Journal of Theoretical and Applied Information Technology
30th April 2026. Vol.104. No.8

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

OPTIMIZING SEARCH SPACE FOR DYNAMIC SFC ROUTING: A MINIMUM SPANNING TREE-BASED GREY WOLF ALGORITHM

ZAHIDA SHARIF [1], MUHAMMED BASHEER JASSER [2], ANGELA AMPHAWAN [3], KOK-LIM ALVIN YAU [4], TSE-KIAN NEO [5]

[1,2] School of Computing and Artificial Intelligence, Faculty of Engineering and Technology, Sunway University, No. 5, Jalan University, Bandar Sunway, 47500 Selangor Darul Ehsan, Malaysia

[3] Smart Photonics Research Laboratory, Faculty of Engineering and Technology, Sunway University, No. 5, Jalan University, Bandar Sunway, 47500 Selangor Darul Ehsan, Malaysia.

[4] Faculty of Engineering and science, Department of computing, University of Tunku, Abdul Rahman (UTAR), Malaysia

[5] Center for Innovative and Immersive Technology, Multimedia University, Cyberjaya, Malaysia.

zeerah53@gmail.com [1], basheerj@sunway.edu.my [2], angelaa@sunway.edu.my [3]

ABSTRACT

In Network Function Virtualization Infrastructure (NFVI), the dynamic nature of Service Function Chain (SFC) mapping poses significant challenges, particularly for routing optimization. Fluctuating network conditions and user demands rapidly expand the routing search space, making real-time identification of optimal paths computationally expensive. An effective routing strategy therefore requires the construction of an efficient search space that ensures full node connectivity while selecting only qualitative links in terms of latency, bandwidth, and resource availability. Achieving these objectives simultaneously remains a major challenge, as conventional algorithms designed for rigid or strong tree structures are often unsuitable for dynamic chain orchestration in NFV environments. To address this issue, this work focuses on routing search space optimization by reducing computational overhead and eliminating infeasible routing options from the discrete solution space. A modified Grey Wolf Optimization (GWO) algorithm is proposed, specifically tailored for discrete routing scenarios in dynamic SFC mapping. The proposed approach employs a discrete initialization strategy, enforces a Minimum Spanning Tree (MST) based connectivity constraint to guarantee end-to-end reachability, and integrates a penalty function to suppress redundant or overlapping routing solutions. This design ensures that optimization is performed over a compact, connectivity-aware, and quality-driven search space. Simulations results validate the effectiveness of the proposed search space optimization strategy, demonstrating significant performance improvements across different network scales. The proposed approach achieves up to 91% reduction in execution time and 82.8% reduction in end-to-end delay, confirming its suitability for real-time dynamic SFC routing. In addition, improved bandwidth efficiency is observed, reflected by reductions of **51**%, **59**%, and **66**% in average available bandwidth for small, medium, and large topologies, respectively. These quantitative gains highlight the benefits of pre-optimizing the routing search space and confirm its effectiveness for scalable and adaptive NFV environments.

Keywords: *Routing Optimization; Service Function Chaining (SFC); Search Space Optimization; Swarm Intelligence; Latency Optimization*

1 INTRODUCTION

Dynamic Service Function Chain (SFC) mapping is the process of mapping and routing service functions in real-time due to the changes that occur continuously in the network, such as changes in the traffic, failures, and changes in the performance demands [1]. In contrast to static mapping technologies, dynamic SFC mapping

requires ongoing adaptation to keep the service quality and efficiency up [2]. This necessitates the use of smart algorithms that can react to the prevailing network conditions in making placement and routing decisions. One of the biggest problems in this respect is the uncertainty of routing decisions [3], particularly as the search space increases with the different availability of resources and the necessity to prevent congestion in real-time [4]. Consequently, dynamic routing turns out to be one of the most computationally intensive parts of contemporary network management [5]. To solve such issues, different kinds of search spaces are applied in network optimization problems, including Virtual Network Function (VNF) placement, routing, and resource allocation. Search spaces may be continuous to enable the fine-grained tuning of parameters such as bandwidth and energy consumption [6], or discrete (typically binary), such as the binary decision of whether to activate a link [7]. Combinatorial search spaces look at different groupings of resources or paths that are required in routing and scheduling problems [8]. The disjointed feasible solutions in discontinuous spaces are scattered, and it is hard to move smoothly between solutions [9], and fragmented search spaces are parted into disjoint regions that may conceal global optima because of the existence of local solutions [10]. Discrete search spaces in which all possible solutions are well defined and separated are especially applicable in routing and VNF assignment problems, where there are predetermined alternatives to choose among a finite set [11]. Enhancement of the efficiency and flexibility of dynamic Service Function Chain (SFC) routing also depends on optimizing the underlying discrete search space [12]. Computational efforts can be better concentrated by searching for the solution in the most promising regions' leading to lower overhead, faster convergence, and more scalable and high-quality solutions [1], [13], [14], [15], [16], [17]. Several studies have proposed approaches to address this issue. For example,' Liao et al. [1] and Wei et al. [2] utilize Deep Reinforcement Learning (DRL) techniques with mechanisms such as action masking and temporal learning guidance. These enhancements help limit the decision-making process to only viable actions. Cai et al. [3] and Tam et al. [11] employ swarm intelligence with adaptive exploration and beam search for dynamic environments. Mao et al. [14]' cluster paths and use offline learning to reduce routing overhead. Zhao et al. [18] transform SFC chains and use hybrid metaheuristics to accelerate search. Farshin and Sharifian [15] filter infeasible links and apply hybrid ACS-GWO optimization. Ng and Mahmoodi [4] reduce orchestration time through offline clustering. Yu et al. [19] and Escheikh & Taktak [5] apply DRL with priority-based and QoS-guided exploration. Appari [12] and Wang et al. [20] utilize single and multi-agent models to focus learning on viable routing options. Khaghkhagh and Mahmoodi [21] use distributed A2C for edge-aware decision-making. Wei et al. [22] combines simulated annealing and quantum genetic search to target high potential solutions. Zhang et al. [13]' use greedy based genetic initialization, while El Amri and Meddeb [23] enforce strict MILP constraints. Diab et al. [16]' precomputes all paths and evaluates only relevant ones at runtime.

Despite recent advancements, optimizing the search space in dynamic networks remains a significant challenge due to increasing complexity and scale. SFC mapping requires balancing optimal VNF placement with efficient routing, while dynamic conditions necessitate real-time adaptation [24, 25, 26]. As the number of nodes and links grows, the search space expands, making it computationally intensive to find optimal solutions [15, 16, 17, 27]. Existing methods often lack flexibility, struggle with scalability, and yield suboptimal outcomes. Most focus on SFC placement and routing techniques without directly addressing how search space optimization impacts routing efficiency [1], [13], [14], [25], [26], [15], [16], [17], [27]. Although MST-based algorithms help reduce the search space, they may overlook higher-performance paths by optimizing only lower-dimensional factors.

While several related studies attempt to manage computational complexity (of the proposed solutions) by treating search space optimization as an additional phase for SFC placement and routing' they still leave critical questions unanswered. First, these studies do not provide results on the effectiveness of this search space optimization phase' leaving it unclear to what extent the search space is reduced or how it benefits dynamic SFC mapping. This lack of

empirical results raises questions about the necessity and efficiency of dedicating computational resources to this optimization phase. If search space optimization is not considered a major issue, it is unclear why resources are allocated to this additional step. Second, these approaches do not consider the dynamic environment for SFC placement and routing optimization. Given the recognized importance of search space optimization in dynamic SFC mapping, this paper specifically aims to address these concerns. It introduces an adapted swarm intelligence-based optimization algorithm (combination of GWO and MST), analyzes existing techniques and evaluates the effects of search space optimization supporting results, providing clarity on its real-world implications and long-lasting impacts. The key contributions of this work are as follows:

1. This study presents MST-GWO' an adapted version of the Grey Wolf Optimizer to optimize search space in discrete network environments for dynamic SFC routing. The algorithm incorporates a discrete initialization scheme and a revised position update mechanism to achieve a more effective balance between global exploration and local exploitation.

2. To enhance the quality of solutions' a Minimum Spanning Tree (MST) constraint and a penalty function are integrated into the MST-GWO algorithm. These additions ensure that selected paths maintain network-wide connectivity while minimizing redundancy' leading to more stable routing outcomes and fewer inefficient configurations.

3. The effectiveness of the proposed algorithm is validated through detailed search space analysis and extensive simulations on both realistic and synthetic network models. Comparisons with standard routing techniques such as Dijkstra and BFS confirm MST-GWO's superiority in reducing computational overhead and improving resource utilization.

The remainder of this paper is organized as follows: Section 2 presents the problem description and related work. Section 3 provides an overview of the traditional GWO algorithm. Section 4 introduces the proposed Minimum Spanning Tree Grey Wolf Optimization (MSTGWO) algorithm and solution representation. Section 5 covers performance evaluation and comparative analyses followed by a discussion in Section 6. Finally, Section 7 concludes the paper and outlines future research directions.

2 PROBLEM DESCRIPTION AND RELATED WORK

2.1 System Model

The substrate network is modeled as an undirected graph G(N′L) where N represents the set of substrate network nodes (e.g., servers switches) and L represents the set of communication links interconnecting these nodes. Each link $(i,j) \in L$ establishes a connection between node i and node j ' and is characterized by a set of physical and performance-related parameters:

- Propagation Speed $v_{pp}(i,j)$: The speed at which a signal propagates through the link between nodes i and j, influenced by the transmission medium.
- Distance $\text{dist}_{(i,j)}$: The physical length of the link connecting node i and node j.
- Available Bandwidth $Bw_{(i,j)}^{\text{Avl}}$: The unused bandwidth on the link, indicating how much data can be transmitted without exceeding capacity.
- Propagation Delayed $pp(i,j)$: The time it takes for a signal to traverse the link, calculated as

$$d_{pp}(i,j) = \frac{\text{dist}_{(i,j)}}{v_{pp}(i,j)} \quad (1)$$

- Power Consumption Rate $W_{(i,j)}^{Bw}$: The power consumption per unit of bandwidth utilization on the link.

These parameters are randomly distributed across the links to simulate a realistic network environment where some links may be of lower quality in terms of delay' bandwidth' or power efficiency. The notations used in this study are summarized in Table 1.

Table 1: Summary of Notations

Indices	Definition
N	Number of nodes, indexed by $N = \{n_1, n_2, \dots, n_k\}$
L	Number of links, indexed by $L = \{(i,j)_1, (i,j)_2, \dots, (i,j)_k\}$
$v_{pp}(i,j)$	Propagation speed of link (i,j)
$\text{dist}_{(i,j)}$	Distance/length of link (i,j)
$Bw_{(i,j)}^{\text{Avl.}}$	Available bandwidth at link (i,j)
$d_{pp}(i,j)$	Propagation delay of link (i,j)
$W_{(i,j)}^{Bw}$	Power consumption per unit of bandwidth usage at link (i,j)
Decision Variables	
$x_{(i,j)}$	1 if the link is selected; 0 otherwise
$y_{(i,j)}$	1 if the link is feasible; 0 otherwise

In the context of Service Function Chaining (SFC), where a sequence of Virtual Network Functions (VNFs) must be connected over the physical network, not all links are equally viable. Some may be infeasible due to inadequate bandwidth' high latency or other limitations. Therefore, the system must be optimized to minimize the use of infeasible links' thereby reducing the search space for routing and improving the quality of service (QoS) across the network.

2.2 Problem Formulation

The main goal is to optimize the search space for SFC routing minimizing the number of infeasible links in the substrate network while ensuring that:

- The selected subset of links connects all network nodes'
- The selected links exhibit minimal latency.
- A predefined number (k) of links are selected.

To formalize this two key binary decision variables are defined:

- $x_{(i,j)} \in \{0,1\}$: Indicates whether link (i,j) is selected (1) or not (0).
- $y_{(i,j)} \in \{0,1\}$: Indicates the feasibility of link (i,j). A value of 1 means feasible; 0 indicates infeasibility due to network constraints.

Objective Function:

The primary objective is to minimize the number of selected infeasible links. This is expressed through the product $x_{(i,j)} \cdot y_{(i,j)}$, which will be 1 only if the link is selected and infeasible. To ensure connectivity, a penalty function is also incorporated in cases where the selected links do not connect all nodes. The objective function is defined as

$$min \ \sum(i,j) \in L \ x\_\{ij\} \ y\_\{ij\} + \beta \cdot max\left(0, |N| - 1 - \sum(i,j) \in L \ x_{\{ij\}}\right) \cdot f \quad (2)$$

Where

- β is a penalty coefficient that determines the severity of the penalty for unconnected network topologies.
- $f \in \{0,1\}$ is a binary penalty flag defined as

$$f = \begin{cases} 1, & \text{if } \sum_{(i,j)\in L} x_{(i,j)} < |N| - 1 \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

The penalty term encourages the optimization process to prefer connected subgraphs that span all nodes.

Constraints:

Each link is either selected or not, and is either feasible or not:

$$x_{(i,j)} \in \{0,1\}, y_{(i,j)} \in \{0,1\}, \forall (i,j) \in L \quad (4)$$

The number of selected links should be exactly k :

$$\sum_{(i,j)\in L} x_{(i,j)} = k \quad (5)$$

At least $|N| - 1$ links must be selected to connect all nodes in a spanning structure:

$$\sum_{(i,j)\in L} x_{(i,j)} \geq |N| - 1 \quad (6)$$

The selected links must be chosen from among the top k links with the smallest propagation delays:

$$d_{pp}(i,j)_1 \leq d_{pp}(i,j)_2 \leq \cdots \leq d_{pp}(i,j)_k \quad (7)$$

This constraint ensures that the selected links have the lowest possible propagation delay, thereby promoting faster data transmission and improved quality of service (QoS). The above formulation balances link quality, resource feasibility, and network connectivity within a constrained selection space. The classical Minimum Spanning Tree (MST) approach provides the foundational idea for the connectivity requirement in constraint (6), ensuring that the selected links form a connected topology [28]. However, MST-based approaches typically focus on minimizing the number of links or total cost, while ignoring important performance parameters such as delay, bandwidth, and power consumption, which may lead to suboptimal service function chaining (SFC) routing decisions.

To address this shortcoming, the proposed model incorporates a delay-based link selection strategy (Constraint 7), selecting the top k links with the lowest propagation delays. However, this introduces a new challenge: these k links may not form a connected graph, potentially leaving nodes isolated. This challenge is addressed defining a penalty mechanism in the objective function. When the selected subset of links fails to satisfy the connectivity constraint, the penalty term $\beta \cdot \max\left(0, |N| - 1 - \sum_{(i,j)\in L} x_{(i,j)}\right) \cdot f$ becomes active and penalizes solutions that do not meet the connectivity requirement, guiding the algorithm to replace redundant or unhelpful links with those that contribute to full network connectivity.

2.3 Related Work

The NP-hard nature of the dynamic Service Function Chain (SFC) mapping problem has driven extensive research toward mitigating its large and complex search space [1], [3], [5], [11], [13,,15], [17,,27]. Existing approaches primarily reduce search complexity implicitly by embedding pruning and filtering mechanisms within routing, placement, or learning based orchestration frameworks. Deep reinforcement learning based solutions leverage action masking, reward shaping, and state awareness to avoid infeasible routing and placement decisions, thereby constraining the explored solution space [1], [2], [4], [5], [19]. Swarm intelligence-based methods manage search complexity through clustered exploration, beam search, sub swarm strategies, or adaptive exploitation, retaining only high-quality candidate solutions and discarding redundant or suboptimal paths during optimization [3], [11], [14], [18], [27]. Heuristic and graph-based techniques further reduce routing complexity by pre selecting shortest or feasible paths using BFS, DFS, Dijkstra based filtering, or path clustering, ensuring that only valid routing candidates are evaluated in subsequent stages [14], [25], [27]. Optimization and MILP based frameworks apply feasibility constraints, greedy initialization, or iterative refinement to confine the search to cost and latency efficient regions of the solution space [13, 17, 23]. Additionally, offline pre computation and learning based dimensionality reduction techniques store or classify feasible routing paths in advance, significantly lowering online computational overhead [4], [16], [20,,22], [26]. Although these strategies effectively reduce computational burden, search space optimization is largely treated as a supporting mechanism rather than a standalone routing focused phase, and its direct impact on subsequent route selection and overall SFC performance is rarely quantified.

The optimization focuses existing studies on Service Function Chain (SFC) mapping primarily centers on reducing computational complexity through indirect search space management and resource efficiency improvements [17],[25],[27]. Several works aim to shrink the routing or placement search space by precomputing feasible paths or excluding infeasible links, thereby limiting the number of candidate solutions considered during optimization [15],[26]. Other approaches emphasize efficient resource allocation and dynamic resource management to enhance scalability under varying network

conditions [17],[25]. Additionally, some studies focus on reducing redundant computations and improving computational efficiency through heuristic filtering or hybrid optimization strategies [16],[27]. While these efforts contribute to lowering overhead and improving scalability, the optimization focus largely remains fragmented and secondary to placement or deployment objectives, rather than providing a dedicated and quantifiable routing search space optimization framework.

Notable examples include Cai et al. [3], who introduced the Improved Sparrow Search Algorithm (ISSA); Zhao et al. [18], who applied an Improved Cuckoo Search (ICS); Farshin and Sharifian [15], who combined the Ant Colony System (ACS) with a Chaotic Grey Wolf Optimizer (GWO); and Wu and Zhou [27], who proposed a two-stage method integrating Breadth-First Search (BFS) with Improved Ant Colony Optimization (IACO). While these approaches contribute significantly to metaheuristic-based network optimization, they do not fully address the specific limitations discussed earlier. Modern optimization challenges, however, can be handled efficiently using swarm intelligence techniques. Our study of search space optimization through swarm algorithms indicates that swarm-based methods, particularly GWO, provide a promising approach for systematically excluding infeasible links from the search space.

To address these gaps, this paper proposes a GWO-based strategy specifically designed to optimize the routing search space for dynamic SFC routing. The Minimum Spanning Tree (MST) technique remains a foundational approach in selecting the minimum essential network structure [29]. MST constructs a tree that links all nodes using the least cumulative edge weight, effectively eliminating unnecessary connections and narrowing the search space[30]. This makes MST particularly valuable in network design and routing, where computational efficiency is crucial. Common algorithms for generating MSTs include Prim's [43], Kruskal's [42], and Boruvka's [39] algorithms. Prim's algorithm grows the MST by starting from any node and repeatedly adding the shortest edge that links to a new node [31]. In contrast, Kruskal's algorithm sorts all edges by weight and adds them incrementally to form the MST, ensuring no cycles occur [32]. Boruvka's algorithm, known for its suitability in parallel computing, expands multiple trees concurrently by choosing the shortest outgoing edge from each component until a unified MST emerges [33]. Each method has its strengths: Prim's excels in dense networks with numerous connections, Kruskal's is ideal for sparse graphs and benefits from efficient union-find structures, and Boruvka's is advantageous in distributed systems due to its inherent parallelism. The question arises here in discrete problem can MST ensure connectivity when we require desired no of links in the search space? Secondly, are the selected links being qualitative based on min delay.? Despite their strengths, MST algorithms primarily optimize for edge weight and do not consider additional network characteristics such as resource efficiency and quality of services, etc. This limitation means that while MSTs are effective for basic connectivity, they may fall short in complex or dynamic environments where additional performance metrics are critical.

Despite the central role of routing in dynamic Service Function Chain (SFC) mapping, existing studies implicitly assume that optimal or near-optimal routes can be identified from an effectively unbounded routing space without explicitly optimizing that space itself. This raises a fundamental and unresolved question: how can the shortest feasible path be efficiently discovered among an exponential number of routing alternatives when latency minimization is the primary objective? Current routing approaches neither define nor quantify the routing search space reduction achieved prior to optimization, nor do they evaluate how the absence of such pre-optimization affects the quality, feasibility, and computational efficiency of subsequent route selection. Consequently, latency optimization is performed over excessively large and poorly structured routing spaces, rendering many proposed solutions impractical for real-time deployment in dynamic NFV environments.

3 OVERVIEW OF TRADITIONAL GWO ALGORITHM

The Grey Wolf Optimization algorithm (GWO), introduced by Mirjalili and Lewis in 2014 [34], draws inspiration from the social dominance hierarchy observed in grey wolves, the apex predators in their ecosystem. The social hierarchy

within a pack of grey wolves, which includes alpha (α), beta (β), delta (δ), and omega (ω) wolves, each with distinct roles. The hunting behavior of grey wolves encompasses two primary phases: exploration and exploitation. The pseudo code of GWO algorithm is represented in Table 2

Table 2: Pseudo code of basic GWO algorithm

Step	Description
-	Initialize the grey wolf population X_i, where $i = 1,2, \dots, n$
-	Initialize parameters a, A, and C
1	Calculate the fitness of each search agent
2	$X_\alpha \leftarrow$ the best search agent
3	$X_\beta \leftarrow$ the second-best search agent
4	$X_\delta \leftarrow$ the third best search agent
5	While $t <$ MaxIter do
6	.....For each search agent do
7	..Update the position of the current search agent using the position update equation
8	.....End for
9	Update parameters a, A, and C
10	Calculate the fitness of all search agents
11	Update X_α, X_β, and X_δ
12	$t = t + 1$
13	End while
14	Return X_α

During the exploration phase, the search agents employ the coefficient vectors **A** and **C** to diverge from the prey and explore the search space globally. The exploration mechanism is mathematically modeled by computing the distance vector **D** between the prey's position vector $\mathbf{X}_p$ and the position vector of a search agent **X**. The agents update their positions using the vectors **A** and **C**, where the control parameter **a** decreases linearly from 2 to 0 over the course of iterations. Additionally, stochasticity is introduced through random vectors $\mathbf{r}_1$ and $\mathbf{r}_2$, which enhance population diversity and prevent premature convergence. In the exploitation phase, the algorithm progressively converges toward the optimal solution by intensifying the attack on the prey. As the search agents approach the optimal value, the parameter a continues to decrease, resulting in a corresponding reduction in the magnitude of **A**. Consequently, the values of **A** are constrained within the interval $[-2\mathbf{a}, ,2\mathbf{a}]$.

4 MINIMUM SPANNING TREE GREY WOLF OPTIMIZATION(MSTGWO)

4.1 Solution Representation

The substrate network considered for routing optimization comprises VNF nodes and links. The input values and hyperparameters for the algorithm are detailed in the Table. 3. The number of nodes and their distances from each other are reflected in this substrate network as an (N x N) dimensional matrix. The goal of the routing optimization issue is to minimize the search space based on the stipulated rules. The Grey Wolf Optimization (GWO) mathematical model uses each wolf to reflect a possible VNF placement solution in the search space. In order to solve this problem by means of the Minimum Spanning Tree Grey Wolf Optimization (MST-GWO) algorithm, a certain encoding and decoding scheme is proposed. Where every object of a network (e.g., nodes, links) is given a distinct integer ID in this encoding scheme. This ID contains detailed information regarding available or needed resources (e.g., propagation speed of the link, available bandwidth, power consumption rate per unit of bandwidth used, etc.). All these details determine the fitness value (F) of the network element.

The MSTGWO algorithm starts with the random initialization of wolves and chooses identifiers (IDs) that refer to particular elements of the network, i.e., links or nodes. These IDs are encodings of possible solutions. After initialization, the algorithm reads each of the selected IDs to extract its fitness value (e.g., delay). This assessment directs the process of updating the position based on a modified

mechanism suggested in discrete routing environments. As an example, suppose that a wolf first chooses ID 8 which is associated with a certain network link. The algorithm considers the delay of the associated link and adds it to the dynamics of the search. The exploration stage involves the algorithm generating diverse coverage of the solution space, which encourages extensive search. In the subsequent exploitation phase, wolves having the best fitness are selected to narrow down and converge on the best pathways. These solutions with good performance are re-encoded as network element IDs. Such a strategy corresponds to real-life networking experiences, where objects, such as links and nodes, are usually identified with some sort of unique identifier, e.g., IP addresses [41].

4.2 Overview of Proposed MSTGWO Algorithm

Gray Wolf Optimizer (GWO) was created to tackle continuous optimization, but the task of routing discussed in the present paper is discrete by its nature. To fill this gap, the proposed MSTGWO algorithm introduces some important changes, which are described in Table 3, directly addressing the initialization and position update mechanism to be discrete to fit the search space of routing. The search procedure is organized into two broad phases: Exploration and exploitation. In this arrangement, individual wolf agents are possible solutions. The agent that has the lowest fitness value is the most optimal. In the exploration phase, the algorithm tests a wide variety of solutions, measures each in terms of fitness and then ranks the solutions accordingly. The exploitation stage picks a subset of the highest-performing agents (k wolves) to narrow down the search around the promising areas. In order to improve the quality of solutions further and ensure network connectivity, the algorithm adds a constraint in the form of Minimum Spanning Tree (MST). This guarantees the selected wolves to create a connected structure of the search space. Also, a penalty is added to control the redundancy of solutions and to impose the choice of a specific number of exploitation wolves. Such penalty control combined with MST restraints makes an effective compromise between exploitation and exploration.

4.3 The Initialization Process

In the MSTGWO algorithm, each wolf (x) represents a possible solution to the routing problem. The population size of search agents is considered equal to the number of links in the network. Each search agent has a position in the search space surrounding the promising areas, and each position corresponds to a potential link between nodes. The nodes in the network are identified as promising areas in the search space. In the adapted initialization equation, certain terms from the original GWO are replaced to suit the discrete nature of the problem. Specifically, "rnd" and "dim" are replaced with "pick-rnd" and "ID of link", respectively. Furthermore, the upper bound "ub" and lower bound "lb" are not required to be calculated. This is because, unlike the continuous optimization context, there is no need to generate random solutions within a range. Instead, each search agent picks links randomly, which are then evaluated during the exploration and exploitation phases. The initial position x_i (link ID) represented by each search agent using the adapted equation refers to randomly selecting a link ID from the set of all available links IDs. This approach ensures that each search agent starts with a unique link ID, representing a potential solution that will be further evaluated through the algorithm's exploration and exploitation processes.

$$x_i = \text{pick\_rnd (ID of link)} \quad (8)$$

where $i = 1,2,3,\ldots,n$ denotes the index of the solution associated with the x_i search agent, with each agent selecting one potential link. The term "ID of link" functions as a discrete variable representing a specific link in the network. Each selected link ID corresponds to a particular connection and carries detailed information about the available resources, such as bandwidth, latency, or capacity. The complete operational logic of the MSTGWO algorithm is illustrated in the flowchart (see Fig. 1 and Table 4).

4.4 Position Update Process for Routing

In the original GWO, the position update mechanism is based on the positions of alpha, beta, and delta wolves. In nature as per the social hierarchy, alpha (α), beta (β), and delta (δ) wolves are dominant. They are familiar with the position of the prey in the search space. Therefore,

the whole population follows these three types of wolves and updates their positions. Whereas, parameters and A and C are used for continuous optimization in the original GWO and updated in each iteration. But, in the MSTGWO, the parameters and coefficient position vectors of traditional GWO are not needed to be initialized.

Table 3: The Key Differences And Significance Of The Proposed MST-GWO Algorithm Compared To The Original GWO Algorithm

Aspect	Original GWO Algorithm	Proposed MST-GWO	Significance of the Proposed Adaptations
Optimization Problem	Designed for continuous optimization problems	Adapted for discrete search space optimization problems, specifically for routing	Allows application to discrete search space problems, making the MST-GWO suitable for network routing optimization tasks
Initialization Process	Generates random solutions within a specified interval for each variable	Randomly selects solutions (link IDs) from a predefined range, representing network links	Tailors the algorithm to the nature of the problem, improving relevance and efficiency
Position Update Mechanism	Utilizes positions of alpha, beta, and delta wolves based on social hierarchy	Introduces a novel update process with exploration and exploitation phases specific to routing	Ensures better exploration and exploitation in the discrete context, enhancing optimization
Exploration Phase	Global search using coefficient vectors A and C to diverge from prey	Fitness value (propagation delay) based exploration, categorizing wolves by best fitness	Improves global search effectiveness in network routing by focusing on propagation delay
Exploitation Phase	Convergence towards the optimal value, updating positions based on dominant wolves	MST-based constraint and penalty function for selecting and optimizing best k wolves	Ensures comprehensive search space coverage and eliminates overlapping, optimizing exploitation
Categorization of Wolves	Alpha (), beta (), delta (), and omega () wolves based on fitness values	Alpha-1, Alpha-2, ..., Alpha-k based on min. fitness values, scalable to the network's size	Provides a more granular, scalable categorization, enhancing performance for large networks
Objective	Minimizes a continuous function	Minimizes infeasible solutions in network links, ensuring connectivity and optimal routing	Directly addresses the needs of network routing problems, making the algorithm more effective
Constraint Handling	N/A	Uses MST-based constraint for connectivity and a penalty function to avoid overlaps	Balances link quantity and quality, optimizing both connectivity and link performance

The novel position update mechanism is introduced for the optimization of the search space of solutions for SFC routing. The new position update process consists of two phases; first is the position update phase for exploration wolves, and the second phase is the position update phase for exploitation wolves. Since the optimization problem focuses on minimization, a wolf with a higher fitness value is considered farther from the prey and thus more suitable for exploration. Conversely, a wolf with the best (i.e., minimum) fitness value, indicating proximity to the prey, is regarded as part of the global best solutions. In the following, the exploration and exploitation phases are discussed in detail.

4.4.1 Exploration

As a population of N wolves is initialized, each wolf undergoes the exploration phase. In the natural environment, wolves know where the prey exists and move in those directions to search for it. Wolves that are far from the prey have the worst fitness values and are considered for global exploration, while wolves that are closer to the prey have the best (minimum) fitness values and are considered for local exploitation. These two groups capture their respective areas by updating their positions according to the position update schemes for exploration and exploitation to find the prey (the optimal solution). In the proposed position update process, after initialization, each wolf undergoes exploration.

Once the entire search space has been explored, later on some of the wolves that were responsible for exploration are chosen for

Table 4: Pseudo Code Of Proposed MSTGWO Algorithm

Step	Description	Step	Description
-	**Input**: The substrate network as an undirected graph $G(L, N)$	17	**If** $\sum_{(i,j)\in L} x_{(i,j)} < \lvert N\rvert - 1$, **assign** $f = 1$
-	**Output:** The optimized search space (network) for Service Function Chain (SFC) routing	18	**Otherwise, assign** $f = 0$
1	**Initialize** N **wolves**	19	**Compute penalty: Penalty** $(x) = \beta \cdot \max(0, \lvert N\rvert - 1 - \sum_{(i,j)\in L} x_{(i,j)}) \cdot f$
2	**For each wolf** i**, assign a random position** x_i = **pick_rnd(ID of link)**	20	**Compute total fitness:** $F_{\text{total}}(x) = F(x_{i_{\text{th}}}) + \text{Penalty}(x)$
3	**Calculate the fitness value for each** $x_{i_{\text{th}}}$: $F(x_{i_{\text{th}}}) = d_{pp}(i, j)$	21	**Update coverage of optimized search space**
4	**End initialization**	22	**Minimize objective:** $\sum_{(i,j)\in L} x_{(i,j)} y_{(i,j)} + \beta.\max(0, \lvert N\rvert - 1 - \sum_{(i,j)\in L} x_{(i,j)}) \cdot f$
5	**For exploration,** iterate over each wolf $x_{i_{\text{th}}}$ in the population	23	**Subject to constraints:** $x_{(i,j)} \in \{0, 1\}, y_{(i,j)} \in \{0, 1\} \forall (i, j) \in L$
6	Categorize wolves based on their fitness values F	24	**Ensure sum constraint:** $\sum_{(i,j)\in L} x_{(i,j)} = k \forall (i, j) \in L$
7	**Assign the first best wolf as** x_{a1}	25	**Repeat exploration and exploitation until convergence criterion is met**
8	**Assign the second-best wolf as** x_{a2}	26	**Update positions of wolves based on fitness improvement**

9	**Assign the third best wolf as x_{a3}**	27	**Recompute fitness for updated positions**
10	Assign remaining wolves in descending order of fitness	28	**Update x_{a1}, x_{a2}, x_{a3} as per new fitness values**
11	**End exploration phase**	29	**Apply MST and penalty again on updated top wolves**
12	**For exploitation, select top k wolves based on best fitness values**	30	**Compute updated total fitness $F_{\text{total}}(x)$**
13	**Apply Minimum Spanning Tree (MST) to selected wolves**	31	**Enhance search space coverage using updated solutions**
14	**Ensure connectivity: $\sum_{(i,j)\in L} x_{(i,j)} \geq \lvert N\rvert - 1$**	32	**Continue iterative process until g_{best} wolf is determined**
15	**Sort links in MST: $\{d_{pp}(i,j)_1 \leq d_{pp}(i,j)_2 \leq \cdots \leq d_{pp}(i,j)_k\} \subseteq L$**	33	**Return optimized search space based on g_{best} wolf, ensuring network connectivity**
16	**For each solution x, check penalty condition**		

exploitation based on their best (minimum) fitness values that were acquired during the exploration phase. The propagation delay is considered as the fitness value of the wolves. The fitness value of the wolves can be calculated using the following equation:

$$F_{x_{i_{\text{th}}}} = d_{pp}(i,j) \quad (9)$$

Where, F is the fitness value of the i_{th} wolf $x_{i_{\text{th}}}$ and $d_{pp}(i,j)$ is the propagation delay. In the search space, each wolf represents a part of the solution to the problem. In the original GWO, the wolves are categorized into four classes: alpha (α), beta (β), delta (δ), and omega (ω) based on their fitness values from minimum to maximum. However, in the scenario of a discrete search space, particularly when the number of values extends into the tens and hundreds, it becomes challenging to nominate only four categories. The routing problem can scale from small to large networks, increasing the number of nodes and links. Therefore, the social hierarchy of the original GWO needs adaptation as shown in Fig 1.

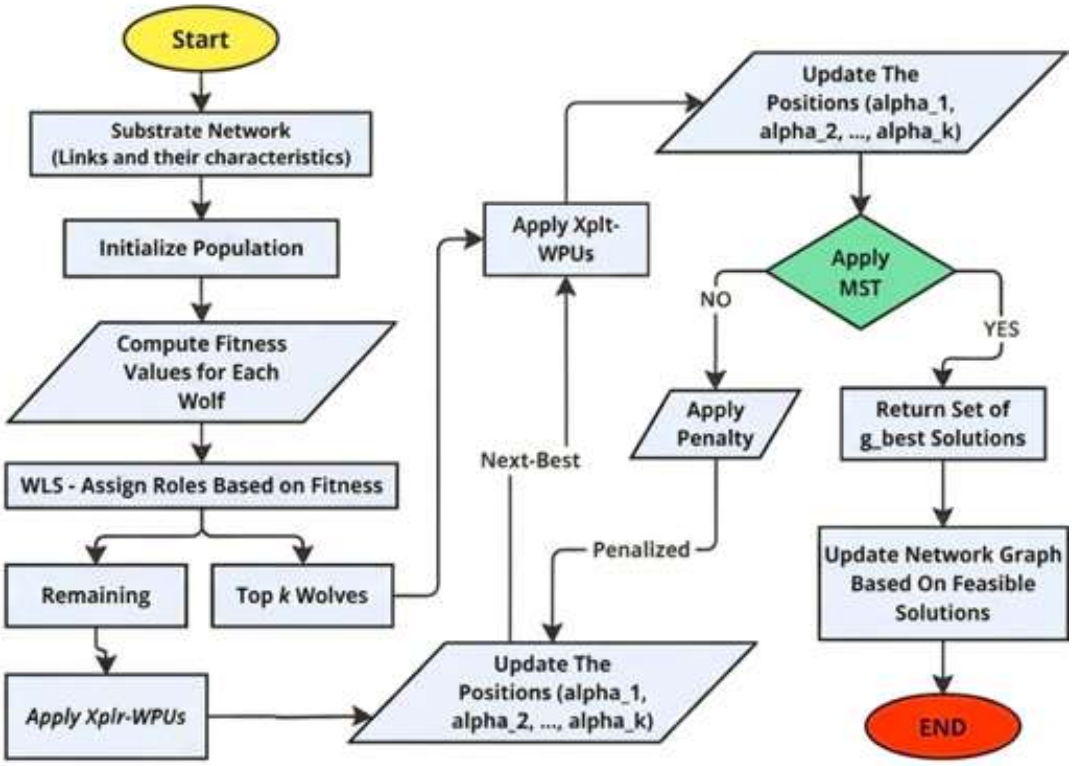

Fig. 1: Flowchart Of MSTGWO Algorithm

In the proposed scheme, the adapted hierarchy can encompass a set of the best wolves extending up to N dimensions, eliminating the limitation of categorizing solutions into alpha (α), beta (β), delta (δ), and omega (ω). The wolves are categorized into ranks from α_1 to α_k based on their fitness values. According to the adapted hierarchy, the positions of the wolves are updated sequentially in the order of α_1, α_2, and α_k by comparing their fitness values (sorting the links in ascending order based on the minimum propagation delay)

4.4.2 Exploitation

When the algorithm moves to the exploitation stage, a subgroup (k) of the wolves with the highest fitness values is chosen. It is assumed that these wolves correspond to the most promising areas in the search space and that they need to fulfill the conditions of choosing precisely k links with the least possible propagation delay. Nonetheless, this selection may occasionally result in an overlap that does not cover the search space sufficiently in promising directions. This drawback has been

referred to as poor or limited exploitation capacity, and it is an indication that the algorithm is struggling to find the right balance between optimality and diversity. To address this problem, the algorithm adds a new refinement layer that includes a Minimum Spanning Tree (MST) constraint and a penalty function. These elements cooperate to impose connectivity and disincentivize overlapping choices to enhance the quality of the final solution set.

The constraint on the MST guarantees that the chosen k wolves explore the entire important regions of the search space upon the exploitation phase. This limitation imposes connectivity, and the selected wolves cover the entire search space interestingly. Nonetheless, this constraint is a guarantee of coverage but not of best minimum fitness value of these wolves being globally the best. A penalty function is used to adjust the

selection process so that only the globally optimal solutions are selected. This penalty term is meant to penalize overlapping wolves, especially when part of the promising areas remains uncovered. The weight of this penalty determines how serious it is, and it is a multiple of the difference between the number of wolves chosen and the number required to cover all promising areas. This function helps to eliminate overlapping wolves from the selected k wolves and incorporates additional (next best) wolves if necessary. The penalty function adjusts the selection process to ensures that all promising areas are covered when selecting k best wolves for exploitation, thus balancing the goals of effective exploitation and performance optimization. Once the exploration and exploitation processes are completed, the algorithm provides an optimized search space of feasible solutions that effectively covered all promising areas, adhering to the specified objectives and constraints.

4.5 Time Complexity Analysis

The proposed algorithm for optimized search space construction in SFC routing demonstrates a time complexity of $O(m\log\log m + k \cdot m\log\log m)$, where m is both the number of links in the substrate network and the number of initialized grey wolves (candidate solutions), chosen deliberately to enable solution generation in a single iteration. The initial random position assignment and fitness evaluation for each wolf occur in linear time with respect to the population size, while sorting the population for exploration contributes $O(m\log\log m)$, complexity. In the exploitation phase, each selected wolf undergoes evaluation using a customized Minimum Spanning Tree (MST), rather than relying directly on classical algorithms such as Kruskal's or Prim's. The proposed approach checks whether the selected set of links meets the essential connectivity requirement specifically with at least ($|N|-1$), links. If this condition is violated, a penalty is introduced to the fitness function to discourage infeasible solutions. The process of verifying connectivity and evaluating the selected links requires sorting and comparison operations, leading to a time complexity of $O(m\log\log m)$, per wolf, where m denotes the number of links. Additionally, the space complexity of MSTGWO is $O(m^2+n^2)$, where n is the number of substrate nodes. This memory is primarily used for maintaining each wolf's binary representation in the search space, storing their respective fitness values, and managing a precomputed matrix of link propagation delays.

5 PERFORMANCE EVALUATION

This section evaluates the performance of the proposed optimization algorithm by detailing the simulation setup, parameter configurations, and selected performance metrics. The goal is to demonstrate how effectively and efficiently the algorithm enhances network performance, reduces computation time, and improves both Quality of Service (QoS) and resource utilization across different network topologies.

5.1 Simulation Setup

Simulations were conducted using MATLAB on an HP laptop powered by an Intel Core(TM) i5-3320M CPU running at 2.60 GHz , with 4 GB of RAM. To thoroughly evaluate the proposed optimization algorithm, we tested it across five distinct network topologies (two derived from real-world infrastructure and three randomly generated). For the real-world scenarios, we adopted the network setups described by [35], including the US and Pan-European topologies. These configurations reflect typical NFV (Network Function Virtualization) environments designed to support ultra–Reliable Low Latency Communications (RLLC) applications. Table 3 outlines the number of nodes and links for each topology. We treated the search space as discrete, assigning integer IDs to every node and link. Each ID encapsulates network characteristics such as server and switch roles, VNF assignments, routing elements, available bandwidth, propagation speed, power consumption per unit of bandwidth, and link distance. Alongside the real-world examples, we designed three random topologies of increasing size to assess the algorithm's scalability. In these topologies, each successive network doubled the number of nodes (from 14 up to 224). We calculated the number of links using the formula $L=\frac{N(N-1)}{2}$ ensuring full connectivity potential between nodes. These diverse topologies allowed for a robust assessment of the MSTGWO algorithm's adaptability across various network scales. The simulation aimed to enhance network performance by optimizing the routing search

space prior to dynamic SFC mapping. The focus was on reducing computation time, improving Quality of Service (QoS), and increasing the efficiency of resource usage. To ensure reliable and consistent results, each network topology underwent 100 simulation runs. We adopted the parameter values from [35], ensuring they reflect current trends in network link characteristics. Table. (5), provides a detailed summary of both the topological configurations and the algorithmic parameters used in the simulations.

5.2 Performance Metrics

The existing literature often treats search space optimization as a preliminary step within the broader scope of Service Function Chain (SFC) placement and routing. However, this critical phase has not been independently and thoroughly investigated, particularly in the context of dynamic SFC routing where proactive measures are essential. Most prior studies omit a dedicated analysis or performance validation of this phase, leaving a gap in understanding its actual impact on network optimization. While several works highlight the theoretical advantages of search space optimization such as reduced computation time [1], [13], [14], [26], [15], [25], [16], [17], [27], improved bandwidth availability [35, 15, 27], and reduced end-to-end latency [35] These claims often lack empirical support specific to the search space optimization phase itself.

Table 5: Simulation Parameters For SFC Optimization

Description	Input Values
Topological Parameters	
Number of Nodes in US network and Pan-European network topologies, respectively	{14,28}
Number of Nodes in random network topologies, respectively	{56,112,224}
Number of Links in the US network, Pan-European network, and random network topologies	$L = \frac{N(N-1)}{2}$
Propagation speed of link (km/sec)	[200000]
Distance/length of link (km), random	[350-4000]
Available bandwidth at the links (Gbps), random	[100,500]
Power consumption per unit of bandwidth usage at the link (Watt/Mbps), random	[0.2,0.5]
MSTGWO Parameters	
Number of Wolves (Population)	Equal to the number of links
Number of Runs	100

to address this gap, the present study proposes a dedicated optimization strategy using the MST-GWO algorithm. This paper evaluates the effectiveness of the proposed approach measuring computation time, available bandwidth, and end-to-end delay. The subsequent subsections define each metric and explain its role in the performance evaluation framework

5.2.1 Average execution time

The average execution time represents the execution time of the proposed algorithm takes in MATLAB to identify optimal routing solutions for a given network topology. Several factors influence this metric, including the number of nodes and links, the computational processes executed during each iteration, and the

complexity introduced by various optimization parameters and constraints. To ensure consistency and reliability, the algorithm executes 100 simulation runs for each of the five selected network topologies. This includes evaluating all possible links and nodes in the search space to determine the optimal routing paths. The final average execution time is then computed from these runs.

5.2.2 Average end-to-end delay

The average end-to-end delay (D) generally includes four components: transmission delay (d_t), queuing delay (d_q), processing delay (d_{pr}), and propagation delay (d_{pp}) [38, 26]. However, this study evaluates the network before traffic generation and transmission, queuing, and processing delays are considered negligible in end-to-end delay measurements [27]. Since data transmission through links does not occur prior to search space optimization, the simulation excludes dynamic traffic modeling and distribution, focusing solely on propagation delay allows for a more accurate and isolated assessment of each link's inherent latency characteristics [36]. Propagation delay refers to the time a signal takes to travel from the source to the destination over a specific link. This is calculated using the link's physical distance and the signal's propagation speed [40]. The following formula expresses this relationship for average propagation delay

$$d_{pp}^{(\text{Avg.})} = \frac{\sum_{(i,j)\in L} d_{pp}(i,j)}{|L|} \qquad (10)$$

where $d_{pp}^{(\text{Avg.})} = \frac{\sum_{(i,j)\in L} d_{pp}(i,j)}{|L|}$ represents the calculation of the propagation delay of the selected link between nodes i and j , and L is the total number of selected links.

5.2.3 Average available bandwidth

Average available bandwidth measures the amount of unused capacity on a link, expressed in Mbps (Megabits per second). It reflects the additional data that can be transmitted over the link without causing congestion.

$$Bw_{(i,j)}^{(\text{Avg.})} = \frac{\sum_{(i,j)\in L} Bw_{(i,j)}^{\text{Avl.}}}{|L|} \qquad (11)$$

This formula calculates the average available bandwidth of the links selected after optimization. $Bw_{(i,j)}^{(\text{Avg.})} = \frac{\sum_{(i,j)\in L} Bw_{(i,j)}^{\text{Avl}}}{|L|}$ represents the average available bandwidth in the network between nodes i and j , and List the total number of selected links

5.3 Baseline

The baseline results are established using well-known and widely used graph/topology computation algorithms, such as Dijkstra and Breadth-First Search (BFS), as referenced in existing studies. To collect the baseline results, highly connected underlying networks are considered, where each node is connected to others with the maximum possible number of links, as described in detail earlier. During the simulations, direct links between nodes are deliberately avoided when finding the shortest path. This approach aligns with realistic network scenarios, where direct links from the source node to the destination node are not always feasible. Notably, prior works have primarily discussed the theoretical benefits of search space optimization but have not thoroughly investigated its practical effects, resulting in a lack of benchmark results in this domain. This paper addresses this gap by conducting an in-depth investigation and presenting benchmark results to serve as a foundation for future research. We collected baseline results from the network topologies before applying the MSTGWO based search space optimization. These baseline values serve as benchmark for measuring the effectiveness of the proposed optimization technique. Specifically, they allow us to quantify improvements in computation time required for shortest path discovery, as well as enhancements in average end-to-end delay and bandwidth utilization following the application of the MSTGWO algorithm.

5.3.1 Breadth First Search (BFS)

Breadth-First Search (BFS) is a fundamental algorithm used to determine the shortest path between two nodes in unweighted network graphs. It explores nodes level by level, ensuring that all nodes within the same connected component are identified [27]. The algorithm starts from a designated source node and traverses all its immediate neighbors before progressing to

the next level of adjacent nodes. To manage this traversal efficiently, BFS employs a queue data structure. As each node is visited, it is added to the queue, and the algorithm proceeds iteratively by dequeuing nodes and exploring their neighbors until it either reaches the target node or exhausts all reachable nodes.

5.3.2 Dijkstra

Dijkstra's algorithm effectively determines the shortest path from a source node to all other nodes in a weighted graph, particularly when all edge weights are non-negative [36, 37]. The process begins by assigning a distance of zero to the source node and infinity to all other nodes, marking every node as unvisited. The algorithm then repeatedly selects the unvisited node with the smallest tentative distance, examines its neighbors, and updates their distances if it identifies a shorter path through the current node. Once a node has been visited, it is marked accordingly to avoid revisiting until all nodes have been visited or the shortest path to a target node has been identified.

5.4 Results and Analysis

The following subsections present assessment of the MSTGWO optimization algorithm. The results reveal marked enhancements in execution time, end-to-end delay, and bandwidth efficiency across different network topologies.

5.4.1 Execution Time

This study thoroughly examines how search space optimization impacts execution time across networks of different sizes. It evaluates performance by applying two widely used algorithms; BFS and Dijkstra, both before and after implementing the optimization.

A. Execution Time Using BFS Algorithm for Path Finding

The experimental analysis reveals that MSTGWO significantly reduces execution time around 34% in smaller networks and up to 91% in larger topologies, as shown in Fig. 2 and 3.

These improvements stem from MSTGWO's ability to guide BFS toward the most relevant nodes while avoiding unnecessary computations. The most notable gains appear in large-scale scenarios, underscoring the scalability of the proposed method.

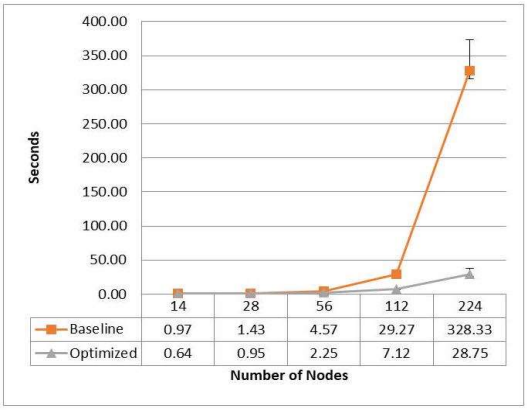

Fig. 2: Comparison of Pathfinding Time for BFS Algorithm in Baseline vs. Optimized Search Space

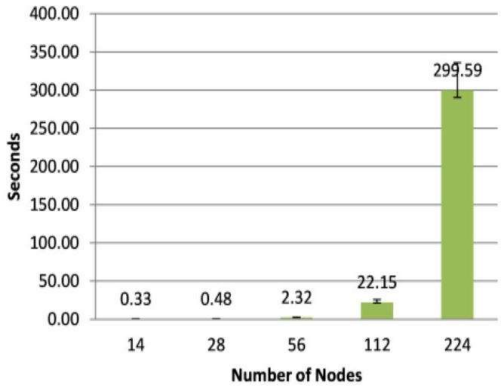

Fig. 3: Average Minimized Pathfinding Time of BFS Algorithm Across Five Networks with Search Space Optimization

This study also explores how the MSTGWO algorithm improves search space efficiency and overcomes the limitations of Breadth-First Search (BFS), particularly in largescale networks where BFS's exhaustive nature leads to high computational costs.

B. Execution Time Using Dijkstra Algorithm for Path Finding

This study demonstrates that search space optimization significantly reduces Dijkstra's algorithm's computational costs while preserving its shortest-path optimality. Execution time improvements range from 30% (small networks) to 91% (large networks) as shown in (Fig. 4 and Fig. 5) are achieved by focusing on relevant paths and minimizing unnecessary comparisons.

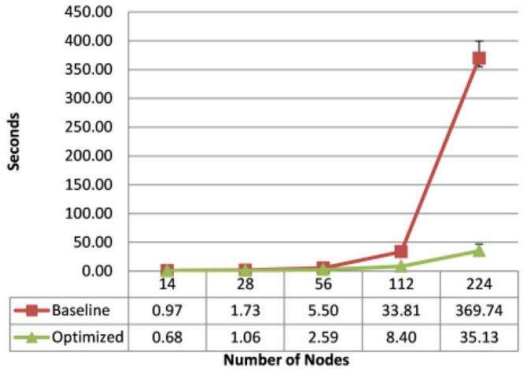

Fig. 4: Comparison Of Pathfinding Time For Dijkstra Algorithm In Baseline Vs. Optimized Search Space

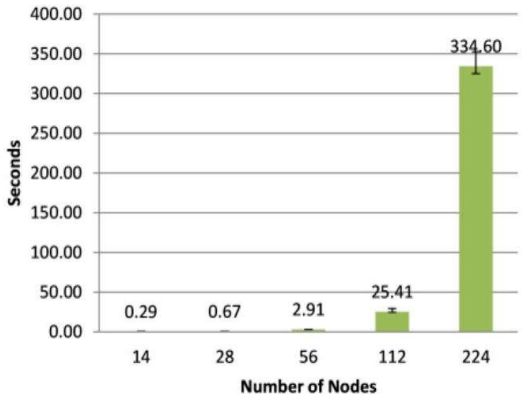

Fig. 5: Average Minimized Pathfinding Time Of Dijkstra Algorithm Across Five Networks With Search Space Optimization

These reductions enhance the algorithm's adaptability to complex networks, yielding practical benefits including faster data transmission and reduced computational overhead. The consistent performance gains across network sizes confirm the optimization's effectiveness for scalable routing solutions, enabling both Dijkstra's and BFS algorithms to better meet modern networking demands.

C. Execution Time of The Proposed MSTGWO Algorithm

The MSTGWO (Minimum Spanning Tree and Grey Wolf Optimization) algorithm is specifically designed to leverage search space optimization, introducing an innovative approach that achieves significant efficiency in execution times across networks of varying sizes. The execution times for MSTGWO in optimizing search spaces are 2.7, 4.6, 11.6, 39.5, and 158.1 seconds for 14, 28, 56, 112, and 224 -node networks, respectively, as illustrated in Fig.6. Even as network complexity scales, MSTGWO maintains comparatively low execution times, demonstrating its adaptability and robustness across diverse network environments. The integration of minimum spanning tree heuristics with Grey Wolf Optimization allows MSTGWO to keep computational costs minimal while delivering highly effective link optimization.

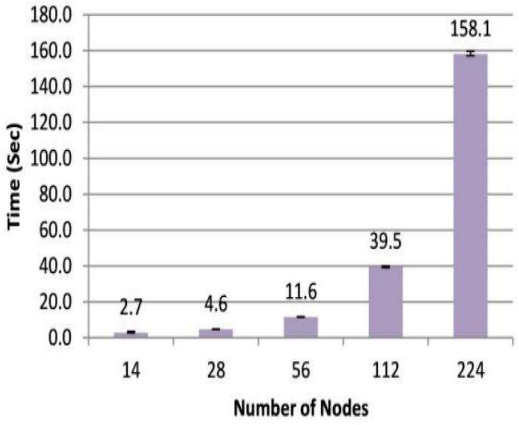

Fig. 6: Average Execution Time Of Proposed Algorithm For Search Space Optimization Across Five Network Configurations

5.4.2 Average End-to-End Delay

Search space optimization significantly improves end-to-end delay, a critical latency metric for real-time applications like multimedia streaming and IoT. The results show a marked improvement in end-to-end delay following search space optimization across various network sizes.

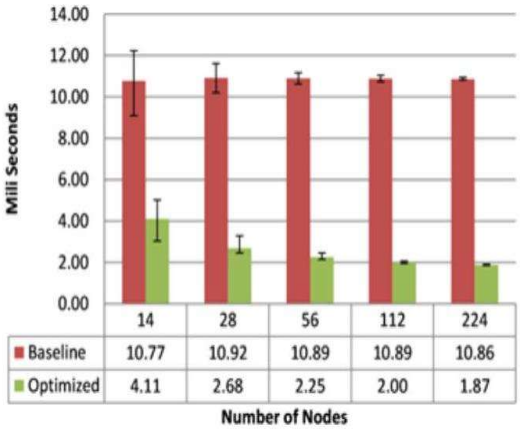

Fig. 7: Comparison Of Average Delay (Ms) In Baseline Vs. Optimized Networks Search Spaces

The optimization resulted in delay reductions of approximately 61.8% in small networks, with larger networks achieving reductions of up to 82.8%, as shown in Fig. 7 and Fig 8. This substantial improvement reflects the algorithm's ability to streamline pathfinding by avoiding unnecessary routes, effectively reducing the time packets need to travel from source to destination.

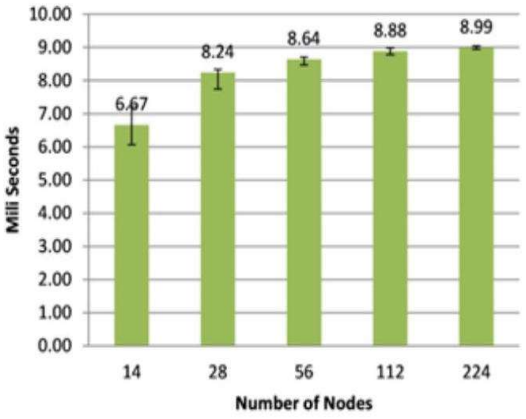

Fig. 8: Average Optimized Delay (Ms) Using Search Space Optimization Across Five Networks

5.4.3 Average Available Bandwidth

This study measured bandwidth changes across network sizes before and after search space optimization. Pre-optimization bandwidth neared full capacity (100%), while post-optimization showed reductions of 51% (small), 59% (medium), and 66% (large networks) (Fig. 9).

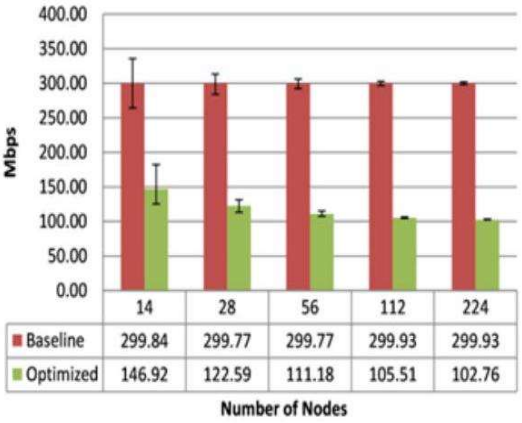

Fig. 9: Comparison Of Average Available Bandwidth In Baseline Vs. Optimized Networks

These intentional reductions support our URLLC optimization goals, prioritizing low latency and reliability over bandwidth availability. The optimization strategically focuses on minimum-delay paths, channels data through direct high-utilization routes, concentrates traffic on optimal paths and eliminates underutilized links. This traffic concentration on optimized paths reduces unused bandwidth, improving effective utilization by prioritizing latency/ reliability critical routes. Though available bandwidth decreases, throughput and efficiency for critical data flows improve through reduced redundancy and streamlined routing. In contrast, unoptimized networks, which may include more long or redundant routes, maintain higher levels of available bandwidth but also experience increased delays and greater computational complexities due to the broader range of routing options. This study's focus, however, is on optimizing the search space specifically for URLLC applications, prioritizing delay reduction as shown in Fig 10.

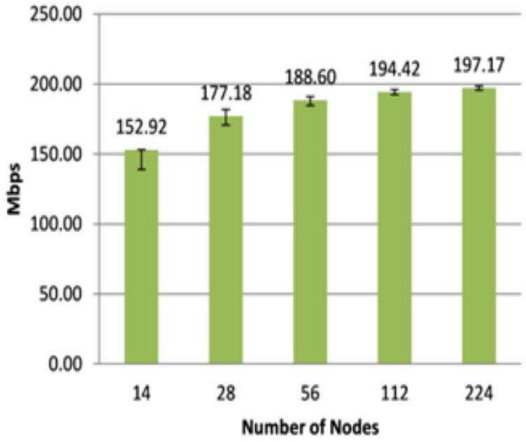

Fig. 10: Average Optimized Bandwidth Using Search Space Optimization Across Five Networks

In scenarios where increasing bandwidth availability is critical such as in multi-domain routing environments a trade-off between delay and bandwidth could be introduced. For example, core links (used in external multi-domain routing) might be differentiated from access links (used internally within a domain), balancing delay-sensitive paths with broader bandwidth needs.

5.5 Additional Simulation Observations

These analyses are based on additional simulations conducted by varying the number of links in the network to evaluate their impact on key performance parameters. This section systematically examines the influence of search space optimization on execution time, average end-to-end delay, and available bandwidth,

alongside assessing the performance of the proposed MSTGWO algorithm for search space optimization across diverse network sizes and search space configurations. The simulations leverage five predefined network configurations (as detailed earlier), with search spaces categorized into three classes: small, medium, and large, based on the ratio of links to nodes. This categorization allows a structured evaluation of how different search space sizes affect network performance and the computational efficiency of the optimization process

- Small Search Space: Links are equal to nodes.
- Medium Search Space: Links are 1.5 times the number of nodes, aligning with ratios for the US and Pan-European networks.
- Large Search Space: Links are doubled the number of nodes

These results specifically examine the performance ratios reported in related studies and analyze any deviations to uncover broader performance trends.

5.5.1 Execution Time

The discussion focuses on pathfinding time for BFS and Dijkstra on optimized search spaces and the optimization time for MSTGWO.

A. Average Pathfinding of BFS and Dijkstra

Pathfinding execution time is closely tied to the size of the search space being optimized. As the search space grows, both Breadth-First Search (BFS) and Dijkstra's algorithm require more time to execute due to the increased number of routing possibilities.

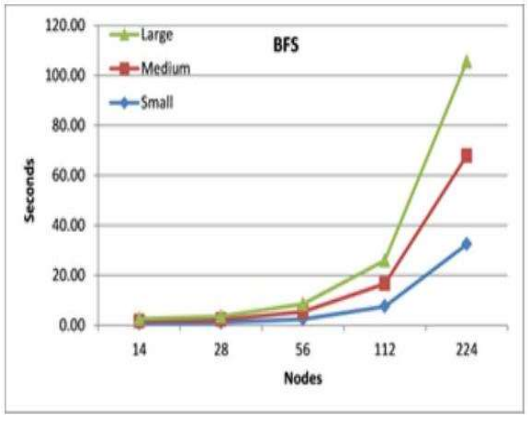

Fig. 11: Average Pathfinding Time Of BFS

- Breadth-First Search (BFS): BFS performs efficiently in smaller search spaces but shows a marked increase in execution time as the search space expands (Fig. 11). This is because the algorithm must process a larger number of nodes and links, which significantly raises computational demand.
- Dijkstra Algorithm: Similarly, Dijkstra's algorithm demonstrates rising execution times in larger search spaces (Fig. 12). The presence of more routing paths and weighted edges amplifies the algorithms complexity, contributing to longer computation durations.

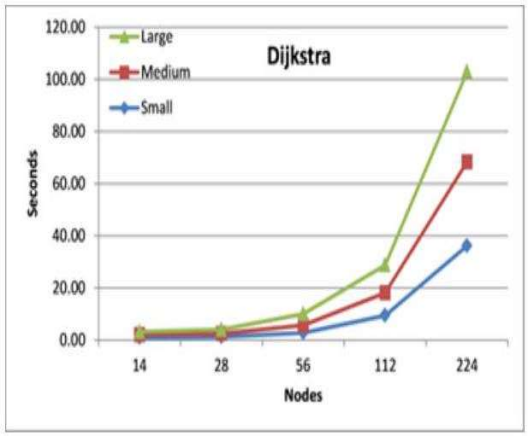

Fig. 12: Average Pathfinding Time Of Dijkstra

B. Execution Time of the Proposed MSTGWO

The execution time of the MSTGWO algorithm reflects the computational effort involved in optimizing the search space for dynamic Service Function Chain (SFC) mapping. As the size of the search space increases, the algorithm evaluates a greater number of possible configurations, which in turn leads to longer execution times as shown in fig 13.

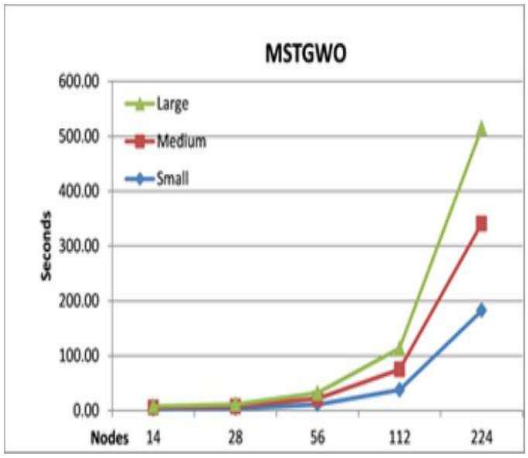

Fig. 13: Average Execution Time Of MSTGWO For Search Space Optimization

- Smaller Search Spaces: In smaller search spaces, MSTGWO exhibits faster execution times as it processes fewer links and nodes.
- Larger Search Spaces: As the search space expands, the execution time increases substantially. This increase results from the additional routing options that MSTGWO must evaluate, leading to higher computational overhead.

5.5.2 Average End-to-End Delay

1. Decreased Search Space: Smaller search spaces show significant improvement in end-to-end delay due to shorter and more direct routing paths as shown in fig.14.
2. Increased Search Space: Larger search spaces marginally increase delay due to the complexity of routing through more links.

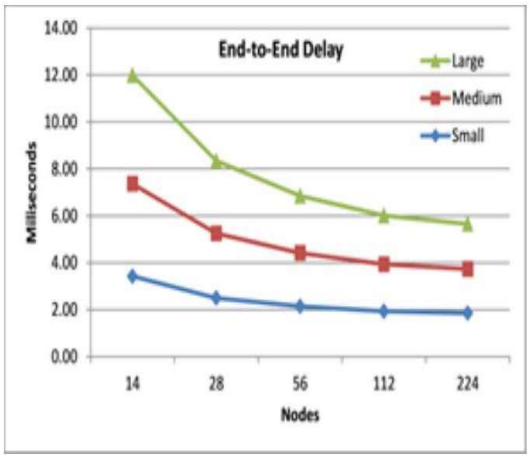

Fig. 14: Average End-To-End Delay

5.5.3 Average Available Bandwidth

1. Decreased Search Space: Smaller search spaces result in limited available bandwidth as the fewer links constrain the network's capacity to accommodate higher traffic (Fig.15).
2. Increased Search Space: Larger search spaces maximize bandwidth utilization due to increased connectivity.

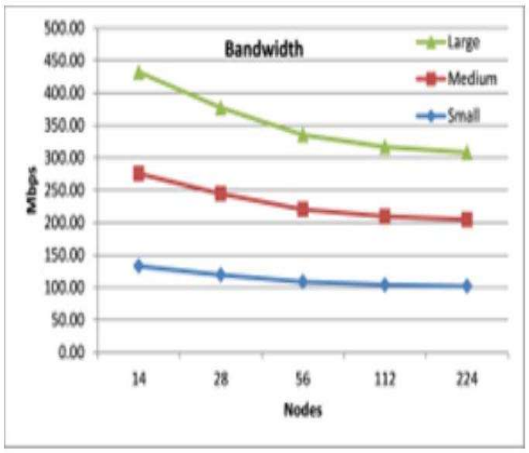

Fig. 15: Average Available Bandwidth In The Network

5.5.4 Key Insights on Balanced Search Space Sizing

- Decreased Search Space: Limiting the search space (e.g., to the number of nodes) accelerates BFS and Dijkstra execution by reducing routing options. However, this restricts connectivity and resource allocation, impairing scalability for high-demand scenarios. While beneficial for latency-sensitive tasks, the constrained bandwidth and flexibility render it unsuitable for large-scale deployments.
- Increased Search Space: Increasing the search space (e.g., double the nodes) enhances resource utilization and user capacity but at the cost of computational efficiency. The added routing complexity degrades optimization performance, elevating execution times for MSTGWO and pathfinding algorithms.

Aligned with US and Pan-European network benchmarks, medium-sized search spaces balance execution speed, resource efficiency, and end-to-end delay. This configuration validates the prescribed ratio's efficacy, demonstrating superior trade-offs between computational load, latency,

and adaptability. Minor result variations may stem from limited simulation runs in this ancillary analysis. While the core study focused on the established ratio, these findings underscore the need for ratio-adherent optimization in dynamic networks. Future work could explore extended simulations to reinforce these insights.

6 DISCUSSIONS

Optimization strategy with the primary objective of minimizing infeasible links in the routing search space while preserving efficient and strongly connected topologies for dynamic SFC mapping. The initial phase introduced an aggressive search space reduction mechanism, which successfully eliminated a large number of infeasible routing options but resulted in partial network disconnections due to excessive link pruning. To resolve this limitation and ensure the targeted attainment of a feasible number of links, a Minimum Spanning Tree (MST),,based connectivity constraint was integrated into the Grey Wolf Optimization (GWO) framework. This constraint guaranteed full network connectivity and produced a loop-free baseline topology without relying on classical deterministic MST algorithms, thereby maintaining flexibility within the metaheuristic optimization process.

While the MST constraint ensured structural feasibility, it exposed limitations in link quality selection, particularly with respect to end-to-end delay and bandwidth utilization. To address this, a delay-sensitive constraint was incorporated to prioritize qualitative links that satisfy latency requirements critical to SFC routing. Although this enhancement significantly improved latency performance, it also reduced routing diversity and produced overly simplified topologies that did not fully capture the complexity of real NFV infrastructures.

To balance these competing objectives, a multi-criteria optimization strategy was formulated, combining (i) controlled search space size, (ii) delay-aware link selection, and (iii) a penalty-based mechanism to enforce the desired level of redundancy. The penalty term guided the optimization toward solutions that met the targeted number of links, avoiding both over pruned and excessively dense topologies. This approach enabled the algorithm to reconcile connectivity, link quality, and routing robustness, particularly in scenarios where low-delay links alone could not sustain end-to-end SFC connectivity.

The resulting MSTGWO framework demonstrated substantial performance gains by effectively preoptimizing the routing search space. Execution time was reduced by up to 91%, and average end-to-end delay decreased by up to 82.8%, confirming that the elimination of infeasible links and the selection of high-quality routing paths directly enhance computational efficiency and QoS. Additionally, the observed 51,,66% bandwidth reallocation across network scales reflects a deliberate concentration of traffic over critical SFC paths, which is well aligned with the stringent requirements of latency-sensitive applications such as URLLC. Overall, the results confirm that targeted link selection and controlled search space optimization are key enablers for efficient and scalable SFC routing in dynamic NFV environments.

7 FUTURE WORK AND OPEN RESEARCH ISSUES IN SEARCH SPACE OPTIMIZATION

The proposed MSTGWO-based search space optimization framework is highly relevant to real-world telecommunications networks, where Service Function Chain (SFC) orchestration must be performed under strict latency, reliability, and scalability constraints.

In 5G and emerging 6G core networks, dynamic SFCs are routinely instantiated to support services such as ultra-Reliable Low-Latency Communication (URLLC), enhanced Mobile Broadband (eMBB), and network slicing. In such environments, the routing search space grows rapidly due to frequent traffic fluctuations and service mobility. The ability of MSTGWO to eliminate infeasible links early, while preserving full connectivity through a controlled number of links, directly supports real-time SFC orchestration within stringent control-plane timing requirements.

In Software-Defined Wide Area Networks (SDWANs) and carrier backbone networks, routing decisions often span geographically distributed nodes with heterogeneous link qualities. Here, ensuring that only qualitative links (low latency, high reliability) participate in SFC routing is critical to maintaining service-level agreements (SLAs). MSTGWOs penalty-

based link control mechanism enables operators to avoid both over-provisioned and overly sparse topologies, allowing traffic to be steered through stable and high-performing paths without exhaustive path enumeration.

The framework is also applicable to edge-cloud and multi-access edge computing (MEC) deployments, where latency-sensitive services such as autonomous driving support, augmented reality, and real-time video analytics require fast and reliable chaining of virtualized functions across edge and core nodes. By constraining the routing search space to a compact and connectivity-aware structure, MSTGWO ensures rapid decision-making while maintaining robustness against topology changes, which is essential in distributed edge environments.

Beyond telecommunications, the proposed approach is relevant to data center networks and cloud service infrastructures, where virtual network functions must be dynamically chained to support scalable services. In these scenarios, reducing infeasible routing options and enforcing a targeted number of high-quality links leads to improved resource utilization and faster service deployment. Overall, MSTGWO provides a practical, scalable, and QoS-driven routing framework that aligns well with the operational requirements of modern, dynamic networked systems.

The proposed MSTGWO-based search space optimization framework is highly applicable to real world telecommunications systems and other emerging networked infrastructures that require dynamic and latency-aware SFC orchestration. In 5G and upcoming core networks, services such as ultra-Reliable Low Latency Communication (URLLC), enhanced Mobile Broadband (eMBB), and network slicing rely on rapid and reliable SFC routing under constantly changing traffic conditions. By minimizing infeasible links and enforcing a targeted number of high-quality connections, MSTGWO enables efficient real-time orchestration while maintaining strict Quality of Service (QoS) guarantees.

In Smart City environments, where heterogeneous services such as intelligent traffic management, public safety surveillance, smart healthcare, and IoT data aggregation coexist, network conditions are highly dynamic and resource constrained. MSTGWOs ability to pre-optimize the routing search space and prioritize qualitative, low-delay links ensures timely delivery of mission-critical data while avoiding unnecessary routing complexity. The controlled redundancy introduced through penalty-based mechanisms also improves resilience against link failures, which is essential for large-scale urban infrastructures.

Similarly, in Unmanned Aerial Vehicle (UAV)-assisted networks, where aerial nodes support temporary coverage, disaster recovery, or data collection, network topology changes frequently due to mobility and energy constraints. In such scenarios, exhaustive routing exploration is impractical. MSTGWO effectively restricts routing decisions to feasible and connectivity preserving paths, enabling fast SFC establishment while adapting to rapidly changing link conditions. This makes the framework particularly suitable for UAV-enabled edge computing and aerial relay networks supporting latency-sensitive applications.

Overall, by systematically reducing infeasible routing options, achieving a balanced number of links, and selecting qualitative paths, MSTGWO offers a scalable and deployment-ready solution for dynamic SFC routing across telecommunications, smart city infrastructures, UAV-assisted networks, and other next generation networked systems.

8 CONCLUSION

This paper explores the routing search space optimization method by proposing the MSTGWO algorithm, a novel approach designed to identify qualitative links and systematically reduce infeasible paths for efficient routing of SFC orchestration. By constraining the search space through an MST-based connectivity mechanism and swarm-guided exploration, MSTGWO effectively eliminates redundant routing options in dynamic NFV environments.

The results demonstrate that this focused reduction has a direct, measurable impact on performance across diverse network topologies. Specifically, MSTGWO achieves up to a 91% reduction in execution time and an 82.8% reduction in end-to-end delay, confirming its suitability for latency-sensitive, real-time SFC orchestration in URLLC applications. Future work will involve extending the algorithm to support multi-domain and multi-tenant

environments, further enhancing its applicability to large-scale networks with heterogeneous QoS requirements.

ACKNOWLEDGEMENTS

This work is conducted at Sunway University and supported by the Ministry of Higher Education, Malaysia, under the Fundamental Research Grant Scheme (FRGS) FRGS/1/2022/STG07/SYUC/01/1.

Also, the authors used some AI tools only to check grammar and improve the clarity of the manuscript text. All research design, data analysis, interpretations, and conclusions are solely the work of the authors.

DECLARATION

Conflict of interest: The authors have no Conflict of interest to declare that are relevant to the content of this article. Disclosure Statement: The authors declare that there are no potential conflicts of interest with respect to the research, authorship, and publication of this article.

Data Availability Statement: The study is based on simulation data generated by the authors using standard network parameters and benchmark topologies reported in related literature. No publicly available datasets were used or created. The simulation parameters and settings are available from the corresponding author upon reasonable request.

REFERENCES

[1] Liao, C., Chen, J., Gao, D., Huang, X., Liu, S., & Qian, D. (2025). JRLAO: Joint Resource and Latency-Aware SFC Optimized Orchestration in NFV-Enabled Networks. IEEE Transactions on Cognitive Communications and Networking.

[2] Wei, W., Wang, Q., Gu, H., Zheng, D., Zhang, N., & Wu, C. (2025). An Adaptive Service Function Chains Mapping With Multi-Task Deep Reinforcement Learning. IEEE Transactions on Network Science and Engineering.

[3] Zhuge, B., Cai, X., Zhang, Z., Ren, Q., Dong, L., Jiang, X., ... & Lu, L. (2025). Service function chain mapping method based on delay guarantee. Cluster Computing, 28(5), 1-19.

[4] Hisyam Ng, H. A., & Mahmoodi, T. (2024). Machine Learning-Driven Dynamic Traffic Steering in 6G: A Novel Path Selection Scheme. Big Data and Cognitive Computing, 8(12), 172.

[5] Escheikh, M., & Taktak, W. (2024). Online QoS/QoEDriven SFC Orchestration Leveraging a DRL Approach in SDN/NFV Enabled Networks. Wireless Personal Communications, 137(3), 1511-1538.

[6] Lee, K. S., &Geem, Z. W. (2005). A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Computer methods in applied mechanics and engineering, 194(36-38), 3902-3933.

[7] Demirci, S., &Sagiroglu, S. (2019). Optimal placement of virtual network functions in software defined networks: A survey. Journal of Network and Computer Applications, 147, 102424.

[8] Kotthoff, L. (2016). Algorithm selection for combinatorial search problems: A survey. In Data mining and constraint programming: Foundations of a cross-disciplinary approach (pp. 149-190). Cham: Springer International Publishing.

[9] Afrasiabi, S. N. (2023). Algorithmic Solutions for Virtual Network Function Migration in Cloud (Doctoral dissertation, Concordia University).

[10] Ikhelef, I. A. (2024). Optimization of VNF placement and chaining according to NFV/SDN paradigms (Doctoral dissertation, Universitä Paris-Nord-Paris XIII).

[11] Tam, N. T., & Binh, H. T. T. (2024). Subswarm-guided ant colony optimization with enhanced pheromone update mechanism and beam search for VNF placement and routing. Applied Soft Computing, 153, 111263.

[12] Appari, K. K. Q-Learning-Based Path Selection for Service Function Chaining in Network Function Virtualization Environments. constraints, 12, 13.

[13] Zhang, C., He, Q., Li, F., Wang, X., Garg, S., Han, M. S. H. Z., & Yuan, W. (2025). GAI based Resource and QoE Aware Service Placement in Next-Generation Multi-domain IoT Networks. IEEE Transactions on Cognitive Communications and Networking.

[14] Mao, Y., Shang, X., & Yang, Y. (2023, May). Ant colony based online learning algorithm for service function chain deployment. In IEEE INFOCOM 2023-IEEE Conference on Computer Communications (pp. 1-10). IEEE.

[15] Farshin, A., & Sharifian, S. (2019). A modified knowledgebased ant colony algorithm for virtual machine placement and simultaneous routing of NFV in distributed cloud architecture. The Journal of Supercomputing, 75, 5520-5550.

[16] Diab, K., Lee, C., &Hefeeda, M. (2020, October). Oktopus: Service chaining for multicast traffic. In 2020 IEEE 28th International Conference on Network Protocols (ICNP) (pp. 1-11). IEEE.

[17] Addis, B., Belabed, D., Bouet, M., &Secci, S. (2015, October). Virtual network functions placement and routing optimization. In 2015 IEEE 4th International Conference on Cloud Networking (CloudNet) (pp. 171-177). IEEE.

[18] Zhao, S., Kang, Q., Wang, J., Hu, H., & Fu, Y. (2023). Intelligent Deployment of DelaySensitive Service Function Chain Based on Parallelization and Improved Cuckoo Search Algorithm. Wireless Communications and Mobile Computing, 2023(1), 6683900.

[19] Yu, X., Wang, R., Hao, J., Wu, Q., Yi, C., Wang, P., Niyato, D. (2024). Priority-aware deployment of autoscaling service function chains based on deep reinforcement learning. IEEE Transactions on Cognitive Communications and Networking, 10(3), 1050-1062.

[20] Wang, S., Yuen, C., Ni, W., Guan, Y. L., & Lv, T. (2022). Multiagent deep reinforcement learning for cost-and delaysensitive virtual network function placement and routing. IEEE Transactions on Communications, 70(8), 5208-5224. [21] Khoshkholghi, M. A., & Mahmoodi, T. (2022). Edge intelligence for service function chain deployment in NFV-enabled networks. Computer Networks, 219, 109451.

[22] Wei, S., Zhou, J., & Chen, S. (2022). Delay-aware multipath parallel SFC orchestration. IEEE Access, 10, 120035-120055.

[23] El Amri, A., & Meddeb, A. (2024, October). Optimal Routing for Competitive Service Providers in Network Virtualization Context. In 2024 IEEE/ACS 21st International Conference on Computer Systems and Applications (AICCSA) (pp. 1-7). IEEE.

[24] Sharif, Z., Jasser, M. B., Yau, K. L. A., & Amphawan, A. (2025). Advancements and challenges in latency-optimized joint SFC placement and routing: a comprehensive review and future perspectives. International Journal of System Assurance Engineering and Management, 1-34.

[25] Jin, P., Fei, X., Zhang, Q., Liu, F., & Li, B. (2020, July). Latency-aware VNF chain deployment with efficient resource reuse at network edge. In IEEE INFOCOM 2020-IEEE conference on computer communications (pp. 267-276). IEEE.

[26] Pei, J., Hong, P., Xue, K., Li, D., Wei, D. S., & Wu, F. (2020). Two-phase virtual network function selection and chaining algorithm based on deep learning in SDN/NFV-enabled networks. IEEE Journal on Selected Areas in Communications, 38(6), 1102-1117.

[27] Wu, Y., & Zhou, J. (2021). Dynamic service function chaining orchestration in a multi-domain: a heuristic approach based on SRv6. Sensors, 21(19), 6563.

[28] Asgarian, M., & Mirjalily, G. (2024). A comparative analysis of twostage approaches for embedding network function virtualization enabled multicast services. IET Communications, 18(13), 778-788.

[29] Zukhri, Z., Paputungan, I. V., Handayani, N. S., &Satriadi, V. (2021, February). Penalty Strategy in The Fitness Function of Grey Wolf Optimizer for Minimum Spanning Tree Problem. In IOP Conference Series: Materials Science and Engineering (Vol. 1077, No. 1, p. 012071). IOP Publishing.

[30] Fuchs, K. A Probabilistic Approach to Discover Heterogeneous Network Topologies.

[31] Godinho, N. P. L. (2017). Algorithms for the Min-Max Regret Minimum Spanning Tree Problem (Master's thesis, Universidade de Coimbra (Portugal)).

[32] Katsigiannis, A., Anastopoulos, N., Nikas, K., &Koziris, N. (2012, May). An approach to parallelize Kruskal's algorithm using helper threads. In 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (pp. 1601-1610). IEEE.

[33] Qiao, W. B., & Cräput, J. C. (2019). GPU implementation of Borśvkas algorithm to Euclidean minimum spanning tree based on Elias method. Applied Soft Computing, 76, 105-120.

[34] Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. Advances in engineering software, 69, 46-61.

[35] Shokouhifar, Mohammad. "FH-ACO: Fuzzy heuristic-based ant colony optimization for joint virtual network function placement and routing." Applied Soft Computing 107 (2021): 107401.

[36] Xing, H., Zhou, X., Wang, X., Luo, S., Dai, P., Li, K., & Yang, H. (2019). An integer encoding grey wolf optimizer for virtual network function placement. Applied Soft Computing, 76, 575-594.

[37] Khoshkholghi, Mohammad Ali, et al. "Service function chain placement for joint cost and latency optimization." Mobile Networks and Applications 25.6 (2020): 2191-2205.

[38] Sharif, Z., Jasser, M. B., Yau, K. L. A., & Amphawan, A. (2022, October). Towards Latency Aware Multi-joint Optimization Method for VNF Placement and SFC Routing Via Swarm Intelligence. In 2022 IEEE 12th International Conference on Control System, Computing and Engineering (ICCSCE) (pp. 25-30). IEEE

[39] Nešetřil, Jaroslav & Milková, Eva & Nešetřilová, Helena. (2001). Otakar vka on minimum spanning tree problem Translation of both the 1926 papers, comments, history. Discrete Mathematics. 233. 3-36. 10.1016/S0012-365X (00)00224-7.

[40] Dwaraki, A., & Wolf, T. (2016, August). Adaptive servicechain routing for virtual network functions in software-defined networks. In Proceedings of the 2016 workshop on Hot topics in Middleboxes and Network Function Virtualization (pp. 3237).

[41] Komisarek, M., Pawlicki, M., Kozik, R., Houbowicz, W., & Chora, M. (2021). How to effectively collect and process network data for intrusion detection?. Entropy, 23(11), 1532.

[42] Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical society, 7(1), 48-50.

[43] Prim, R. C. (1957). Shortest connection networks and some generalizations. The Bell System Technical Journal, 36(6), 1389-1401.