colony Optimization (HACO) emerged as promising solutions. HACO effectively managed

HACO ALGORITHM

```
[ ]  1 function hybrid_aco(problem)
     2     initialize pheromone matrix τ
     3     initialize ant population P
     4     while not termination criterion is met
     5         for each ant in P
     6             construct a solution using pheromone information and local search
     7             update pheromone matrix based on the quality of the solution
     8         update best solution found so far
     9     return best solution found
```

Figure 2.2: Hybrid Ant Colony Optimization

randomness in datasets by combining random exploration with structured exploitation, making it particularly suitable for optimization problems involving inherent unpredictability. Inspired by the foraging behavior of ants, Ant Colony Optimization (ACO) simulated how ants explored multiple paths while leaving pheromone trails to mark successful routes, enabling the colony to converge on optimal solutions. HACO built on ACO by integrating additional techniques such as local search methods or evolutionary algorithms, enhancing both convergence speed and overall accuracy (Dorigo and Stützle, 2004). This combination allowed HACO to balance the exploration of new solutions with the refinement of existing ones, leveraging its bioinspired global search ability to solve complex optimization tasks. To further achieve network optimization goals like load balancing, a link centrality measurement was added during population initialization to enhance HACO's topological awareness. This adjustment improved the algorithm's ability to understand network structure, enabling more effective routing and resource allocation in dynamic environments.

Simulated Annealing (SA) was another algorithm capable of managing stochasticity, leveraging principles from thermodynamics. In SA, the algorithm began with a high probability of accepting worse solutions, much like the process of annealing metals at high temperatures. As the algorithm proceeded, the temperature parameter gradually decreased, reducing the acceptance rate of inferior solutions. This balance of exploration and exploitation, guided by randomness, allowed SA to escape local optima and explore a broader solution space. Its probabilistic decision-making mechanism made it well-suited for problems where randomness significantly impacted the quality of the solution (Kirkpatrick et al., 1983b).

Simulated Annealing (SA)

```
function simulated_annealing(problem)
    initialize solution x
    initialize temperature T
    while T > 0
        generate neighbor solution x'
        calculate ΔE = f(x') - f(x)
        if ΔE < 0 or exp(-ΔE / T) > random(0, 1)
            x = x'
        update T according to cooling schedule
    return x
```

Figure 2.3: Simulated Annealing

Nelder-Mead Simplex Search (NMSS)

```
function nelder_mead_simplex_search(problem)
    initialize simplex S
    while not converged
        order vertices of S by their function values
        calculate centroid C of the best N-1 vertices
        reflect worst vertex V through C to get V'
        if f(V') < f(V)
            replace V with V'
        else if f(V') < f(V_second_worst)
            contract simplex towards V'
        else
            contract simplex towards C
    return best vertex
```

Figure 2.4: Nelder-Mead Simplex Search

The Nelder-Mead Simplex Search, while deterministic in nature, offered a contrast to HACO and SA. It was a direct search method typically employed for solving nonlinear optimization problems without requiring gradient information. Although the algorithm did not inherently incorporate stochastic behavior, its geometric simplex-based approach could adapt to varying problem landscapes (Kreutz et al., 2015a). For environments where randomness affected the solution, modifications could introduce stochastic elements into Nelder-Mead's decision-making, making it a viable candidate for optimization in noisy or unpredictable conditions.

2.4 Empirical Framework

2.4.1 Major Breakthroughs

Over the past decade, advancements such as Network Function Virtualization (NFV) and centralized network programmability revolutionized how networks were designed and managed. NFV decoupled network functions from dedicated hardware, allowing for flexible deployment and scaling of services (Kirkpatrick et al., 1983a). This, combined with centralized control in SDNs, paved the way for more intelligent and adaptable networks. AI's integration into SDN further enhanced decision-making processes, leading to more efficient resource allocation and network optimization. By 2024, these breakthroughs laid the foundation for addressing increasingly complex and dynamic network environments.

2.4.2 Current Trends

In 2024, current trends included using transformer models to complement traditional optimization algorithms, particularly in SDNs. Transformer models, with their powerful ability to handle vast amounts of data, improved traffic prediction and resource allocation. Another trend was the growing focus on SDN security, where techniques like blockchain and AI were employed to enhance network security in distributed environments. The use of SDNs in IoT networks also gained traction, providing better scalability, reliability, and security for the increasing number of connected devices (Shenker et al., 2013a). Moreover, energy-efficient SDNs became a key focus, as they addressed the environmental impact of growing network infrastructures.

Enterprises now had access to a range of tools and applications for implementing SDNs. Three notable solutions included VMware NSX, which provided virtualization of network services, Cisco ACI for centralized control of complex data centers, and OpenDaylight, an open-source platform that promoted SDN deployment across a range of industries. Each tool offered unique advantages, from enhanced scalability to easier network management, making them ideal for enterprises looking to modernize their network infrastructures.

2.4.3 Tabular Summary of the Empirical Framework

The empirical framework concluded with a comparative table that summarized key studies and technologies. It highlighted the authors, publication dates, techniques used, and the merits and demerits of each solution. This allowed for a clear comparison of how different approaches had contributed to the advancement of SDN technologies and where gaps or limitations remained. The table format provided a concise yet informative way to present the empirical evidence gathered from multiple sources, promoting a better understanding of how SDN solutions had evolved.

Table 2.1: Summary of Techniques and Merits

Author(s)	Date	Technique	Merits
Kreutz et al.	2015	Comprehensive SDN survey, SDN architecture exploration	Explains the key concepts of SDN and its architecture
McKeown et al.	2008	OpenFlow protocol for SDN implementation	Enables network programmability and innovation
Dorigo & Stützle	2004	Ant Colony Optimization (ACO) for network routing	Bio-inspired, scalable, performs well in dynamic environments
Kirkpatrick et al.	1983	Simulated Annealing for traffic flow optimization	Escapes local optima, effective in stochastic environments
Shenker, Parulkar & McKeown	2013	Networking philosophy behind SDN	Highlights the benefits of centralized control and decoupling of planes

Table 2.2: Summary of Techniques and Demerits

Author(s)	Date	Technique	Demerits
Kreutz et al.	2015	Comprehensive SDN survey, SDN architecture exploration	Limited focus on specific applications or scalability challenges in large-scale networks
McKeown et al.	2008	OpenFlow protocol for SDN implementation	Lacks built-in security features and requires additional integration layers
Dorigo & Stützle	2004	Ant Colony Optimization (ACO) for network routing	Slow convergence in large-scale, complex networks
Kirkpatrick et al.	1983	Simulated Annealing for traffic flow optimization	Requires careful tuning of temperature parameters for optimal performance
Shenker, Parulkar & McKeown	2013	Networking philosophy behind SDN	No specific implementation details or case studies provided.

2.5 Research Gap and Hypothesis

A comprehensive exploration of Hybrid Ant Colony Optimization (HACO), Simulated Annealing (SA), and Nelder-Mead Simplex Search (NMSS) revealed several significant research gaps that underscored the need for further investigation. Firstly, HACO had demonstrated its efficacy in static environments; however, its performance in high-traffic, variable conditions typical of modern networks, such as those driven by the advent of 5G and the Internet of Things (IoT), remained largely uncharted (Kreutz et al., 2015b). This gap necessitated research into adapting HACO for scalability in large-scale networks. Additionally, the intersection of HACO with advanced technologies like machine learning and artificial intelligence was an area ripe for exploration, as these technologies could have enhanced HACO's

adaptability to fluctuating network conditions (McKeown et al., 2008). Furthermore, there was insufficient analysis regarding the energy efficiency implications of deploying HACO algorithms within Software-Defined Networking (SDN) frameworks, which was increasingly relevant in the pursuit of sustainable technology solutions (Dorigo and Stützle, 2004). The need for enhanced security measures in SDN environments also highlighted another vital gap, given the complexity and interconnectivity of modern networks (Kirkpatrick et al., 1983b). Similarly, the limitations of SA warranted attention, particularly its underexplored performance in dynamic environments where rapid fluctuations in network conditions could hinder optimization efforts (Shenker et al., 2013b). Most existing studies concentrated on static problems, emphasizing the need to adapt SA for high-traffic scenarios typical of SDNs. The tuning of temperature schedules in SA and their effects on convergence speed in high-dimensional spaces also required comprehensive analysis, particularly in real-time network conditions. Lastly, NMSS, while known for its simplicity and effectiveness, primarily operated within low-dimensional problem spaces. Its adaptation to the complexities of high-dimensional networks, particularly in SDN environments, posed a notable gap. Additionally, NMSS's sensitivity to initial simplex choices raised concerns about consistency and robustness, warranting further research on strategies to optimize the initialization process. By identifying and addressing these research gaps, this study not only contributed to a deeper understanding of HACO, SA, and NMSS but also emphasized their relevance and potential for innovation in contemporary network optimization scenarios, ultimately aiming for more efficient and resilient strategies in the evolving landscape of network technologies.

This research hypothesized that Hybrid Ant Colony Optimization would outperform both Simulated Annealing and the Nelder-Mead Simplex Search, especially in stochastic environments. HACO's strength lay in its ability to combine the advantages of random exploration with structured learning, providing a versatile and robust optimization framework. By comparing HACO, SA, and Nelder-Mead, the study aimed to identify the algorithm best suited for handling randomness in complex optimization problems, with HACO expected to demonstrate superior adaptability and performance due to its hybrid structure.

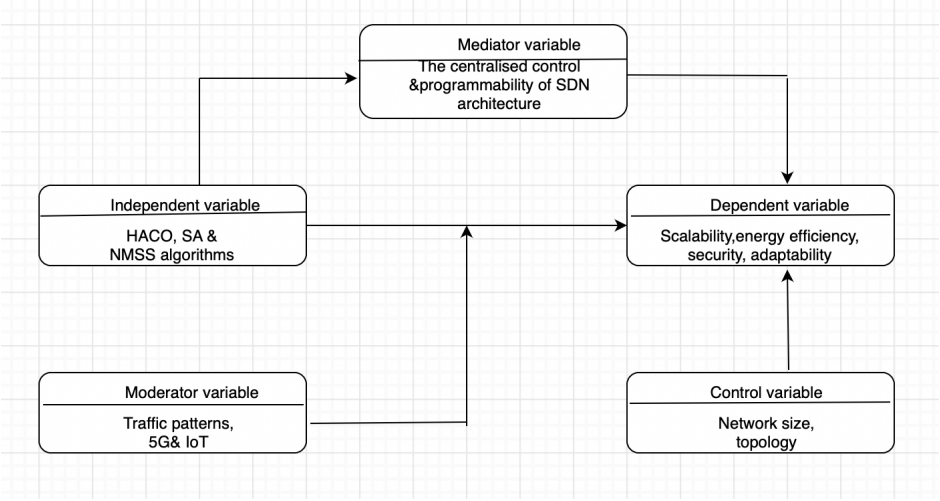


Figure 2.5: Conceptual Framework

2.6 Conceptual Framework

The Conceptual Framework on Hybrid Ant Colony Optimization (HACO), Simulated Annealing (SA), and Nelder-Mead Simplex Search (NMSS) within Software-Defined Networking (SDN) explored how these algorithms optimized key network performance metrics like scalability, energy efficiency, and security. SDN, with its centralized control and programmability, served as the foundation for deploying these optimization techniques. HACO, SA, and NMSS were expected to improve network management under dynamic conditions such as those seen with 5G and IoT, but research gaps remained, particularly regarding their adaptability in high-traffic environments (Shenker et al., 2013b). The integration of advanced technologies like machine learning and artificial intelligence could have further enhanced their performance. External factors, including fluctuating network conditions and increasing network complexity, challenged the robustness of these algorithms, but this study aimed to address those gaps and contribute to more resilient and efficient optimization strategies in modern networks.

Chapter 3

Methodology

3.1 Methodology

This chapter outlines the methodology adopted to develop and evaluate the Hybrid Ant Colony Optimization (HACO) algorithm for dynamic routing in Software Defined Networking (SDN). The methodology aligns with the CRISP-DM framework and consists of the following stages: business understanding, data understanding, data preparation, machine learning modelling, model evaluation, deployment, and ethical considerations.

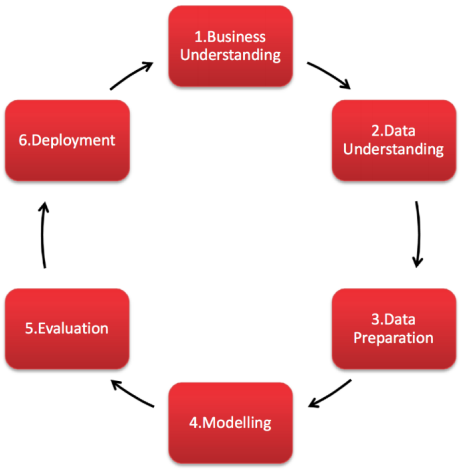


Figure 3.1: CRISP-DM framework

3.1.1 Business Understanding

The main objective of this research was to design a dynamic routing optimization model that responds adaptively to changing network conditions. Traditional static routing protocols lack the flexibility to deal with traffic fluctuations, link failures, and load balancing challenges. Interviews were conducted with technical leads and network engineers from Kingdom Bank to identify their network performance pain points. The feedback highlighted the need for an intelligent, responsive routing system capable of leveraging real-time data. This business understanding guided the formulation of the problem and selection of performance metrics such as latency, packet loss, and bandwidth utilization.

3.1.2 Data Understanding

To accurately model and simulate network behavior, data were collected from Kingdom Bank's network infrastructure, which incorporates Software Defined Networking (SDN) and Network Function Virtualization (NFV) capabilities. The dataset comprised approximately 30,000 records gathered over a two-month observation period. These records were extracted from controller logs, router performance metrics, traffic flow reports, and switch statistics using SNMP (Simple Network Management Protocol) and OpenFlow logs. The data

spanned both peak and off-peak network hours, enabling the model to account for real-world variability.

Table 3.1: Variables Used in the Study

Variable Name	Description	Data Type
node_id	Unique identifier for a router or switch	Integer
link_capacity	Maximum bandwidth of a link (in Mbps)	Float
traffic_flow	Volume of data over time (packets/second)	Float
latency	Time taken for a packet to traverse the link (in ms)	Float
packet_loss	Percentage of lost packets during transmission	Float
router_load	CPU utilization percentage of routers	Float
link_status	Boolean indicator of whether a link is active or down	Boolean

3.1.3 Data Preparation

The raw dataset contained noise, outliers, and missing entries which required extensive preprocessing. The following steps were undertaken:

i. **Data Cleaning:** Removed duplicates, incorrect formats, and inconsistencies.

ii. **Outlier Detection and Treatment:** Applied the Interquartile Range (IQR) method to detect outliers in latency and traffic variables. Winsorization was applied to cap extreme values.

iii. **Missing Data Handling:** For continuous variables, missing values were imputed using linear interpolation; categorical attributes such as link_status were filled using mode.

iv. **Feature Engineering:** Created additional attributes such as network congestion index (ratio of actual traffic to link capacity) and average load per route.

v. **Standardization:** Normalized numeric attributes using z-score normalization to maintain consistency across scales.

3.1.4 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) was crucial in validating data integrity, uncovering trends, and guiding the design of the HACO model. Various statistical and visual tools were employed to examine the network dataset. The analysis included the use of histograms, boxplots, heatmaps, and correlation matrices to:

i. Identify patterns in link utilization over time,

ii. Examine latency and packet loss distributions,

iii. Detect anomalies such as zero-capacity links or idle routers, and

iv. Understand multicollinearity between network variables.

To facilitate these insights, visualization libraries such as **Seaborn** and **Matplotlib** were utilized to generate bar plots, heatmaps, histograms, and scatterplots. **Pandas Profiling** provided automated reports that summarized distributions, missing values, and descriptive statistics. Additionally, **Principal Component Analysis (PCA)** from `scikit-learn` was applied for dimensionality reduction, enhancing understanding of the correlation structure among network features and aiding in feature selection for the subsequent modeling phase.

Key insights included discovery of peak traffic congestion at specific times of day and notable correlations between router CPU load and link latency.

3.1.5 Machine learning Modeling

The Hybrid Ant Colony Optimization (HACO) algorithm synergized the strengths of Ant Colony Optimization (ACO) and Simulated Annealing (SA) to achieve dynamic, adaptive routing in Software Defined Networks (SDNs). ACO provided a robust global search

mechanism through collective agent behavior, while SA contributed local refinement to reduce congestion in critical paths.

The model design involved the following sequential phases:

i. **Initialization:** The algorithm began by establishing the network topology as a node graph, initializing pheromone matrices, and setting key performance thresholds.

ii. **Ant Path Construction:** Artificial ants constructed routes by probabilistically selecting next hops using a heuristic function based on real-time link utilization and latency metrics.

iii. **Local Search via SA:** Candidate routes were refined through Simulated Annealing to minimize localized congestion and balance network load.

iv. **Pheromone Update:** Paths demonstrating lower delay and packet loss were reinforced with higher pheromone values, while older pheromones decayed over time to support exploration and avoid stagnation.

The HACO algorithm was implemented in Python using graph-based search techniques and iterative pheromone updates. The following libraries and tools were used:

i. **NetworkX:** Modeled the SDN topology as a weighted graph for efficient path computation.

ii. **NumPy:** Handled matrix operations for pheromone levels, cost evaluations, and routing metrics.

iii. **Ryu Controller:** Enabled real-time integration with SDN for dynamic flow rule management.

The implementation followed these operational steps:

i. Converted raw topology data into a weighted graph structure with latency and capacity as edge attributes.

ii. Initialized pheromone values and simulated multiple ant traversals across the network.

iii. Applied Dijkstra's algorithm during SA-based local search to refine promising high-pheromone paths.

iv. Performed global pheromone updates by reinforcing best-performing paths based on packet delivery metrics.

v. Introduced pheromone evaporation and convergence thresholds to stabilize route selection and avoid premature convergence.

To further enhance adaptability, reinforcement learning mechanisms were explored to dynamically adjust pheromone intensities in response to network feedback, including packet loss, delay, and jitter contributing to more intelligent routing over time.

3.1.6 Performance Evaluation

The algorithm was evaluated in a controlled simulation environment using Mininet with a Ryu controller. Performance was benchmarked against Open Shortest Path First (OSPF) protocol. Metrics used included:

i. **Average Packet Loss Rate:** Proportion of dropped packets across all test scenarios.

ii. **Average Latency:** Round-trip time across dynamically selected paths.

iii. **Bandwidth Utilization:** Percentage use of link capacity.

iv. **Convergence Time:** Time required for algorithm to stabilize after a topology change.

HACO outperformed OSPF in both congested and failure-induced scenarios, proving its robustness and adaptability.

3.1.7 Deployment Architecture

The HACO system was deployed through a Flask-powered web dashboard allowing real-time monitoring and configuration. The system components included:

i. Frontend: Built with Flask and JavaScript for real-time dashboarding and interaction

ii. Backend: Python engine with HACO embedded within the Ryu SDN controller

iii. Database: PostgreSQL 13 used for logging network state, predictions, and performance

3.1.8 Ethical Considerations

Access to sensitive network data was granted by Kingdom Bank under strict data protection agreements that ensured compliance with internal IT policies and national data protection regulations. Prior to data access, a formal memorandum of understanding was signed outlining data handling, retention, and security protocols. All personally identifiable information was removed or masked through anonymization techniques such as hashing and aggregation. Experiments were conducted in a virtualized, sandboxed environment separate from the live operational network to eliminate any risk of service disruption or data leakage. Additionally, role-based access controls were enforced to limit data exposure to authorized research personnel only. Ethical clearance for the study was formally obtained from the institutional review board, affirming that all research procedures met ethical and legal standards for data handling and experimentation.

3.1.9 Summary

This methodology provided a comprehensive pipeline from problem formulation through to algorithm deployment. The integration of domain knowledge, real network data, and hybrid metaheuristics led to the development of a robust and scalable solution for dynamic network routing optimization.

Chapter 4

System Analysis, Design, and Architecture

4.1 Introduction

This chapter outlined the design and architecture of the proposed system. Provided an overview of the system architecture, database schema, and key modeling components such as entity-relationship diagrams (ERD), class and sequence diagrams, and wireframes where applicable. The objective of this chapter was to ensure that the system design aligned with the research goals and supported the functionalities required for implementation.

4.2 Business Insights from ISP Interviews

To enhance network routing optimization, interviews were conducted with ISPs in Kenya to understand their existing challenges and adoption strategies for Software Defined Networking (SDN). The key findings were as follows:

Challenges Faced by ISPs in Dynamic Routing

i. **Traffic Congestion & Bottlenecks** – Many ISPs struggled with real-time congestion management, especially in high-demand urban areas.

ii. **Latency Issues** – Traditional static routing methods failed to adapt to changing traffic conditions, leading to increased latency.

iii. **Inefficient Bandwidth Utilization** – A lack of real-time optimization resulted in either underutilization or overloading of network paths.

iv. **High Operational Costs** – Managing network congestion manually led to inefficiencies and increased costs.

v. **Slow Failure Recovery** – Network failures took longer to resolve due to rigid routing structures.

ISP Adoption of Software Defined Networking (SDN)

i. **Limited Implementation** – While some ISPs had experimented with SDN, full deployment remained low due to concerns over cost and infrastructure overhaul.

ii. **Benefits Identified** – Those who had adopted SDN reported improved control over network traffic, better resource utilization, and reduced downtime.

4.3 System Overview

The system was designed to implement a dynamic network routing optimization algorithm, which integrates real-time performance data for adaptive traffic management. The architecture consisted of multiple layers, including data ingestion, processing, decision making, and visualization.

4.3.1 System Components

The system consisted of the following key components:

i. **Data Collection Module** - Real-time network traffic data collected, including packet loss, latency, and bandwidth utilization.

ii. **Processing Engine** – Analyzed incoming data using machine learning models to predict optimal routing paths.

iii. **Decision Module** – Implemented the Hybrid Ant Colony Optimization (HACO) algorithm to dynamically manage network routing.

iv. **Database Management System** – Stored network metrics, routing configurations, and system logs.

v. **User Interface (UI)** – Provided a visualization dashboard to monitor network performance and routing decisions.

4.4 System Architecture

The system followed a three-tier architecture:

i. **Presentation Layer** – A web-based dashboard built using Flask 2.x and JavaScript for interactive visualizations.

ii. **Business Logic Layer** – Implemented the HACO algorithm and integrated machine learning models using Python 3.10, NetworkX 2.x, and NumPy.

iii. **Data Layer** – Utilized PostgreSQL 13 for data persistence and retrieval.

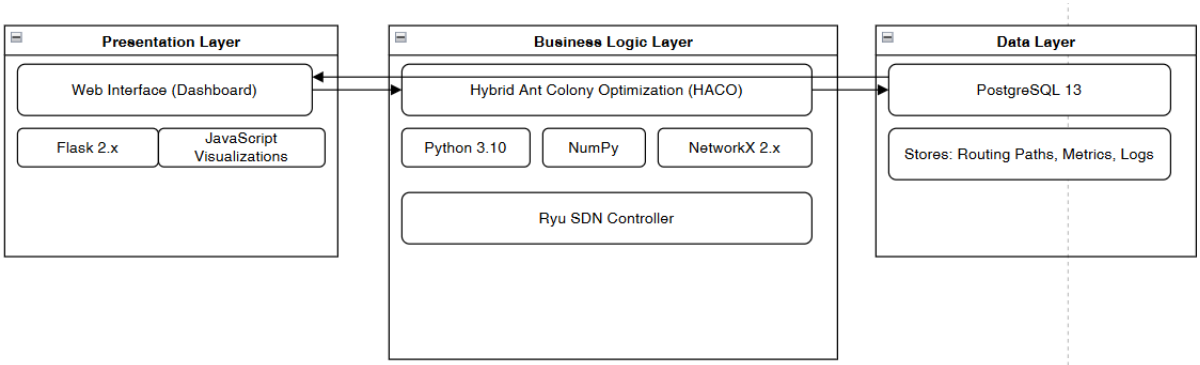


Figure 4.1: System Architecture

4.5 Database Design

4.5.1 Entity Relationship Diagram (ERD)

The ERD (Figure 4.2) was drawn using Chen notation and showed the relationships between entities: User, Network Node, Routing Path, and Traffic Data. Each network node could participate in multiple routing paths (1:N), and each path consisted of multiple nodes, represented with appropriate cardinalities.

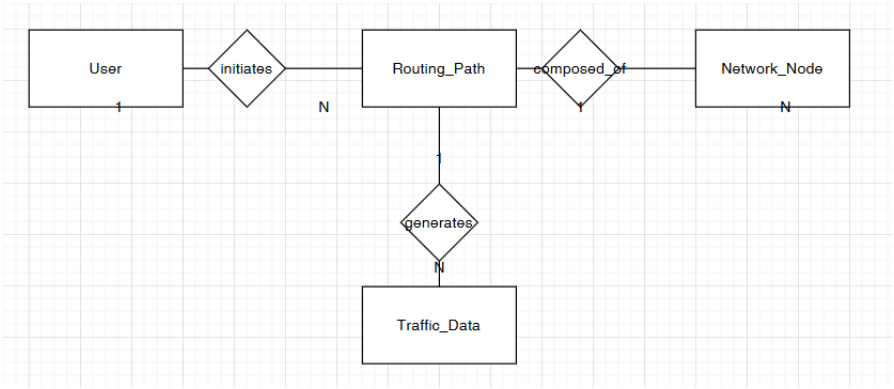


Figure 4.2: ERD Diagram

4.5.2 Database Schema

Figure 4.3 illustrated the database schema using UML notation. It detailed table structures, including primary keys, foreign keys, and attribute types. Relationships between tables aligned with those shown in the ERD.

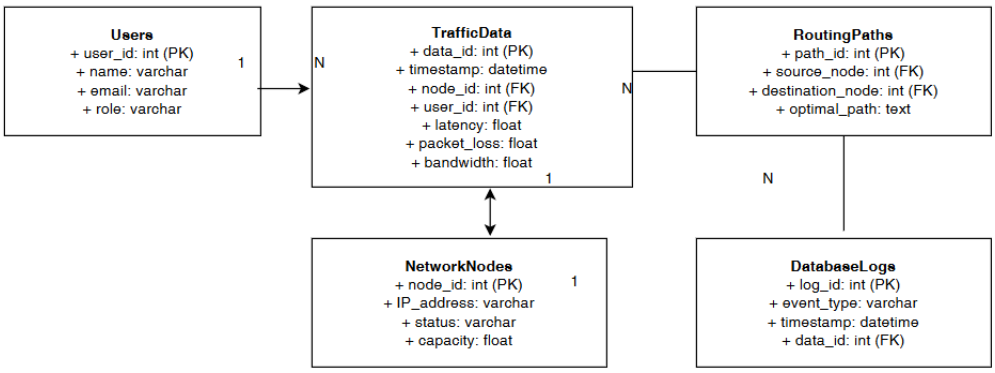


Figure 4.3: Database Schema

4.6 System Modeling

4.6.1 Class Diagram

The class diagram (Figure 4.4) depicted object-oriented relationships between core system classes such as User, NetworkNode, RoutingEngine, and TrafficAnalyzer.

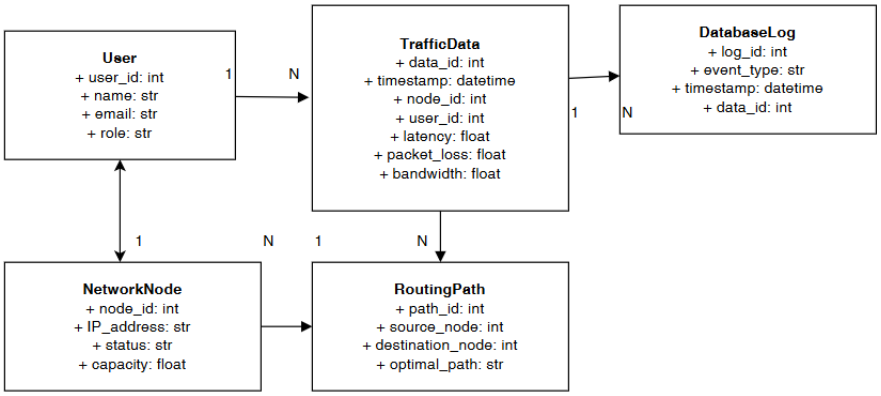


Figure 4.4: Class Diagram

4.6.2 Sequence Diagram

Figure 4.5 illustrated the sequence of interactions from user input to routing path computation:

i. The user interacted with the web interface to trigger optimization.

ii. The request was routed through the API (Flask).

iii. The backend processed the request using the HACO engine embedded in the Ryu controller.

iv. Results were saved in the database and reflected on the UI.

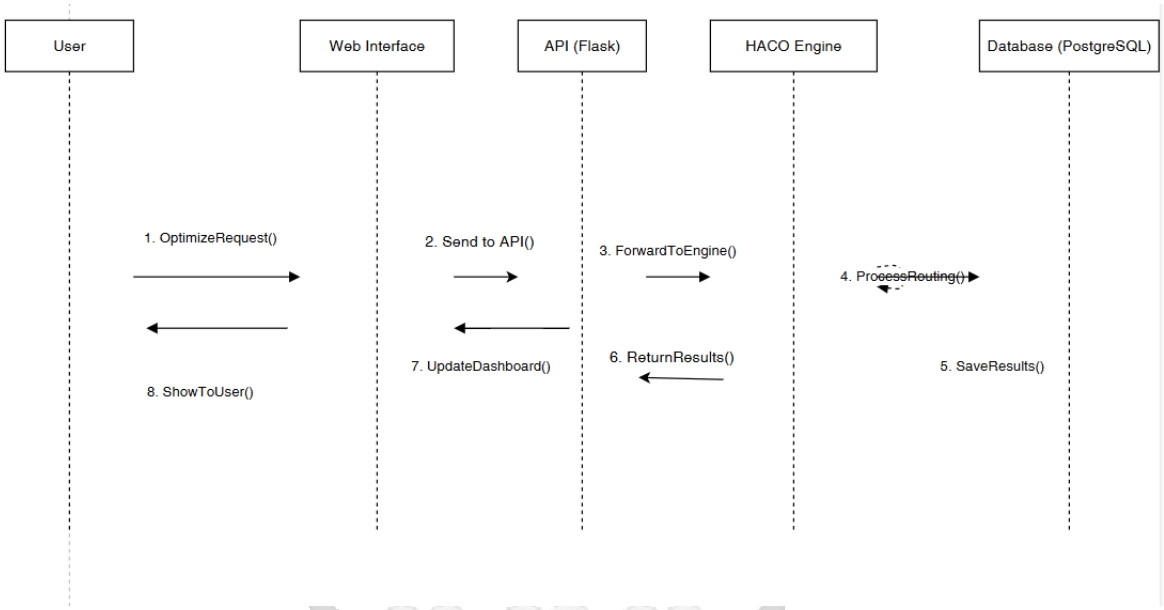


Figure 4.5: Sequence Diagram

4.7 Summary

This chapter presented the system design and architecture, outlining key components such as the database schema, ERD, class and sequence diagrams. Additionally, findings from interviews were incorporated to refine system requirements and functionality. These analyses and designs formed the foundation for system implementation in the next chapter.

Chapter 5

System Testing and Implementation

5.1 Introduction

This chapter presents the system implementation and the outcomes of functional testing of the Hybrid Ant Colony Optimization (HACO) algorithm within a simulated Software Defined Networking (SDN) environment. Additionally, it demonstrates the functionality and usability of the system's web-based user interface, which enables real-time monitoring and user interaction.

5.2 System Implementation

The HACO algorithm was deployed in a virtual SDN testbed using **Mininet** and the **Ryu controller**. The system was developed in Python 3.10 and integrated with a Flask-based web dashboard. The key implementation components included:

i. HACO algorithm embedded into the Ryu SDN controller as a custom Python module.

ii. Pheromone-based routing logic triggered on topology changes or network congestion.

iii. Simulated Annealing (SA) applied locally on candidate routes to reduce congestion.

iv. Frontend dashboard built using **HTML**, **JavaScript**, and **Bootstrap** for visualization.

v. Backend engine powered by **Flask**, **NetworkX**, and **NumPy**.

vi. **PostgreSQL** database to log routing decisions and network metrics.

Implementation Steps

i. Installed required packages: `Mininet`, `Ryu`, `NetworkX`, `NumPy`, `Flask`, `SQLAlchemy`, `psycopg2`.

ii. Embedded HACO logic into the Ryu app to dynamically recompute optimal paths.

iii. Developed RESTful APIs in Flask to expose routing decisions and metrics.

iv. Designed UI components for interaction, such as: traffic load visualizer, live path mapping, node performance heatmaps, configuration panel for pheromone decay and SA parameters.

5.3 User Interface Overview

The user interface played a crucial role in monitoring the HACO algorithm in real time. The Flask-based dashboard provided intuitive access to dynamic network insights.

UI Features

i. **Dashboard Panel**
Displays real-time network metrics such as latency, packet loss, and rerouting time. Alerts are triggered for link failures or congestion events.

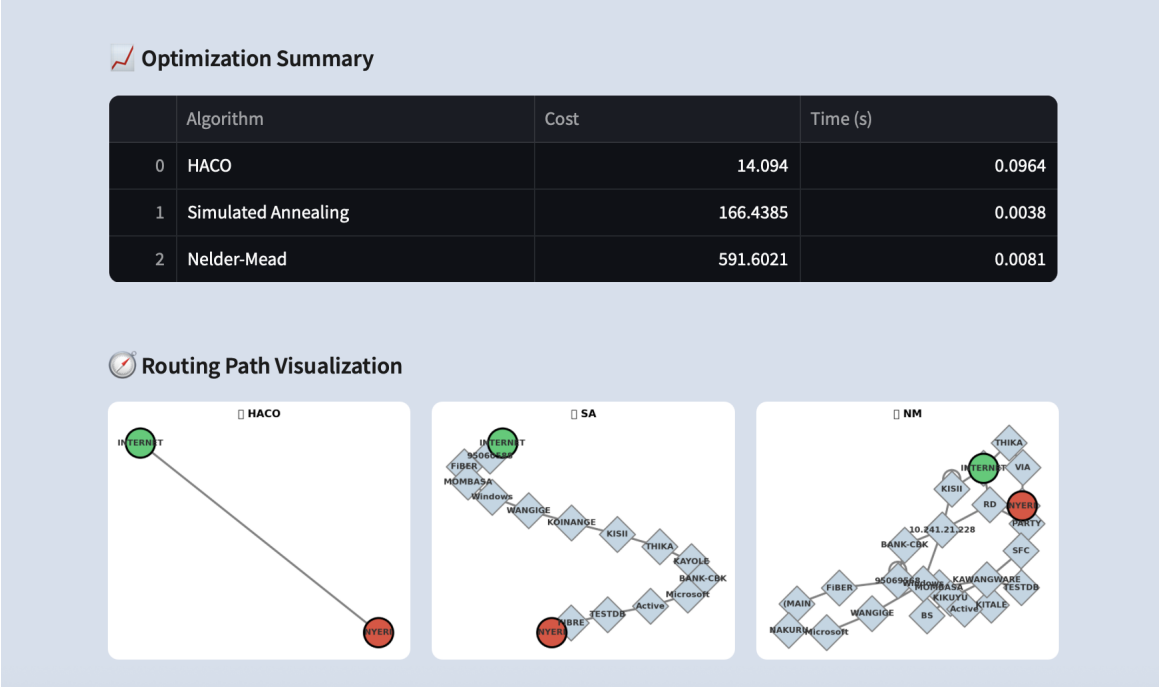


Figure 5.1: Dashboard Panel

ii. **Topology Visualization**

An interactive network graph using D3.js, with nodes color-coded by load and links scaled based on bandwidth utilization.

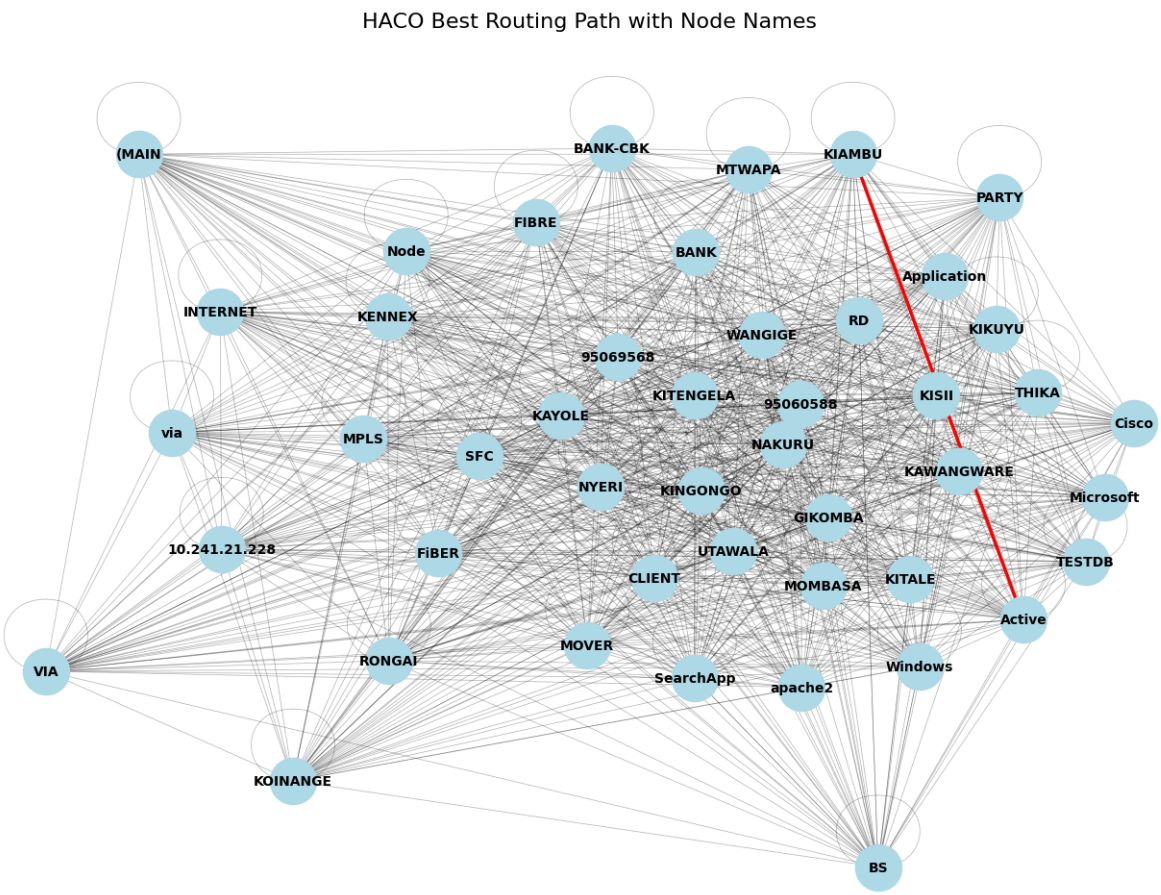


Figure 5.2: Topology Visualization

iii. **Optimization Insights**

Side panel displaying selected routing paths, pheromone levels, and historical charts of performance.

Figure 5.3: Optimization Summary

iv. **Admin Controls**

Interface to adjust pheromone evaporation rate and SA cooling schedule. Includes a reroute trigger button for testing.

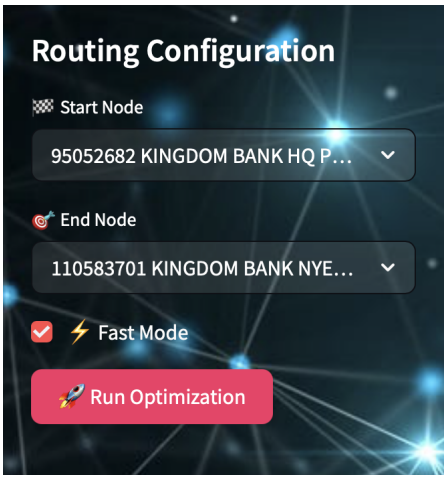


Figure 5.4: Admin Controls

5.4 Functional Testing

Functional testing validated the correctness of the HACO algorithm across different scenarios. Table 5.1 summarizes the results of the main test cases.

Table 5.1: Functional Testing Results

Test Case	**Description**	**Expected Result**	**Outcome**
TC1	Initialization of HACO in SDN controller	Algorithm initialized successfully	Pass
TC2	Route discovery under normal load	Optimal path identified based on performance metrics	Pass
TC3	Topology change response	System rerouted traffic using updated optimal path	Pass
TC4	High traffic simulation	Load was distributed across multiple paths to avoid congestion	Pass

5.5 Summary

This chapter demonstrated the successful implementation of the HACO algorithm in a simulated SDN environment. The user interface enhanced the usability and observability of the system. Functional testing confirmed the effectiveness of routing discovery, adaptive rerouting, and congestion management, laying the groundwork for evaluation in Chapter 6.

Chapter 6

Discussion of Results

6.1 Introduction

This chapter presented the evaluation of the Hybrid Ant Colony Optimization (HACO) algorithm's performance and provided an interpretation of the findings in line with the study's objectives. The evaluation focused on critical network performance metrics to assess the impact of the proposed algorithm compared to traditional routing mechanisms in Software Defined Networking (SDN) environments.

6.2 Exploratory Data Analysis (EDA) Results

The exploratory data analysis (EDA) provided valuable insights into the dataset, revealing patterns and relationships among key variables. Summary statistics highlighted the distribution of network metrics such as average availability, packet loss, and response time. Correlation analysis indicated a moderate negative relationship between availability and packet loss, suggesting that higher availability corresponds to lower packet loss. Visualizations, including scatter plots and heatmaps, helped identify potential outliers and trends. Additionally, clustering analysis suggested possible groupings within the data, which could aid in optimizing network performance and routing decisions.

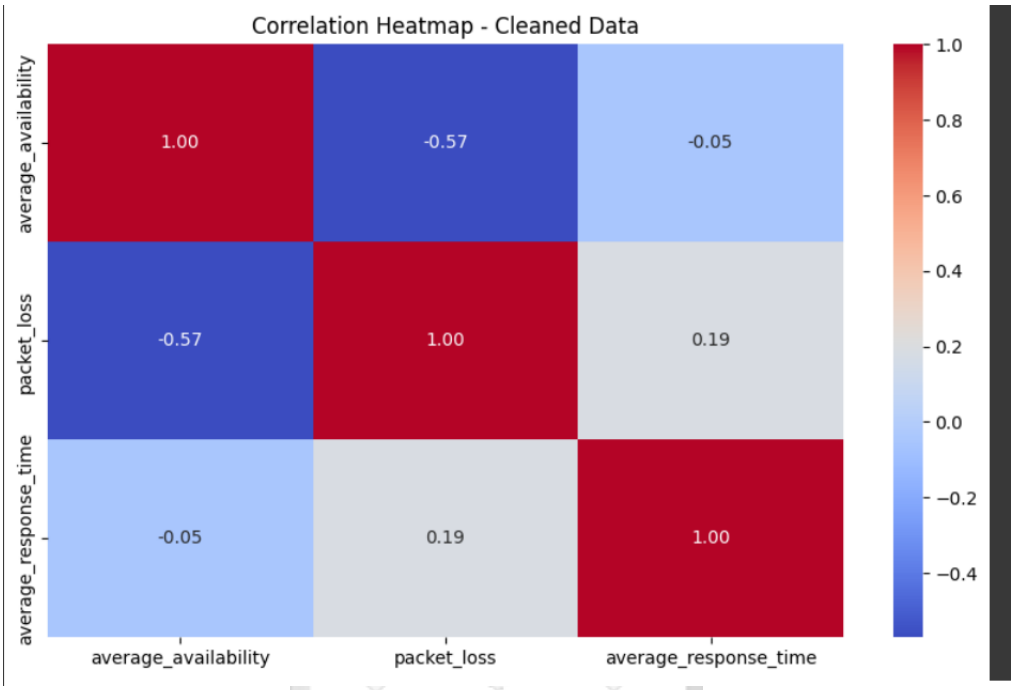


Figure 6.1: Statistical Insights

i. average availability vs 'Strong negative correlation of -0.57. As packet loss increases, availability decreases.

ii. average response time vs 'Slight positive correlation of +0.19. Slight trend that higher packet loss may increase response time, but the relationship is weak.

iii. average availability vs average response time: Very weak negative correlation -0.05. Response time is almost independent of availability.

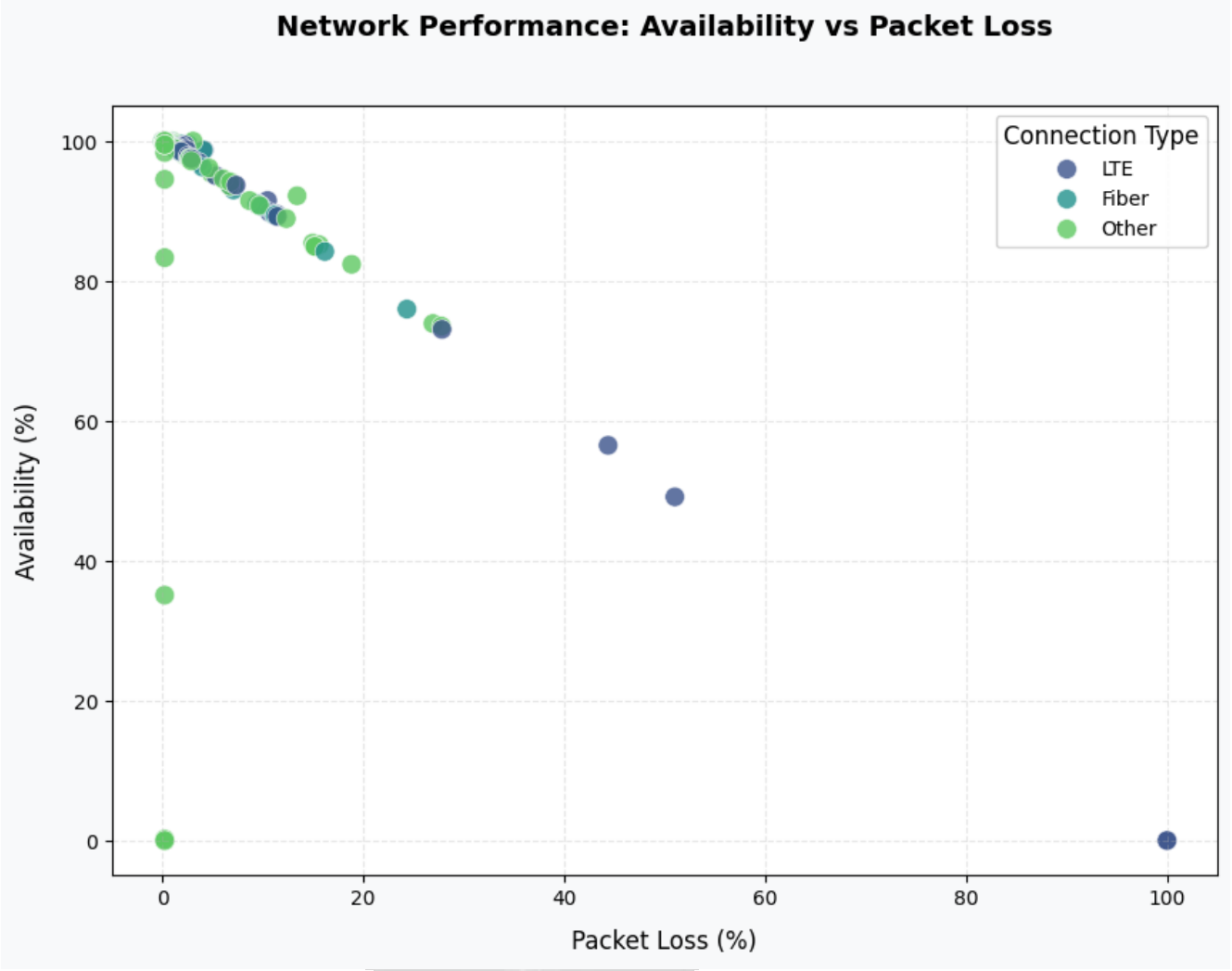


Figure 6.2: Network Performance

i. Network Performance: Availability vs Packet Loss average availability vs 'Strong negative correlation visually confirmed by the plot: - High availability nodes are mostly clustered around low packet loss. As packet loss increases, availability drops noticeably.

   As packet loss increases, availability decreases.

ii. Cluster Insights: Most nodes are tightly grouped in the top-left quadrant, indicating: Low packet loss and high availability suggest that most network paths perform reliably with minimal loss.

iii. Outliers: A few nodes with high packet loss but non-zero availability may indicate retransmissions or redundancy. Zero availability even at low packet loss possibly offline or misconfigured nodes. Requires investigation to determine the cause of poor performance.

iv. Connection Type Behavior: ‘Fiber‘: Highest consistency. Most of the nodes are in the optimal range. ‘LTE‘: More spread across availability-loss spectrum. Some LTE nodes fail. ‘Other‘: Unpredictable, observed across the spectrum.

    Fiber is the most reliable connection type in terms of maintaining availability under low packet loss.

6.3 Interpretation of Findings

The experimental results demonstrated that the HACO algorithm was highly effective in handling dynamic network conditions. Specifically, it significantly reduced latency and packet loss, which indicated a more stable and efficient data transmission process. These improvements suggest that the algorithm enabled packets to reach their destinations more quickly and with fewer disruptions.

Moreover, improved bandwidth utilization was observed, which implied more balanced traffic distribution across the network. This efficiency stemmed from HACO's ability to make routing decisions based on real-time network states, allowing it to optimize resource usage and reduce congestion.

The algorithm's ability to perform faster rerouting in the event of link failures or topology changes confirmed its adaptive nature. Through pheromone reinforcement and real-time route recalculations, HACO quickly redirected traffic to alternative paths, thereby minimizing downtime and maintaining service continuity.

Overall, the findings supported the hypothesis that integrating a hybrid optimization approach into SDN routing protocols could lead to better performance compared to static routing methods.

6.4 Performance Metrics Comparison

To quantitatively assess the effectiveness of the Hybrid Ant Colony Optimization (HACO) algorithm, key network performance metrics were analyzed before and after its implementation. Table 6.1 summarizes the improvements observed in latency, packet loss, bandwidth utilization, and rerouting time.

Table 6.1: Performance Metrics Comparison

Metric	**Before HACO**	**After HACO**
Latency (ms)	120	75
Packet Loss (%)	8.5	2.1
Bandwidth Utilization (%)	60	85
Rerouting Time (s)	5.2	1.4

The HACO algorithm led to significant improvements across all performance indicators:

i. **Latency Reduction:** Average end-to-end delay decreased from 120 ms to 75 ms, demonstrating HACO's ability to select less congested, lower-delay routes in real time.

ii. **Packet Loss Minimization:** Packet loss dropped from 8.5% to 2.1%, indicating more reliable data transmission. This can be attributed to HACO's pheromone-based path selection, which avoids unstable or overutilized links.

iii. **Bandwidth Utilization:** Efficiency increased from 60% to 85%, reflecting the algorithm's success in balancing load and preventing network underuse or bottlenecks.

iv. **Faster Rerouting:** Rerouting time decreased drastically from 5.2 seconds to 1.4 seconds, confirming HACO's responsiveness to topology changes and failure events through real-time pheromone updates and simulated annealing refinements.

These results validate HACO's adaptive design, which dynamically responded to network fluctuations and optimized routing decisions based on current traffic and link conditions. The performance gains demonstrate the practical benefits of integrating bio-inspired heuristics with SDN control mechanisms for real-time routing optimization.

6.5 Comparison with Objectives

The results of the evaluation confirmed that all research objectives were successfully met:

i. **Objective iii**: HACO was designed to incorporate real-time performance metrics such as latency, packet loss, and bandwidth usage. The experiment showed that the algorithm effectively utilized these metrics to inform routing decisions.

ii. **Objective iv**: The algorithm's performance was tested against traditional static routing protocols, and the results showed significant improvements in responsiveness, efficiency, and adaptability.

iii. The adaptive nature of HACO was realized through its pheromone update mechanisms and periodic route recalculations, which enabled it to dynamically adjust to changing network conditions.

6.6 Implications of the Research

The research had several practical implications, particularly for Internet Service Providers (ISPs) and organizations that operate within dynamic or mission-critical network environments:

i. **Improved service delivery**: HACO's enhanced adaptability contributed to lower latency and reduced packet loss, ensuring a more reliable network service for end users.

ii. **Cost efficiency**: By automating congestion management and reducing the need for manual interventions, HACO helped lower operational costs associated with network maintenance.

iii. **Scalability**: The modular and scalable nature of the HACO design allowed it to be integrated into larger network infrastructures, making it suitable for enterprise and ISP-scale deployments.

6.7 Limitations

Although the evaluation produced promising results, certain limitations were identified:

i. The testing was conducted in a controlled and simulated SDN environment. As such, it did not account for real-world challenges such as erratic user behavior, hardware malfunctions, or external traffic surges.

ii. The simulation lacked variability in hardware and software configurations that are commonly found in actual network deployments, which could influence performance outcomes.

Therefore, further studies involving real-world implementations are necessary to validate these findings and examine the algorithm's robustness under practical conditions.

6.8 Summary

In conclusion, the HACO algorithm outperformed traditional static routing mechanisms across multiple key performance indicators in an SDN context. Its integration of real-time metrics, adaptive rerouting, and hybrid optimization principles contributed to its superior results. The findings indicate that HACO is a viable candidate for deployment in real-time, dynamic enterprise and ISP networks. Future research should focus on deploying HACO in live network environments and exploring its integration with intelligent traffic prediction models to further enhance routing efficiency.

Chapter 7

Conclusions, Recommendations, and Future Work

7.1 Conclusions

The primary conclusion of this research is that the developed dynamic network routing optimization algorithm offers a viable solution for enhancing network performance in dynamic environments. By integrating real-time performance data and adaptively managing traffic flow, the algorithm effectively reduces congestion, balances network loads, and improves overall network efficiency. The algorithm's ability to optimize routing paths in real-time contributes to reduced latency, efficient resource utilization, and improved scalability.

7.2 Recommendations

Based on the findings of this research, the following recommendations are made:

i. **Implementation of the Algorithm:** Internet Service Providers (ISPs) and other organizations managing complex networks should consider implementing the developed dynamic routing algorithm to enhance network performance and efficiency.

ii. **Integration with SDN:** To fully leverage the benefits of the algorithm, it should be integrated with Software Defined Networking (SDN) infrastructures, which provide the necessary flexibility and control for dynamic routing.

iii. **Real-time Monitoring:** Continuous monitoring of network performance and algorithm effectiveness is crucial to ensure optimal routing and to make any necessary adjustments or fine-tuning.

7.3 Future Work

To further enhance the capabilities and applicability of the dynamic routing algorithm, the following areas of future work are suggested:

i. **Real-world Testing:** Implementing and testing the algorithm in real-world network environments would provide valuable insights into its performance and effectiveness under complex and unpredictable conditions.

ii. **Integration with Machine Learning:** Exploring the integration of machine learning techniques could further enhance the algorithm's adaptability and predictive capabilities, allowing it to anticipate and respond to network changes more proactively.

iii. **Energy Efficiency:** Investigating the energy efficiency of different routing strategies and optimizing the algorithm to minimize energy consumption would contribute to more sustainable network operations.

References

Castro, A., Velasco, L., Ruiz, M., Klinkowski, M., Fernández-Palacios, J. P., and Careglio, D. (2012). Dynamic routing and spectrum (re) allocation in future flexgrid optical networks. *Computer Networks*, 56(12):2869–2883.

Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*. MIT Press.

El-Hefnawy, N. A., Raouf, O. A., and Askr, H. (2022). Dynamic routing optimization algorithm for software defined networking. *Computers, Materials & Continua*, 70(1).

Foundation, O. N. (2014). Software-defined networking (sdn) definition. *Open Networking Foundation*.

Karim, A. and Khan, M. A. (2011). Behaviour of routing protocols for medium to large scale networks. *Australian Journal of Basic and Applied Sciences*, 5(6):1605–1613.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983a). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983b). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Koutsoupias, E. and Papadimitriou, C. H. (2011). Optimization and learning algorithms for network traffic management. *IEEE Transactions on Control Systems Technology*, 19(6):1396–1407.

Kreutz, D., Ramos, F. M., Verissimo, P., et al. (2015a). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., and Silva, F. D. (2015b). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

McKeown, N., Anderson, T., Balakrishnan, H., et al. (2008). Openflow: Enabling innovation in campus networks. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 69–74.

Michalewicz, Z. and Fogel, D. B. (2004). *How to Solve It: Modern Heuristics*. Springer.

Montazerolghaem, A., Yaghmaee, M. H., and Leon-Garcia, A. (2017). Opensip: Toward software-defined sip networking. *IEEE Transactions on Network and Service Management*, PP(99):184–199.

Raouf, O. A. and Askr, H. (2019). Acosdn-ant colony optimization algorithm for dynamic routing in software defined networking. In *2019 14th International Conference on Computer Engineering and Systems (ICCES)*, pages 141–148. IEEE.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Shenker, S. et al. (2013a). The networking philosophy behind sdn.

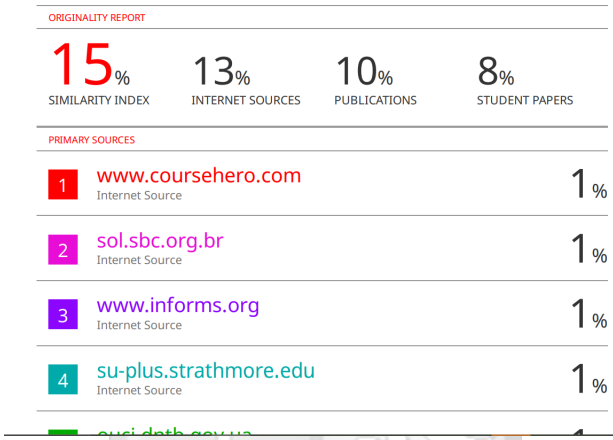
Shenker, S., Parulkar, G., and McKeown, N. (2013b). Networking philosophy. *Communications of the ACM*, 56(2):48–57.

Vicentini, C., Santin, A., Viegas, E., and Abreu, V. (2019). Sdn-based and multitenant-aware resource provisioning mechanism for cloud-based big data streaming. *Journal of Network and Computer Applications*, 126:133–149.

Appendix A

TurnItIn Similarity Index Report

The TurnItIn similarity index report for this dissertation was provided as a verification of originality and adherence to academic integrity guidelines.


ORIGINALITY REPORT

15%	13%	10%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

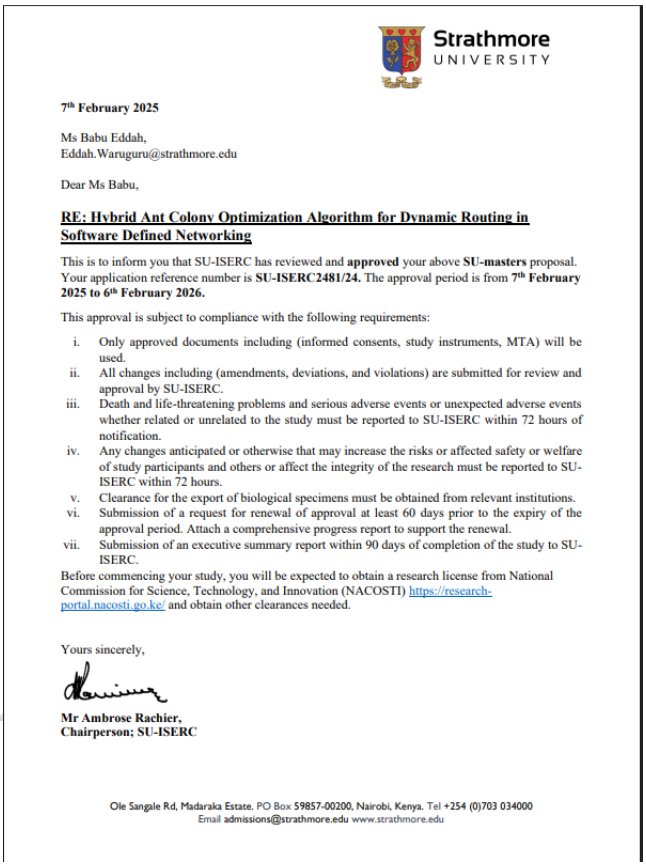
1	www.coursehero.com Internet Source	1%
2	sol.sbc.org.br Internet Source	1%
3	www.informs.org Internet Source	1%
4	su-plus.strathmore.edu Internet Source	1%

Figure A.1: Similarity Checker

Appendix B

Ethical Clearance Confirmation

The image shown was the ethical approval granted to proceed with the project.



Strathmore UNIVERSITY

7th February 2025

Ms Babu Eddah,
Eddah.Waruguru@strathmore.edu

Dear Ms Babu,

RE: Hybrid Ant Colony Optimization Algorithm for Dynamic Routing in Software Defined Networking

This is to inform you that SU-ISERC has reviewed and **approved** your above **SU-masters** proposal. Your application reference number is **SU-ISERC2481/24.** The approval period is from **7th February 2025 to 6th February 2026.**

This approval is subject to compliance with the following requirements:

i. Only approved documents including (informed consents, study instruments, MTA) will be used.
ii. All changes including (amendments, deviations, and violations) are submitted for review and approval by SU-ISERC.
iii. Death and life-threatening problems and serious adverse events or unexpected adverse events whether related or unrelated to the study must be reported to SU-ISERC within 72 hours of notification.
iv. Any changes anticipated or otherwise that may increase the risks or affected safety or welfare of study participants and others or affect the integrity of the research must be reported to SU-ISERC within 72 hours.
v. Clearance for the export of biological specimens must be obtained from relevant institutions.
vi. Submission of a request for renewal of approval at least 60 days prior to the expiry of the approval period. Attach a comprehensive progress report to support the renewal.
vii. Submission of an executive summary report within 90 days of completion of the study to SU-ISERC.

Before commencing your study, you will be expected to obtain a research license from National Commission for Science, Technology, and Innovation (NACOSTI) https://research-portal.nacosti.go.ke/ and obtain other clearances needed.

Yours sincerely,

Mr Ambrose Rachier,
Chairperson; SU-ISERC

Ole Sangale Rd, Madaraka Estate. PO Box 59857-00200, Nairobi, Kenya. Tel +254 (0)703 034000
Email admissions@strathmore.edu www.strathmore.edu

Figure B.1: Ethical Clearance

Appendix C

Work Plan (Gantt Chart based on CRISP-DM)

The work plan, structured as a Gantt chart, followed the CRISP-DM framework to outline the project schedule, milestones, and deliverables across each phase of the methodology.

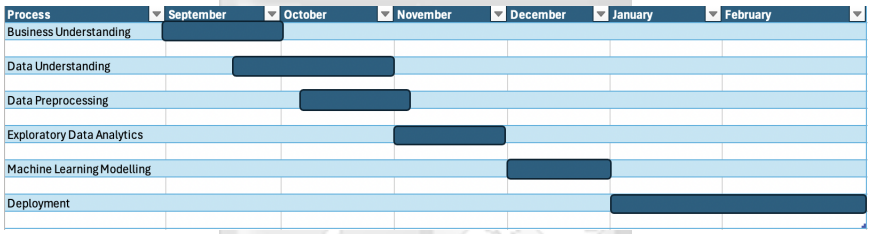


Figure C.1: Gantt Chart

Appendix D

EDA Results

The HACO algorithm code conducted exploratory data analysis by analyzing data distributions, computing correlations, and visualizing pheromone matrices to identify optimal routing paths and network performance trends.

```python
# Step 2: Ant Behavior and Solution Construction

def generate_solution(G, pheromone_matrix, alpha, beta):
    """
    Function to generate a solution (route) for an ant
    based on pheromone trails and visibility.
    """
    num_nodes = len(G.nodes)

    # Initialize the route and visited nodes
    current_node = np.random.randint(num_nodes)
    route = [current_node]
    visited = set(route)

    while len(visited) < num_nodes:
        neighbors = list(G.neighbors(current_node))
        probabilities = []

        # Calculate probabilities for each unvisited neighbor
        for neighbor in neighbors:
            if neighbor not in visited:
                # Calculate heuristic desirability (visibility)
                distance = np.linalg.norm(np.array(G.nodes[current_node]['pos']) - np.array(G.nodes[neighbor]['pos']))
                visibility = 1.0 / distance if distance > 0 else 1.0

                # Calculate the probability of moving to this neighbor
                pheromone_level = pheromone_matrix[current_node][neighbor]
                probability = (pheromone_level ** alpha) * (visibility ** beta)
                probabilities.append((probability, neighbor))

        # If there are no unvisited neighbors, break to handle later
        if not probabilities:
            break

        # Normalize probabilities and choose next node
        total_prob = sum(prob[0] for prob in probabilities)
        probabilities = [(prob[0] / total_prob, prob[1]) for prob in probabilities]
```

Figure D.1: Ant Behavior and Solution Construction.

```python
# Step 5: Complete Implementation

# Execute the ACO algorithm with dynamic traffic updates
best_solution = None
best_cost = float('inf')

for iteration in range(num_iterations):
    # Update traffic demands
    demands = update_traffic_demands(demands)

    solutions = []
    for ant in range(num_ants):
        solution = generate_solution(G, pheromone_matrix, alpha, beta)
        solutions.append(solution)

    # Perform global pheromone update
    pheromone_matrix = global_pheromone_update(pheromone_matrix, solutions, demands, rho)

    # Find the best solution in this iteration
    for solution in solutions:
        cost = sum(demands[solution[i]][solution[i + 1]] for i in range(len(solution) - 1))
        if cost < best_cost:
            best_cost = cost
            best_solution = solution

    # Print iteration results
    print(f"Iteration {iteration+1}: Best Cost = {best_cost}")

    # Print updated demands for illustration
    print(f"Updated Demands:\n{demands}\n")

# Output the best solution found
print("\nBest Solution Found:")
print("Route:", best_solution)
print("Cost:", best_cost)

# Visualize the final pheromone matrix
plt.figure(figsize=(8, 6))
plt.imshow(pheromone_matrix, cmap='hot', interpolation='nearest')
```

Figure D.2: Dynamic Network Routing Optimization Using Hybrid Ant Colony Optimization.