applied sciences

MDPI

Article

DDS: A Delay-Based Differentiated Service Virtual Network Embedding Algorithm

Jiamin Tian [1,2], Xuewen Zeng [1,2] and Xiaodong Zhu [1,*]

[1] National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, No. 21, North Fourth Ring Road, Haidian District, Beijing 100190, China
[2] School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, No. 19(A), Yuquan Road, Shijingshan District, Beijing 100049, China
* Correspondence: zhuxd@dsp.ac.cn

Abstract: Network virtualization (NV) is considered a promising technology that may solve the problem of Internet rigidity. The resource competition of multiple virtual networks for shared substrate network resources is a challenging problem in NV called virtual network embedding (VNE). Existing approaches do not consider the differences between multi-tenant requests and adopt a single embedding method, resulting in poor performance. This paper proposes a virtual network embedding algorithm that distinguishes the network types requested by tenants. This method divides virtual network requests into ordinary requests and delay-sensitive requests according to the delay constraints, provides personalized mapping strategies for different networks, and flexibly responds to the resource requirements and quality of service (QoS) requirements of the virtual network. The simulation results show that, compared with other algorithms, the proposed algorithm improves the request acceptance ratio by about 2% to 15% and the substrate network resources are more effectively utilized.

Keywords: network virtualization; virtual network embedding; NP-hard; heuristic algorithm; integer linear programming

Citation: Tian, J.; Zeng, X.; Zhu, X. DDS: A Delay-Based Differentiated Service Virtual Network Embedding Algorithm. *Appl. Sci.* **2022**, *12*, 9897. https://doi.org/10.3390/app12199897

Academic Editor: Alexandros-Apostolos Boulogeorgos

Received: 25 August 2022
Accepted: 26 September 2022
Published: 1 October 2022

1. Introduction

With the rise of various new network services and applications, the demands of users for the network show a diversified trend. However, deploying new businesses under the traditional network architecture requires not only redeployment of hardware but also consensus among device providers, which is difficult to cope given the diverse, customized, and differentiated applications. The proposal of network virtualization is to overcome the impasse of the Internet [1]. As one of the important technologies for promoting current network innovation [2], network virtualization has been actively applied in many research testbeds and projects [3], such as CABO [4], 4WARD [5], and G-Lab [6]. Through network virtualization, multiple isolated networks can be virtualized in the shared underlying network infrastructure. Each virtual network can customize the network structure or run different network protocols according to business requirements. Different users can use their network resources independently without interfering with each other, thereby improving the utilization of network resources, satisfying the diverse demands of users, and realizing an elastic network.

In network virtualization, the Internet service providers (ISPs) in traditional networks are divided into infrastructure providers (InPs) and service providers (SPs). This decoupling method that separates resources and services is more flexible. InPs are responsible for providing underlying infrastructure and managing underlying network resources. SPs rent physical resources from InPs, build and operate virtual networks, and offer services through virtual networks. Embedding the requests of SPs onto underlying infrastructure depends on efficient embedding algorithms.

The software defined network (SDN) [7] proposed by Professor Nick McKeown of Stanford University in the United States realizes the separation of forwarding and control. The centralized control plane can flexibly program the network and easily obtain a global view, which provides a natural advantage for the realization of network virtualization. Facing the diversified and personalized network requirements, Hu [8] proposed a full-dimensional defined polymorphic smart network architecture. In the polymorphic network environment, the resource requirements of diversified services are different. Specifically, in the substrate network, this is manifested as competition for the same resources between different networks—in other words, how to effectively allocate the substrate network resources to virtual network requests and satisfy the constraints of nodes and links at the same time. This problem is called virtual network embedding (VNE) and is one of the key problems of network virtualization. The definition of node constraint is that the resource requirements of the virtual node must not exceed the physical resources of the mapped entity node. The definition of link constraint is that the resource requirements of the virtual link must not exceed the physical resources of the mapped entity link.

VNE aims to provide an optimal resource allocation scheme for each virtual network, which proves to be an NP-hard problem [9]. In the existing research, the solutions to VNE problems can be divided into three categories [10], namely exact solutions, heuristic solutions, and meta-heuristic solutions. The exact solution method is used in [11–14] to solve the VNE problem. The exact solution uses mixed integer programming (MIP) or integer linear programming (ILP) to establish a mathematical model, which is commonly solved by solution software such as GLPK [15] and CPLEX [16]. However, the computational complexity of the exact solution increases exponentially with the expansion of the network scale. In contrast, heuristic solutions can solve the virtual network embedding scheme in a shorter time, but the mapping results are approximate optimal solutions. Some typical studies [17–22] focus on designing different heuristic algorithms to obtain better embedding solutions. Some meta-heuristic solutions, like simulated annealing [23], particle swarm optimization [24], ant colony optimization [25], or genetic algorithm [26], are also used in the field of VNE.

However, in real network scenarios, different tenants have different network requirements. For example, a virtual network that provides VoIP services has high CPU, medium bandwidth, and low link propagation delay requirements, and a virtual network that provide P2P services has medium CPU, medium bandwidth, and no propagation delay constraints [3]. Considering that different virtual network requests have different resource requirements and QoS requirements, a single virtual network embedding method cannot effectively serve differentiated requests.

To solve the inefficient utilization of underlying network resources caused by the single virtual network embedding method, this paper proposes an algorithm that distinguishes network types and adopts different mapping strategies to efficiently map differentiated virtual network requests. The advantage of this method is that it provides a personalized mapping scheme for virtual network requests by classifying network types and flexibly responds to the resource requirements and QoS requirements of the virtual network. Experimental results show that our method can accept more virtual network requests and improve the utilization of underlying network resources.

The main contributions of this paper are as follows:

1. For the sake of efficiently mapping differentiated virtual network requests, maximizing the utilization of substrate network resources, and improving the success ratio of virtual network embedding, we classify virtual network requests. The classification standard of virtual network is whether the virtual network has link delay constraints, which is divided into ordinary requests and delay-sensitive requests.
2. A two-stage embedding sub-algorithm is used for ordinary virtual network requests. In the node mapping stage, the resource-rich nodes are selected based on the greedy strategy, and the adjacent link resources are considered in the resource measurement to improve the success ratio of the next stage. Moreover, the consumption balance

between different resources of nodes is considered to avoid the early exhaustion of a single resource affecting the embedding of subsequent virtual requests. Then the k-shortest path algorithm [27] is used for link mapping.

3. A constrained integer linear programming sub-algorithm is used for delay-sensitive virtual network requests. Inspired by Ref. [14], we construct candidate node set and candidate link set to reduce the solution scale of linear programming, aiming to find the optimal embedding solution with delay constraints in polynomial time.

The rest of this paper is organized as follows. The related work is introduced in Section 2. In Section 3, we introduce the problem and model of virtual network embedding. In Section 4, we describe the proposed VNE algorithm in detail. The Section 5 shows the evaluation results. Finally, Section 6 concludes this paper.

2. Related Work

In order to solve the NP-hard problem of VNE, scholars have proposed many heuristic-based algorithms. According to the relationship between node mapping and link mapping, these can be divided into one-stage embedding algorithms and two-stage embedding algorithms. The one-stage algorithm embeds virtual nodes and virtual links at the same time. Ref. [17] proposed a subgraph isomorphism detection algorithm which splits the VNR into subgraphs and embeds nodes and links at the same time, but this will lead to a lot of backtracking. Yu et al. [18] proposed a classic heuristic two-stage algorithm that has been used so far. Nodes are sorted according to the CPU and the total adjacent link bandwidth, and nodes in the virtual network are greedily mapped to the substrate nodes. Yu introduces the concept of path splitting. In the next stage, links are mapped by the k-shortest path algorithm [27] or the multi-community flow (MCF) algorithm [28]. This algorithm had a profound impact on the research of virtual network mapping algorithms. Ref. [29] effectively reduces the virtual network request rejection rate through path splitting. Ref. [19], considering node degree, and [20], considering time window, are variants of [18]. Inspired by the Google PageRank algorithm, Ref. [21] applies a Markov random walk model to rank nodes according to resources and topological attributes. The link mapping method is similar to [18]. However, these algorithms only consider the node capacity and link bandwidth and do not consider the complexity of network resources in the actual network, so they cannot be extended to practical network applications.

Chowdhury et al. [11,12] used MIP for mathematical modeling in the VNE problem for the first time, and relaxed integer constraints to obtain linear programming. They proposed two node-embedding algorithms: deterministic rounding and random rounding. When the solution of node embedding is feasible, the MCF or shortest path algorithm is used to embed virtual links. Melo [13] proposed an algorithm to solve VNE using ILP with the goal of minimizing resource consumption and load balancing, adding node distance and link propagation delay constraints, but it is not described in detail in the paper. As far as we know, pure ILP has high computational complexity. Cao [14] proposed an ILP algorithm based on candidate assistance, which reduces the computational complexity of linear programming by adding location and delay constraints to limit the range of solutions, enabling linear programming to be applied in medium-scaled networks. Ref. [22] proposed a three-dimensional virtual network embedding model based on computing, storage, and bandwidth resources. Compared with the previous VNE model that only considers node computing resources and link bandwidth resources, this model is closer to the real network attributes, such as SDN flow table resources and ICN node content cache, all need to occupy node storage resources. In addition, considering some services with QoS requirements, such as telemedicine, online game, Internet of Vehicles, industrial Internet remote control, and other delay-sensitive services, link delay should also be considered as a network attribute in the model.

Ref. [30] combined with the novel neural network of the graph convolution network, proposed a virtual network embedding algorithm based on deep reinforcement learning. Ref. [31] introduces simulated annealing into the particle swarm optimization algorithm

to solve the VNE problem and proposes a particle initialization allocation strategy for the shortcomings of the particle swarm optimization algorithm in initializing particles' positions. Ref. [32] formulated the VNE problem as multi-objective integer linear programming and designed an algorithm based on an artificial immune system to solve this problem. Compared with exact solutions and heuristic solutions, meta-heuristic solutions are not the mainstream of VNE research [3].

Generally speaking, the resource requirements and QoS requirements of the virtual network requests of different tenants are differentiated. However, the current VNE algorithm adopts the same mapping algorithm for the virtual network request types of different tenants. This single mapping method will lead to the waste of substrate resources and low mapping efficiency. Therefore, this paper comprehensively considers node computing, storage, link bandwidth, and delay constraints, and proposes a virtual network embedding algorithm for differentiated tenant requests.

3. Virtual Network Embedding Problem Formulation

3.1. Network Model

When modeling the VNE problem, the previous approaches either do not consider link propagation delay or only consider node CPU resources, which is insufficient to a certain extent. In order to be closer to the actual network environment, this paper comprehensively considers node computing resources, storage resources, link bandwidth resources, and link propagation delay constraints in order to model the substrate network and virtual network.

3.1.1. Substrate Network

The notations and meanings of the virtual network embedding problem are shown in Table 1. The substrate network is modeled as a weighted, undirected graph $G_s = (N_s, L_s, A_n^s, A_l^s)$ where N_s represents the sets of substrate nodes, L_s represents the sets of substrate links, and A_n^s represents the sets of substrate nodes attributes, which mainly refers to the computing resources (such as CPU resources) and storage resources (such as flow table resources) of substrate nodes. A_l^s represents the sets of substrate links attributes, which mainly refers to the bandwidth and delay of substrate links.

Table 1. Notations in VNE.

Notation	Description
G_s/G_v	substrate/virtual network
N_s/N_v	the set of substrate/virtual nodes
L_s/L_v	the set of substrate/virtual links
n_i^s	substrate node
n_j^v	virtual node
l_{mn}^s	substrate link from node m to node n
l_{pq}^v	virtual link from node p to node q
p_{mn}^s	substrate path from node m to node n
$R_c(n_i^s)$	available computing resource of substrate node n_i^s
$c(n_i^s)$	computing resource of substrate node n_i^s
$c(n_j^v)$	computing demand of virtual node n_j^v
$R_s(n_i^s)$	available storage resource of substrate node n_i^s
$s(n_i^s)$	storage resource of substrate node n_i^s
$s(n_j^v)$	storage demand of virtual node n_j^v
$R_b(p_{mn}^s)$	available bandwidth resource of substrate path p_{mn}^s
$R_b(l_{mn}^s)$	available bandwidth resource of substrate link l_{mn}^s
$b(l_{mn}^s)$	bandwidth resource of substrate link l_{mn}^s
$b(l_{pq}^v)$	bandwidth demand of virtual link l_{pq}^v
$delay(p_{mn}^s)$	path propagation delay of substrate path p_{mn}^s
$delay(l_{mn}^s)$	link propagation delay of substrate link l_{mn}^s
$delay(l_{pq}^v)$	link propagation delay constraint of virtual link l_{pq}^v

In addition, P_s represents the set of all loop-free paths in the substrate network. A path contains one or more substrate links. p^s_{mn} represents a path from substrate node n^s_m to substrate node n^s_n. P^s_{mn} represents the set of paths from substrate node n^s_m to substrate node n^s_n.

3.1.2. Virtual Network

The virtual network is also modeled as a weighted undirected graph $G_v = (N_v, L_v, C^v_n, C^v_l)$, where N_v represents the sets of virtual nodes, L_v represents the sets of virtual links, C^v_n represents the sets of virtual nodes constraints, which mainly refers to the computing and storage requirements of virtual nodes, and C^v_l represents the sets of virtual link constraints, which mainly refers to the bandwidth requirements of virtual links. The virtual network link constraints with delay requirements should also include delay constraints.

3.2. Mapping Model

Based on the above description, the virtual network embedding problem can be abstracted as the mapping operation M of embedding a virtual network G_v into an entity network G_s, which is expressed as follows:

$$M(G_v) : (N_v, L_v) \rightarrow (N'_s, L'_s),\ N'_s \in N_s, L'_s \in L_s, \tag{1}$$

Therefore, the VNE problem can be divided into two sub-problems: virtual node mapping (VNM) and virtual link mapping (VLM). VNM and VLM can be solved separately or together.

3.3. Measurements of Substrate Network Resources

The available computing resource of a substrate node $R_c(n^s)$ is defined as follows:

$$R_c(n^s) = c(n^s) - \sum_{n^v \in N} c(n^v), \tag{2}$$

The available storage resource of a substrate node $R_s(n^s)$ is defined as follows:

$$R_s(n^s) = s(n^s) - \sum_{n^v \in N} s(n^v), \tag{3}$$

where n^s denotes a substrate node and n^v denotes a virtual node; N represents all virtual nodes mapped to n^s.

The available bandwidth resource of a substrate link is defined as follows:

$$R_b(l^s) = b(l^s) - \sum_{l^v \in L} b(l^v), \tag{4}$$

where l^s denotes a substrate link and l^v denotes a virtual link; L represents all virtual links mapped to l^s. Note that a virtual link can be mapped to one or more substrate links, which depends on the hops between the substrate nodes mapped by both ends nodes of the virtual link. The available bandwidth resources of a substrate path are defined as follows:

$$R_b(p^s) = \min_{l^s \in p^s} R_b(l^s), \tag{5}$$

where p^s denotes a substrate path including the substrate link l^s.

3.4. Performance Metrics

There are many performance metrics to evaluate the virtual network embedding algorithm. Our goal is mainly to improve the success ratio of virtual network requests and reduce the cost of virtual network embedding. The following is a detailed introduction to several commonly used metrics in research.

3.4.1. VNR Acceptance Ratio

The VNR acceptance ratio is defined in Equation (6). It is determined by the number of successfully embedded virtual networks and the number of total virtual network requests, which can directly reflect the ability of the virtual network embedding algorithm, and it is an important metric to evaluate the virtual network embedding algorithm.

$$R_{\text{accept}} = \frac{N_{\text{success}}}{N_{\text{total}}}, \tag{6}$$

where N_{success} represents the number of successfully embedded virtual network requests, and N_{total} represents the number of total virtual network requests.

3.4.2. Revenue–Cost Ratio

The revenue of a virtual network is defined as:

$$R(G_v) = \sum_{n_j^v \in N_v} (\alpha \cdot c(n_j^v) + \beta \cdot s(n_j^v)) + \lambda \cdot \sum_{l_{pq}^v \in L_v} b(l_{pq}^v), \tag{7}$$

Equation (7) indicates that the revenue of VNR is the sum of the computing and storage requirements of all virtual nodes of the VNR and the bandwidth requirements of the virtual link. Weight factor (α, β, λ) is used to balance different types of network resources.

The cost of a virtual network is defined as:

$$C(G_v) = \sum_{n_j^v \in N_v} (\alpha \cdot c(n_j^v) + \beta \cdot s(n_j^v)) + \lambda \cdot \sum_{l_{pq}^v \in L_v} \sum_{l_{mn}^s \in L_s} y_{mn}^{pq} \cdot b(l_{pq}^v), \tag{8}$$

Equation (8) indicates that the cost of VNR is all nodes and link resources occupied by the VNR, which is 1 when the substrate link l_{mn}^s allocates resources to the virtual link l_{pq}^v and 0 in other cases.

The revenue–cost ratio provides a metric of revenue and cost which is used to evaluate the utilization efficiency of the substrate network resources, and its value is less than or equal to 1. If each virtual network embedding algorithm can successfully map a virtual network to the substrate network, then the revenue of the virtual network is the same, and the algorithm with a larger revenue–cost ratio indicates that the substrate network resources are consumed less. Equation (9) gives the definition of revenue–cost ratio.

$$R/C(G_v) = \frac{R(G_v)}{C(G_v)}, \tag{9}$$

3.4.3. Resource Utilization

In order to qualify the resource consumption of a node or a link in the substrate network, the resource utilization is defined as follows:

$$U(R_s) = \frac{\text{occupied}(R_s)}{\text{total}(R_s)}, \tag{10}$$

where R_s could be the computing or storage resource of the substrate node or the bandwidth resource of substrate link, $\text{occupied}(R_s)$ represents the consumed resources, and $\text{total}(R_s)$ represents the total amount of resources.

4. Proposed Algorithm

4.1. Virtual Network Request Classification

In order to flexibly respond to virtual network requests with differentiated resource requirements and QoS requirements, we propose a virtual network embedding algorithm that distinguishes network types. The algorithm flow chart is shown in Figure 1. When a virtual network request arrives, the virtual network type is first classified. According to

whether the virtual network has delay constraints, virtual network requests are divided into ordinary requests and delay-sensitive requests, and the sub-algorithms mentioned in Sections 4.2 and 4.3 are used to solve them, respectively. Before embedding the virtual network, it is necessary to judge whether the substrate network resources satisfy the demands and constraints of the virtual network. After the embedding is successful, update the substrate network, wait for the arrival of next virtual network request, and start timing the virtual network lifetime. When the virtual network request leaves, release the occupied substrate network resources and update the substrate network.

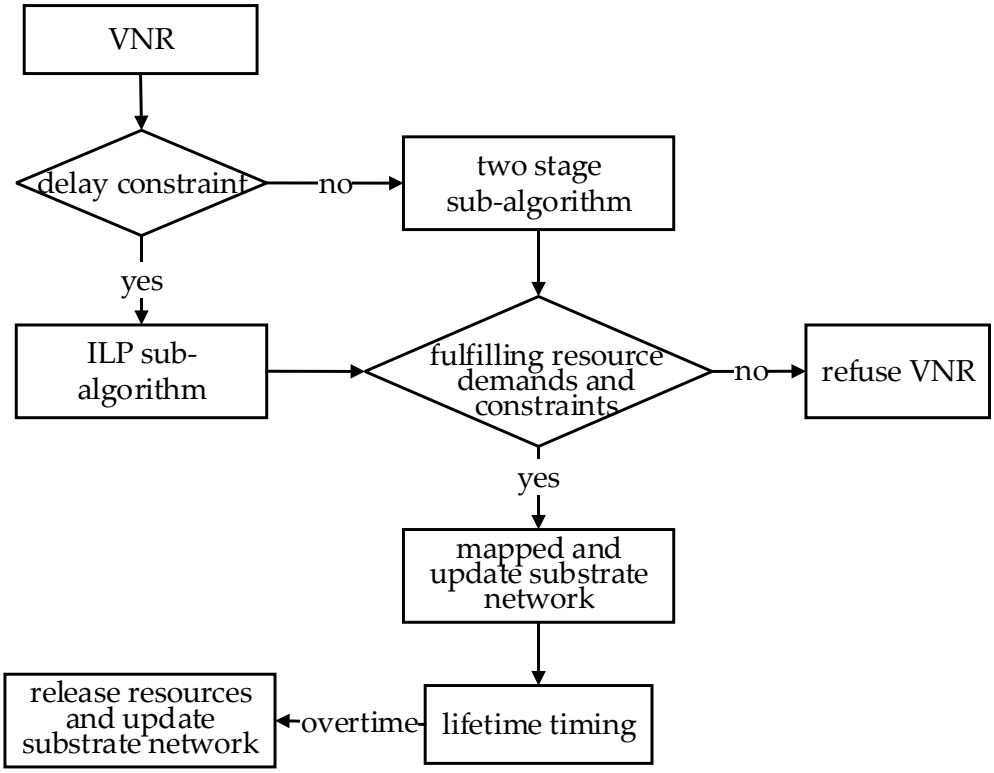

Figure 1. Flow Chart of VNE Algorithm.

4.2. Two Stage Embedding Sub-Algorithm

In this section, we propose a node mapping algorithm based on greedy strategy, which considers both node resource value and node resource balance. For ease of understanding, Algorithms 1 and 2 give the pseudocode of the algorithms. We measure nodes value and sort the virtual nodes and substrate nodes in non-increasing order. Simply considering node resources is not conducive to the success of link mapping in the next stage, and adjacent link resources should be additionally considered when measuring the node resources' value, which helps to improve the success rate of link mapping. When sorting nodes, we comprehensively consider node computing resources, node storage resources, and adjacent link resources and expand them based on the article [18], defining the resource value of nodes as follows:

$$NRV(n_i^s) = (R_c(n_i^s) + R_s(n_i^s)) \times \sum_{l_s \in L_s(n_i^s)} R_b(l_s), \tag{11}$$

where $L_s(n_i^s)$ represents the set of all substrate links set adjacent to node n_i^s.

Algorithm 1 Node Mapping Algorithm.

Input: N_s, N_v
Output: the node mapping results for N_v
1: **for all** virtual nodes $n_j^v \in N_v$ **do**
2: $candidates(n_j^v) \leftarrow$ substrate nodes fulfilling computing demand $c(n_j^v)$ and store demand $s(n_j^v)$
3: **for all** substrate nodes $n_i^s \in candidates$ **do**
4: calculate *nodeVal* by Algorithm 2
5: sort *nodeVal*
6: **end for**
7: $chosen(n_j^v) \leftarrow$ the substrate node with maximum *nodeVal* in *candidates*
8: **end for**
9: perform node mapping

Algorithm 2 Calculate Node Value Algorithm.

Input: N_s, $candidates(n_j^v)$, n_j^v
Output: *nodeVal*
1: **for all** substrate nodes $n_i^s \in candidates$ **do**
2: calculate $NRV(n_i^s)$ (defined in Equation (11))
3: calculate RB_c (defined in Equation (12))
4: calculate RB_s (defined in Equation (13))
5: $bal(n_i^s) \leftarrow \max(RB_c/RB_s, RB_s/RB_c)$
6: $nodeVal \leftarrow NRV(n_i^s)/bal(n_i^s)$
7: **end for**

In addition, considering that the virtual node has different node resource requirements, it may lead to the excessive use of a certain resource. Due to the Cask Effect, it is difficult to satisfy the demands of other nodes which affects the success rate of embedding. In order to solve this problem, the load consumption balance of the two resources should also be considered. Define the node computing resource demand balance as follows:

$$RB_c = R_c(n_i^s)/c(n_j^v), \tag{12}$$

Similarly, the node storage resource demand balance is:

$$RB_s = R_s(n_i^s)/s(n_j^v), \tag{13}$$

The resource demand balance reflects the relationship between the remaining resources of the substrate node and the virtual node demand. The substrate node with the closest RB_c and RB_s should be selected first to avoid the imbalance of resource consumption and help to accommodate more virtual network requests. Figure 2 gives a typical example. The numbers in the circles represent the available computing and storage resources, the numbers over the links represent the available bandwidth, and the numbers in the hexagon represent the computing and storage requirements. When selecting the substrate node for virtual node a, the NRV of A is $(20+40)\times(10+20+10) = 2400$, and the NRV of B is $(30+30)\times(10+20+10) = 2400$. Obviously, if A is selected for embedding, A can no longer provide services for subsequent virtual networks due to the exhaustion of computing resources, even though the available storage resource of A is still abundant. After combining the balance factor, the *nodeVal* of A is $2400 \div (2 \div 1) = 1200$, and the *nodeVal* of B is $2400 \div (1.5 \div 1.5) = 2400$. B is selected for embedding. In this case, both A and B can continue to provide services for the subsequent virtual networks.

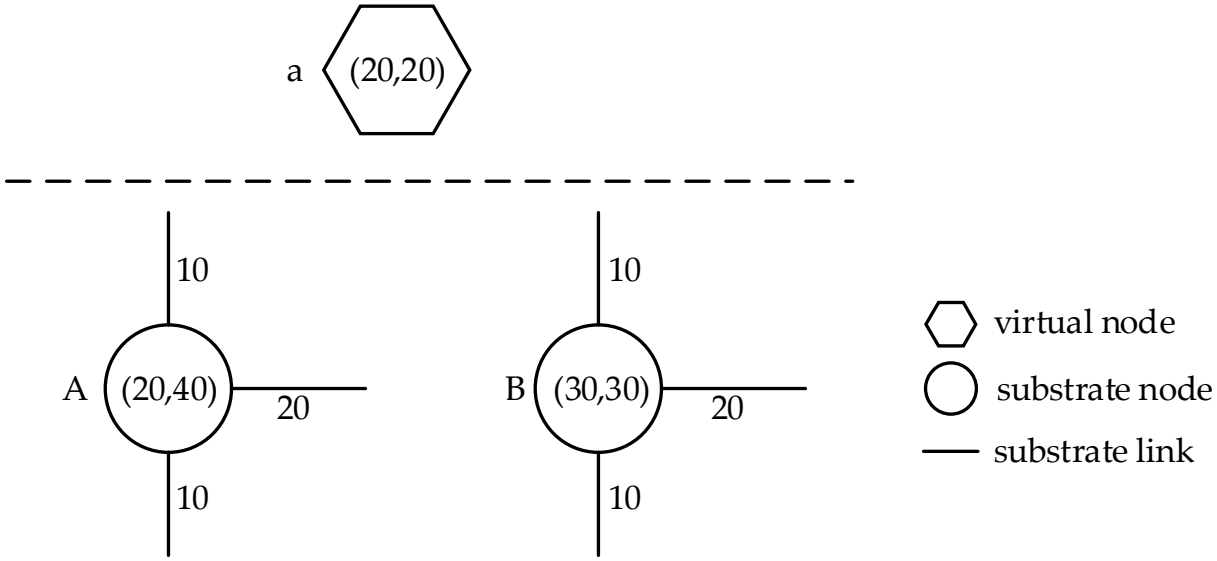

Figure 2. An example of virtual node mapping.

In link mapping, some algorithms assume support for path splitting, which means that when a single substrate path cannot satisfy the virtual links' requirements, a virtual link is allowed to be allocated to multiple substrate paths. Although path splitting can improve the utilization of the substrate links, it will cause some additional burdens, such as the switch needing to reorganize packets, the additional delay caused by multiple links, and the additional flow table resource used to save flow rules in SDN. Therefore, path splitting is not considered in this paper. When the node mapping is over, the virtual link mapping is transformed into a resource-constrained shortest path solution problem between fixed nodes. Algorithm 3 describes our link mapping algorithm.

Algorithm 3 Link Mapping Algorithm.

Input: N_s, N_v, k, node mapping results for N_v
Output: the link mapping results for N_v
1: **for all** virtual links $l_{pq}^v \in L_v$ **do**
2: find two substrate nodes n_{src}^s and n_{dst}^s corresponding to link l_{pq}^v from N_v
3: search k-shortest paths from n_{src}^s to n_{dst}^s with increasing k
4: **if** exist *path* satisfying $R_b(path) \geq b(l_{pq}^v)$ **then**
5: perform link mapping
6: **else**
7: reject N_v
8: **end if**
9: **end for**

4.3. ILP Sub-Algorithm

This section introduces a constrained integer linear programming algorithm that we use to solve delay-sensitive virtual network requests. The algorithm first calculates the candidate node set and the candidate link set to reduce the scale of the ILP solution and then calculates the optimal solution of the embedding under the objective function. Algorithm 4 describes the selection process of candidate sets. D in Line 2 represents the location constraints of the virtual network. We assume that the position of the node is defined by coordinates (x, y). Equation (14) defines the distance between nodes.

$$\text{dist}(n_j^v, n_i^s) = \sqrt{(x_j^v - x_i^s)^2 + (y_j^v - y_i^s)^2}, \tag{14}$$

Algorithm 4 Candidate Sets Construction Algorithm.

Input: N_s, N_v
Output: candidate node set $canN$, candidate link set $canL$

1. **for all** virtual nodes $n_j^v \in N_v$ **do**
2. **if** dist$\left(n_j^v, n_i^s\right) \leq$ D && $c(n_j^v) \leq R_c(n_i^s)$ &&$s(n_j^v) \leq R_s(n_i^s)$
3. $can_{n_j^v}.add(n_i^s)$
4. **end if**
5. $canN.add(can_{n_j^v})$
6. **end for**
7. **for all** virtual links $l_{pq}^v \in L_v$ do
8. select candidate substrate links of l_{pq}^v which end points are in the corresponding candidate node set and fulfilling the bandwidth and delay constraints of l_{pq}^v, add to $canL$.
9. **end for**

The candidate node set should be selected within the distance constraints and satisfy the computing and storage requirements of virtual nodes. When constructing the candidate link set, for each virtual link, select the substrate links whose bandwidth and delay between the substrate candidate nodes corresponding to both ends nodes of the link satisfy the constraints and add them to candidate link set, to reduce the calculation scale. The next integer linear programming algorithm seeks the best embedding scheme from the candidate sets. The ILP model is described as follows:

1. Variables

$$x_i^j = \begin{cases} 1, \text{ if } n_j^v \text{ is mapped into } n_i^s \\ 0, \text{ else} \end{cases}, \tag{15}$$

$$y_{mn}^{pq} = \begin{cases} 1, \text{ if } l_{pq}^v \text{ is mapped into } l_{mn}^s \\ 0, \text{ else} \end{cases}, \tag{16}$$

2. Restrictions

(1) virtual node constraints

Equation (17) is designed to ensure that each virtual node maps to only one substrate node.

$$\forall n_j^v \in N_v, \sum_{n_i^s \in N_s} x_i^j = 1, \tag{17}$$

(2) substrate node constraints

Equation (18) aims to ensure that each substrate node can only accept the mapping of one virtual node from the same virtual network.

$$\forall n_i^s \in N_s, \sum_{n_j^v \in N_v} x_i^j \leq 1, \tag{18}$$

(3) virtual link constraints

Equation (19) is meant to ensure that each virtual link is allocated to one substrate path, and that the path endpoint corresponds to the substrate node selected by the virtual link endpoint.

$$\forall l_{pq}^v \in L_V, \forall n_m^s \in N_s, \sum_{n_n^s \in N_s} y_{mn}^{pq} - y_{nm}^{pq} = x_m^p - x_n^p, \tag{19}$$

(4) Node resource constraints

The chosen substrate node must satisfy the computing and storage requirements of the virtual node, as defined in Equations (20) and (21).

$$\forall n_i^s \in N_s, \sum_{n_j^s \in N_v} x_i^j \cdot c(n_j^v) \leq R_c(n_i^s), \tag{20}$$

$$\forall n_i^s \in N_s, \sum_{n_j^s \in N_v} x_i^j \cdot s(n_j^v) \leq R_s(n_i^s), \tag{21}$$

(5) link bandwidth constraints

The chosen substrate link must be able to provide sufficient bandwidth to satisfy the bandwidth requirements of the virtual link, as defined in Equation (22).

$$\forall l_{mn}^s \in L_v, \sum_{l_{pq}^v \in L_v} y_{mn}^{pq} \cdot b(l_{pq}^v) \leq R_b(l_{mn}^s), \tag{22}$$

(6) link propagation delay constraints

Equation (23) is designed to ensure that the delay of the chosen substrate link $delay(l_{mn}^s)$ does not exceed the propagation delay limit of the virtual link $delay(l_{pq}^v)$.

$$\forall l_{pq}^v \in L_v, \sum_{l_{mn}^s \in L_v} y_{mn}^{pq} \cdot delay(l_{mn}^s) \leq delay(l_{pq}^v), \tag{23}$$

3. objective

$$\min : \sum_{l_{pq}^v \in L_v} \sum_{l_{mn}^s \in L_s} (y_{mn}^{pq} \cdot b(l_{pq}^v) + y_{mn}^{pq} \cdot delay(l_{mn}^s)), \tag{24}$$

For a virtual network embedding request, the substrate node resources cost of different embedding strategies is the same, but due to different substrate paths chosen by different embedding strategies, the link resources cost is different. Therefore, the definition of the objective function only considers the link cost.

5. Performance Evaluation

In this section, we first describe the evaluation environment and then present our performance evaluation results. Our performance evaluation includes the VNR acceptance ratio, revenue–cost ratio, node resource utilization, and link resource utilization.

5.1. Evaluation Environment and Settings

To evaluate our proposed algorithm, we perform simulations in the following evaluation environment: Intel (R) core (TM) i7-10875H CPU, 32.0 GB RAM. The open-source tool GLPK [15] is used to solve integer linear programming.

We implement a discrete event simulator to simulate the arrival and departure of virtual networks. In the substrate network topology, the number of nodes is set to 40, and each pair of substrate nodes is randomly connected with a probability 0.5, which corresponds to this medium-scaled network. The setting is consistent with [14]. Nodes are randomly distributed within a grid of (20 × 20). The computing resources and storage resources of the substrate nodes are uniformly distributed between 50 and 100 units, the bandwidth resources of the substrate links are uniformly distributed between 50 and 100 units, and the propagation delay of the substrate links is set to be uniformly distributed between 1 and 3 units.

The arrival of virtual network requests follows Poisson distribution. In the experiment, the results of the arrival rate from 2 to 6 virtual network requests per 100 time units are evaluated. The virtual network survival time follows the exponential distribution, the average value is set to 1000 time units. The number of virtual nodes is uniformly distributed between 2 and 8, and the probability of connectivity between virtual nodes is 0.5. The requirements of computing resources and storage resources of virtual nodes are uniformly distributed between 1 and 20, and the bandwidth requirements of virtual links are uniformly distributed from 1 to 50. The propagation delay of virtual links for delay-sensitive requests is limited to 2 to 8 uniformly distributed. The value of D is set to 5. The ratio of ordinary virtual network requests to delay-sensitive virtual network requests

is 1:1. The weight factor α, β, λ is set to 1. In the experiment, the simulation time is set to 10,000 time units.

In our experiments, in order to avoid extreme situations such as a very high VNR acceptance ratio or very low VNR acceptance ratio, we conducted 10 experiments for each virtual network request arrival ratio, and each experiment generated a new set of VNRS and a new substrate network. We recorded the arithmetic average of the results of 10 operations as the final result.

In order to evaluate the algorithm performance, we compared the proposed algorithm with [14,18,21,22] in the simulation experiment. We marked them as CAN-ILP, GR-SP, RW-SP, and RCR-VNE, respectively, and kept the parameters consistent. The objective functions of two types of virtual network requests in [14] are CAN-CF and CAN-CLDF, respectively.

5.2. Evaluation Results

In this section, the simulation results and the analysis of the experimental results are given. The VNR acceptance ratio, average revenue, average cost, revenue–cost ratio, node computing resource utilization, node storage resource utilization, and link bandwidth resource utilization are evaluated and analyzed, respectively.

5.2.1. Virtual Network Request Acceptance Ratio

The virtual network request acceptance ratio is one of the most important metrics for evaluating the VNE algorithm. Figure 3 reflects the result that the virtual network request acceptance ratio of each algorithm changes with the virtual network request arrival ratio. As can be seen from the figure, as the number of VNRs per unit time increases, the request success ratio gradually decreases. This is consistent with our intuitive prediction because the substrate network resources are limited. With the increase in the number of virtual network requests, the acceptance ratio will inevitably decline. Obviously, our algorithm has a better acceptance ratio than other algorithms. When the virtual network request arrival ratio is 2 per 100 time units, our algorithm and that of [14] have acceptance ratios of 98.977 and 97.098%, respectively, while other algorithms are below 80%. Our algorithm shows the best acceptance ratio at any arrival ratio, with a maximum improvement of about 15% compared with other algorithms. Ref. [14] performs well when the arrival ratio of virtual network requests is low, but the acceptance ratio decreases as the arrival ratio increases. This is because the location constraints limit the range of solutions, which causes some virtual nodes to have no suitable candidate nodes to embed, thus reducing the virtual network requests' success ratio to a certain extent.

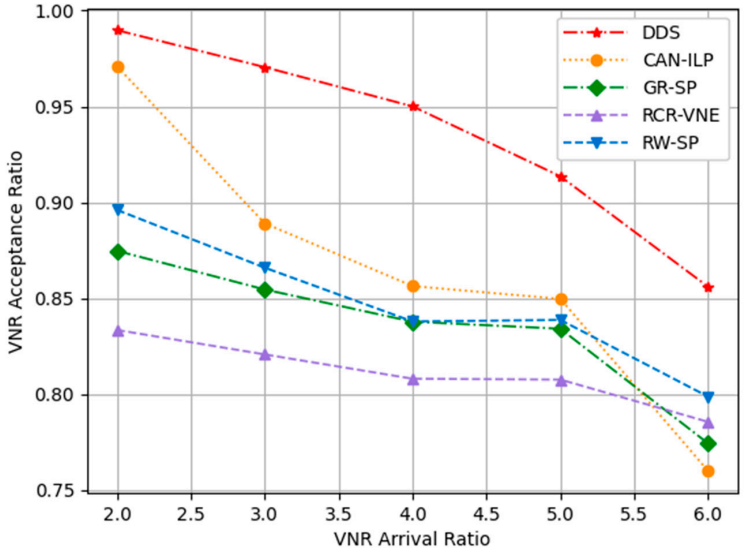

Figure 3. VNR acceptance ratio under various VNR arrival ratios.

5.2.2. Revenue and Cost

Figures 4–6 describe the average revenue, average cost, and revenue–cost ratio under different virtual network request arrival ratios. It can be seen that the algorithm of [14] has the minimum average cost and the maximum revenue–cost ratio, the revenue–cost ratio is 93.998% to 98.516%. This is because it adopts integer linear programming to solve based on minimizing cost, so it performs well in the average cost. Compared with other algorithms, our algorithm has the lowest average cost and the maximum revenue–cost ratio. The revenue–cost ratio is 79.515% to 83.781%, which is about 5% to 10% higher than other algorithms. Our algorithm has the largest average revenue among all algorithms. The above indicates that our algorithm effectively utilizes the substrate network resources and saves the available resource space for subsequent virtual network requests.

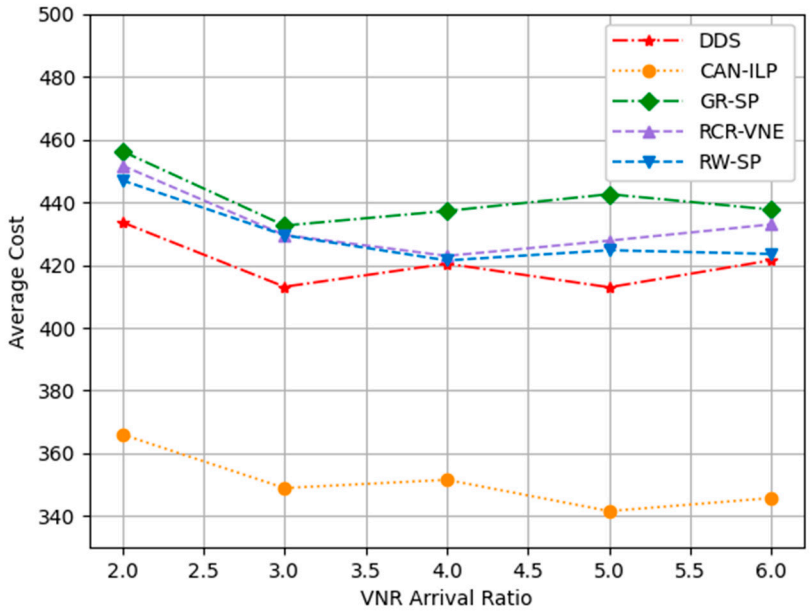

Figure 4. Average cost under various VNR arrival ratios.

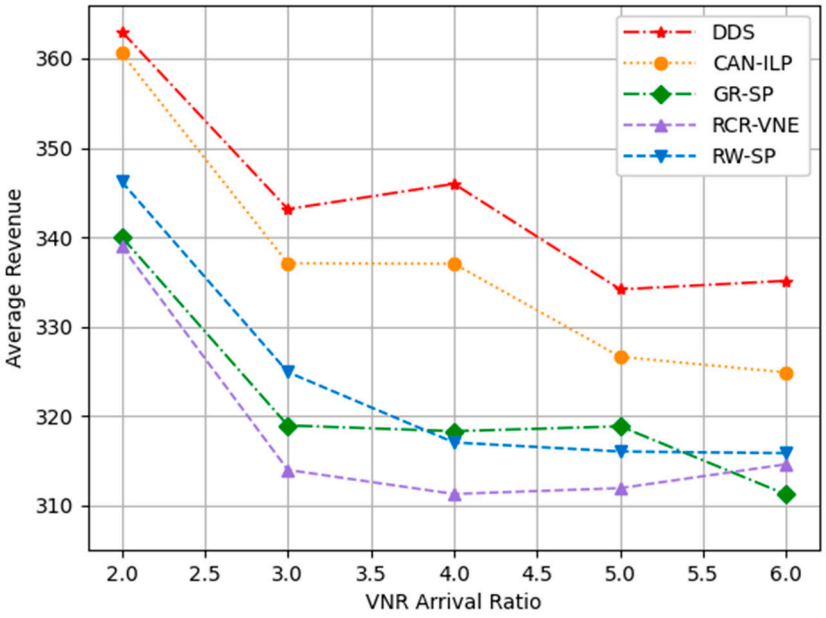

Figure 5. Average revenue under various VNR arrival ratios.

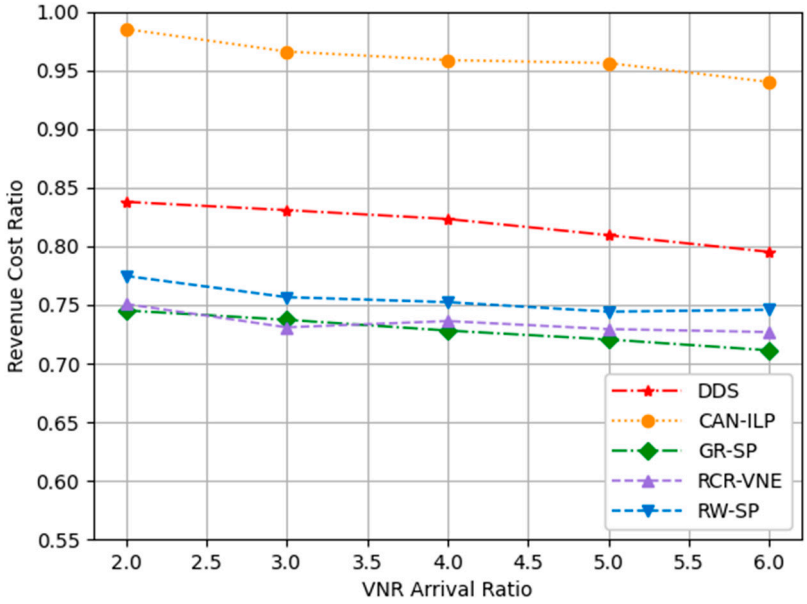

Figure 6. Revenue–cost ratio under various VNR arrival ratios.

5.2.3. Resource Utilization

Figures 7 and 8 describe the node computing resource utilization and storage resource utilization, while Figure 9 describes the link bandwidth resource utilization. With the increase of the virtual network request arrival ratio, the node resource utilization and link resource utilization of all algorithms increase accordingly. Compared with other algorithms, our algorithm has the highest resource utilization. Our node resource utilization is approximately 80%, while other algorithms are less than 70%, indicating an optimization of 10%. The link resource utilization is approximately 30%. The reason for the low link resource utilization in all algorithms is that the existence of the link propagation delay constraint and path splitting is not supported. The high resource utilization is because our algorithm can receive more virtual network requests than other algorithms and can use the substrate network resources for mapping more efficiently.

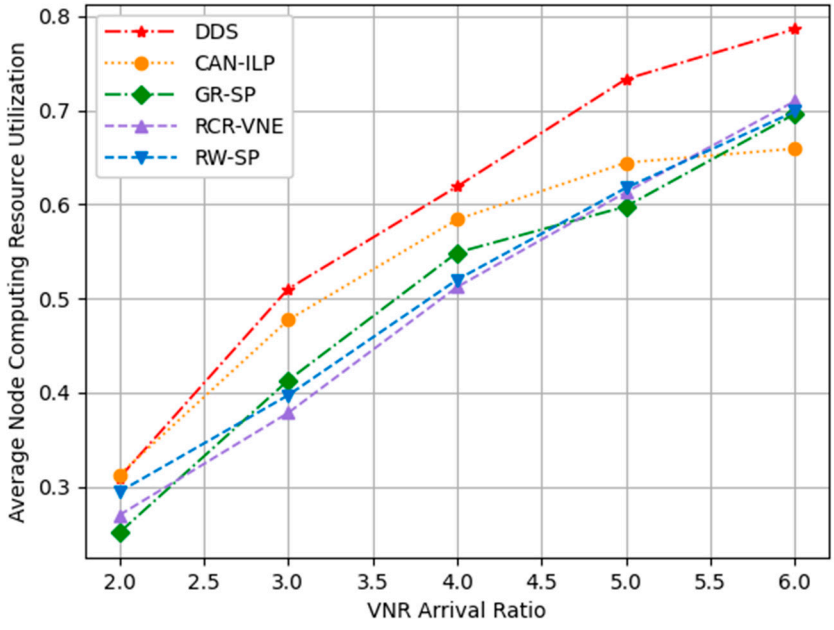

Figure 7. Average node computing resource utilization under various VNR arrival ratios.

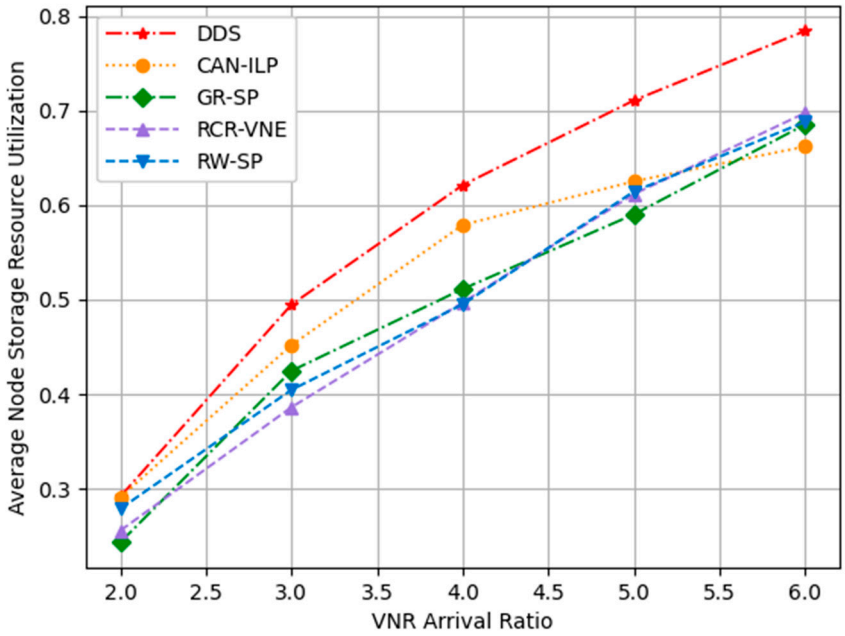

Figure 8. Average node storage resource utilization under various VNR arrival ratios.

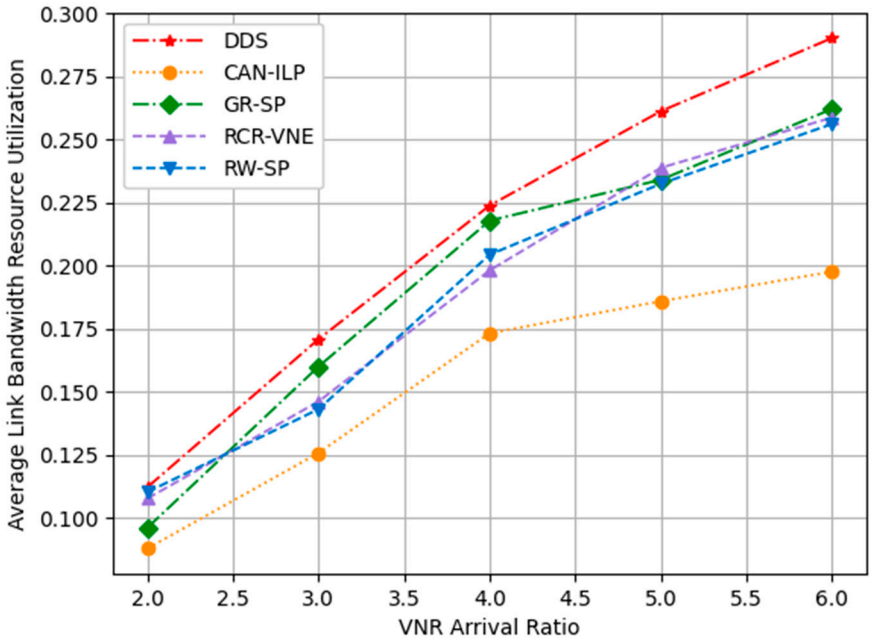

Figure 9. Average link bandwidth resource utilization under various VNR arrival ratios.

5.2.4. Comparison with the Optimal Solution

PURE-ILP, the virtual network embedding algorithm based on ILP is an exact and optimal algorithm. However, the NP-hard nature of PURE-ILP leads to high computational complexity, making it unsuitable for medium- or large-scaled network scenarios. We established a small network topology to evaluate the deviation between the proposed algorithm and the optimization algorithm. The number of substrate nodes is set to 20. D is set to 10. Other parameters are consistent with the description in Section 5.1. Figure 10 shows the numerical results of the VNR acceptance rate versus arrival rate. The acceptance ratio of the proposed algorithm VNR is slightly lower than that of the optimal solution, and the maximum deviation is about 3%, which shows that the proposed algorithm can also play a better role in small networks.

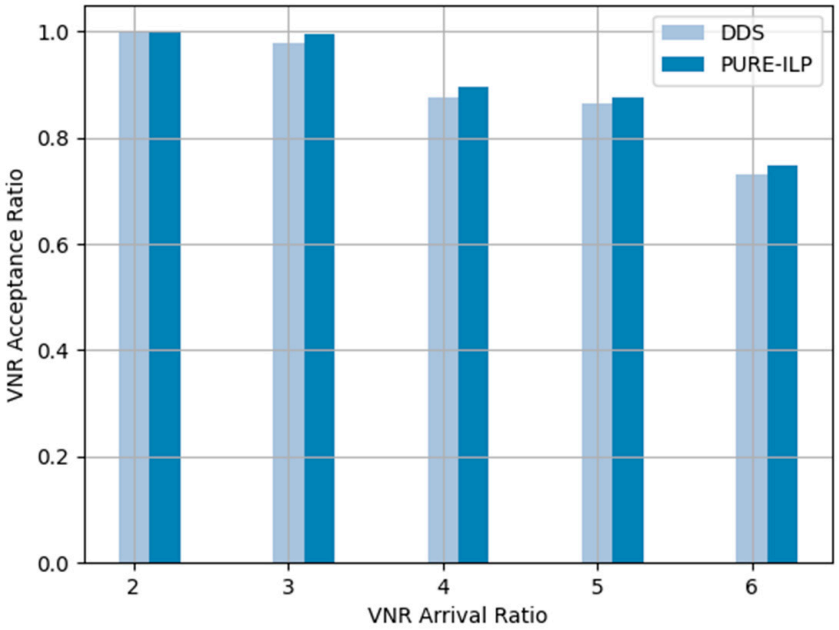

Figure 10. Comparison of DDS with PURE-ILP under various VNR arrival ratios.

5.2.5. Computational Complexity

This section analyzes the computational complexity of the algorithm. In a two-stage embedding sub-algorithm, node mapping is a polynomial algorithm in terms of and $|\text{node}^v|$, and [18,22] are as well. Node mapping in [21] is a polynomial time algorithm in terms of $|\text{node}^s|$, $|\text{node}^v|$, and $\max\{1, -\log \varepsilon\}$ (ε is a desired precision). The k-shortest path link mapping algorithm can be solved in $O((|\text{node}^s| + |\text{link}^s|)\log(|\text{node}^s| + k))$ time [27]. The computational complexity of link mapping is the same as in [18,21,22]. PURE-ILP has exponential time complexity. The solution increases exponentially with an increase in network size. However, our constrained integer linear programming sub-algorithm reduces the number of binary variables in linear programming by constructing a candidate node set and candidate link set. Ref. [14] proved that the mapping can be completed in a limited time by this method. Thus, the computational complexity of our algorithm is between a heuristic algorithm and a linear programming algorithm.

5.2.6. Discussion

The proposed algorithm provides a practical choice for the use of VNE in the real world with a model close to the real network and a good virtual request acceptance ratio and resource utilization. In fact, different virtual network requests have different resource requirements and QoS requirements. The advantage of our algorithm is that we have modeled this key problem and designed embedding strategies for different virtual network requests under different constraints and different optimization objectives, rather than using multi-objective optimization functions, which leads to the needs of tenants being met in a differentiated way and makes full use of the substrate network resources. The above experimental results fully verify the effectiveness of the proposed algorithm.

6. Conclusions

In order to flexibly respond to virtual network requests with differentiated resource requirements and QoS requirements, we propose a virtual network embedding algorithm that distinguishes network types. We first divide virtual network requests into ordinary requests and delay-sensitive requests according to the low latency requirements of virtual networks. Next, we design different embedding strategies for different types of virtual network requests. For ordinary requests, we adopt a heuristic two-stage embedding sub-

algorithm based on greedy strategy and considering resource consumption balance. We use a constrained integer linear programming sub-algorithm to solve delay-sensitive requests.

We conducted a comprehensive simulation and evaluation of the proposed algorithm. Experimental results showed that our algorithm outperforms the compared algorithm. Our algorithm can accept more virtual network embedding requests, reduce virtual network embedding cost, and maximize the utilization of substrate network resources. In addition, the task of placing service function chain (SFC) in network function virtualization (NFV) [33] can be seen as a virtual network embedding problem with network function constraints. Therefore, the proposed algorithm in this paper can also be applied to other network environments such as NFV.

In the next research stage, there are still some aspects to be improved. In order to improve the persuasiveness of the algorithm, it is meaningful to evaluate the algorithm in realistic traffic and network scenarios based on a real SDN substrate infrastructure and tenant requests. The network environment is complex and changeable, and the traffic in the network will change dynamically. In a future work, we should also consider resource reallocation [34] in the case of dynamic network traffic. A preliminary idea is to combine the global view of SDN, consider the state of resources allocated before, and reconfigure network elements that do not satisfy resource requirements to maintain network stability. Furthermore, considering the complexity of the actual network, the classification criteria for virtual network requests may not be comprehensive. In the future, it is important to consider other classification criteria for the actual network environment and propose personalized embedding algorithm research for differentiated networks, such as the cost demand network, delay demand network, bandwidth demand network, etc.

Author Contributions: Conceptualization, J.T., X.Z. (Xuewen Zeng), and X.Z. (Xiaodong Zhu); methodology, J.T. and X.Z. (Xiaodong Zhu); software, J.T.; validation, J.T., X.Z. (Xuewen Zeng), and X.Z. (Xiaodong Zhu); formal analysis, J.T.; investigation, J.T.; resources, J.T.; data curation, J.T.; writing—original draft preparation, J.T.; writing—review and editing, J.T., X.Z. (Xuewen Zeng), and X.Z. (Xiaodong Zhu); visualization, J.T.; supervision, X.Z. (Xuewen Zeng) and X.Z. (Xiaodong Zhu); project administration, X.Z. (Xiaodong Zhu); funding acquisition, X.Z. (Xuewen Zeng) and X.Z. (Xiaodong Zhu). All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Strategic Leadership Project of Chinese Academy of Sciences: SEANET Technology Standardization Research System Development (Project No. XDC02070100) and Goal-Oriented Project Independently Deployed by Institute of Acoustics, Chinese Academy of Sciences (Project No. MBDX202114).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We would like to express our gratitude to Xuewen Zeng and Xiaodong Zhu for their meaningful support in this work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Anderson, T.; Peterson, L.; Shenker, S.; Turner, J. Overcoming the internet impasse through virtualization. *Computer* **2005**, *38*, 34–41. [CrossRef]
2. Wang, X.; Tian, Y. Research on Network Virtualization for Software Defined Networking. *J. Netw. New Media* **2017**, *6*, 1–6, 23. [CrossRef]
3. Cao, H.; Wu, S.; Hu, Y.; Liu, Y.; Yang, L. A survey of embedding algorithm for virtual network embedding. *China Commun.* **2019**, *16*, 1–33. [CrossRef]
4. Feamster, N.; Gao, L.; Rexford, J. How to lease the Internet in your spare time. *ACM SIGCOMM Comput. Commun. Rev.* **2007**, *37*, 61–64. [CrossRef]
5. Bless, R.; Werle, C. Network virtualization from a signaling perspective. In Proceedings of the 2009 IEEE International Conference on Communications Workshops, Dresden, Germany, 14–18 June 2009; pp. 1–6. [CrossRef]

6. Schwerdel, D.; Günther, D.; Henjes, R.; Reuther, B.; Müller, P. German-lab experimental facility. In *Future Internet Symposium*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 1–10. [CrossRef]
7. McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 69–74. [CrossRef]
8. Hu, Y.; Cui, Z.; Li, Z.; Dong, Y.; Cui, P.; Wu, J. Construction technologies of polymorphic network environment based on codesign of domain-specific software/hardware. *J. Commun.* **2022**, *43*, 3–13. [CrossRef]
9. Andersen, D.G. Theoretical approaches to node assignment. Unpublished Manuscript. 2002. Available online: http://www.cs.cmu.edu/~{}dga/papers/andersen-assign.ps (accessed on 26 May 2022).
10. Fischer, A.; Botero, J.F.; Beck, M.T.; Meer, H.D.; Hesselbach, X. Virtual network embedding: A survey. *IEEE Commun. Surv. T utor.* **2013**, *15*, 1888–1906. [CrossRef]
11. Chowdhury, N.M.M.K.; Rahman, M.R.; Boutaba, R. Virtual network embedding with coordinated node and link mapping. In Proceedings of the IEEE INFOCOM, Rio de Janeiro, Brazil, 19–25 April 2009; pp. 783–791. [CrossRef]
12. Chowdhury, M.; Rahman, M.R.; Boutaba, R. Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Trans. Netw.* **2012**, *20*, 206–219. [CrossRef]
13. Melo, M.; Sargento, S.; Killat, U.; Timm-Giel, A.; Carapinha, J. Optimal virtual network embedding: Node-link formulation. *IEEE Trans. Netw. Serv. Manag.* **2013**, *10*, 356–368. [CrossRef]
14. Cao, H.; Zhu, Y.; Zheng, G.; Yang, L. A novel optimal mapping algorithm with less computational complexity for virtual network embedding. *IEEE Trans. Netw. Serv. Manag.* **2017**, *15*, 356–371. [CrossRef]
15. GLPK (GNU Linear Programming Kit). Available online: http://ftp.gnu.org/gnu/glpk/ (accessed on 16 December 2020).
16. IBM ILOG CPLEX Optimization Studio. Available online: https://www.ibm.com/docs/zh/icos/12.9.0?topic=cplex (accessed on 5 March 2021).
17. Lischka, J.; Karl, H. A virtual network mapping algorithm based on subgraph isomorphism detection. In Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures, Barcelona, Spain, 17 August 2009; pp. 81–88. [CrossRef]
18. Yu, M.; Yi, Y.; Rexford, J.; Chiang, M. Rethinking virtual network embedding: Substrate support for path splitting and migration. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 17–29. [CrossRef]
19. Cui, H.; Kong, F.; Liu, Y. A new algorithm of virtual network embedding based on minimum node stress and adjacent principle. In Proceedings of the 2012 IEEE Globecom Workshops, Anaheim, CA, USA, 3–7 December 2012; pp. 792–796. [CrossRef]
20. Nguyen, L.D.; Kim, N.; Kim, S.; Kim, C.-K. RT-VNE: A real-time strategy for Virtual Network Embedding towards resource efficiency. In Proceedings of the 2017 International Conference on Information Networking (ICOIN), Nang, Vietnam, 11–13 January 2017; pp. 185–190. [CrossRef]
21. Cheng, X.; Su, S.; Zhang, Z.; Wang, H.; Yang, F.; Luo, Y.; Wang, J. Virtual network embedding through topology-aware node ranking. *ACM SIGCOMM Comput. Commun. Rev.* **2011**, *41*, 38–47. [CrossRef]
22. Zhang, P.; Yao, H.; Liu, Y. Virtual network embedding based on computing, network, and storage resource constraints. *IEEE Internet. Things J.* **2017**, *5*, 3298–3304. [CrossRef]
23. Kirkpatrick, S.; Gelatt, C.D., Jr.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [CrossRef]
24. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 27 November 1995–1 December 1995; pp. 1942–1948. [CrossRef]
25. Dorigo, M. Optimization, Learning and Natural Algorithms. Ph.D. Thesis, Politecnico di Milano, Milan, Italy, 1992.
26. Holland, J.H. *Adaptation in Natural and Artificial Systems*; MIT Press: Cambridge, MA, USA, 1992.
27. Eppstein, D. Finding the k shortest paths. *SIAM J. Comput.* **1998**, *28*, 652–673. [CrossRef]
28. Kolliopoulos, S.G.; Stein, C. Improved approximation algorithms for unsplittable flow problems. In Proceedings of the 38th Annual Symposium on Foundations of Computer Science, Miami Beach, FL, USA, 20–22 October 1997; pp. 426–436. [CrossRef]
29. Nassar, B.O.; Tachibana, T. Path Splitting for Virtual Network Embedding in Elastic Optical Networks. *Int. J. Comput. Netw. Commun.* **2018**, *10*, 01–13. [CrossRef]
30. Yan, Z.; Ge, J.; Wu, Y.; Li, L.; Li, T. Automatic virtual network embedding: A deep reinforcement learning approach with graph convolutional networks. *IEEE J. Sel. Areas Commun.* **2020**, *38*, 1040–1057. [CrossRef]
31. Zhang, P.; Hong, Y.; Pang, X.; Jiang, C. VNE-HPSO: Virtual network embedding algorithm based on hybrid particle swarm optimization. *IEEE Access* **2020**, *8*, 213389–213400. [CrossRef]
32. Zhang, Z.; Su, S.; Lin, Y.; Shuang, K.; Xu, P. Adaptive multi-objective artificial immune system based virtual network embedding. *J. Netw. Comput. Appl.* **2015**, *53*, 140–155. [CrossRef]
33. Eramo, V.; Lavacca, F.G. Optimizing the Cloud Resources, Bandwidth and Deployment Costs in Multi-Providers Network Function Virtualization Environment. *IEEE Access* **2019**, *7*, 46898–46916. [CrossRef]
34. Eramo, V.; Tosti, A.; Miucci, E. Server Resource Dimensioning and Routing of Service Function Chain in NFV Network Architectures. *J. Electr. Comput. Eng.* **2016**, *2016*, 7139852. [CrossRef]