sensors

MDPI

Article

Optimizing Traffic Engineering for Resilient Services in NFV-Based Connected Autonomous Vehicles

Tuan-Minh Pham [1,2,*], Thi-Minh Nguyen [3]

[1] Faculty of Computer Science, Phenikaa University, Hanoi 12116, Vietnam
[2] A&A Green Phoenix Group JSC, Phenikaa Research and Technology Institute (PRATI), Hanoi 11313, Vietnam
[3] Department of Technology, Dong Nai Technology University, Bien Hoa City 760000, Vietnam; nguyenthiminh02@dntu.edu.vn
* Correspondence: minh.phamtuan@phenikaa-uni.edu.vn

Abstract: The massive amount of data generated daily by various sensors equipped with connected autonomous vehicles (CAVs) can lead to a significant performance issue of data processing and transfer. Network Function Virtualization (NFV) is a promising approach to improving the performance of a CAV system. In an NFV framework, Virtual Network Function (VNF) instances can be placed in edge and cloud servers and connected together to enable a flexible CAV service with low latency. However, protecting a service function chain composed of several VNFs from a failure is challenging in an NFV-based CAV system (VCAV). We propose an integer linear programming (ILP) model and two approximation algorithms for resilient services to minimize the service disruption cost in a VCAV system when a failure occurs. The ILP model, referred to as TERO, allows us to obtain the optimal solution for traffic engineering, including the VNF placement and routing for resilient services with regard to dynamic routing. Our proposed algorithms based on heuristics (i.e., TERH) and reinforcement learning (i.e., TERA) provide an approximation solution for resilient services in a large-scale VCAV system. Evaluation results with real datasets and generated network topologies show that TERH and TERA can provide a solution close to the optimal result. It also suggests that TERA should be used in a highly dynamic VCAV system.

Keywords: NFV; VCAV; resilient service; optimization; reinforcement learning; connected autonomous vehicles

Citation: Pham, T.-M.; Nguyen, T.-M. Optimizing Traffic Engineering for Resilient Services in NFV-Based Connected Autonomous Vehicles. *Sensors* **2021**, *21*, 8446. https://doi.org/10.3390/s21248446

Academic Editors: Cristina Cervelló-Pastor, Francisco Valera, Jorge Navarro, Jasone Astorga

Received: 18 November 2021
Accepted: 14 December 2021
Published: 17 December 2021

1. Introduction

Recently, emerging Internet of Things (IoT) applications, such as connected autonomous vehicles (CAV), smart home, mobile augmented reality, smart agriculture, became increasingly popular [1]. A CAV system relies upon computer vision using a series of video cameras, radars, and Light Detection and Ranging (LIDAR) that allow the car to perceive the world around it. The system processes a massive amount of data collected from sensors to provide its application services composed of several application functions, including video capturing, sensor fusion, object tracking, localization, path planning, and control components. For example, a video camera on an autonomous car could generate hundreds of gigabytes in an hour of driving for a 720p video. A critical issue for a CAV system is how to transfer and process the massive amount of data generated daily in a timely fashion.

Network Function Virtualization (NFV) has been raised as a promising approach to tackling this issue [2]. A virtualized network function (VNF), including a traditional network function and general computation task, can be deployed as an instantiable software component running in a commercial off-the-shelf server. A VNF instance can be placed in edge devices to enable application services with low latency. Several VNFs on different edge and cloud devices can be connected as a service function chain (SFC) to enable real-time and flexible services. While an NFV-based CAV system, referred to as VCAV, is able to

provide a flexible CAV service with low latency, it is challenging to protect a service from any system failure.

The issue of resilient services has been discussed in a specification published by The European Telecommunications Standards Institute (ETSI) [3]. The main challenge of providing a resilient service in a VCAV system is to optimize the placement and routing of VNFs in response to a failure in an NFV infrastructure (NFVI). Previous researches have considered various techniques to address several aspects of the resilient service problem [4–9]. However, all previous approaches could not be applied to a VCAV system due to the high dynamics of VCAV data traffic. In addition, most previous work assumes a fixed mapping of routing paths onto NFVI and implicit paths connected between VNFs in an SFC, which is not practical. Our work aims to optimize traffic engineering, including the VNF placement and routing for resilient services in a VCAV system, to minimize the service disruption cost, considering the dynamics of routing paths and service function chaining.

The main contributions of this paper are three-fold:

- We proposed an integer linear programming (ILP) model for the resilient service problem, referred to as TERO. The TERO model provides the optimal VNF placement and routing for a set of service demands when a failure occurs in a VCAV system.
- We developed a heuristic algorithm (i.e., TERH) and Reinforcement Learning (RL) based algorithm (i.e., TERA) to find an approximation solution for the resilient service problem in an extensive network. The approximation solution provided by TERH and TERA is close to the optimal solution. In comparison with TERH, TERA can achieve a similar cost with significantly reduced time in a dynamic failure scenario.
- We validate our proposed models and algorithms in real datasets and generated network topologies. The evaluation results suggest that TERA should be used to minimize the service disruption cost of a VCAV system concerning the high dynamics of data traffic.

The rest of the paper is organized as follows. Section 2 reviews some related works. Section 3 describes the system and states the optimization problem of traffic engineering for resilient services in a VCAV system. Section 4 presents the TERO model that provides the optimal traffic engineering for resilient services, including VNF placement and routing when a failure occurs in a VCAV system. Section 5 describes the TERH and TERA algorithms based on heuristic and reinforcement learning to find an approximation solution for the resilient service problem. We present the evaluation of our proposed model and algorithms in In Section 6. Finally, the conclusion is presented in Section 7.

2. Related Work

Network Function Virtualization (NFV) has been raised as a potential approach to a flexible and efficient solution for processing a massive volume of data in an IoT system. For the evolution of an IoT system with NFV, we refer the readers to [10]. The reliability of services on the Internet and in an IoT system is a crucial problem that has been widely studied (e.g., [11–16]). However, existing solutions are not suitable for an NFV-based IoT system, where functional modules can be deployed in different data centers and connected together to create a flexible service.

The challenge of developing a solution for resilient services in an NFV-based IoT system is to find the optimal resource allocation for data routing and processing when a failure occurs in a distributed system. So far, a few studies have considered the design of a resilient NFV-based IoT system [17–20]. Huang et al. devised a proactive fail-over mechanism based on failure prediction to enhance the resilience of NFV services deployed in a distributed edge network [17]. Ergenc et al. analyzed the complexity and boundaries of the problem as well as developed heuristics to increase the fault tolerance of an IoT network when there are some node and link failures [18]. Bakhshi et al. proposed a mathematical model for an SDN-based fault-tolerant architecture in an IoT environment [19]. Sanabria et al. used machine learning techniques to provide prediction and alert capabilities for telemedicine applications [20]. They proposed a hybrid Edge/Cloud architecture training of the deep learning prediction model. Several optimization models have been proposed

for resilient services in the Mobile Edge Computing [21]. However, these solutions lack considering service function chaining that is a key feature of NFV. In addition, the dynamic routing path has not been tackled while it is an essential feature of a CAV system. Some studies have discussed an efficient design for data processing and routing in a VCAV system (e.g., [22–24]) However, a solution to the traffic engineering problem for resilient services in a VCAV system has not been provided.

This paper offers a new ILP formulation for a traffic engineering solution, including the VNF placement and routing when node or link failures occur in a VCAV system. Moreover, we take into account the dynamic routing path at the request time. We also propose two algorithms based on heuristics and reinforcement learning to provide an approximation solution in a large-scale VCAV system.

3. System Description

In a VCAV system, system and safety functions are deployed locally in an autonomous vehicle. Light workload functions such as planning and sensor fusion can be run in edge nodes. Some functions that require a heavy computational task and process a massive volume of data collected from many cars can be implemented in the cloud layer (Figure 1). These functions can be connected in an order list to create an application service. An example of an SFC in a VCAV system is sensor fusion, world model, behavior generation, planning, and vehicle control. A VCAV system allocates its resource in the edge and cloud layers for a set of service demands required by vehicles. When a failure occurs, system resources are rapidly reallocated to maintain application services supplied to vehicles.

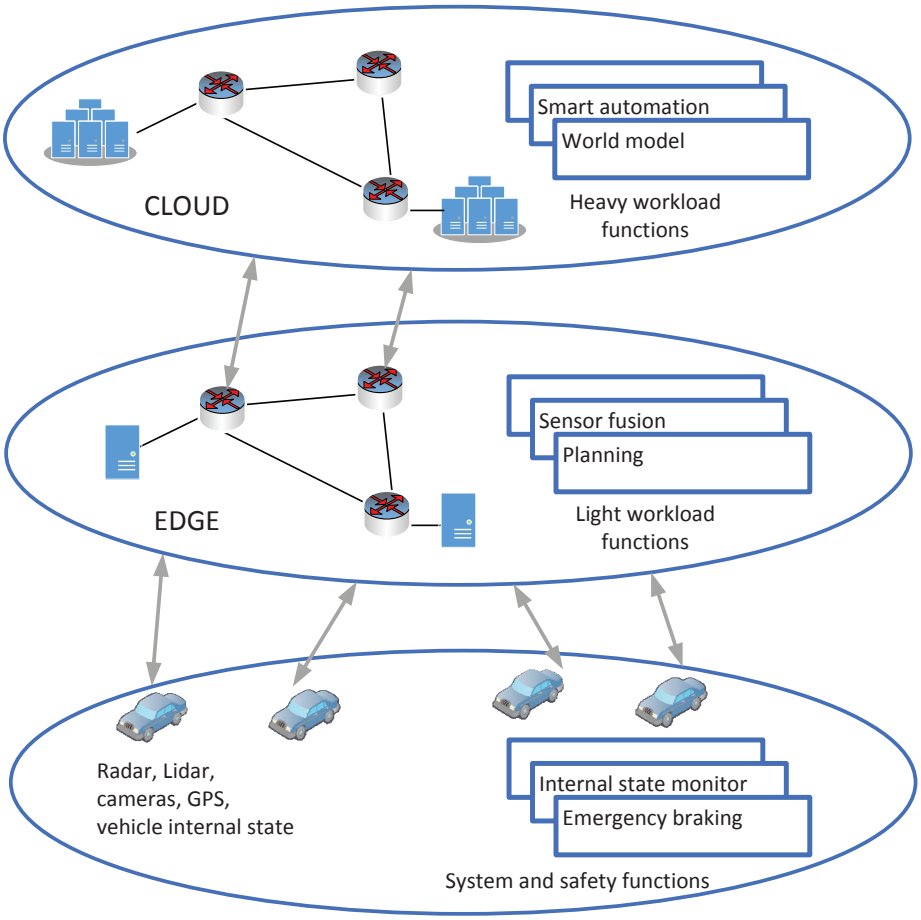

Figure 1. Functional components in an NFV-based connected autonomous vehicle.

A CAV system based on NFV includes three main elements: the NFV infrastructure (NFVI), the VNFs, and the management and orchestration of NFVs (MANO). NFVI consists

of the shared and virtualized resources of physical networking, computing, and storage. A VNF can be any functional module of a VCAV system, e.g., sensor fusion, world model, and planning. The MANO element handles all automatic processes for loading and managing VNFs. A traffic engineering solution for resilient services can be incorporated into the NFV architecture as a part of MANO.

We represent a VCAV system as a directed graph $G = (V, E)$ where V and E denote the set of physical nodes and links. We define r_v^n to be node v's resource capacity, and r_e^l to be link e's bandwidth capacity. The beginning node and ending node of link e are denoted by i_e and j_e. The node's processing resource considered in this work is the number of CPU cores. We can use similar formulas of the processing resource in the model to include additional types of resources (e.g., memory, storage). We represent different network topologies by setting the parameters of links and nodes in the model. We denote by F the set of VNF types. η_u is the number of cores required by VNF type $u \in F$ to process a traffic volume. The routing delay β_v is the time duration needed by node v to route an amount of traffic. The processing delay μ_{vu} is the time duration required to provide VNF type u at node v. We denote by $\mathbf{w} = (w_e)$ the weight vector of NFVI where w_e is an integer number representing link e's weight. We define λ_v^n to be the failure state of node v, and λ_e^l to be the failure state of link e. $\lambda_v^n = 0$ if node v fails, otherwise $\lambda_v^n = 1$. $\lambda_e^l = 0$ if link e fails, otherwise $\lambda_e^l = 1$. A link failure can be caused by hardware problems, software issues (e.g., too many connections, configuration changes, denial of service attacks), or the mobility of vehicles.

We define $\Omega = \{S_i\}$ to be all system-supported SFC. An SFC is denoted by $S_i = (u_{i1}, \ldots, u_{ij}, \ldots, u_{in})$ where u_{ij} is the jth VNF of SFC S_i. The service demand set is denoted by $\Gamma = \{d\}$. The parameter set of service demand $d \in \Gamma$ includes arrival node s_d, departure node t_d, SFC $S_d \in \Omega$, SFC delay α_d, and bandwidth volume b_d. An arrival node is an NFV node that provides an entry of a service demand into a VCAV system. A departure node is an NFV node at which the demand traffic leaves a VCAV system. A middle node is an NFV node between an arrival node and a departure node on an SFC path realizing a service demand. An NFV node either provides a VNF instance or routes traffic of a service demand.

When a failure happens, a VCAV system needs to modify some paths of service demands and VNF placement on these paths to meet the requirement of service demands and avoid an overload of some nodes. The process is referred to as the traffic engineering problem for resilient services. Optimizing VNF placement and routing could significantly impact the cost efficiency and performance of a VCAV system. The problem is stated as follows:

Problem 1 (Traffic Engineering for Resilient Services (TER))**.** *Given a VCAV system G, find a traffic engineering solution for fulfilling a service demand set Γ, in order to minimize the system interruption when failures occur under constraints on service functions chaining and the restriction rule of routing reallocation.*

4. Optimization Model for Resilient Services

We propose an optimization model based on ILP to find the optimal result of the TER problem. The model is referred to as TERO. The main variables of TERO are as follows:

- $\mathbf{x}_2 = (x_{2ed})$ is the routing solution satisfying the service demand set when a failure occurs in a VCAV system. If demand d uses link e, $x_{2ed} = 1$, otherwise, $x_{2ed} = 0$.
- $\mathbf{y}_2 = (y_{2vdi})$ is the VNF placement solution in the failure state. If node v provides the ith VNF of demand d, $y_{2vdi} = 1$, otherwise, $y_{2vdi} = 0$.

We summarize the main mathematical notations of TERO in Table 1.

Table 1. Summary of main notations.

Input Parameters	
$G = (V, E)$	A directed graph representing a VCAV system where V and E is denoted the set of physical nodes and physical links, respectively.
r_v^n	Node v's resource capacity
r_e^l	Link e's bandwidth capacity
i_e	Link e's beginning node
j_e	Link e's ending node
F	The VNF type set
η_u	The number of CPU cores required by VNF type $u \in F$ to process a volume of data traffic
Ω	The system-supported SFC set: $\Omega = \{S_i\}$, $S_i = \left(u_{i1}, \ldots, u_{ij}, \ldots, u_{in}\right)$; u_{ij} is the jth VNF of SFC S_i.
$\Gamma = \{d\}$	The service demand set
s_d	Demand d's arrival node
t_d	Demand d's departure node
b_d	Demand d's bandwidth volume
α_d	Demand d's SFC delay
S_d	Demand d's SFC
β_v	The routing delay for a traffic unit at node v
μ_{vu}	The processing delay of VNF type u at node v
$\gamma_{vv'u_{di}}$	The moving cost when moving ith VNF of demand d from v to v'
$\rho_{vv'}$	The cost of the minimum-weight path between v and v'
κ_u	The size of the state and data of a VNF type u
λ_v^n	The failure state of node v
λ_e^l	The failure state of link e
$\mathbf{y}_1 = (y_{1vdi})$	The current VNF placement solution: If node v provides the ith VNF of SFC S_d, $y_{1vdi} = 1$, otherwise, $y_{1vdi} = 0$
$\mathbf{x}_1 = (x_{1ed})$	The current routing solution: If demand d uses link e, $x_{1ed} = 1$, otherwise, $x_{1ed} = 0$
$\mathbf{w} = (w_e)$	The weight vector of NFVI where w_e is link e's weight.
Output variables	
$\mathbf{x}_2 = (x_{2ed})$	The routing solution for satisfying demands in the failure state: If demand d uses link e, $x_{2ed} = 1$, otherwise, $x_{2ed} = 0$
$\mathbf{y}_2 = (y_{2vdi})$	The VNF placement solution in the failure state: If node v provides the ith VNF of SFC S_d, $y_{2vdi} = 1$, otherwise, $y_{2vdi} = 0$
Auxiliary variables	
$l_{v_1v_2}$	The length of the path from node v_1 to node v_2
θ	A large number
y_{2vdi}^{σ}	If a node between s_d and v on the path realizing demand d provides the ith VNF of demand d (i.e., u_{di}), $y_{2vdi}^{\sigma} = 1$, otherwise $y_{2vdi}^{\sigma} = 0$.
$\bar{y}_{2edi}^{\sigma}$	If link e is on the path realizing demand d, and a node between s_d and i_e on the path provides u_{di}, $\bar{y}_{2edi}^{\sigma} = 1$, otherwise $\bar{y}_{2edi}^{\sigma} = 0$.
λ_e'	$\lambda_e' = 1$ if and only if a failure occurs on link e, at node i_e, or at node j_e.
φ_{ed}	$\varphi_{ed} = 0$ if and only if $\lambda_e' = 1$ and link e is on the path realizing demand d.
φ_d^{σ}	$\varphi_d^{\sigma} = 0$ if and only if there exists a link or node failure on the path used by demand d.
$z_{vv'di}$	$z_{vv'di} = 1$ if and only if the ith VNF of SFC S_d is moved from node v to node v', otherwise, $z_{vv'di} = 0$.

4.1. Service Function Chaining Routing

The four conditions of service function chaining routing in a VCAV system are as follows: the flow balance, function provision, function chain, and delay constraints. The flow balance condition guarantees to conserve the flow traffic of a service demand along its path. The function provision condition assures that the VCAV system provides all VNFs of a service demand. The function chain condition guarantees that all VNFs of a service demand are connected in sequence. The delay constraint assures the fulfillment of the end-to-end delay of an SFC.

We define $l_{v_1v_2}$ to be the length of the path from node v_1 to node v_2. Let θ be a large number. The balance condition is as follows:

$$\sum_{\{e:i_e=v\}} x_{2ed} - \sum_{\{e:j_e=v\}} x_{2ed} = 0, \quad \forall d, \forall v, v \neq s_d, v \neq t_d, \tag{1}$$

$$\sum_{\{e:i_e=s_d\}} x_{2ed} = 1, \quad \forall d, \tag{2}$$

$$\sum_{\{e:j_e=t_d\}} x_{2ed} = 1, \quad \forall d, \tag{3}$$

$$(x_{2ed} - 1)\theta \leqslant l_{i_e t_d} - w_e - l_{j_e t_d} \leqslant (1 - x_{2ed})\theta, \quad \forall d, \forall e. \tag{4}$$

Equation (1) guarantees that there is one entering flow and one leaving flow at a middle node on the path of a service demand. Equation (2) assures that there is one leaving flow at the departure node of a service demand. Equation (3) assures that there is one entering flow at the arrival node of a service demand. Equation (4) guarantees that there is no cycles in a service demand path.

The function provision condition is as follows:

$$\sum_v y_{2vdi} = 1, \quad \forall d, \forall i, \tag{5}$$

$$y_{2vdi} \leqslant \sum_{\{e:i_e=v \text{ or } j_e=v\}} x_{2ed}, \quad \forall v, \forall d, \forall i. \tag{6}$$

Equation (5) assures that the VCAV system provides all VNFs required by a service demand. Equation (6) ensures that the VCAV system only selects a node on the path of demand d to allocate a VNF for the demand.

To represent the function chain condition, we add two additional binary variables y^{σ}_{2vdi} and $\bar{y}^{\sigma}_{2edi}$. If a node between s_d and v on the path realizing demand d provides the ith VNF of demand d (i.e., u_{di}), $y^{\sigma}_{2vdi} = 1$, otherwise $y^{\sigma}_{2vdi} = 0$. If link e is on the path realizing demand d, and a node between s_d and i_e on the path provides u_{di}, $\bar{y}^{\sigma}_{2edi} = 1$, otherwise $\bar{y}^{\sigma}_{2edi} = 0$. The constraint is as follows:

$$y_{2vdi} \leqslant y^{\sigma}_{2vd(i-1)}, \quad \forall e, \forall d, \forall i \geqslant 1, \tag{7}$$

$$y^{\sigma}_{2vdi} = y_{2vdi} + \sum_{\{e:j_e=v\}} \bar{y}^{\sigma}_{2edi}, \quad \forall e, \forall d, \forall i, \tag{8}$$

$$x_{2ed} + y^{\sigma}_{2i_e di} - 1 \leqslant \bar{y}^{\sigma}_{2edi} \leqslant x_{2ed}, \quad \forall v, \forall d, \forall i, \tag{9}$$

$$\bar{y}^{\sigma}_{2edi} \leqslant y^{\sigma}_{2i_e di}, \quad \forall e, \forall d, \forall i. \tag{10}$$

Equation (7) guarantees that node v supplies demand d with u_{di} if and only if $u_{d(i-1)}$ is fulfilled by either node v or its preceding node that belongs to the demand d's path. Equation (8) guarantees that $y^{\sigma}_{2vdi} = 1$ if and only if u_{di} is delivered by a node between s_d and v and the node belongs to the path realizing demand d. Note that we have the sum of $\bar{y}^{\sigma}_{2edi}$ on the right-hand side of Equation (8) because there might be several incoming links of node v. Equations (9) and (10) assures that $\bar{y}^{\sigma}_{2edi} = 1$ if and only if link e belongs to the

path realizing demand d, and VNF u_{di} is deployed at either i_e or its preceding node that belongs to the demand d's path.

The SFC delay represents the sum of the routing delay and VNF processing delay at every node that belongs to the demand path. We express the condition as follows:

$$\sum_e \beta_{i_e} x_{2ed} b_d + \sum_v \mu_{vu_{di}} \sum_i y_{2vdi} \leqslant \alpha_d, \quad \forall d. \tag{11}$$

4.2. Restriction Rule in Flow Reallocation

First, the demand traffic cannot routed through a failed node or link. The condition is as follows:

$$\sum_d b_d x_{2ed} \leqslant r_e^l \lambda_e^l, \quad \forall e, \tag{12}$$

$$\sum_{v,d,i} b_d y_{2vdi} \eta_{u_{di}} \leqslant r_v^n \lambda_v^n, \quad \forall v. \tag{13}$$

Equation (12) guarantees that the total traffic of all demands passing through a link cannot surpass its bandwidth capacity. Equation (13) guarantees that the number of cores that a node allocates to the VNFs of all demands cannot surpass the node capacity. Note that when a node and link fail, the system loses all capacity of the node and link.

Second, the resource allocation for a service demand without failures on its paths should not be changed. We introduce three binary variables λ'_e, φ_{ed} and φ_d^σ. $\lambda'_e = 1$ if and only if a failure occurs on link e, at node i_e, or at node j_e. $\varphi_{ed} = 0$ if and only if $\lambda'_e = 1$ and link e is on the path realizing demand d. $\varphi_d^\sigma = 0$ if and only if there exists a link or node failure on the path used by demand d. Let $\mathbf{x}_1 = (x_{1ed})$ be the current routing solution. If demand d uses link e, $x_{1ed} = 1$, otherwise, $x_{1ed} = 0$. The condition is given by:

$$\lambda_e^l \leqslant {\lambda'}_e, \quad \forall e, \tag{14}$$

$$\lambda_{i_e}^n \leqslant {\lambda'}_e, \quad \forall e, \tag{15}$$

$$\lambda_{j_e}^n \leqslant {\lambda'}_e, \quad \forall e, \tag{16}$$

$${\lambda'}_e \leqslant \lambda_e^l + \lambda_{i_e}^n + \lambda_{j_e}^n, \quad \forall e, \tag{17}$$

$$\varphi_d^\sigma x_{1ed} \leqslant x_{2ed}, \quad \forall d, \forall e, \tag{18}$$

$$\varphi_d^\sigma \leqslant \varphi_{ed}, \quad \forall d, \forall e, \tag{19}$$

$$1 - {\lambda'}_e \leqslant \varphi_{ed}, \quad \forall d, \forall e, \tag{20}$$

$$1 - x_{1ed} \leqslant \varphi_{ed} \leqslant 2 - x_{1ed} - {\lambda'}_e, \quad \forall d, \forall e. \tag{21}$$

Equations (14)–(17) guarantee that $\lambda'_e = 1$ if and only if we have either $\lambda_e^l = 1$, $\lambda_{i_e}^n = 1$, or $\lambda_{j_e}^n = 1$, and $\lambda'_e = 0$ if and only if we have $\lambda_e^l = 0$, $\lambda_{i_e}^n = 0$, and $\lambda_{j_e}^n = 0$. Equation (18) guarantees that the routing solution for demand d does not change if there is no failures on its path. Equation (19) ensures that $\varphi_d^\sigma = 0$ if and only if $\varphi_{ed} = 0$ for one of links along the path used by demand d. Equations (20) and (21) ensure that $\varphi_{ed} = 0$ if and only if $\lambda'_e = 1$ and $x_{1ed} = 1$.

4.3. Objective Function

Our objective is to minimize the service disruption cost. The service disruption cost of a service demand is the cost of moving its VNF state and data to a new node. It is in proportion to the time required to provide all services normally. Its unit of measurement is a derived unit of time. We denote by $\gamma_{vv'u_{di}}$ the cost when moving ith VNF of demand d

from v to v'. Let $\rho_{vv'}$ be the cost of the minimum-weight path from v to v'. κ_u is the size of the state and data of a VNF type u.

The service disruption cost of a VNF instance is given by:

$$\gamma_{vv'u_{di}} = \rho_{vv'}\kappa_{u_{di}}. \tag{22}$$

We add an additional variable $z_{vv'di}$ to compute the service disruption cost in a VCAV system when a failure occurs. In a failure state, if the ith VNF of SFC S_d is moved from node v to node v', $z_{vv'di} = 1$, otherwise, $z_{vv'di} = 0$. Let $\mathbf{y}_1 = (y_{1vdi})$ be the current VNF placement solution. If node v provides the ith VNF of SFC S_d, $y_{1vdi} = 1$, otherwise, $y_{1vdi} = 0$. The constraints on the value of $z_{vv'di}$ are given by:

$$z_{vv'di} \leqslant y_{1vdi}, \quad \forall v, \forall v', \forall d, \forall i, \tag{23}$$

$$z_{vv'di} \leqslant y_{2v'di}, \quad \forall v, \forall v', \forall d, \forall i, \tag{24}$$

$$y_{1vdi} + y_{2v'di} - 1 \leqslant z_{vv'di}, \quad \forall v, \forall v', \forall d, \forall i. \tag{25}$$

Equations (23)–(25) guarantee that v', $z_{vv'di} = 1$ if and only if we have $y_{1vdi} = 1$ and $y_{2v'di} = 1$, otherwise, $z_{vv'di} = 0$.

The service disruption cost of a resource allocation solution when a failure happens is given by:

$$U = \sum\nolimits_{v,v',d,i} z_{vv'di}\gamma_{vv'u_{di}}. \tag{26}$$

4.4. ILP Model for Resilient Services

The TER problem is to find a traffic engineering solution for minimizing a cost function of service disruption when a failure occurs in a VCAV system. The TERO model provides the optimal VNF routing and placement for the TER problem in a failure state. The formulation of the TERO model includes the objective function given by Equation (26) and the constraints given by Equations (1)–(25).

5. Approximation Algorithms

In the previous section, we proposed the TERO model to obtain the optimal solution for traffic engineering in a VCAV system when a failure occurs. An ILP solver is not able to handle a scenario with hundreds of nodes and thousands of demands since the number of variables in TERO comes to billions in such a large scenario. Hence, we propose two algorithms based on a heuristic approach and reinforcement learning to find an approximation solution for the TER problem in a large-scale VCAV system. The two algorithms use the similar input parameters of the TER problem, which are presented in Table 1.

5.1. Heuristic Algorithm

We propose a heuristic algorithm, namely TERH, based on the Simulated Annealing (SA). In TERH, we develop the structure of the resource allocation solution and the function of neighborhood selection for the TER problem. SA is a heuristic technique that finds the optimum for a global optimization problem [25]. The search method accepts a worse scenario with a certain probability of overcoming a local optimum.

We represent a resource allocation solution for a service demand set in a VCAV system as a list of tuples, which is denoted by $O_m = ((d, i, v) : d \in D, i \in S_d, v \in V)$. The solution shows that node v provides the ith VNF of demand d. The details of the TERH algorithm are presented in Algorithm 1.

Algorithm 1 Simulated Annealing-based approximation algorithm for TER

1: **function** TERH(G, Γ, $\mathbf{y}_1$)
2: Initialize T, T_0, T_n, ϕ, τ
3: Find an initial solution O_m
4: $O_m^* \leftarrow O_m$
5: Compute $\mathbf{y}_2^*$ from O_m^*
6: **while** $T \geq T_n$ **do**
7: **for** $n \leftarrow 1$ to ϕ **do**
8: **repeat**
9: $(d, i, v) \leftarrow$ a random tuple in O_m
10: $v' \leftarrow$ a random node in V
11: $O_m^{'} \leftarrow \text{REPLACE}(d, i, v, v', O_m)$
12: Compute $\mathbf{x}_2'$ and constraints in a failure scenario
13: **until** O_m' is feasible
14: Compute $\mathbf{y}_2$ from O_m
15: Compute $\mathbf{y}_2'$ from O_m'
16: **if** $U(\mathbf{y}_2', \mathbf{y}_1) < U(\mathbf{y}_2, \mathbf{y}_1)$ **then**
17: $O_m \leftarrow O_m'$
18: **if** $U(\mathbf{y}_2', \mathbf{y}_1) < U(\mathbf{y}_2^*, \mathbf{y}_1)$ **then**
19: $O_m^* \leftarrow O_m'$
20: $\mathbf{x}_2^* \leftarrow \mathbf{x}_2'$
21: $\mathbf{y}_2^* \leftarrow \mathbf{y}_2'$
22: **end if**
23: **else**
24: $\Delta \leftarrow U(\mathbf{y}_2', \mathbf{y}_1) - U(\mathbf{y}_2, \mathbf{y}_1)$
25: $\varepsilon \leftarrow$ a random number between 0 and 1
26: **if** $\exp(-\Delta/T) > \varepsilon$ **then**
27: $O_m \leftarrow O_m'$
28: **end if**
29: **end if**
30: **end for**
31: $T \leftarrow C(T)$
32: **end while**
33: **return** $\mathbf{y}_2^*$, $\mathbf{x}_2^*$
34: **end function**

The algorithm contains two main loops. The outer loop is controlled by the temperature parameter T, the start temperature parameter T_0, the stop temperature parameter T_n, and the cooling function $C(T)$. For each T, the algorithm runs an inner loop that uses a neighborhood function to move from the current solution to another. T decreases by $C(T)$ after one iteration of the outer loop. The algorithm completes its solution search when T is smaller than T_n.

We define ϕ to be the number of iterations of the inner loop. Let O_m be an initial solution. We use the most common cooling function $C(T) = \tau T$, for some parameter τ from interval (0, 1). The initial temperature is the maximal cost difference between any two neighbor solutions. The end temperature typically is close to zero.

In the neighborhood selection (i.e., line 8–12), we define the Replace(d, i, v, v', O_m) operator that substitutes node v' for node v. We use the Replace operator for a random tuple $(d, i, v) \in O_m$ and random target node v' repeatedly until we find a feasible solution.

In the inner loop, if the objective value of a neighborhood solution is less than that of a current solution, the iteration continues with the neighborhood solution as TERH is moving towards a better solution (i.e., lines 16–22). Otherwise, TERH randomly accepts the neighborhood solution with a probability in order to overcome local optimization (i.e., lines 24–28). The acceptance probability decreases with T for a given value of Δ. Hence, the uphill movement is more uncommon in a successive inner loop. After ϕ iterations, the inner loop finishes its solution search. After the temperature is decreased, the inner

loop is started again. The approximation of TERH's solution can be controlled by adjusting the number of iterations ϕ and cooling function $C(T)$.

5.2. Reinforcement Learning Based Approximation Algorithm

We propose a Soft Actor-Critic (SAC) based approximation algorithm, called TERA, to solve the TER problem in a large-scale VCAV system. SAC is a variant of actor-critic methods for reinforcement learning. It aims to maximize expected rewards and entropy in a large-scale continuous action space [26]. While earning as many rewards as possible, it attempts to take actions as randomly as possible. This encourages the search process to discover the environment, which accelerates training and decreases the probability of going back to a visited action.

The mathematical formulation of SAC is a Markov decision process with a set of parameters including the state space $\mathcal{M}$, action space $\mathcal{A}$, probability density p and reward function r. The probability density p, defined by $\mathcal{M} \times \mathcal{M} \times \mathcal{A} \to (0, \infty]$, is the probability of the next state $\mathbf{m}_{t+1} \in \mathcal{M}$ given the current state $\mathbf{m}_t \in \mathcal{M}$ and action $\mathbf{a}_t \in \mathcal{A}$. The reward r, defined by $\mathcal{M} \times \mathcal{A} \to [r_{\min}, r_{\max}]$ is an environment reward of a state transition. SAC seeks a policy $\omega(\mathbf{m}_t|\mathbf{a}_t)$ for maximizing the learning objective. The learning objective is the expected sum of rewards and the policy's entropy. SAC uses the hyperparameter λ, namely temperature, to adjust the association between the reward and the entropy in the learning objective. Let h be the entropy function with regard to the policy ω. The formulation of the learning objective of SAC is as follows:

$$J(\omega) = \sum_t \mathbb{E}_{(\mathbf{m}_t, \mathbf{a}_t)}[r(\mathbf{m}_t, \mathbf{a}_t) + h(\omega(\cdot|\mathbf{m}_t))\lambda] \tag{27}$$

The primary step of developing a solution based on SAC is to formulate the three key parameters: The state space, action space, and reward function. In TERA, the state space should represent how a set of demands is satisfied when a failure happens. Hence, we formulate it by a list of tuples, $\mathbf{m}_t = \{(d, u, v)\} \cup \{\lambda_v^n\} \cup \left\{\lambda_v^l\right\}$. An element of $\mathbf{m}_t$ show that node v provides VNF u of service demand d in a failure scenario. An action in TERA makes a movement between states, representing a possible resource allocation solution. We represent an action by $\mathbf{a}_t = \left(v_1, v_2, \ldots, v_{|\mathbf{m}_t|}\right)$ where $v_i \in V$ is a resource allocation solution for the ith tuple in the action space. As TERA optimizes the learning policy to maximize the learning objective, we use the objective function $U_a = -U$ to compute the reward of a solution. Hence, we can evaluate the solution's cost efficiency and learning policy produced by TERA for minimizing the service disruption cost.

We present the main steps of TERA in Algorithm 2. The actor network returns a resource allocation action according to an input state. The NFV environment runs the action to move to a new state. The critic network uses the new state, its reward and the previous state to compute the advantage of the new state, which is used to update the weights of the actor and critic networks. The role of the critic network is the actor's loss function. We implement the actor and critic networks as neural networks. We will discuss some details of selecting their parameters in Section 6.1.

Algorithm 2 Learning-based approximation algorithm for TER

1: **function** TERA(G, Γ, $\mathbf{y}_1$)
2: *actor* ← Initialize the actor network
3: *critic* ← Initialize the critic network
4: *env* ← Initialize the VCAV environment
5: **for** $i = 1, 2, \ldots, \phi_a$ **do**

Algorithm 2 *Cont.*

$\mathbf{a}_t = \left(v_1, v_2, \ldots, v_{|\mathbf{m}_t|}\right) \leftarrow actor(\mathbf{m}_t)$
$\mathbf{m}_{t+1} = \{(d, u, v)\} \cup \{\lambda_v^n\} \cup \left\{\lambda_v^l\right\}, \mathbf{r}_{t+1} \leftarrow env(\mathbf{a}_t, \mathbf{m}_t)$
$\delta \leftarrow critic(\mathbf{m}_t, \mathbf{m}_{t+1}, \mathbf{r}_{t+1})$
Use δ to update the actor and critic networks
end for
return *actor*
end function

6. Evaluation

We evaluate the service disruption cost and computation time of our proposed solution approaches for traffic engineering in a failure scenario of a VCAV system. We used the optimal solution obtained by TERO as a baseline solution for evaluating the approximation solution achieved by TERH and TERA.

6.1. Scenarios and Parameters Setting

Our objective is to evaluate the performance of TERO, TERH and TERA with respect to the service disruption cost and computation time when we consider various network topologies. The three main evaluation questions are as follows: What is the gap between the optimal results and approximation solutions? How do different solution approaches respond to the dynamics of failure scenarios? Can TERH and TERA efficiently provide a VNF placement and routing solution in a large-scale scenario when a failure occurs? We use eight topologies in our evaluation. Note that it is the diversity and size of topologies that affect the answer to our questions rather than a specific topology. The first topology, referred to as Abilene, is the US backbone network composed of 12 nodes and 15 links, described in the Abilene dataset [27]. The second topology, namely Geant, is the Europe backbone network of 22 nodes and 36 links, presented in the Geant dataset [28]. The other topologies are synthetic topologies based on random graph generation algorithms, including the Barabási-Albert (BA), Waxman (WA), Erdős-Rényi (ER) models [29]. We create a small topology composed of 50 nodes and a large topology composed of 200 nodes for each random graph generation algorithm. The random graph generation tool is FNSS [30]. A BA topology is created with four nodes at first. A new node is added by connecting to four preceding nodes. The link density probability used to create a WA topology is 0.9. The edge generation probability used to create an ER topology is 0.2. We denote the small and large BA topologies by BA1 and BA2, the small and large WA topologies by WA1 and WA2, and the small and large ER topologies by ER1 and ER2. In a failure scenario, we randomly generate one node and link failure in a network topology.

We randomly create 15 demands in the Abilene and Geant topologies and 100 service demands in the BA, WA, ER topologies. The arrival and departure nodes of a service demand are randomly selected. The SFC delay is varied between one and thirty milliseconds. The range of the bandwidth demand is between 1 Gbps and 5 Gbps. We consider four types of VNFs. The number of CPU cores demanded by a VNF type for one volume of traffic is varied between one and two cores. The SFC of a service demand is randomly selected in four VNF types. We assign a bandwidth value of 80 Gbps to the capacity of all links. The edge and cloud nodes are randomly selected. The cloud node capacity is 200 cores. The edge node capacity is 50 cores. At a node, the processing delay of a VNF and the routing delay for a traffic unit is randomly generated between 10 and 100 microseconds. The value of link weight is varied between 1 and 3.

We now look at how to choose hyperparameters for the implementation of our proposed algorithms. In TERA, the temperature hyperparameter is automatically configured as described in [31]. We chose two layers for the actor and critic networks because we did not obtain a better policy when the number of layers increases beyond two. After running TERA with a varying number of neurons, we chose 32 neurons for each layer of the actor

and critic networks because the policy did not significantly improve when we used a bigger value.

In TERH, the value of the end temperature is 0.1. For each temperature, the number of neighbor selections is $\phi = 100$. For comparison purposes, we select the parameter τ of the cooling function so that the iteration number of TERH and that of TERA is similar. The parameter τ is computed as follows:

$$\tau = \left(\frac{T_n}{T_0}\right)^{\frac{1}{\phi_a}}, \tag{28}$$

where $\phi_a = 8000$ since TERA can obtain a steady policy after eight thousand iterations.

We used an x86 computer in our evaluation. Its hardware configuration is a four-core 2.60 GHz Intel processor with 8 GB memory and an NVIDIA GeForce GTX 850M card. We solved TERO in CPLEX [32]. We implemented TERH in Java and TERA in Python with TensorFlow [33].

6.2. Evaluation Results

First, we compare the performance of different solution approaches when a failure scenario is fixed. In a fixed failure scenario, we compute the service disruption cost and computation time in only one failure scenario. We consider limited-size scenarios, including the Abilene, Geant, BA1, WA1, and ER1 topologies, to compare approximation solutions with optimal results. Figure 2a shows that TERO is better than TERH and TERA in terms of the service disruption cost, but the difference is marginal. We also observe that TERH and TERA can archive similar service disruption costs after 8000 iterations. In Figure 2b, the computation time of TERA and TERH is higher than that of TERO, and the computation time of TERA is slightly higher than that of TERH.

Second, we compare the performance of different solution approaches when a failure scenario is changed. Specifically, we consider 8000 failure scenarios in our evaluation. We compute the service disruption cost and computation time in each failure scenario and plot their average value. Figure 3a shows that TERO, TERH, and TERA archive similar service disruption costs. In Figure 3b, we use a base 10 logarithmic scale for the y-axis and a linear scale for the x-axis to illustrate a variation in the computation time of TERO, TERH, and TERA. The figure shows that the computation time of TERA is significantly smaller than that of TERO and TERH. It is because TERA can remember its policy learned from previous data while TERO and TERH are required to solve the TER problem for an individual failure scenario.

Finally, we evaluate the TERH and TERA performance in a large-scale VCAV system when a failure scenario is changed. Figure 4 plots the service disruption cost and computation time for the BA2, WA2, and ER2 topologies with 200 nodes. In such large-scale topologies, CPLEX cannot solve the TERO model to find the optimal solution. In Figure 4b, we use a base 10 logarithmic scale for the y-axis and a linear scale for the x-axis to plot the computation time. We observe that TERA is significantly faster than TERH. The service disruption cost of TERH is slightly smaller than that of TERA, but it is negligible. It suggests that we should use TERA to protect service demands from a failure in a real-time VCAV system.

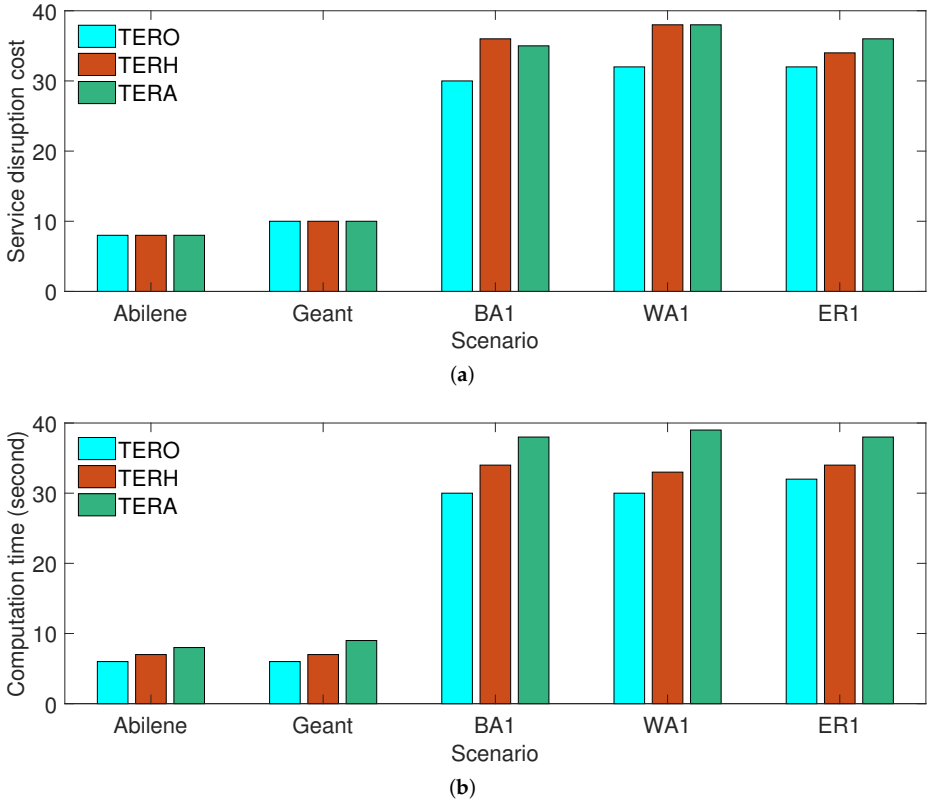

Figure 2. Performance comparison of TERO, TERH and TERA when a failure scenario is fixed. (**a**) Service disruption cost; (**b**) Computation time.

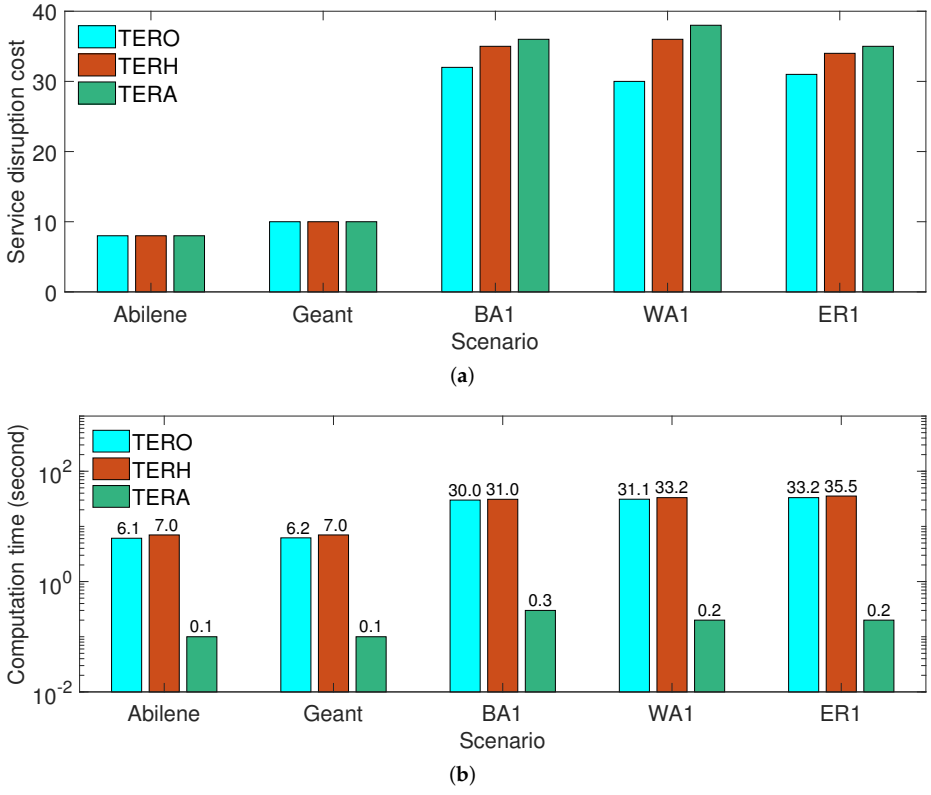

Figure 3. Performance comparison of TERO, TERH and TERA when a failure is changed. (**a**) Service disruption cost; (**b**) Computation time.

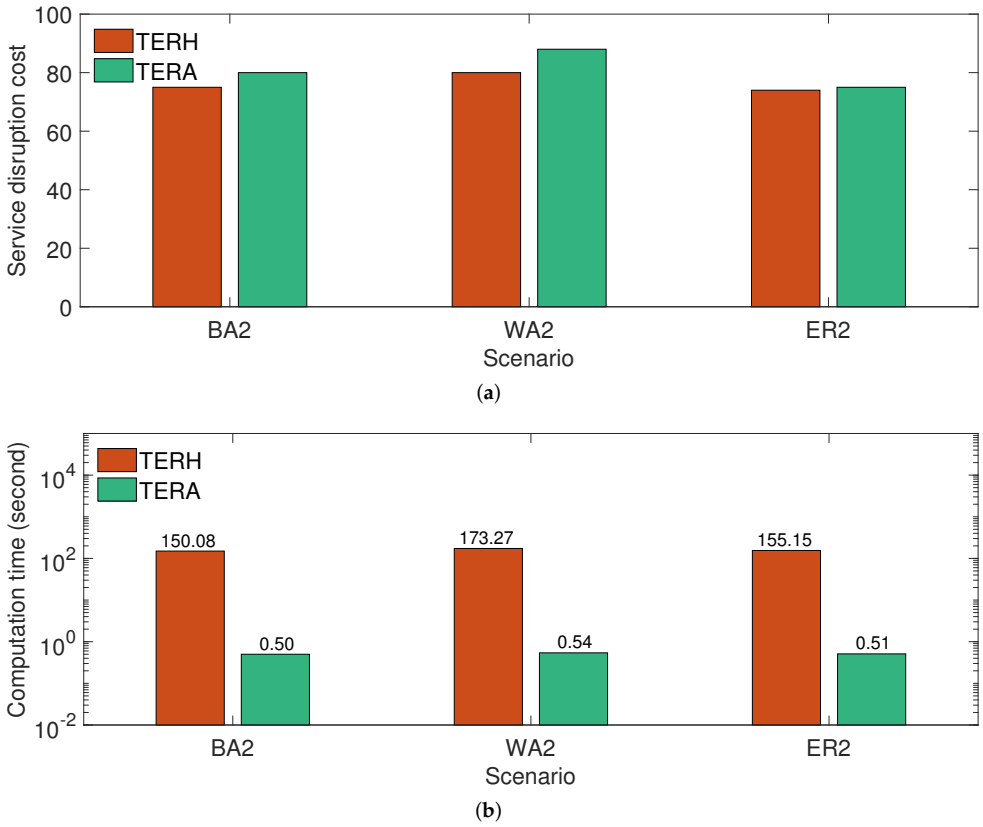

Figure 4. The TERA and TERH performance in a large-scale VCAV system. (**a**) Service disruption cost; (**b**) Computation time.

7. Conclusions

We studied the optimization problem of traffic engineering for resilient services in a VCAV system. We proposed an ILP model (i.e., TERO) to find the optimal VNF placement and routing when a node or link failure occurs. The model captures essential features of NFV such as service function chaining, the restriction rule of resource reallocation, and the exact placement and routing solution for the service demand set. We developed the TERH and TERA approximation algorithms based on heuristics and reinforcement learning to provide an efficient traffic engineering solution for resilient services in a large-scale VCAV system. The evaluation results show that TERO, TERH, and TERA can protect service demands from node and link failures. The approximation results provided by TERH and TERA are very close to the optimal results. The results also suggest that a network service provider should consider TERA to provide resilient services in a real-time VCAV system. Possible directions for extending our work comprise the consideration of various network technologies supporting a VCAV system, an evaluation of other network topologies and performance metrics, or an optimization model of a resilient service with a federation of several VCAV providers as in [9,34].

Author Contributions: Conceptualization, T.-M.P.; methodology, T.-M.P.; software, T.-M.P.; validation, T.-M.P. and T.-M.N.; formal analysis, T.-M.P.; investigation, T.-M.P.; writing—original draft preparation, T.-M.P. and T.-M.N.; writing—review and editing, T.-M.P. and T.-M.N.; supervision, T.-M.P.; project administration, T.-M.P.; funding acquisition, T.-M.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.02-2020.13.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: The authors are sincerely grateful to the anonymous reviewers for many constructive helpful comments.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Lv, Z.; Xiu, W. Interaction of Edge-Cloud Computing Based on SDN and NFV for Next Generation IoT. *IEEE Internet Things J.* **2020**, *7*, 5706–5712. [CrossRef]
2. ETSI. *Network Functions Virtualisation (NFV): Architectural Framework, GS NFV 002 V1.2.1*; ETSI: Sophia Antipolis, France, 2014.
3. ETSI. *Network Functions Virtualisation (NFV): Resiliency Requirements, GS NFV-REL 001 V1.1.1*; ETSI: Sophia Antipolis, France, 2015.
4. Marotta, A.; Kassler, A. A Power Efficient and Robust Virtual Network Functions Placement Problem. In Proceedings of the 28th International Teletraffic Congress (ITC 28), Würzburg, Germany, 12–16 September 2016; Volume 1, pp. 331–339.
5. Marotta, A.; Zola, E.; D'Andreagiovanni, F.; Kassler, A. A Fast Robust Optimization-based Heuristic for the Deployment of Green Virtual Network Functions. *J. Netw. Comput. Appl.* **2017**, *95*, 42–53. [CrossRef]
6. Hmaity, A.; Savi, M.; Musumeci, F.; Tornatore, M.; Pattavina, A. Protection strategies for virtual network functions placement and service chains provisioning. *Networks* **2017**, *70*, 373–387. [CrossRef]
7. Wen, R.; Feng, G.; Tang, J.; Quek, T.Q.S.; Wang, G.; Tan, W.; Qin, S. On Robustness of Network Slicing for Next-Generation Mobile Networks. *IEEE Trans. Commun.* **2019**, *67*, 430–444. [CrossRef]
8. Bian, S.; Huang, X.; Shao, Z.; Gao, X.; Yang, Y. Service Chain Composition with Failures in NFV Systems: A Game-Theoretic Perspective. In Proceedings of the IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6.
9. Pham, T.M.; Fdida, S.; Nguyen, T.T.L.; Chu, H.N. Modeling and analysis of robust service composition for network functions virtualization. *Comput. Netw.* **2020**, *166*, 106989. [CrossRef]
10. Pham, T.M.; Nguyen, T.T.L. Optimization of Resource Management for NFV-Enabled IoT Systems in Edge Cloud Computing. *IEEE Access* **2020**, *8*, 178217–178229. [CrossRef]
11. Wang, Y.; Wang, H.; Mahimkar, A.; Alimi, R.; Zhang, Y.; Qiu, L.; Yang, Y.R. R3: Resilient Routing Reconfiguration. *SIGCOMM Comput. Commun. Rev.* **2010**, *40*, 291–302. [CrossRef]
12. Cho, S.; Elhourani, T.; Ramasubramanian, S. Independent Directed Acyclic Graphs for Resilient Multipath Routing. *IEEE/ACM Trans. Netw.* **2012**, *20*, 153–162. [CrossRef]
13. Yang, B.; Liu, J.; Shenker, S.; Li, J.; Zheng, K. Keep Forwarding: Towards k-link failure resilient routing. In Proceedings of the IEEE Conference on Computer Communications (INFOCOM 2014), Toronto, ON, Canada, 27 April–2 May 2014; pp. 1617–1625.
14. Power, A.; Kotonya, G. Providing Fault Tolerance via Complex Event Processing and Machine Learning for IoT Systems. In Proceedings of the 9th International Conference on the Internet of Things, Bilbao, Spain, 22–25 October 2019; ACM: New York, NY, USA, 2019.
15. Wang, S.; Yuan, J.; Li, X.; Qian, Z.; Arena, F.; You, I. Active Data Replica Recovery for Quality-Assurance Big Data Analysis in IC-IoT. *IEEE Access* **2019**, *7*, 106997–107005. [CrossRef]
16. Goudarzi, M.; Wu, H.; Palaniswami, M.; Buyya, R. An Application Placement Technique for Concurrent IoT Applications in Edge and Fog Computing Environments. *IEEE Trans. Mob. Comput.* **2021**, *20*, 1298–1311. [CrossRef]
17. Huang, H.; Guo, S. Proactive Failure Recovery for NFV in Distributed Edge Computing. *IEEE Commun. Mag.* **2019**, *57*, 131–137. [CrossRef]
18. Ergenc, D.; Rak, J.; Fischer, M. Service-Based Resilience for Embedded IoT Networks. In Proceedings of the 2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Valencia, Spain, 29 June–2 July 2020; pp. 540–551. [CrossRef]
19. Bakhshi Kiadehi, K.; Rahmani, A.M.; TSabbagh Molahosseini, A. A fault-tolerant architecture for internet-of-things based on software-defined networks. *Telecommun. Syst.* **2021**, *77*, 1572–9451. [CrossRef]
20. Sanabria-Russo, L.; Serra, J.; Pubill, D.; Verikoukis, C. CURATE: On-Demand Orchestration of Services for Health Emergencies Prediction and Mitigation. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 438–445. [CrossRef]
21. Luo, Q.; Hu, S.; Li, C.; Li, G.; Shi, W. Resource Scheduling in Edge Computing: A Survey. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 2131–2165. [CrossRef]
22. Si, W.; Starobinski, D.; Laifenfeld, M. A Robust Load Balancing and Routing Protocol for Intra-Car Hybrid Wired/Wireless Networks. *IEEE Trans. Mob. Comput.* **2019**, *18*, 250–263. [CrossRef]
23. Malik, F.M.; Khattak, H.A.; Almogren, A.; Bouachir, O.; Din, I.U.; Altameem, A. Performance Evaluation of Data Dissemination Protocols for Connected Autonomous Vehicles. *IEEE Access* **2020**, *8*, 126896–126906. [CrossRef]
24. Muhammad, A.; Saqib, M.; Song, W.C. Sensor Virtualization and Data Orchestration in Internet of Vehicles (IoV). In Proceedings of the 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM), Bordeaux, France, 17–21 May 2021; pp. 998–1003.
25. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by Simulated Annealing. *Science* **1983**, *220*, 671–680. [CrossRef]

26. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Proceedings of the 35th International Conference on Machine Learning, Stockholm Sweden, 10–15 July 2018; Volume 80, pp. 1856–1865.
27. Pióro, M.; Wessäly, R. SNDlib. Available online: http://sndlib.zib.de (accessed on 8 December 2021).
28. Uhlig, S.; Quoitin, B.; Lepropre, J.; Balon, S. Providing public intradomain traffic matrices to the research community. *SIGCOMM Comput. Commun. Rev.* **2006**, *36*, 83–86. [CrossRef]
29. Drobyshevskiy, M.; Turdakov, D. Random Graph Modeling: A Survey of the Concepts. *ACM Comput. Surv.* **2019**, *52*, 131. [CrossRef]
30. Saino, L.; Cocora, C.; Pavlou, G. A Toolchain for Simplifying Network Simulation Setup. In Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques (SIMUTOOLS '13), Cannes, France, 5–7 March 2013; ICST: Brussels, Belgium, 2013.
31. Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. Soft Actor-Critic Algorithms and Applications. *arXiv* **2019**, arXiv:1801.01290.
32. IBM. IBM ILOG CPLEX Optimizer. Available online: https://www.ibm.com/analytics/cplex-optimizer/ (accessed on 8 December 2021).
33. Google Brain Team. TensorFlow. Available online: https://www.tensorflow.org (accessed on 8 December 2021).
34. Pham, T.M.; Chu, H.N. Multi-Provider and Multi-Domain Resource Orchestration in Network Functions Virtualization. *IEEE Access* **2019**, *7*, 86920–86931. [CrossRef]