Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.051871

ARTICLE

Improving Network Availability through Optimized Multipath Routing and Incremental Deployment Strategies

Wei Zhang[1] **and Haijun Geng**[2,*]

[1]Computer Application Teaching and Research Section, China University of Labor Relations, Beijing, 100048, China

[2]School of Automation and Software Engineering, Shanxi University, Taiyuan, 030006, China

*Corresponding Author: Haijun Geng. Email: genghaijun@sxu.edu.cn

Received: 18 March 2024 Accepted: 10 May 2024 Published: 18 July 2024

ABSTRACT

Currently, distributed routing protocols are constrained by offering a single path between any pair of nodes, thereby limiting the potential throughput and overall network performance. This approach not only restricts the flow of data but also makes the network susceptible to failures in case the primary path is disrupted. In contrast, routing protocols that leverage multiple paths within the network offer a more resilient and efficient solution. Multipath routing, as a fundamental concept, surpasses the limitations of traditional shortest path first protocols. It not only redirects traffic to unused resources, effectively mitigating network congestion, but also ensures load balancing across the network. This optimization significantly improves network utilization and boosts the overall performance, making it a widely recognized efficient method for enhancing network reliability. To further strengthen network resilience against failures, we introduce a routing scheme known as Multiple Nodes with at least Two Choices (MNTC). This innovative approach aims to significantly enhance network availability by providing each node with at least two routing choices. By doing so, it not only reduces the dependency on a single path but also creates redundant paths that can be utilized in case of failures, thereby enhancing the overall resilience of the network. To ensure the optimal placement of nodes, we propose three incremental deployment algorithms. These algorithms carefully select the most suitable set of nodes for deployment, taking into account various factors such as node connectivity, traffic patterns, and network topology. By deploying MNTC on a carefully chosen set of nodes, we can significantly enhance network reliability without the need for a complete overhaul of the existing infrastructure. We have conducted extensive evaluations of MNTC in diverse topological spaces, demonstrating its effectiveness in maintaining high network availability with minimal path stretch. The results are impressive, showing that even when implemented on just 60% of nodes, our incremental deployment method significantly boosts network availability. This underscores the potential of MNTC in enhancing network resilience and performance, making it a viable solution for modern networks facing increasing demands and complexities. The algorithms OSPF, TBFH, DC and LFC perform fast rerouting based on strict conditions, while MNTC is not restricted by these conditions. In five real network topologies, the average network availability of MNTC is improved by 14.68%, 6.28%, 4.76% and 2.84%, respectively, compared with OSPF, TBFH, DC and LFC.

KEYWORDS

Multipath routing; network availability; incremental deployment schemes; genetic algorithm

Nomenclature

MNTC	Multiple Nodes with at Least Two Choices
TBFH	Two Best First Hops algorithm
OSPF	Open Shortest Path First
IS-IS	Intermediate System to Intermediate System
ISPs	Internet Service Providers
MPLS	Multi-Protocol Label Switching
ECMP	Equal Cost Multiple Paths
MRC	Multiple Routing Configurations
FCP	Failure Carrying Packet
IETF	Internet Engineering Task Force
IPFRR	IP Fast Reroute
LFA	Loop-Free Alternates
LFC	Loop-Free Condition
NPC	Node Protection Condition
DC	Downstream Condition
SPT	Shortest Path Tree
GA	Genetic Algorithm
SID	Segment ID

1 Introduction

The demand for the Internet has undergone significant transformation with its rapid development and advancement. Initially, the Internet was primarily utilized for non-real-time applications such as email communication and file transmission. However, over time, its capabilities have expanded to encompass real-time applications that require prompt and reliable network connectivity. These include online stock trading, remote surgery, multiplayer games, and instant messaging services.

As the Internet has become a medium for such diverse and demanding applications, routing availability [1,2] has emerged as a crucial metric to assess network performance. Routing availability refers to the ability of the network to maintain stable and efficient routing paths between source and destination nodes. It ensures that data packets are delivered to their intended destinations in a timely and reliable manner, minimizing delays, packet losses, and network congestion.

In today's interconnected world, where real-time applications play a pivotal role in various industries and daily lives, routing availability is paramount. It directly impacts the quality of user experience, business operations, and overall network reliability. Therefore, it has become an essential indicator for evaluating and improving network performance.

However, in real-world scenarios, network failures are inevitable occurrences that can significantly impact network performance and reliability. To address these challenges, intra-domain routing protocols such as Open Shortest Path First (OSPF) and Intermediate System to Intermediate System (IS-IS) are widely employed. While these protocols provide efficient routing mechanisms, they still face challenges when it comes to failure recovery.

During the repair process of these protocols, routing loops and black holes [3,4] may emerge, which can lead to packet loss and delay. Routing loops occur when packets constantly circulate within the network without reaching their destination, while black holes refer to areas in the network where packets may get trapped or disappear. Furthermore, the convergence time, which is the duration it takes for the network to stabilize after a failure, can be considerably long.

In the event of a failure, if a routing protocol is used to repair the issue, the repair time can range from a few seconds to tens of seconds. During this period, a significant number of packets may be discarded or delayed, resulting in a decrease in network performance and reliability. For Internet service providers (ISPs), such failures can have significant consequences. ISPs may be unable to meet the promised service quality levels [5–7], leading to dissatisfied customers and a potential loss of reputation and earnings.

Therefore, it is crucial for ISPs to have efficient failure recovery mechanisms to minimize the impact of network failures on their services. This involves employing advanced routing protocols, implementing fault tolerance mechanisms, and regularly monitoring and updating network infrastructure to ensure optimal performance and reliability.

The industry generally adopts passive recovery and route protection schemes [8,9] to improve network availability. Passive recovery speeds up route convergence by adjusting the default parameters of the routing protocol. However, this may cause route oscillation and network instability. Given a network topology, a route protection scheme calculates in advance the standby next-hop for all nodes to reach the destination address according to the no-loop rule, and these are used to forward the affected packets in case of network failure, which reduces the network interruption time and the packet loss rate and improves network availability. According to the method of forwarding packets, routing protection schemes can be divided into non-hop-by-hop and hop-by-hop forwarding. The former requires auxiliary mechanisms, such as Not-via [10], tunnel [11], Multi-Protocol Label Switching (MPLS) [12], and Segment Routing [3]. Both the MPLS and Segment Routing are two sophisticated networking technologies that significantly enhance network availability. MPLS operates by assigning unique labels to individual data packets upon their entry into the network. The working principle of the MPLS protocol is primarily based on label switching. The working mechanism of the MPLS protocol can be briefly summarized in the following steps: (1) Label Distribution: When a data packet enters the MPLS network, the first router (Ingress Router) assigns a unique label to it and attaches this label to the header of the packet. (2) Label Forwarding: Routers determine the next-hop router based on the label of the data packet and forward it to that router. Each router makes optimal routing decisions according to its label forwarding table. (3) Packet Transmission: The data packet traverses the network according to the label forwarding table until it reaches its destination. During transmission, routers quickly forward packets based on the labels, without the need to parse and process the IP header of each packet, thus reducing processing time and network latency. (4) Label Popping: When the data packet reaches its destination, the last router (Egress Router) removes the label from the packet and delivers it to the destination device.Through these steps, the MPLS protocol enables rapid packet forwarding and flexible routing choices, greatly enhancing the availability of the network. Segment Routing introduces the concept of source routing. Segment Routing introduces the concept of source routing. The fundamental principle of Segment Routing involves dividing the network path into multiple segments, each consisting of one or more nodes and links. Each segment is assigned a unique identifier, known as the Segment ID (SID). When a packet enters the network, the source node creates an ordered list of SIDs, referred to as the Segment List, based on a predetermined path. As the packet traverses the network, each node forwards the packet in the order of the SIDs listed in the Segment List until it reaches its destination. This approach can quickly reroute traffic in the event of failures, thereby enhancing network availability.

We introduce an innovative algorithm that aims to enhance network reliability by assisting nodes in discovering multiple next-hops for each destination. This approach ensures that the majority of nodes maintain at least two paths to their destinations, thus mitigating the risk of network failures and ensuring continuous connectivity.

To assess the efficacy of our proposed algorithm, we conduct rigorous evaluations using both real-world and generated network topologies. Our findings reveal that our algorithm exhibits high reliability, effectively dispersing traffic across multiple paths and minimizing the chances of network congestion or failures. Furthermore, our algorithm is designed to be compatible with the existing link-state routing protocols, ensuring smooth integration with existing network infrastructures. To facilitate the deployment of our algorithm, we propose three incremental deployment strategies. These strategies aim to identify the optimal set of nodes for deployment, ensuring maximum network coverage and minimal resource utilization. By leveraging our proposed algorithm and incremental deployment strategies, network administrators can enhance the reliability and resilience of their networks, providing improved connectivity and service availability to their users.

The remaining sections of this paper are organized as follows: Section 2 presents the background information and related works, Section 3 introduces our proposed algorithm, Section 4 delves into incremental deployment algorithms, Section 5 evaluates the performance of our proposed algorithm across various network topologies, and Section 6 concludes the paper.

2 Background

The earliest routing protection algorithm in the Internet, Equal Cost Multiple Paths (ECMP), which is an option in OSFP/intra-domain routing. ECMP assumes multiple paths with identical minimum cost between source and destination nodes can serve as backups. Despite its simplicity and ease of implementation in OSFP/intra-domain routing, ECMP's key limitation is that all backup paths must have the same cost, limiting its contribution to route availability. Attempts to address this by configuring the shortest path with the same cost through link weight adjustments have proven computationally intractable due to the NP-hard nature of the problem [13]. This complexity arises from the need to consider all possible link weight combinations to achieve the desired cost, challenging efficient implementation in large-scale networks. While ECMP offers basic routing protection, its limitations highlight the need for more advanced routing protection algorithms in modern networks.

The routing deflection algorithm introduced in [14] aimed to enhance route availability by meticulously computing potential next-hops using a loop-free rule. This approach prevented routing loops and employed label technology for flexible message forwarding, improving routing efficiency. However, its implementation is complex and costly, involving careful planning of network topology and additional hardware requirements. These challenges limit its practicality in large-scale networks. Consequently, there is a need for more cost-effective and practical routing algorithms to enhance route availability.

Multiple Routing Configurations (MRC) [15] aim to improve network resilience by maintaining alternative routing options for each router in case of failures. This approach involves strategic adjustment of link weights and comprehensive consideration of single failures. However, implementing MRC poses challenges due to high computing and storage requirements, especially in large-scale networks. On the other hand, the Failure Carrying Packet (FCP) [16] incorporates fault information encountered during message forwarding into its header. By analyzing this information, nodes can construct a new topology and recalculate the shortest loop-free path for reliable message delivery.

This approach enhances network resilience and ensures efficient routing. In [17], authors present an algorithm, named TBFH, which computes the Two Best First Hop disjoint paths.

The Internet Engineering Task Force (IETF) has put forward the IP Fast Reroute Framework (IPFRR) as a solution to minimize the impact of network failures. The aim is to reduce the packet loss rate caused by these failures and shorten the network interruption time as much as possible. IPFRR provides a structured approach to fast rerouting, ensuring that packets can be quickly redirected around failed links or nodes. One of the simpler implementations of IPFRR is Loop-Free Alternates (LFA), which has been widely adopted by router manufacturers [18,19]. RFC 5286, a key document published by the IETF, outlines the basic framework of IPFRR and specifies the implementation details of LFA. These include Loop-Free Condition (LFC), the Node Protection Condition (NPC), and the Downstream Condition (DC).

However, there are two primary challenges in implementing LFA:

Firstly, the complexity of LFA is relatively high. Each router needs to construct multiple shortest path trees, including the shortest path tree with itself as the root and the shortest path trees with all its neighbors as the roots. As the number of nodes increases, so does the complexity of this task. This results in higher implementation costs, as routers need to have sufficient processing power to handle the additional computations.

Secondly, the route availability with LFA is lower compared to other routing mechanisms. The single-fault protection rate of LFA is approximately 50%, meaning that in the event of a failure, only half of the traffic can be rerouted using LFA. This can lead to partial network outages or increased latency for affected traffic.

Despite these challenges, the benefits of IPFRR and LFA in particular are significant, especially in mission-critical networks where minimizing downtime and packet loss is crucial. Future research and improvements in routing algorithms could potentially address these issues, further enhancing the reliability and performance of IPFRR-based solutions.

3 MNTC Algorithm

3.1 Basic Idea

We explain the core idea of the MNTC algorithm through an example. The left side of Fig. 1 shows a network topology with six routers and eight links. We use the Dijkstra algorithm to build a shortest-path tree with a as the root and assign a sequence number to each node to indicate the order in which they are added to the tree, as shown on the right side of Fig. 1.

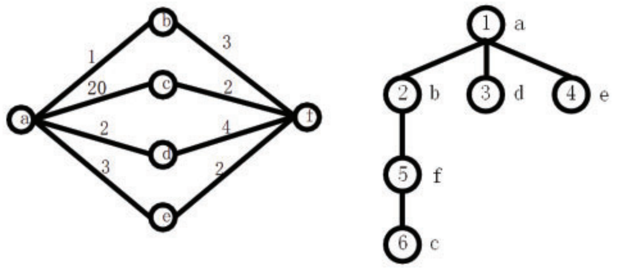

Figure 1: An example

If all nodes in the network can build exactly the same tree, e.g., the same shortest path tree (SPT) with the same root, they will have the same sequence number assigned to each node. To deliver a packet to node a, a node can forward it to any neighbor whose sequence number in T_a is less than its own. This will never introduce a loop, because the sequence number of a node on the induced forwarding path can only be reduced. For example, if a packet destined for a arrives at c, then c can select f or a as its next-hop to forward these packets to, because their serial numbers are less than that of c. Similarly, f can choose b, d, or e, but not c, as its next-hop. Thus, although these nodes make single-hop forwarding decisions in a distributed and independent manner, the multi-hop forwarding path made by these decisions can ensure no loop.

We delve deeper into the core principle of the MNTC algorithm by way of an illustrative example. Consider the network topology depicted on the left side of Fig. 1, which comprises six routers and eight links. To establish a foundation for routing decisions, we utilize the renowned Dijkstra algorithm to construct a shortest-path tree (SPT) rooted at node a. This SPT not only identifies the most direct routes between nodes but also assigns a unique sequence number to each node. These sequence numbers, as shown on the right side of Fig. 1, serve as indicators of the order in which nodes were included in the tree. A crucial aspect of this algorithm is the consistency of the SPT across all nodes in the network. Imagine if all nodes were able to build the exact same SPT, rooted at the same node. In such a scenario, the sequence numbers assigned to each node would be identical for all nodes in the network. This homogeneity in numbering is essential for ensuring loop-free routing.

To efficiently deliver a packet destined for node a, any node in the network can forward it to any of its neighbors that have a lower sequence number than itself in the SPT. This approach is guaranteed to prevent loops, as the sequence numbers of nodes encountered on the forwarding path can only decrease. For instance, if a packet bound for node a reaches node c, c can choose to forward it to either f or a since both have sequence numbers lower than c. Similarly, f can select b, d, or e as its next-hop, excluding c due to its higher sequence number. Remarkably, even though these forwarding decisions are made in a distributed and autonomous manner by individual nodes, the cumulative effect of these decisions across multiple hops ensures a loop-free forwarding path. This fundamental property of the MNTC algorithm enables efficient and reliable routing in complex network topologies.

3.2 Notations and Algorithm Specifications

We define some symbols and summarize them in Table 1. The network is modeled as a graph, $G = (V,E)$, where V and E are the sets of nodes and edges, respectively, in the network. Each link $(u,v) \in E$ has a cost $L(u,v)$. The notation $r(u,v)$ denote the failure probability of link (u,v), and the link failure events are statistically independent. For a destination d, T_d is the SPT, with d as the root.

Table 1: Notations

$G = (V,E)$	Graph with node set V and edge set E
$L(u,v)$	Direct link cost between nodes u and v
$r(u,v)$	Link failure probability between nodes u and v
T_d	Shortest path tree rooted at node d
$A(G,M)$	Network availability

Given a network graph G, our goal is to achieve high network availability by maximizing the number of nodes with at least two next-hops at each destination. We formally define network availability $A(G,M)$ [20] in the Eq. (4), where M is the node set for deploying MNTC, as an indicator to evaluate the capability of our scheme.

The end-to-end availability of a source-destination $(s\text{-}d)$ pair is defined as the probability that packets can be correctly forwarded from the source (s) to the destination (d). This metric is crucial in network reliability analysis as it quantifies the likelihood of successful packet delivery in the presence of potential failures. Let's assume that there are n forwarding paths from the source (s) to the destination (d). Each path, represented by $p_i(s,d)$, has a certain probability of being available or functional. The availability of a path is typically determined by the availability of the links along that path. If a link fails, the entire path may become unavailable. If we let $A_i\ (s,d)$, represent the availability of the i-th path $p_i(s,d)$, then the end-to-end availability can be expressed as:

$$P\left(A_i\left(s,d\right)\right)=\prod_{\forall(m,n)\in pi(s,d)} r\left(m,n\right) \tag{1}$$

According to the inclusion-exclusion principle, the end-to-end availability of a source-destination pair can be expressed as a sum of probabilities that involves the availability of individual paths as well as the interactions between them. This principle allows us to account for the overlapping and interdependencies among different paths, which is crucial in accurately calculating the overall availability. The end-to-end availability of a source-destination pair can be expressed as:

$$A\left(s,d,M\right)=\sum_{k=1}^{n}\left(-1\right)^{(k-1)}Sk \tag{2}$$

where Sk is the sum of the probabilities that a unique set of k paths from s to d are simultaneously working, i.e.,

$$Sk=\sum_{i<j<\cdots<k} P\left(Ai\cap Aj\cdots\cap Ak\right)=\sum_{i<j<\cdots<k}\left(\prod_{(m,n)\in pi(s,d)\cup pj(s,d)\cup\cdots\cup pk(s,d)} r\left(m,n\right)\right) \tag{3}$$

Then, network availability can be expressed as the availability of all source-destination node pairs divided by $|V|$ multiplied by $|V|$–1, where $|V|$ represents the total number of nodes in the network.The formula is expressed as:

$$A\left(G,M\right)=\frac{\sum_{s,d\in V,s\neq d} A\left(s,d\right)}{|V|*\left(|V|-1\right)} \tag{4}$$

3.3 Algorithm

Algorithm 1 presents a detailed execution process for the MNTC algorithm. The algorithm takes the network topology and a destination address d as input. The ultimate output is a sequence number assigned to each node in the network. The sequence number for the destination node d is initialized as 1 (line 1), serving as the starting point for the subsequent steps. Two sets, U and M, are introduced to assist in the algorithm's execution. Set U represents the collection of nodes that have been incorporated into the SPT rooted at node d, denoted as T_d. Set M, on the other hand, encompasses the nodes that have a direct link to set U in the topology T_d. The notation $NL(u,M)$ refers to the count of links between a given node u and the nodes belonging to set M. In line 2, the destination node d is added to set U, marking it as the root of the SPT. Subsequently, Dijkstra's algorithm is employed to compute the SPT rooted at node d (line 3). Dijkstra's algorithm is a well-known technique for finding the shortest paths from a source node to all other nodes in a graph, making it suitable for this context.

The algorithm then enters an iterative process. In each iteration, a node u belonging to set M and satisfying the condition $NL(u,M) \geq 2$ is considered for addition to the SPT. If multiple such nodes exist, the one with the smallest sequence number is selected for inclusion (lines 6–7). This ensures that nodes with a higher degree of connectivity are prioritized, while also maintaining a consistent numbering scheme. If no node satisfying the $NL(u,M) \geq 2$ condition exists, the algorithm falls back to adding the node with the smallest sequence number from set M to the SPT (lines 6–7). This ensures that the SPT remains connected and comprehensive, even when there are no nodes with a high degree of connectivity.

After adding a node u to the SPT, its sequence number is updated accordingly (line 8). The specific manner of this update depends on the algorithm's internal logic and may involve considering the sequence numbers of neighboring nodes or other relevant factors. Node u is then added to set U, reflecting its inclusion in the SPT (line 9). Finally, the value of the counter i is incremented by 1 (line 10). This counter likely serves to track the progress of the algorithm or keep track of the number of iterations performed. By incrementing it at the end of each iteration, the algorithm ensures that it can terminate when all eligible nodes have been processed and added to the SPT. The overall process described by Algorithm 1 is designed to construct an SPT rooted at the destination node d that takes into account the connectivity and sequence numbering of nodes in the network. This SPT serves as a critical component in the MNTC algorithm, enabling efficient routing and network operations.

Algorithm 1: MNTC

```
Input $G = (V, E)$
Output Destination $d$
1  $d.sequence \leftarrow 1$
2  $U = \{d\}$
3  Compute $T_d$
4    $i \leftarrow 2$
5  While $i \leq |V| - 1$ do
6    $M \leftarrow \{m \in V \setminus U | \forall (m, n) \in T_d, n \in U\}$
7    select a node $u \in M$
8    $u.sequence \leftarrow i$
9    $U \leftarrow U \cup \{u\}$
10   $i \leftarrow i + 1$
```

4 Optimal Incremental Deployment Problem

We deliberate on the deployment of MNTC within our network infrastructure. Given its unique capability of incremental deployment, the challenge lies in determining which nodes to prioritize for the deployment of the algorithm. This paper mainly solves the problem of how to select nodes from the network to deploy the MNTC algorithm using the incremental deployment method.

The deployment problem can be outlined as follows: The optimal deployment scenario involves finding the most suitable set of nodes in a given network $G(V,E)$ to deploy MNTC. This selection must consider the number of nodes to be deployed, denoted by k. The objective is to maximize network availability by carefully selecting the nodes where MNTC will be implemented. The constraint condition of this problem is the number of deployment nodes, K. For each node a, the value of deploying the node is represented as $V(a)$, and the objective is to maximize network availability. Each node can either be deployed with the MNTC algorithm or remain undeployed. This problem is analogous to the classic Knapsack Problem. In both cases, there is a limit on the number of items/nodes that can be selected (capacity in the Knapsack Problem and the number of deployment nodes k in this problem). Similarly, each item/node has an associated value (value of an item in the Knapsack Problem and the deployment value $V(a)$ in this problem). The objective is to select the optimal combination of items/nodes to maximize the total value within the given constraints. Since the problem is computationally complex and NP-hard, we introduce three heuristic algorithms to tackle it efficiently. These algorithms aim to provide practical solutions that balance network availability and deployment costs. These three algorithms include the traditional greedy algorithm (Algorithm 2), the simulated annealing algorithm (Algorithm 3), and the genetic algorithm (Algorithm 5). Both the greedy algorithm and the simulated annealing algorithm operate by selecting a node that offers the greatest contribution to network availability at each iteration, thereby facilitating the deployment of the MNTC algorithm.There are two reasons for considering the Genetic Algorithm to solve the incremental deployment problem. Firstly, it is a powerful optimization technique that can effectively handle complex problems with multiple variables and constraints. Secondly, it is a heuristic-based approach that can find near-optimal solutions.

4.1 Greedy Heuristic Algorithm

Since the optimal deployment problem is categorized as NP-complete, it becomes computationally intractable to perform a straightforward exhaustive search, especially when dealing with large values of k and complex topological structures. This means that traditional search algorithms would take an unreasonably long time to find a solution, making them unsuitable for practical applications. Therefore, alternative approaches such as approximation algorithms or heuristic methods are often employed to address this problem efficiently.

Definition 1. Let a be an arbitrary node within the network. The benefit of node a in relation to node set M, which has already been deployed, is denoted by:

$$B(a, M) = A(G, M \cup a) - A(G, M) \tag{5}$$

Formula (5) demonstrates the extent to which network availability is improved after node a deploys the MNTC algorithm.

Relying on Definition 1, we introduce the OptimalDeploy1() function to determine the optimal nodes for implementing the MNTC deployment algorithm. This algorithm requires the execution of k cycles, during which the node that offers the greatest benefit is chosen to deploy Algorithm 2. Specifically, in line 3 of the code, the function identifies the most beneficial node and deploys Algorithm 2 accordingly. The selection criteria are based on various factors such as node capabilities, network topology, and the specific requirements of the MNTC algorithm. By utilizing OptimalDeploy1(), we aim to maximize the overall performance and efficiency of the deployment process.

Algorithm 2: OptimalDeploy1(G,k)

```
1 $M \leftarrow \phi$
2 **While** $M \leq k$ **do**
3     $a \leftarrow \max(B(a, M))$
4     $M \leftarrow M \cup a$
5 **retrun** $M$
```

Drawing inspiration from the simulated annealing, we introduce an enhanced approximate algorithm based on Algorithm 2. The cornerstone of this enhanced approach is the selection of M, which represents the solution generated by Algorithm 2. Our objective is to identify the optimal solution and accept it if it offers superior network availability compared to the previous solution or if the system temperature T remains sufficiently high.

As the algorithm progresses, we introduce a gradual reduction in the system temperature T. This reduction ensures that the system increasingly gravitates towards achieving a high-quality local minimum. This strategy is crucial in preventing premature convergence to suboptimal solutions.

Algorithm 3 addresses the optimal deployment challenge. It takes k as input and outputs the optimal deployment strategy M. Initially, we select a candidate set of M and employ the MNTC algorithm, setting an appropriate initial system temperature (lines 1–2). A tuple, denoted as (i, j), represents a pair of nodes such that i and j are connected. We unconditionally accept *(i, j)* if it outperforms the previous solution or if a randomly generated temperature value is below the current system temperature T (lines 4–5). This mechanism ensures that the algorithm can effectively escape local minima, ensuring a more comprehensive search for the global optimum.

By incorporating these improvements, our enhanced algorithm offers a more robust and efficient approach to addressing the optimal deployment challenge within complex network environments.

Algorithm 3: OptimalDeploy2(G,k)

```
1 $M \leftarrow OptimalDeploy\,(G, k)$
2 $T \leftarrow T_0$
3 repeat
4    if $A\,(G, (M - c) \cup c') > A\,(G, M) \vee T > random\,(T_0)$ then
5          $M \leftarrow (M - c) \cup c'$
6    else
7          $T \leftarrow T - 1$
8 until $a \leftarrow \max\,(B\,(a, M))\; T > T_0 \wedge$ no such $(c, c')$ exists
9 retrun $M$
```

***Theorem 1*:** The time complexity of Algorithm 2 is $O(k * (n + A))$, where n is the number of nodes in the network.

***Proof*:** The time complexity of computing $A(G,M)$ is assumed to be $O(A)$. The time complexity of the Algorithm 2 can be analyzed in a simpler manner as follows:

Initialization: The initialization step is a constant-time operation, typically $O(1)$.

Main Loop: The while loop iterates until the set M contains k nodes. In the worst case, this loop will execute k times.

Finding Optimal Node: Inside the loop, the algorithm finds the node a that maximizes $B(a, M)$. This step has a time complexity of $O(n)$ since the graph has n nodes.

Updating the Set M: Adding a node to the set M has a time complexity that depends on the implementation of the set. In the best case, it could be $O(1)$. However, even if the worst-case time complexity for this operation is considered, it would still be dominated by the k factor from the loop.

Therefore, the overall time complexity of the algorithm can be expressed as $O(k * (n + A))$. This means that the algorithm's performance depends primarily on the number of nodes k to be deployed, the number of nodes n in the network. □

***Theorem 2*:** The time complexity of Algorithm 3 is at lease $O(k * (n + A) + T_0 * A)$, where n is the number of nodes in the network.

Proof: The time complexity of computing $A(G,M)$ is assumed to be O(A). Assuming that OptimalDeploy(G,k) is implemented using Algorithm 2, which has a time complexity of $O(k * (n + A))$. The time complexity of the algorithm can be analyzed as follows:

Initialization: The initial call to OptimalDeploy(G,k) has a complexity of $O(k * (n + A))$.

Main Loop: The main loop iterates until a stopping condition is met. In each iteration, it computes the value function $A(G, M)$ for the current deployment M and also computes $A(G, (M - c) \cup c')$ for a potential swap of nodes c and c'. Both of these computations have a complexity of O(A).

Randomness and Iterations: The number of iterations in the loop is not fixed but depends on the initial value of T_0 and the probability of accepting improvements based on a random condition. As a result, The maximum number of iterations is T_0.

Since the number of iterations is not fixed, we can say that the time complexity is at least $O(k * (n + A) + T_0 * A)$. The exact value of Iterations depends on the specific instance of the problem and the behavior of the algorithm. □

4.2 Genetic Algorithm

The genetic algorithm (GA), drawing upon the principles of evolutionary biology, serves as a powerful instrument in seeking approximate solutions to intricate search problems. Its core lies in biologically inspired mechanisms such as inheritance, natural selection, mutation, and sexual reproduction, iteratively guiding a population of potential solutions towards optimal outcomes. It is worth noting that GA do not inherently consider constraints, which can lead to the generation of infeasible solutions during the evolutionary search process, especially through crossover and mutation operations. To address this issue, we propose Algorithm 4, which incorporates a loss function to ensure that only feasible solutions are considered.

Definition 2. The decrease in network availability resulting from the elimination of node a relative to already deployed set of nodes M is denoted by:

$$L(a, M) = A(G, M) - A(G, M - a) \tag{6}$$

Formula (6) indicates the degree to which network availability is reduced when node a is removed from the set of nodes deployed with the MNTC algorithm.

Algorithm 4: Repair(P,k)

```
1 While $|P| > k$ do
2     $a \leftarrow \min(L(a, M))$
3     $M \leftarrow M - a$
4 retrun $P$
```

In the context of addressing the deployment problem, GA finds a natural fit in our proposed Algorithm 5.

Algorithm 5: GeneticDeploy(G,k)

```
1 Initialize the parameters
2 Construct initial population $G(0)$ according to Algorithm 3
3 Evaluate all individuals in $G(0)$
4 $t \leftarrow 0$
```

(Continued)

Algorithm 5 (continued)

```
5  While $t < maxgen$ do
6      $t \leftarrow t + 1$
7      select $G(t)$ from $G(t-1)$
8      alter $G(t)$ using variation operator
9      for $P \in G(t)$ do
10         if $|P| > k$ then
11             Repair$(P, k)$
12 Evaluate all individuals in $G(t)$
13 retrun $M$
```

Firstly, the deployment problem is represented through a binary string framework. Each binary string, consisting of zeros and ones, represents a unique deployment configuration, indicating whether a specific node is deployed or not. For instance, in a scenario with 10 candidate nodes, a binary string like [1 0 1 1 0 0 0 0 0 0] signifies that the first, third, and fourth nodes are deployed, while the remaining nodes are not.

Secondly, the initial population of potential solutions is generated through a strategic process. We employ a graph search strategy, akin to breadth-first search, to establish an ordered sequence of nodes. Subsequently, a set of random binary strings of a fixed size (n) is generated to represent these potential solutions. Algorithm 3 plays a pivotal role in facilitating the generation of this initial population.

Thirdly, the fitness function holds a crucial position in the genetic algorithm. It assigns a numerical value to each solution, effectively quantifying its quality or fitness. In our specific case, the fitness function is tailored to maximize network availability. A high fitness value indicates a solution that aligns well with our objectives, whereas a low fitness value suggests that the solution may not be effective and should be discarded during the evolutionary process.

Lastly, genetic operators play a fundamental role in guiding the evolution of the solution population over time. Selection, crossover, and mutation operators are introduced, each contributing to the generation of new and potentially improved solutions. The selection operator identifies promising individuals from the current population, while the crossover operator combines genetic information from parent solutions to create offspring with novel characteristics. Mutation, on the other hand, introduces random changes to the genetic makeup of individuals, increasing the diversity of the population and preventing premature convergence to suboptimal solutions.

By integrating the genetic algorithm into our proposed solutions in this manner, we leverage its ability to efficiently search for optimal deployment configurations within a complex problem domain. This approach not only enhances the likelihood of finding effective solutions but also promotes the exploration of diverse solutions, thus strengthening the robustness and adaptability of our overall solution framework.

***Theorem* 3:** The time complexity of Algorithm 4 is $O((N-k) * A)$, where N is the initial size of P, k is the number of nodes deployed the Algorithm MNTC, time complexity of computing $A(G,M)$ is assumed to be $O(A)$.

***Proof*:** The time complexity of the Repair(P,k) algorithm can be analyzed as follows:

While Loop: The main structure of the algorithm is a while loop that continues until the size of the set P is less than or equal to k. The loop will iterate as many times as necessary to satisfy this condition.

Finding the Minimum Element: In each iteration of the loop, the algorithm finds the minimum element a from a set of values returned by $L(a,M)$. This step will also have a time complexity of $O(A)$.

Updating the Set: After finding the minimum element a, the algorithm removes it from the set M. This operation typically has a constant time complexity, assuming that removing an element from the set is a direct operation.

Termination and Return: The loop continues until the size of P is reduced to k or less. Once the loop terminates, the algorithm returns the modified set P.

Therefore, the overall time complexity of the algorithm can be expressed as $O((N-k) * A)$. □

***Theorem 4*:** The time complexity of Algorithm 4 is $O(k * (n + A) + T_0 * A + maxgen * n * (N-k) * A$.

***Proof*:** The time complexity of the GeneticDeploy(G,k) algorithm can be analyzed by considering the individual steps and the time complexity of the subroutines involved.

Initial Population Construction: The initial population $G(0)$ is constructed using Algorithm 3, which has a time complexity of $O(k * (n + A) + T_0 * A)$.

Evaluation: Evaluating all individuals in the initial population $G(0)$ has a time complexity of $O(A)$.

Main Loop: The while loop continues until the current generation t reaches the maximum number of generations *maxgen*. The loop performs several operations in each iteration.

Selection: Selecting the individuals for the next generation $G(t)$ from the current generation $G(t-1)$ typically has a time complexity of $O(n)$.

Variation: The population $G(t)$ is altered using a variation operator, which could include crossover and mutation operations. The time complexity of this step is $O(n)$.

Repair: The time complexity of Repair(P,k) is $O((N-k) * A)$.

Termination and Return: Once the loop terminates, the algorithm returns the best individual M found during the evolutionary process. Therefore, overall time complexity can be approximated as $O(k * (n + A) + T_0 * A + maxgen * n * (N-k) * A$. □

5 Performance Evaluation

We conducted a thorough experimental evaluation of the performance of MNTC, a routing protection algorithm designed to maximize network reliability. In this evaluation, we carefully described the network topology, evaluation metrics, and experimental methods to ensure accurate and reproducible results.

To simulate real-world network conditions, we randomly generated the failure probability of each link, within the range of 0 to 0.02. This range represents a wide spectrum of potential link failures, allowing us to comprehensively test MNTC's performance under various scenarios. To assess MNTC's effectiveness, we compared its results with those obtained by three other widely used routing algorithms: OSPF, DC, LFC and TBFH. The idea of the DC rule is as follows: A node can select a neighbor as a backup next hop if the minimum cost from the neighbor to the destination is less than the minimum cost from the node to the destination. These algorithms serve as benchmarks, allowing us to gain insights into MNTC's performance relative to existing solutions. To generate various network topologies for our experiments, we utilized the Brite topology generator. This tool allows us to create networks with different characteristics and complexities, enabling us to thoroughly test MNTC in a wide range of network environments. We selected the parameters for Brite based on real-world network observations and requirements, as detailed in Table 2.

Table 2: BRITE parameters

Model	Waxman
N	20–1000
HS	1000
LS	100
m	2-10
NodePlacement	Random
GrowthType	Incremental
Alpha	0.35
Beta	0.65
BWDist	Heavy Tail
BwMin	100.0
BwMax	1024.0

Our primary evaluation metrics include network availability, path stretch, and partial deployment. By evaluating MNTC using these metrics, we aim to gain a comprehensive understanding of its strengths and weaknesses. The results of our experiments will provide valuable insights into the algorithm's potential to enhance network reliability and efficiency.

Network topology

To experimentally verify the performance of the algorithm, we ran OSPF, DC, LFA, and MNTC on multiple network topologies. The topological structures used in the experiment include the real network Abilene [21], measured topologies [22], and Brite topologies.

1) Abilene is a simple topology consisting of 11 nodes and 14 edges;
2) The measured topologies are published by Rocketfuel and includeing Exodus, Telstra, Tiscali, and Sprint.

5.1 Network Availability

Network availability measures the percentage of the network that remains operational despite link failures. In Section 3, we describe how to quantitatively calculate network availability in detail. Table 3 shows the network availability provided by each of the five actual ISP topologies, from which it can be seen that MNTC has obvious advantages over the other algorithms. For example, in the Exodus topology, MNTC achieves more than 98% network availability.

Table 3 presents a detailed comparison of network availability provided by each of the five real-world ISP topologies. This analysis reveals that MNTC significantly outperforms the other routing algorithms, demonstrating its clear advantages in terms of network reliability. Specifically, in the Exodus topology, MNTC achieves an impressive network availability of over 98%. This high availability percentage indicates that, even under conditions of link failures and network disruptions, MNTC is able to maintain a stable and reliable network connection. This is a remarkable achievement, especially when compared to the performance of other algorithms in the same topology.

The Exodus topology is just one example among the five ISP topologies considered in our evaluation. Similar trends are observed across all topologies, further confirming the superiority of MNTC in terms of network availability. This consistency in performance across different network configurations underscores the robustness and adaptability of MNTC to varying network environments. The high network availability achieved by MNTC is crucial for ensuring reliable network performance. It is particularly important in scenarios where network uptime is critical, such as in enterprise networks, data centers, and other mission-critical infrastructure. By providing such high levels of availability, MNTC enables organizations to rely on their networks for critical operations without the fear of downtime or performance issues.

Table 3: Network availability for real topologies

	Network	The number of nodes	The number of links	Network availability (%)				
				OSPF	TBFH	DC	LFC	MNTC
Real	Abilene	11	28	93.27	96.67	97.12	98.32	99.49
Measured	Exodus	79	294	84.54	90.34	91.25	92.45	98.35
	Telstra	108	306	85.34	92.13	93.23	95.56	97.89
	Tiscali	161	656	82.12	90.34	92.65	94.45	96.89
	Sprint	315	1944	81.76	91.35	93.25	95.45	97.13

The results presented in Table 3 clearly demonstrate the advantages of MNTC over other routing algorithms in terms of network availability. MNTC's ability to maintain high availability levels across various ISP topologies makes it a promising solution for enhancing network reliability and ensuring dependable network performance.

We conducted a thorough analysis of network availability across different topology sizes generated using the Brite topology generator. The results, presented in Fig. 2, provide valuable insights into the relationship between topology size and network availability. As expected, we observed a corresponding decrease in network availability as the topology size increased. This trend is consistent across all routing algorithms evaluated in our study. However, our proposed algorithm, MNTC, demonstrates significant advantages compared to the other algorithms. Specifically, when the topology scale reaches 1200 nodes, MNTC achieves a network availability of more than 92%. This is a noteworthy achievement, as it surpasses the performance of the other algorithms, which provide less than 90% network availability under the same conditions. The gap in performance becomes more evident as the topology size increases, further highlighting the scalability and efficiency of MNTC. The superior performance of MNTC can be attributed to its ability to effectively protect network paths against link failures. MNTC's routing strategies are designed to minimize the impact of link failures on network availability, ensuring that the network remains operational even under conditions of significant disruption.

We conducted a thorough analysis of network availability across different topology sizes generated using the Brite topology generator. The results, presented in Fig. 2, provide valuable insights into the relationship between topology size and network availability. As expected, we observed a corresponding decrease in network availability as the topology size increased. This trend is consistent across all routing algorithms evaluated in our study. However, our proposed algorithm, MNTC, demonstrates significant advantages compared to the other algorithms. Specifically, when the topology scale reaches 1200 nodes, MNTC achieves a network availability of more than 92%. This is a noteworthy achievement, as

it surpasses the performance of the other algorithms, which provide less than 90% network availability under the same conditions. The gap in performance becomes more evident as the topology size increases, further highlighting the scalability and efficiency of MNTC. The superior performance of MNTC can be attributed to its ability to effectively protect network paths against link failures. MNTC's routing strategies are designed to minimize the impact of link failures on network availability, ensuring that the network remains operational even under conditions of significant disruption.

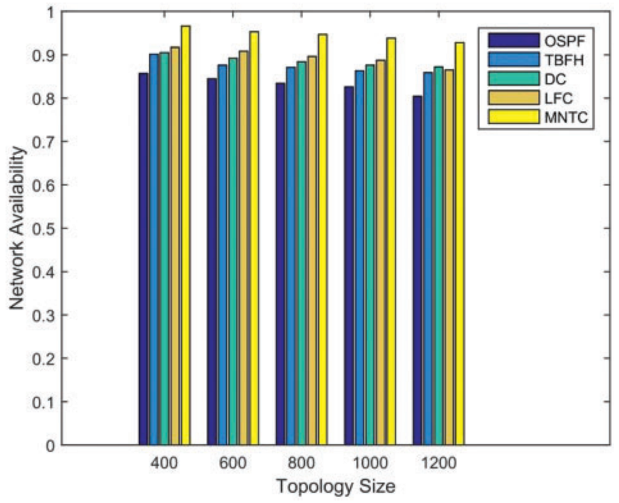

Figure 2: Average node degree = 6

The results presented in Fig. 2 not only confirm the advantages of MNTC but also underscore the importance of considering topology size when evaluating routing algorithms. In practice, organizations often need to scale their networks to meet growing demands, and it is crucial to ensure that the routing algorithm employed can maintain high levels of network availability even as the network size increases.

Our study demonstrates that MNTC offers significant advantages in terms of network availability across different topology sizes generated by Brite. Its ability to maintain high availability levels, even under conditions of large-scale networks, makes it a promising solution for enhancing network reliability and meeting the challenges of network scaling.

We conducted a detailed investigation to understand how network availability is influenced by changes in the average node degree of the topology. The findings, as depicted in Fig. 3, reveal an interesting trend: As the average node degree increases, the network availability also improves. This observation is significant as it highlights the critical role played by node connectivity in determining the overall reliability and performance of a network.

The improved network availability can be attributed to several factors. Firstly, a higher average node degree indicates a more connected network, with more paths available for data transmission. This diversity in paths reduces the dependence on any single link, minimizing the risk of network disruption due to link failures. Secondly, a higher node degree often leads to better load balancing, as data packets can be distributed more evenly across multiple paths, reducing congestion and improving overall network performance.

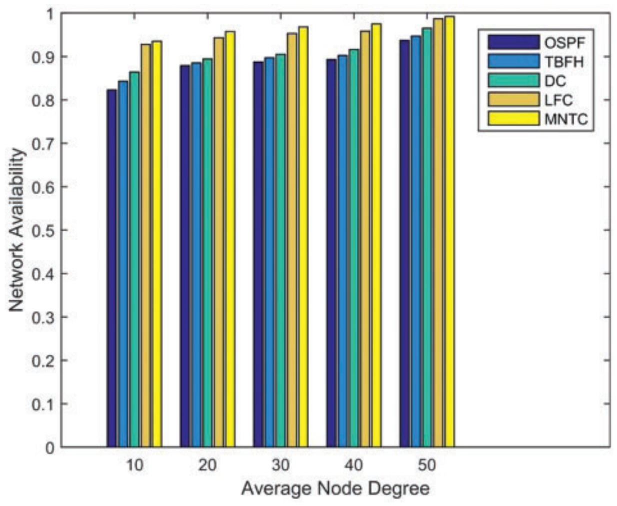

Figure 3: Topology size = 1000

In our evaluation, MNTC demonstrated superior performance compared to the other three algorithms. This is particularly evident in scenarios where the average node degree is high, as MNTC's routing strategies effectively leverage the increased connectivity to maintain high network availability. The ability of MNTC to adapt to changing network conditions and optimize routing decisions based on real-time network information enables it to outperform other algorithms in terms of both network availability and overall performance.

Our findings indicate that an increase in the average node degree of a topology can lead to improved network availability. MNTC's superior performance across different node degrees further underscores its adaptability and effectiveness in maintaining high network availability in diverse network environments.

5.2 *Path Stretch*

Path stretch is the ratio of the path cost of executing an algorithm to that of the default OSPF algorithm. The path cost is the sum of the link cost of all links in the path. In this paper, we use latency as the cost of a link, which is a known parameter of the experimental topology. As can be seen from Figs. 4 and 5, the path stretch of MNTC is basically the same as that of TBFH, DC and LFC. Therefore, excessive path overhead are not introduced. The findings indicate that the path stretch of MNTC is approximately equivalent to those of TBFH, DC and LFC. In other words, the path costs introduced by MNTC do not exceed those of the other two algorithms significantly. This equivalence suggests that the implementation of MNTC does not introduce excessive additional path overhead compared to the other methods. Such a result is favorable because it implies that MNTC achieves comparable routing efficiency without compromising the speed or directness of the network paths used for data transmission.

5.3 *Partial Deployment*

Partial deployment assesses the algorithm's performance when only a subset of nodes is deployed.The MNTC algorithm exhibits a notable characteristic in its compatibility with existing link-state routing protocols, including OSPF and IS-IS. This compatibility allows for the flexible deployment of MNTC within a network, enabling it to be implemented on select nodes rather than requiring a full-scale overhaul. Such partial deployment can be strategically planned to enhance network resilience without necessitating an extensive upgrade of the entire routing infrastructure.

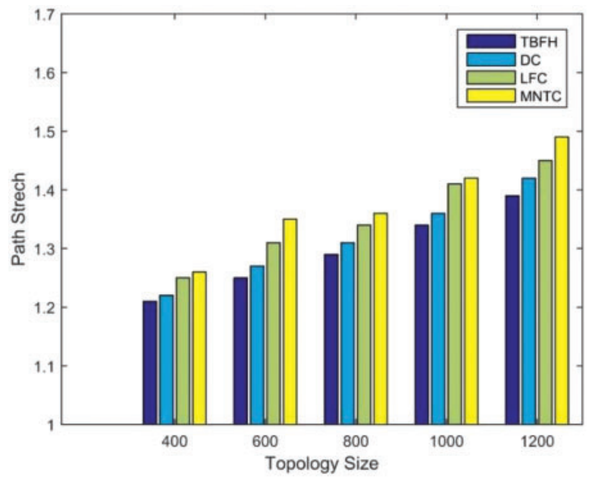

Figure 4: Average node degree = 4

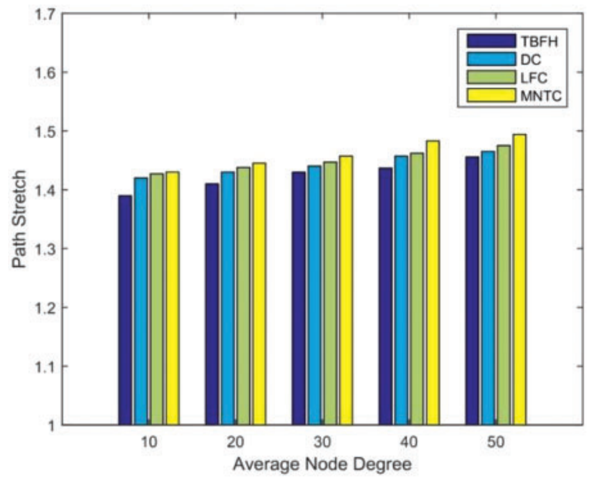

Figure 5: Topology size = 1000

To further investigate the effects of this partial deployment, empirical studies have been conducted, as depicted in Figs. 6–8. These figures provide insights into how varying degrees of node deployment affect both the real and inferred topologies that the routing algorithms utilize. The results indicate a positive correlation between the proportion of nodes employing the MNTC algorithm and the network availability.

Remarkably, even with only 60% of the nodes adopting the MNTC algorithm, there is a substantial improvement in network availability. Moreover, when comparing the performance of different algorithms under similar conditions, the results highlight the clear advantages of the Genetic Algorithm. This implies that the GA-based approach to routing protection not only maintains network stability but also outperforms other methods in enhancing network availability when deployed on a subset of network nodes.

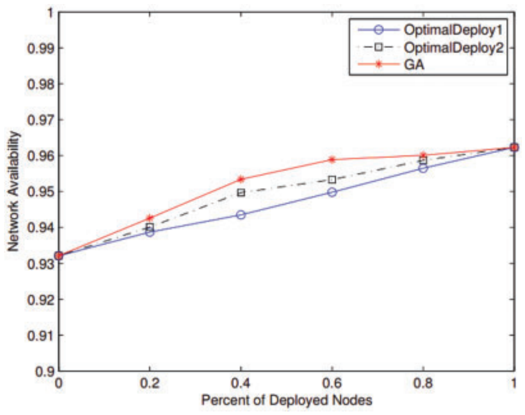

Figure 6: Abilene

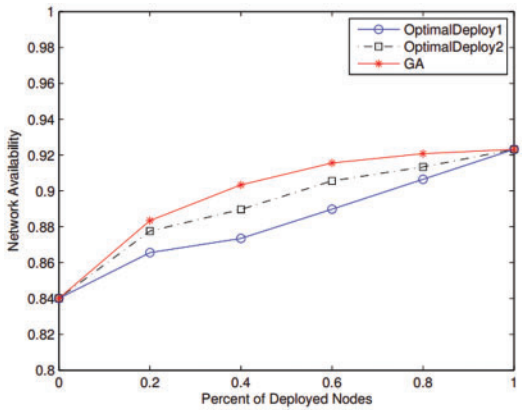

Figure 7: Exodus

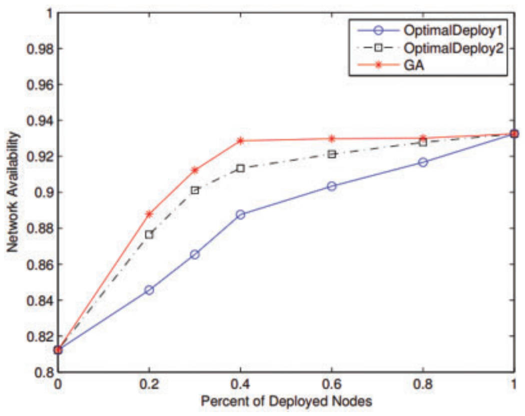

Figure 8: Sprint

6 Discussions

As thoroughly analyzed in the paper, the implementation of the MNTC routing scheme is a pivotal step in enhancing the overall resilience of the network. This advanced approach effectively amplifies network availability by affording each node a minimum of two routing alternatives. This not only alleviates the dependency on a single path but also establishes redundant paths that can be seamlessly activated in the event of failures, thereby greatly bolstering the network's resilience.

To ensure the optimal placement of nodes, we have developed three sophisticated incremental deployment algorithms. These algorithms meticulously assess and select the most suitable set of nodes for deployment, taking into account intricate factors such as node's contribution to network availability, and the complex intricacies of the network topology. By deploying MNTC on a meticulously chosen set of nodes, we can significantly bolster network reliability without the need for a sweeping overhaul of the existing infrastructure.

Rigorous evaluations of MNTC across a diverse array of topological spaces have unequivocally demonstrated its superiority in maintaining exceptional network availability with minimal path dilation. The results are nothing short of impressive, revealing that even when implemented on a mere 60% of nodes, our incremental deployment strategy significantly elevates network availability.

When compared to routing protocols like OSPF, TBFH, DC, and LFC, MNTC exhibits remarkable improvements in network availability. In five representative real-world network topologies, the average network availability of MNTC is augmented by 14.68%, 6.28%, 4.76%, and 2.84%, respectively. This underscores the profound potential of MNTC in fortifying network resilience and performance, making it a cutting-edge solution for modern networks grappling with escalating demands and complexities.

In conclusion, MNTC represents a paradigm shift in network resilience, offering an unparalleled level of robustness and effectiveness that surpasses traditional routing protocols. Its implementation promises to revolutionize network design and operations, ensuring seamless connectivity and unparalleled performance even in the face of failures or adverse conditions.

7 Conclusions

We have introduced a novel routing protection algorithm, MNTC, which is designed to maximize network reliability by leveraging the traditional hop-by-hop forwarding approach. This algorithm offers a significant enhancement to network availability, ensuring smoother and more efficient data transmission. The evaluation results obtained from our extensive testing demonstrate that MNTC exhibits promising performance in enhancing network availability. Its unique approach of providing multiple routing choices to each node greatly reduces the dependency on a single path, thereby enhancing the overall resilience of the network. Furthermore, to ensure optimal placement of nodes, we have proposed three incremental deployment algorithms. These algorithms carefully analyze the network topology and traffic patterns to select the most appropriate set of nodes for deployment. Notably, even when implemented on just 60% of the nodes, our evaluation results show that the network availability is significantly improved. We firmly believe that MNTC, coupled with our incremental deployment algorithms, has the potential to significantly boost network efficiency and reliability. This, in turn, will lead to more robust and dependable network performance, ensuring seamless data transmission and enhanced user experience.

Acknowledgement: The authors would like to acknowledge this work to the Computer Application Teaching and Research Section, China University of Labor Relations, Beijing and the School of

Automation and Software Engineering, Shanxi University, Taiyuan, for providing us with the platform to conduct this valuable research.

Funding Statement: This work is supported by Fundamental Research Program of Shanxi Province (No. 20210302123444), the Research Project at the College Level of China Institute of Labor Relations (No. 23XYJS018), the ICH Digitalization and Multi-Source Information Fusion Fujian Provincial University Engineering Research Center 2022 Open Fund Project (G3-KF2207), the China University Industry University Research Innovation Fund (No. 2021FNA02009), and the Key R&D Program (International Science and Technology Cooperation Project) of Shanxi Province China (No. 201903D421003).

Author Contributions: The authors confirm contribution to the paper as follows: Study conception and design: Wei Zhang, Haijun Geng; data collection: Wei Zhang; analysis and interpretation of results: Haijun Geng; draft manuscript preparation: Wei Zhang, Haijun Geng. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: All data and models are available as per request to the corresponding author.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] M. Chiesa, A. Kamisiński, J. Rak, G. Rétvári, and S. Schmid, "A Survey of fast-recovery mechanisms in packet-switched networks," *IEEE Commun. Surv. Tutorials*, vol. 23, no. 2, pp. 1253–1301, 2021. doi: 10.1109/COMST.2021.3063980.

[2] N. Khan, R. Salleh, A. Koubaa, Z. Khan, M. K. Khan and I. Ali, "Data plane failure and its recovery techniques in SDN: A systematic literature review," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 35, no. 3, pp. 176–201, 2023. doi: 10.1016/j.jksuci.2023.02.001.

[3] A. Anbiah and K. M. Sivalingam, "Efficient failure recovery techniques for segment-routed networks," *Comput. Commun.*, vol. 182, pp. 1–12, 2022. doi: 10.1016/j.comcom.2021.10.033.

[4] J. H. Li, X. G. Qi, W. C. Ma, and L. F. Liu, "Path selection for link failure protection in hybrid SDNs," *Future Gener. Comput. Syst.*, vol. 137, no. 1, pp. 201–215, 2022. doi: 10.1016/j.future.2022.07.016.

[5] Y. Wang, S. X. Feng, H. T. Guo, X. S. Qiu, and H. B. An, "A single-link failure recovery approach based on resource sharing and performance prediction in SDN," *IEEE Access*, vol. 7, pp. 174750–174763, 2019. doi: 10.1109/ACCESS.2019.2957141.

[6] H. J. Geng, H. Zhang, X. G. Shi, Z. L. Wang, and X. Yin, "Efficient computation of loop-free alternates," *J. Netw. Comput. Appl.*, vol. 151, no. 5, pp. 1–12, 2020. doi: 10.1016/j.jnca.2019.102501.

[7] H. J. Geng, H. Zhang, and Y. Y. Zhang, "Efficient routing protection algorithm in large-scale networks," *Comput. Mater. Contin.*, vol. 66, no. 2, pp. 1733–1744, 2022. doi: 10.32604/cmc.2020.013355.

[8] F. L. Verdi and G. V. Luz, "InFaRR: In-network fast rerouting," *IEEE Trans. Netw. Serv. Manag.*, vol. 20, no. 3, pp. 2319–2330, 2023. doi: 10.1109/TNSM.2023.3283459.

[9] K. T. Foerster, A. Kamisinski, Y. A. Pignolet, S. Schmid, and G. Tredan, "Improved fast rerouting using postprocessing," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 1, pp. 537–550, 2022. doi: 10.1109/TDSC.2020.2998019.

[10] G. Enyedi, G. Rétvári, P. Szilágyi, and A. Császár, "IP fast reroute: Lightweight Not-via without additional addresses," in *Proc. INFOCOM*, Rio de Janeiro, Brazil, 2009, pp. 2771–2775.

[11] J. Q. Zheng, H. Xu, X. J. Zhu, G. H. Chen, and Y. H. Geng, "We've got you covered: Failure recovery with backup tunnels in traffic engineering," in *Proc. ICNP*, Singapore, 2016, pp. 1–10.

[12] I. V. Duijn *et al.*, "Automata-theoretic approach to verification of MPLS networks under link failures," *IEEE/ACM Trans. Netw.*, vol. 30, no. 2, pp. 766–781, 2022. doi: 10.1109/TNET.2021.3126572.

[13] A. Sridharan, R. Guerin, and C. Diot, "Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks," *IEEE/ACM Trans. Netw.*, vol. 13, no. 2, pp. 234–247, 2005. doi: 10.1109/TNET.2005.845549.

[14] X. W. Yang and W. David, """Source selectable path diversity via routing deflections," in *Proc. ACM SIGCOMM*, Pisa, Italy, 2006, pp. 159–170.

[15] A. Kvalbein, A. F. Hansen, T. Čičić, S. Gjessing, and O. Lysne, "Fast IP network recovery using multiple routing configurations," in *Proc. INFOCOM*, Barcelona, Catalunya, Spain, 2006, pp. 1–11.

[16] K. Lakshminarayanan *et al.*, "Achieving convergence-free routing using failure-carrying packets," in *Proc. SIGCOMM*, Kyoto, Japan, 2007, pp. 241–252.

[17] P. Mérindol, P. Francois, O. Bonaventure, S. Cateloin, and J. J. Pansiot, "An efficient algorithm to enable path diversity in link staterouting networks," *Comput. Netw.*, vol. 55, no. 5, pp. 1132–1149, 2011. doi: 10.1016/j.comnet.2010.11.005.

[18] G. Rétvári, L. Csikor, J. Tapolcai, G. Enyedi, and A. Császár, "Optimizing IGP link costs for improving IP-level resilience," in *Proc. DRCN*, Krakow, Poland, 2011, pp. 62–69.

[19] G. Rétvári, J. Tapolcai, G. Enyedi, and A. Császár, "Ip fast reroute: Loop free alternates revisited," in *Proc. INFOCOM*, Shanghai, China, 2011, pp. 2948–2956.

[20] R. Terruggia, "Reliability analysis of probabilistic networks," PhD thesis, Universita degli Studi di Torino, Italy, 2010.

[21] H. J. Geng, X. G. Shi, Z. L. Wang, and X. Yin, "A hop-by-hop dynamic distributed multipath routing mechanism for link state network," *Comput. Commun.*, vol. 116, no. 4, pp. 225–239, 2018. doi: 10.1016/j.comcom.2017.12.008.

[22] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP topologies with rocketfuel," *IEEE/ACM Trans. Netw.*, vol. 12, no. 1, pp. 2–16, 2004. doi: 10.1109/TNET.2003.822655.