Performance Modeling and Joint Resource Allocation Algorithms for Online Virtual Network Embedding

Qiao Lu and Changcheng Huang, *Senior Member, IEEE*

***Abstract*—Network Virtualization (NV) has been proposed as an enabling technology of a key value-added service for service providers. While there are a very large number of publications that have proposed various resource allocation algorithms for NV, no effort has been made to estimate the performance of virtual network embedding (VNE) algorithms based on analytical models. In this paper, to assess the blocking probability of virtual link mapping, we propose a novel loss network model with Dynamic Routing And Random Topology (DRART). Moreover, by combining with some existing models, we can estimate the blocking probability for VNE through a creative recursive process. Our model can provide a benchmark for various VNE algorithms. To fill the performance gap between existing resource allocation algorithms and our analytical model, we also propose a distributed Genetic Algorithm (GA) based resource allocation approach that can jointly allocate node and link resources. Our simulation results show that our resource allocation approach can achieve the performance as predicted by our analytical model while meeting stringent online resource allocation requirements.**

***Index Terms*—Network virtualization, virtual network embedding, loss network model, analytical performance estimation.**

I. Introduction

IN A VIRTUALIZED network architecture, traditional Internet service providers are decoupled into service providers and infrastructure providers. Service providers are responsible for deploying and offering customized network services to end users, while infrastructure providers are in charge of maintaining and allocating physical resources to different service providers. Service providers generally deploy and maintain customized end-to-end network services by leasing from the substrate resources of multiple infrastructure providers.

The application of NV leads to the question of how virtualized resources should be realized by underlying infrastructure providers. In order to rationally share the resources of the infrastructure providers and maximize the revenues of service providers, recent research work has made a great effort to find effective solutions of resource allocation problems in NV. The general virtual network allocation problem is also known as the virtual network embedding (VNE) problem. To allocate a virtual network request (VNR) successfully, each virtual node should be mapped into a substrate node where the specific node mapping constraints can be fulfilled, and each virtual link is supposed to be mapped into a substrate path where the specific link mapping constraints have to be satisfied.

VNE problems can be considered as online or offline problems. The offline problems [1] handle a set of static VNRs as a one-time requirement, which is relatively easy to solve. Unlike offline VNE, VNRs in online VNE problems [2], [3], [4] arrive dynamically and stay in the network for a random duration. The infrastructure provider does not know the VNR information, such as arrival time, duration, and topologies before the VNR arrives. In most real-life scenarios, VNE has to be addressed as an online problem that requires an agile and efficient allocation solution.

On the other hand, to our best knowledge, study on the performance estimation of these VNE algorithms based on analytical models does not exist. These algorithms have been evaluated based on simulation results. It should be noted that network topologies in real applications have been evolving fast in the past decades, from traditional carrier networks, to data center networks, and satellite networks. It is hard to find a topology that can represent all applications. Random topology was first adopted in [5] as a way to assess the performances of VNE algorithms through simulation. All topologies including NSF network and structured topologies such as linear and star can be considered as realizations of random topology. By using random topology, performances are averaged over all topologies, making results more objective and comparable. Since [5], random topology has become popular for VNE performance evaluations [5], [6]. However, all these evaluations are still simulation based. There are two major problems with simulation-based solutions. One is that they are highly dependent on the embedding algorithms used and nearly all of the embedding algorithms can be trapped in some sub-optimal solutions that may vary with the approaches used (coordinated vs. uncoordinated), initial candidate selections and the setting of some hyperparameters. The other problem with simulation-based approach is the random errors inherent in the Monte-Carlo method [7]. Warm-up periods can also cause bias. In view of generality and universality, an analytical model is required to serve as a reference

Manuscript received 19 February 2023; revised 16 May 2023; accepted 18 July 2023. Date of publication 28 August 2023; date of current version 7 February 2024. The associate editor coordinating the review of this article and approving it for publication was J.-F. Botero. *(Corresponding author: Qiao Lu.)*

The authors are with the Department of Systems and Computer Engineering, Carleton University, Ottawa, ON K1S 5B6, Canada (e-mail: qiaolu@sce.carleton.ca; huang@sce.carleton.ca).

Digital Object Identifier 10.1109/TNSM.2023.3308065

for comparing different VNE algorithms under random topology.

Recently, VNE is also motivated by some other state-of-the-art research, such as unmanned aerial vehicles (UAVs) [8] and data center migration [9]. UAVs play an important role in supporting 5G and beyond 5G mobile networks. VNE has the potential to effectively manage and enhance UAVs for the upcoming generation of mobile networks by addressing manual, time-consuming UAV network management. One of the most challenging tasks in UAV network management is called UAV dynamic deployment [10]. As its name indicated, the resource allocation problems in the UAV environment highly depend on the status of the UAV trajectory. Therefore, allocating the resource dynamically and assessing different allocation algorithms become difficult issues in this case.

Data center migration is another area where VNE can be used. Migration strategies have been a hot research topic for achieving high availability. Managing the substrate resource workload becomes an essential aspect in the cloud computing environment [11]. The effectiveness of these embedding strategies highly depends on the performance of embedding algorithms. On the other hand, to satisfy a data center's requirement for high availability, computer and communication failures have to be tackled fast and efficiently. These random and unsuspected failures require the migration algorithms to adapt to random and dynamic substrate topology.

Developing an analytical loss network model for VNE is challenging since a blocking event (*i.e.,* a VNE failure) may happen at different levels. A blocking event may happen at the substrate nodes/links level, virtual nodes/links level, and even at the virtual network level. Furthermore, to map a virtual link, there may exist multiple substrate paths with different source-destination pairs in a substrate network. It is important to select a path that can achieve load balancing and minimize blocking. Allocating a virtual link becomes a dynamic routing problem in random topology. To our best knowledge, there is no loss network model incorporating the dynamic routing and random topology requirements so far.

A mathematical model for estimating the VNE performance under dynamic and random topology becomes both tough and promising in light of the aforementioned issues. In this paper, we propose a novel loss network model that can predict the blocking probability of a virtual link with dynamic routing and random topology. By integrating with other existing models through an innovative recursive process, we can predict the performance of VNE with high accuracy. The proposed loss network model can be considered a performance benchmark for various VNE algorithms. We also propose an online coordinated VNE solution using a Genetic Algorithm (GA), which aims to map virtual nodes and links jointly. As our simulation results will show our embedding solution can approach the performance as predicted by our analytical model better than existing representative VNE algorithms.

We summarize our contributions as follows:

- We create a novel loss network model with Dynamic Routing And Random Topology (DRART) to evaluate the blocking probability at the virtual link level;
- We propose a new recursive procedure that integrates different levels for estimating the blocking probability of VNE;
- To fill in the gap between the performance predicted by our analytical model and the performances achieved by various existing VNE algorithms, we develop a Genetic Algorithm (GA) that can allocate virtual nodes and links jointly. For the identification of infeasible mapping, we design a new mechanism based on a two-color graph coloring theory.

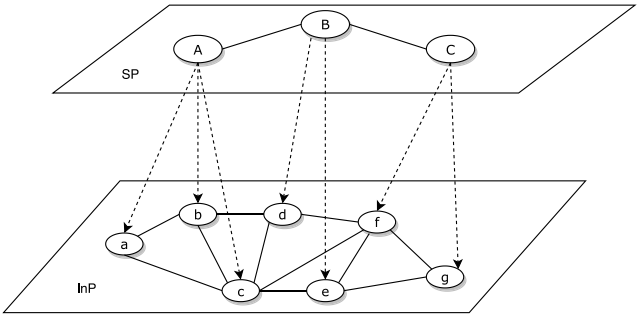

Fig. 1. Mapping a VNR into an SN.

II. Related Work and Background

In this section, we first introduce the related work of VNE and loss network models. Then we give a brief introduction to Genetic Algorithm (GA). Finally, we summarize the problems that have not been solved by current research.

A. Virtual Network Embedding (VNE)

In a VNE problem, a substrate network (SN) and a virtual network (VN) can be any topology such as lines, rings, trees, meshes, etc. In this paper, we consider substrate links and virtual links to be bidirectional. An example of VNE is shown in Fig. 1. InP stands for the infrastructure provider. The VNR as shown in Fig. 1 consists of three virtual nodes (A, B, and C) and two virtual links (A-B, B-C). The dashed lines indicate possible virtual node mapping solutions. Each virtual link should be mapped into a substrate path between the node mapping solutions.

There are four types of resource allocation optimization strategies proposed for NV in recent years. *Exact solutions* formulate VNE problems as ILP or MILP, bringing an exponential run-time cost [12]. The exact solution proposed in [5] aimed to save embedding cost by using the MILP model. The paper [13] developed a multi-objective VNE model by the means of a MILP formulation. This approach modified the previous embedding MILP in [5] to provide an adaptive VNE solution to deal with virtual node and link failures or degradations. Although recently proposed exact solutions were claimed to be the optimal solutions as a baseline for heuristic solutions, these solutions allocated the VNRs based on current residual resources without considering future VNRs. For

such reason, the exact solutions hardly guarantee the global optimal in the long run. This observation provides more space for heuristic algorithms to play.

Heuristic solutions aim to increase placement efficiency as well as improve global optimality. However, most of them focus on specific scenarios [14], thus heuristic solutions are hard to solve other similar problems. Generally, two-stage heuristic VNE algorithms [15] involve a greedy node embedding and a subsequent *k*-shortest path link mapping [16], [17], or a ranking-based node embedding and again the *k*-shortest path link mapping [18].

Metaheuristic solutions have been proposed recently to improve the quality of the performance by escaping from local optima as well to complete embedding in a reasonable time. Previous research [19], [20] allowed improving the quality of candidates to find near-optimal solutions. Therefore, it is indispensable to tailor effective metaheuristic mechanisms to enhance performance.

Reinforcement learning (RL) solutions adopt Markov Decision Process as models to make decisions in the process of interaction with the environment. Many research papers [11], [21], discussing RL solutions, improved the mapping performance. However, it is challenging to apply RL to an online resource allocation scenario. Specifically, training in RL is parametric-based, which is usually implemented in a stationary environment. In dynamic environment, RL requires training and building a model for each specific scenario. As a result, RL has to handle a large number of models. Therefore, the research of RL is confronted with computational complexity problems.

Since VNE approaches handle the allocation for both virtual nodes and virtual links, VNE could be separated into two phases: mapping virtual nodes into substrate nodes and mapping virtual links into substrate paths with multiple connected substrate links. Uncoordinated VNE solves node mapping and link mapping separately in an independent way. This decomposition approach can simplify the algorithmic complexity. However, the complexity is still $\mathcal{NP}$-hard. Moreover, Uncoordinated VNE approaches [16], [22] might result in the situation that virtual neighboring nodes are mapped onto the substrate nodes that are probably far from each other. The long distance between substrate nodes may lead to unexpected longer substrate paths and network fragmentation.

On the contrary, coordinated VNE [23], [24] considered the coordination between node mapping and link mapping. Coordinated VNE has two categories: partial coordinated VNE and fully coordinated VNE. Partial coordinated VNE can be performed to map virtual nodes with the consideration of the relation on link mapping in two separate stages. Alternatively, full coordination of VNE can be achieved by solving the node mapping and link mapping at the same time [25]. Most of the previous research worked on partially coordinated solutions. Even in some papers [26], [27], they claimed virtual nodes and links were mapped in the same stage. However, [26] and [27] solved virtual nodes and links alternately, which were not fully coordinated solutions.

B. Loss Network Model

Loss network models are stochastic models, which study the blocking behavior in all kinds of networks [28]. In this section, we give a brief introduction to loss network models.

1) Generalized Erlang Loss Model: The original loss model was first proposed by Erlang with an M/M/S queue. Erlang loss model describes the blocking probability in a system with S homogeneous servers where request arrivals obey Poisson distribution with mean λ and exponentially distributed holding times with mean $1/\mu$. In the generalized Erlang loss model, requests can have a set R of classes. A class r request requires to hold A_r servers simultaneously with mean holding time $1/\mu_r$ and arrives with rate λ_r. Assuming the number of class r requests is n_r, we define $\vec{n} = \{n_r : r \in R\}$. The set of feasible states $\mathcal{F}$ will be

$$\mathcal{F}(S) = \left\{ \vec{n} \geq 0 : \sum_{r \in R} A_r n_r \leq S \right\} \tag{1}$$

The distribution of busy servers with multi-class requests will be

$$P_g(\vec{n}) = \frac{1}{G(S)} \prod_{r \in R} \frac{\left(\frac{\lambda_r}{\mu_r}\right)^{n_r}}{n_r!}, \tag{2}$$

where

$$G(S) = \sum_{\vec{n} \in \mathcal{F}(S)} \prod_{r \in R} \frac{\left(\frac{\lambda_r}{\mu_r}\right)^{n_r}}{n_r!}. \tag{3}$$

2) Open Loss Network With Fixed Routing: The generalized Erlang loss model can be extended to the generalized loss station with multiple server types. Assume there are S_l servers for server type l, where $l \in \mathcal{L}$. A class r request requires A_{lr} servers simultaneously, for all $l \in \mathcal{L}$. Therefore, the following server constraint has to be met:

$$\sum_{r \in R} A_{lr} n_r \leq S_l, \text{ for all } l \in \mathcal{L}. \tag{4}$$

In an open loss network, the customer requests arrive from and depart to the external environment [29]. An open loss network with fixed routing is equivalent to a generalized loss station with multiple server types. Specifically, each link in the open loss network corresponds to a server type in the generalized loss station. Therefore, the blocking probability of an open loss network could be obtained from (2).

3) Jackson Network: Jackson network is a special class of an open queuing network [30]. In a Jackson network, there are S servers with infinite queues. Request arrival follows the Poisson process and the service time follows the exponential distribution. The state of the network can be defined as the queue size of each server with a set of S-tuples: $(r_1, r_2, \ldots, r_S)$. In a steady state, the states of individual queues are independent. Therefore, the network state could be expressed in a product form:

$$P(r_1, r_2, \ldots, r_S) = \prod_{i=1}^{S} P(r_i) \tag{5}$$

C. Genetic Algorithm

Genetic Algorithm (GA) consists of four operations: initialization, selection, crossover and mutation [31]. In general, GA begins with a population that was generated at random [31].

When it comes to the GA algorithms in VNE, research [32], [33] paid more attention to node mapping. A GA approach for link mapping is far more complicated than node mapping since the virtual link should be mapped into a substrate path that consists of several connected substrate links. Previous research [34] proposed a resource allocation approach (MM-GAPS) based on GA. MM-GAPS solved the link mapping stage with a splittable solution and used GA to determine the splitting ratio of each virtual link. This novel idea aimed to reduce link congestion by updating the splitting ratio. The GA operations have to ensure each virtual link solution is a valid and connected path. One GA approach called Path-Based Genetic Algorithm (PBGA) on link mapping was proposed in [35]. With PBGA, all the GA operations were based on substrate paths. It dramatically reduced the execution time with similar performance on acceptance ratio with the baseline algorithm [5].

D. Problem Statement

There is little research on the analytical performance model for VNE. When it comes to the loss network models with routing, Barry and Humblet [36] investigated the blocking probability with the effects of path length, switch size and interference length in the optical networks. The closest related work by Antunes et al. [37] designed a loss network with fixed routing. However, a loss network model with fixed routing cannot be applied to scenarios with dynamic routing and random topology. Therefore, it is necessary to investigate an analytical loss network solution for the online VNR allocation in a dynamic routing scenario with arbitrary topology.

On the other hand, we found there is little research working on a fully coordinated solution of VNE. As we discussed above, uncoordinated solutions and partial coordinated solutions narrow the searching space of feasible solutions and may miss the global optimum. To match our analytical model, we propose a GA approach that can achieve a fully coordinated solution. We apply the two-color graph coloring theory to solve the mapping conflicts caused by the joint allocation of virtual nodes and virtual links.

III. Network Model

In this section, we elaborate on a general network model used in this paper. Generally, in VNE problems, the input consists of a variable number of VNRs, whereas the SN provides the physical resources in terms of bandwidth and CPU capacity. Appendix C gives a list of notations.

A. Substrate Network (SN)

An SN is represented as a random weighted undirected graph and denoted as $G_s(N_s, E_s)$, where N_s is a set of substrate nodes and $\widetilde{n}_s = |N_s|$ follows a probability distribution P_{N_s} with a maximum value $M_{s,n}$. $\bar{n}_s = \mathbb{E}(\widetilde{n}_s)$ is the average number of substrate nodes. E_s is a set of substrate links and each pair of nodes have the probability $P_{s,l}$ to form a link $e_s \in E_s$. $\widetilde{e}_s = |E_s|$ is the number of substrate links. $\bar{e}_s = \mathbb{E}(\widetilde{e}_s)$ is the average number of substrate links. Each substrate node $n_s \in N_s$ is associated with a CPU capacity value $c(n_s)$ that follows a random distribution $P_{c(n_s)}$ with a maximum CPU capacity C_{n_s} and a location (x_{n_s}, y_{n_s}) following some distribution $P_{L_{n_s}}(x, y)$ in a fixed area *W*. We assume that each substrate link e_s between two substrate nodes has a random bandwidth capacity $b(e_s)$ following the distribution $P_{b(e_s)}$ with a maximum bandwidth capacity B_{e_s}. We also define a substrate path as a set of acyclic substrate links that are connected sequentially. In addition, we use $P^s(m_s, n_s)$ to represent a set of all substrate paths from node m_s to n_s.

B. Virtual Network (VN)

Conventionally, a VN consists of a set of dedicated network service boxes such as firewalls, load balancers and application delivery controllers that are concatenated together to support a specific application [38].

A VNR is represented as a random weighted graph, denoted by $G_v(N_v, E_v, t_a, t_d, D)$. N_v is a set of virtual nodes. $\widetilde{n}_v = |N_v|$ follows a probability distribution P_{N_v} with a maximum value $M_{v,n}$. $\bar{n}_v = \mathbb{E}(\widetilde{n}_v)$ is the average number of virtual nodes. The distance between a substrate node and the virtual node is denoted by $dis(loc(n_v), loc(n_s))$. In practice, there are typically some limitations on which substrate nodes a virtual node can map onto [5]. When we set the distance to very large or infinity, it becomes agnostic of locations. Therefore, the distance limitation makes the model more useful and general. E_v is a set of virtual links and each pair of nodes have the probability $P_{v,l}$ to form a virtual link $e_v \in E_v$. $\widetilde{e}_v = |E_v|$ is the number of virtual links in the VNR. $\bar{e}_v = \mathbb{E}(\widetilde{e}_v)$ is the average number of virtual links. t_a is the arrival time of a VNR, which follows a Poisson process with an arrival rate λ_v. t_d is the duration of the VNR, which follows an exponential distribution with a mean holding time τ_v. Each virtual node $n_v \in N_v$ is associated with a CPU capacity requirement $c(n_v)$ that follows a random distribution $P_{c(n_v)}$ with a maximum CPU capacity requirement C_{n_v} and a location (x_{n_v}, y_{n_v}) following some distribution $P_{L_{n_v}}(x, y)$ in the same fixed area *W*. *D* is a fixed number representing the maximum acceptable distance between a virtual node and its associated substrate nodes. Each virtual node can only be mapped to one substrate node located within the distance *D*, which is called a candidate node for the virtual node. We assume each virtual link e_v between two virtual nodes has a random bandwidth requirement $b(e_v)$ following the distribution $P_{b(e_v)}$ with a maximum bandwidth requirement B_{e_v}. Each virtual link can be mapped to a substrate path if the path connects the two substrate nodes onto which the two virtual nodes are mapped in the node mapping stage. The substrate path is called a candidate path of the virtual link mapping. The selection of a candidate node/path among available ones for a particular node/link mapping depends on the mapping algorithm used.

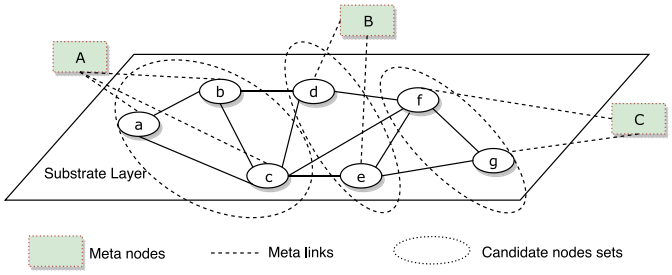

Fig. 2. The augmented graph of Fig 1.

C. Augmented Network

Inspired by [5], an augmented graph $G_{s'} = (N_{s'}, E_{s'})$ is created for the substrate network $G_s = (N_s, E_s)$. We generate a meta-node, called $\mu(n_v)$, for each $n_v \in N_v$. Then we connect all candidate substrate nodes of $n_v \in N_v$ to the meta-node through a bidirectional meta-link with infinite bandwidth. We set $N_{s'} = N_s \cup \{\mu(n_v) | n_v \in N_v\}$ and $E_{s'} = E_s \cup \{(\mu(n_v), n_s) | n_v \in N_v, n_s \in \mathbb{N}_c(n_v)\}$. $\mathbb{N}_c(n_v)$ denotes the set of all candidate nodes for n_v. Fig. 2 is the augmented graph created for Fig. 1. In order to solve the virtual node mapping and virtual link mapping coordinately, generating an augmented graph is the first step.

D. VNE Mapping Constraints

As we talked about in Section I, A VNR is mapped successfully when all the demand requirements of virtual nodes and virtual links are satisfied. In this paper, we use general constraints that have been applied to some previous research [5], [35]. The virtual node mapping constraints include the CPU capacity and a maximum distance radius D of the VNR. The virtual link mapping constraint focuses on the substrate link bandwidth.

E. Objective Function

In VNE problems, we use an objective function to measure if a VNE solution is an acceptable one. In our jointly mapping algorithm, we aim to develop a function to evaluate both node and link mappings together. The objective function should consider all resource usage of a VNR as a whole, as opposed to other heuristic mapping approaches [5] [16]. The heuristic mapping approaches sequentially consider the virtual nodes/links according to a ranking method. In (6), σ is a small positive constant to prevent the zero denominators. f_{uv}^i describes the total amount of flows from u to v for the ith virtual link under a specific mapping scenario. And x_{mw} denotes a binary variable, which has the value 1 if the meta link is activated shown in (8); Otherwise, it is set to 0. We use function $R^e()$ and $R^n()$ to indicate the current available bandwidth and CPU capacity resources on a substrate link/node, respectively. $0 \leq \alpha_{uv} \leq R^e(e_{uv})$ and $0 \leq \beta_w \leq R^n(w_s)$ are parameters to control the significance of load balancing during the placement. $\mathcal{M}()$ is a function that maps a virtual node or a virtual link to a substrate node or a substrate path. $\mathcal{M}(m_v) = w_s$ in (8) represents mapping a virtual node m_v to a substrate node w_s. We define the objective function F_{NLP} in this paper in (6).

$$min \ \ F_{NLP} = \sum_{e_{uv} \in E_S} \frac{\alpha_{uv}}{R^e(e_{uv}) - \sum_i f_{uv}^i + \sigma} + \sum_{w_s \in N_s} \frac{\beta_w}{R^n(w_s) - \sum_{m_v \in N_v} x_{mw} C(m_v) + \sigma} \tag{6}$$

where,

$$f_{uv}^i = B(e_v), \ \text{if} \ \ e_{uv} \in \mathcal{M}(e_v) \cup e_v \in E_v \tag{7}$$

$$x_{mw} = \begin{cases} 1, & \text{if } \mathcal{M}(m_v) = w_s \quad (8a) \\ 0, & \text{otherwise} \quad (8b) \end{cases}$$

Most of the previous solutions chose linear objective functions to accelerate the optimization procedure. In fact, in a GA algorithm, a non-linear objective function has similar computing complexity to a linear one. The paper [39] has shown that a non-linear objective function outperforms a linear one because a non-linear objective function is more sensible when the remaining resources are scarce. With a non-linear function, the value F_{NLP} grows more quickly than a linear one when network costs increase. Furthermore, Equation (6) is designed to restrain the mapping in case the residual resource is limited. A non-linear objective function, such as (6), improves resource utilization, since a small residual substrate resource may cause resource fragmentation. Moreover, a small residual substrate resource is hard to be utilized for future requests under the unsplittable mapping.

IV. Proposed Analytical Model

In this section, we propose a loss network model to estimate the blocking probability of VNE. We first describe our system models in three levels: the substrate node/link level, the virtual link level, and the VN level. Finally, we discuss the proposed recursive process for the estimation of average load that integrates all levels together. To simplify notations, we will omit some arguments of certain functions when the context is clear.

A. Substrate Node/Link Blocking Probability

We start our loss model from the calculation at the substrate node and substrate level. In this subsection, We describe substrate link blocking probability calculation. Substrate node blocking probability can be obtained exactly in the same way.

In most VNE simulation setups, VNRs arrive following the Poisson distribution. We consider the same scenario as other general VNE research. At the substrate link level, we can treat arrivals at a substrate link as random samples from the Poisson arrivals of the VN. Therefore, we assume that the arrivals at the substrate link level still obey Poisson distribution with a mean rate $\lambda_{s,l}$. We name $\lambda_{s,l}$ as the offered load for a substrate link. $\lambda_{s,l}$ is not equal to the VNR arrival rate λ_v, since a VNR is mapped into several substrate links instead of all substrate links. We will discuss our approach to estimate $\lambda_{s,l}$ in Section IV-E.

The holding time of a request received at a substrate link follows the same exponential distribution with the same mean holding time τ_v as the VNR arrival.

There is no existing way to calculate the blocking probability when $b(e_v)$ is a random real number. We found that the generalized Erlang loss model [40] is the closest one that can be used to approximate the substrate blocking probability. The generalized Erlang model requires the capacity of servers and the demand of requests to be discrete as mentioned in Section II-B1. Therefore, we need to quantize the bandwidth demand into a fixed number of classes denoted as R. In this paper, we define the interval size of each class as a constant number κ. Then the demand of bandwidth can be treated as the different classes of requests in the generalized Erlang model. We approximate each class r: $(b(e_v, r), b(e_v, r+\kappa)], 1 \leq r \leq R$ with one bandwidth request $b_r = \int_{b(e_v,r)}^{b(e_v,r+\kappa)} b(e_v) dP_{b(e_v)}$, which is the average bandwidth of the class. The probability that a request has bandwidth b_r is $P_r = \int_{b(e_v,r)}^{b(e_v,r+\kappa)} dP_{b(e_v)}$. The arrival rate within each class is $\lambda_{s,l,r} = \lambda_{s,l} P_r$. The load of each class is $\rho_{s,l,r} = \lambda_{s,l,r} \tau_v$. We also divide the bandwidth capacity of a substrate link $b(e_s)$ into U classes with each class u:$(b(e_s, u), b(e_s, u+\kappa)], 1 \leq u \leq U$. Then the bandwidth capacity of substrate links could be considered multiple classes of servers in the generalized Erlang model. We approximate each class with one bandwidth capacity $b_u = \int_{b(e_s,u)}^{b(e_s,u+\kappa)} b(e_s) dP_{b(e_s)}$. The probability that a substrate link has the capacity b_u is $P_u = \int_{b(e_s,u)}^{b(e_s,u+\kappa)} dP_{b(e_s)}$.

At a specific time, a substrate link hosts h_r requests with bandwidth b_r. We define $\vec{h} = (h_1, \ldots, h_R)$ as a state that includes all the number of different class requests. The set of all feasible states that can be carried by a specific substrate link with bandwidth capacity b_u is shown in (9).

$$\mathcal{F}(b_u) = \left\{ \vec{h} \geq 0 \colon \sum_{r \in R} b_r h_r \leq b_u \right\} \tag{9}$$

According to the generalized Erlang loss model [40], the average blocking probability of the substrate link is:

$$P_{s,l}^{(B)} = \sum_{u \in U} P_u \sum_{r \in R} \left(1 - \frac{G(b_u - b_r)}{G(b_u)} \right) P_r, \tag{10}$$

where

$$G(b_u) = \sum_{\vec{h} \in \mathcal{F}(b_u)} \prod_{r \in R} \frac{\rho_{s,l,r}^{h_r}}{h_r!}. \tag{11}$$

Similarly, we can obtain the substrate node blocking probability $P_{s,n}^{(B)}$ by replacing bandwidth with CPU capacity.

B. The DRART Model for Virtual Link Blocking Probability

In this subsection, we propose a model to estimate the performance at the virtual link level. A virtual link mapping is to find a substrate path that has enough remaining bandwidth capacity at each substrate link to support the virtual link bandwidth requirement. The virtual link mapping fails only when all potential substrate paths are unavailable. In order to determine the blocking probability of a virtual link, we have to solve sub-problems as follows:

- What is the probability that the specifically ordered substrate nodes are connected as a path? In other words, what is the probability that one path exists in a random substrate network?
- How many paths are there between any source and destination? What is the distribution of the number of these paths?
- What is the blocking probability of one path with varied path lengths?

First, we need to get the maximum number of potential paths between any source-destination pair in an SN.

Lemma 1: For any source-destination pair in any substrate network with $\widetilde{n}_s$ nodes, the maximum number of potential paths K_e that have e links between the source and destination nodes is shown in (12).

$$K_e = \frac{(\widetilde{n}_s - 2)!}{(\widetilde{n}_s - e - 1)!} = \prod_{i=2}^{i=e} (\widetilde{n}_s - i) \tag{12}$$

Proof: Starting from the source, the number of potential paths for the first hop is $\widetilde{n}_s - 2$ because the number of intermediate nodes except the source and destination is $\widetilde{n}_s - 2$. For the second hop, there are totally $\widetilde{n}_s - 3$ links available by excluding source, destination, and the first hop node. For the (e)th hop, there is only one link available: the destination. Taking a substrate network with 5 substrate nodes: a, b, c, d, f. The maximum number of potential paths K_2 with 2 links between a and b should be $(5-2)!/(5-2-1)! = 3$. The potential paths are a–c–b, a–d–b and a–f–b, respectively. ■

Each SN can be considered as a sample from the random graph $G_s(N_s, E_s)$ as characterized by the substrate node distribution and the probability for any two nodes to form a link. In order to get virtual link blocking probability, we have to know which paths do actually exist for a source-destination pair in a specific SN.

We assume that all substrate links are independent. Together with the source and destination nodes, the $e-1$ substrate nodes form e independent links. Given $e-1$ substrate nodes $\vec{n}_{s,e-1} = \{n_{s,1}, \ldots, n_{s,e-1}\}$, the probability these nodes form a path in the order given and with a source-destination pair is shown in (13).

$$P_{p,\vec{n}_{s,e-1}} = \left(P_{s,l} \right)^e \tag{13}$$

Lemma 2: For any source-destination pair in any substrate network with $\widetilde{n}_s$ nodes, the probability that there exist exactly k_e paths with length e can be calculated by (14) from the binomial distribution [41].

$$P(k_e) = C_{k_e}^{K_e} \left(P_{p,\vec{n}_{s,e-1}} \right)^{k_e} \left(1 - P_{p,\vec{n}_{s,e-1}} \right)^{K_e - k_e} \tag{14}$$

The binomial distribution is the sum of K_e independent, identically distributed Bernoulli trials. In each Bernoulli trial, we have two possible outcomes: if a source-destination pair has a path with length e or not. Equation (14) then sums up the probability that there exists k_e paths out of K_e potential paths.

Many paths may exist between a source-destination pair, among which shorter paths are more favored for VNE due to

the fact that shorter paths use fewer link resources. However, paths between a source and a destination have more joint links. If a path is blocked, other paths may also become inaccessible since those paths may share the same blocked link due to the fact they share the same source and destination nodes. As we show in Section VI, selecting paths with joint links makes blocking probability higher. In addition, if we consider link-joint paths, the blocking probabilities of two paths may not be independent due to joint links. It is challenging to evaluate the correlation, especially in large substrate topology due to the combinatory growth of the number of scenarios with joint links.

We assume that the blocking probabilities of the link-disjoint paths are independent in this paper. This assumption is based on the assumption that the blocking probabilities of substrate links are independent. This assumption is valid for large and meshly connected networks. A similar assumption was made in [36] for calculating blocking probability in optical networks with fixed routing. If the substrate links are independent, we can assume link-disjoint paths are independent because there are no shared substrate links among link-disjoint paths. We will further justify our assumptions in our simulation results.

We now consider selecting candidate paths from link-disjoint paths only. Before we look at the number of link-disjoint paths that exist in a particular SN, we first think about the maximum number of link-disjoint paths that may exist for all SNs.

Theorem 1: For any source-destination pair in any substrate network with $\widetilde{n}_s$ nodes, the maximum number of potential link-disjoint paths that have e links is $K_{I,e} = \widetilde{n}_s - 2$, where $2 \le e < \frac{\widetilde{n}_s}{2} + 1$.

Proof: See Appendix A-A. ∎

In the following discussion, we assume that the condition $2 \le e < \frac{\widetilde{n}_s}{2} + 1$ is always satisfied, which is typically the case in real networks because the path length e tends to be small for lower blocking probability. While the potential number of link-disjoint paths was calculated in Theorem 1, we need to know the number of link-disjoint paths that actually exist in a specific substrate topology.

Corollary 1: For any source-destination pair in any substrate network with $\widetilde{n}_s$ nodes, the probability that there exist exactly k_e link-disjoint paths with length e has the lower bound as shown in (15).

$$P_I\left(k_e|K_{I,e}\right) \ge C_{k_e}^{K_{I,e}} \left(P_{p,\vec{n}_{s,e-1}}\right)^{k_e} \left(1 - P_{p,\vec{n}_{s,e-1}}\right)^{K_{I,e}-k_e} \tag{15}$$

Proof: See Appendix A-B. ∎

Given the above path-based approach using (15) is intractable, we will try to develop a link-based approach in the following part.

Theorem 2: For any source-destination pair in any substrate network with $\widetilde{n}_s$ nodes, the probability that there exist at least k_e link-disjoint paths with length e can be approximately calculated by (16).

$$P_I\left(k \ge k_e |\ K_{I,e}\right) \approx \widetilde{P}_I\left(k_e|K_{I,e}\right)^2 \check{P}_I\left(k_e|\ K_{I,e}\right), \tag{16}$$

where

$$\widetilde{P}_I\left(k_e|\ K_{I,e}\right) = \sum_{i=k_e}^{K_{I,e}} C_i^{K_{I,e}} {P_{s,l}}^i \left(1 - P_{s,l}\right)^{K_{I,e}-i},$$

and

$$\check{P}_I(k_e|\ K_{I,e}) = \sum_{i=k_e(e-2)}^{L(\widetilde{n}_s-2)} C_i^{L(\widetilde{n}_s-2)} {P_{s,l}}^i (1 - P_{s,l})^{L(\widetilde{n}_s-2)-i}.$$

Proof: See Appendix A-C. ∎

Corollary 2: For any source-destination pair in any substrate network with $\widetilde{n}_s$ nodes, the probability that there exist exactly k_e link-disjoint paths with length e can be estimated as (17).

$$P_I\left(k_e|K_{I,e}\right) = P_I\left(k \ge k_e|\ K_{I,e}\right) - P_I\left(k \ge k_e + 1|\ K_{I,e}\right) \tag{17}$$

Both (15) and (17) can be used to estimate the probability that there exist exactly k_e link-disjoint paths with length e. Equation (17) is more accurate because it covers all selection schemes. Now we move on to look at the probability that there exist multiple link-disjoint paths with different lengths.

Corollary 3: For any source-destination pair in any substrate network with $\widetilde{n}_s$ nodes, the joint probability that there exist exactly $\vec{k} = \{k_e,\ e = 1, \ldots, E_m\}$ link-disjoint paths can be estimated as (18):

$$\begin{aligned} P_I\left(\vec{k}|K_{I,e}\right) &= P_I(k_1, \ldots, k_e|\widetilde{n}_s) \\ &\approx P_I(k_1) \prod_{j=2}^{e} P_I\left(k_j|\ k_{I,j}\right), \end{aligned} \tag{18}$$

where

$$P_I(k_1) = \begin{cases} P_{s,l}, & \text{if } k_1 = 1 \\ 1 - P_{s,l}, & \text{if } k_1 = 0 \end{cases},$$

and

$$k_{I,\,j} = K_{I,j} - k_1 - k_2 - \cdots - k_{j-1}.$$

Proof: Using chain rule, we have:

$$\begin{aligned} P_I\left(\vec{k}|K_{I,e}\right) &= P_I\left(k_1, \ldots, k_e|K_{I,e} = P_I\left(k_1|k_{I,1}\right) P_I(k_2|k_1) \cdots \right. \\ &\quad \left. P_I\left(k_e|k_1, k_2, \ldots, .k_{e-1}\right)\right. \\ &\approx P_I(k_1) \prod_{j=2}^{e} P_I\left(k_j|\ k_{I,j}\right). \end{aligned}$$

The last equation is derived by removing those paths that have been used by shorter paths to simplify the calculation of conditional probabilities. ∎

Given now we know the distribution of available paths, we next focus on calculating the blocking probability for a virtual link.

We assume that the blocking events at each substrate link are independent. Proof of (19) is straight by the fact that a path is blocked if any link of the path is blocked. The independence assumption is necessary to make the calculation tractable:

$$P^{(B)}_{p,\ \vec{n}_{s,e-1}} = 1 - \left(1 - P^{(B)}_{s,l}\right)^e \tag{19}$$

The blocking probability for a virtual link depends on the embedding algorithms used. We consider an embedding algorithm that selects a path from maximum K existing link-disjoint shortest paths with a maximum path length E_m, which is typically used as a benchmark algorithm for comparison. Our approach can be extended to other embedding algorithms if the algorithms are known.

Lemma 3: For any source-destination pair in any substrate network with $\widetilde{n}_s$ nodes, under K shortest paths algorithm with maximum path length E_m, the paths can be identified as:

$$\tilde{k}_e = \begin{cases} k_e, & \text{if } k_e \le K - \sum_{i<e} k_i \\ K - \sum_{i<e} k_i, & \text{if } k_e > K - \sum_{i<e} k_i > 0 \\ 0, & \text{if } K - \sum_{i<e} k_i \le 0 \end{cases}, \quad (20)$$

where

$$e \le E_m.$$

Proof: This equation follows the definition of K shortest paths algorithm with maximum path length E_m. ■

It is interesting to know some statistics about these link-disjoint paths.

Corollary 4: For any source-destination pair in any substrate network with $\widetilde{n}_s$ nodes, under K shortest paths algorithm with maximum path length E_m, the average number of link-disjoint paths utilized can be found in (21).

$$\bar{k}(K,\ E_m) = \sum_{\vec{k}} \sum_{e=1}^{E_m} \widetilde{k}_e P_I\left(\vec{k}|K_{I,e}\right) \quad (21)$$

Corollary 5: For any source-destination pair in any substrate network with $\widetilde{n}_s$ nodes, under K shortest paths algorithm with maximum path length E_m, the average path length of available link-disjoint paths can be calculated by (22).

$$\bar{l}(K,\ E_m) = \sum_{\vec{k}} \sum_{e=1}^{E_m} \frac{\widetilde{k}_e e}{\sum_{e=1}^{E_m} \widetilde{k}_e} P_I\left(\vec{k}|K_{I,e}\right) \quad (22)$$

According to the link-disjoint paths distribution $P_I(\vec{k}|K_{I,e})$, Equation (21) and (22) calculate the expected values of the number of link-disjoint paths and the path length, respectively. We now move on to find the average blocking probability of a virtual link.

Theorem 3: For any source-destination pair, under the K shortest paths algorithm with maximum path length E_m, the average blocking probability of a virtual link can be calculated by (23).

$$P_{v,l}^{(B)}(E_m, K) = \sum_{\widetilde{n}_s=2}^{M_{s,n}} \sum_{\vec{k}} \left[\prod_{e=1}^{E_m} \left(P_{p,\ \vec{n}_{s,e-1}}^{(B)} \right)^{\widetilde{k}_e} \right] P_I\left(\vec{k}|K_{I,e}\right) P_{N_s}(\widetilde{n}_s) \quad (23)$$

Proof: By assuming the blocking events are independent among link-disjoint paths, (23) is based on the fact that a virtual link is blocked if all its candidate paths are blocked. Then (23) can be obtained by summing up all the link-disjoint paths distribution $P_I(\vec{k}|K_{I,e})$ over the distribution of a substrate network $P_{N_s}(\widetilde{n}_s)$. ■

Next, it becomes interesting to know the average path length for accepted virtual link requests.

Corollary 6: For any source-destination pair, under the K shortest paths algorithm with maximum path length E_m, the acceptance probability of a virtual link at a given path length e, when all paths with shorter lengths than e, is blocked can be calculated by (24).

$$P_{v,l}^{(A)}(e,\ K) = P_{v,l}^{(B)}(e-1, K) - P_{v,l}^{(B)}(e, K) \quad (24)$$

Proof: If a virtual link is blocked with path length e, the virtual link must be blocked with path length $e-1$. Therefore, the former is a subset of the latter one. Then, the difference $P_{v,l}^{(B)}(e-1,K) - P_{v,l}^{(B)}(e,K)$ is purely the acceptance gain by increasing maximum path length from $e-1$ to e, which means the acceptance probability with path length e while all paths with lengths smaller than e are being blocked. One of these paths with length e is selected as the mapping path for the virtual link following the shortest paths principle. ■

Corollary 7: For any source-destination pair, under the K shortest paths algorithm with maximum path length E_m, the average path length for accepted virtual link requests can be obtained from (25).

$$\bar{l}_{v,l}(K, E_m) = \sum_{e=1}^{E_m} e P_{v,l}^{(A)}(e,\ K) \quad (25)$$

C. Virtual Node Blocking Probability

In this section, we start to find the acceptance probability of a virtual node.

Lemma 4: The acceptance probability for a virtual node mapping can be calculated as:

$$P_{v,n}^{(A)} = \sum_{\widetilde{n}_s=2}^{M_{s,n}} \sum_{\widetilde{n}_{v,s}=2}^{\widetilde{N}_{v,s}} \left(1 - \left(P_{s,n}^{(B)}\right)^{\widetilde{n}_{v,s}}\right) P(\widetilde{n}_{v,s}|\widetilde{n}_s) P_{N_s}(\widetilde{n}_s), \quad (26)$$

where $P(\widetilde{n}_{v,s}|\widetilde{n}_s)$ is the conditional probability that there are $\widetilde{n}_{v,s}$ candidate nodes for a virtual node given that there are $\widetilde{n}_s$ substrate nodes in the SN. $\widetilde{N}_{v,s}$ denotes the maximum value of $\widetilde{n}_{v,s}$. $P_{s,n}^{(B)}$ is the blocking probability of a substrate node as defined earlier.

Equation (26) came from the fact that the mapping of a virtual node is accepted if any one of its candidate nodes is not blocked and all substrate nodes are independent.

D. Blocking Probability for a VN

We now try to calculate the blocking probability for a specific VNR. We made our efforts to find the probability for the existence of multiple link-disjoint paths connecting a source-destination pair. Finding link-disjoint paths is an important step because link-joint paths connecting a source-destination pair tend to be overlapped very likely due to the fact that link-joint paths share the same source and destination nodes. The paths also have to support the same bandwidth requirement for each virtual link when being selected. Selecting link-joint paths increases blocking probability significantly as

illustrated later in our numerical results. The allocation algorithms typically use link-disjoint paths by deploying variations of shortest-path algorithms.

When we deal with a VNR with multiple virtual links, these virtual links may be mapped onto substrate paths that are partially overlapped. However, these overlaps cause fewer problems because these links have different and independent bandwidth requirements and virtual links in a VNR do not share the same source and destination nodes at the same time. Blocking events are less correlated among paths for different virtual links. To make our analysis tractable, we assume blocking events across all virtual nodes and link mappings are independent. This assumption allows us to have a product-form solution like Jackson networks, which greatly simplifies our analysis.

Theorem 4: Assume a VNR with $\widetilde{n}_v = |N_v|$ nodes and $\widetilde{e}_v = |E_v|$ virtual links. Assume all virtual nodes and virtual links are independent. Then the acceptance probability of the VNR can be calculated as (27).

$$P_n^{(A)}(\widetilde{n}_v, \widetilde{e}_v) = \left(1 - P_{v,l}^{(B)}\right)^{\widetilde{e}_v} \left(P_{v,n}^{(A)}\right)^{\widetilde{n}_v} \tag{27}$$

Proof: This is straight forward by independent assumption. The reason that we could use the average probabilities $P_{v,l}^{(B)}$ and $P_{v,n}^{(A)}$ instead of conditional probabilities in (27) is also due to the independence assumption, which allows us to take expectation operations on each term in (27) separately. ■

Let $L(\widetilde{n}_v) = \frac{\widetilde{n}_v(\widetilde{n}_v - 1)}{2}$ denote the maximum number of virtual links that may exist given $\widetilde{n}_v$ virtual nodes. The average blocking probability can then be calculated.

Corollary 8: The average acceptance probability for an arbitrary VNR can be calculated as (28):

$$\bar{P}_n^{(A)} = \sum_{\widetilde{n}_v=2}^{M_{v,n}} \sum_{\widetilde{e}_v=1}^{L(\widetilde{n}_v)} P_n^{(A)}(\widetilde{n}_v, \widetilde{e}_v) P(\widetilde{e}_v|\widetilde{n}_v) P_{N_v}(\widetilde{n}_v), \tag{28}$$

where $P(\widetilde{e}_v|\widetilde{n}_v)$ is the conditional probability that there exist $\widetilde{e}_v$virtual links given $\widetilde{n}_v$ virtual nodes in a VNR; and $L(\widetilde{n}_v)$ is the maximum number of different links among $\widetilde{n}_v$ nodes.

The conditional probability using binomial distribution is shown in (29):

$$P(\widetilde{e}_v|\widetilde{n}_v) = \begin{cases} C_{\widetilde{e}_v}^{L(\widetilde{n}_v)} (P_{v,l})^{\widetilde{e}_v} (1 - P_{v,l})^{L(\widetilde{n}_v) - \widetilde{e}_v}, \\ \qquad \text{if } \widetilde{e}_v > L_m(\widetilde{n}_v) \\ 1 - \sum\limits_{l > L_m(\widetilde{n}_v)}^{L(\widetilde{n}_v)} C_l^{L(\widetilde{n}_v)} {P_{v,l}}^l (1 - P_{v,l})^{L(\widetilde{n}_v) - l}, \\ \qquad \text{if } \widetilde{e}_v = L_m(\widetilde{n}_v) \end{cases} \tag{29}$$

where $L_m(\widetilde{n}_v)$ is the minimum number of virtual links to make sure that a VNR is actually connected without any isolated nodes.

E. Estimating the Offered Load $\lambda_{s,l}$ for a Substrate Link

At the beginning of Section IV-A, we mentioned that $\lambda_{s,l}$, the offered load for a substrate link, is not the same as λ_v, the VNR arrival rate. This is also true for the offered load of a substrate node $\lambda_{s,n}$. In this section, we discuss how to estimate $\lambda_{s,l}$, $\lambda_{s,n}$.

The offered loads $\lambda_{s,l}$, $\lambda_{s,n}$ should include two parts, the part that is blocked by the substrate node or link and the part that is accepted. We call the part that is accepted and carried by a substrate link as effective loads denoted as $\lambda_{e,l}$, $\lambda_{e,n}$ respectively. As we mentioned at the beginning, there are three levels of blocking events that may happen: the substrate link/node, the virtual link/node, or the VN. A VNR is blocked if any virtual link or node is blocked. It is important to note that the offered loads $\lambda_{s,l}$, $\lambda_{s,n}$ should not include the loads blocked by other links or nodes. Because even a substrate node/link accepts this load, the substrate node/link still may not carry the load. Therefore, we assume the VNRs that have been accepted become the offered loads for the substrate link/node. It should be noted that offered load is not the effective load for a specific node/link due to the fact the offered load may be rejected by this node/link while being accepted by other links/nodes. Following the above discussion, we have:

$$\lambda_{s,l} \approx \lambda_v \bar{P}_n^{(A)}(\lambda_{s,l}, \lambda_{s,n}) \bar{l}(E_m, K, \lambda_{s,l}\,) \bar{e}_v / \bar{e}_s, \tag{30}$$

and

$$\lambda_{s,n} \approx \lambda_v \bar{P}_n^{(A)}(\lambda_{s,l}, \lambda_{s,n}) \bar{n}_v / \bar{n}_s, \tag{31}$$

where we emphasized that the network acceptance probability and average path length for accepted virtual links depend on the offered loads. With rising offered loads, $P_n^{(A)}(\lambda_{s,l}, \lambda_{s,n})$ and $\bar{e}_v(E_m, K, \lambda_{s,l}\,)$ go down monotonically. Therefore, (30) and (31) will converge through a recursive process.

V. Proposed Coordinated Mapping Approach

Most of the existing VNE algorithms are the so-called uncoordinated algorithms, where all virtual nodes are mapped first. The mapped virtual nodes provide a source-destination pair for the virtual link mapping stage. However, mapping virtual nodes and virtual links jointly is likely to produce more optimal solutions because joint mapping allows a virtual link to try multiple source-destination pairs in the SN. Essentially speaking, mapping virtual nodes and links jointly is sampling the source-destination pairs from all possible combinations of virtual node and virtual link mapping. While, uncoordinated two-stage mapping only takes a subset of all possible combinations into account. Therefore, an uncoordinated two-stage mapping can be vulnerable to getting a local optimum.

In this section, we implement a GA algorithm to solve VNE problems with full coordination. As opposed to uncoordinated algorithms, our virtual node mapping and virtual link mapping are generated simultaneously in both the crossover procedure and mutation procedure. Therefore, we can claim our genetic solution encompasses all potential mapping combinations. In the subsections that follow, we illustrate how we achieve full coordination.

In our approach, a chromosome $\boldsymbol{c_i}$ denoted by (32) represents a feasible VNE solution. Since $\widetilde{e}_v$ indicates the number of virtual links for a request, there are totally $\widetilde{e}_v$ genes in a chromosome. Each gene is defined as a path between two meta

Algorithm 1 Coordinated VNE Mapping – GAOne

procedure GENERATING ORIGINAL PATH POOLS
Input: G_s **output:** P^s
  Constructing static K-shortest path pool for each source-destination based on substrate network G_s.
  goto Initialize population
procedure INITIALIZE POPULATION
Input: G_v, G_s, P^s, M **output:** $\mathbb{P}$
  Generating G_s' *by using* G_v *and* G_s
  while *actual_population_size* $< M$ **do**
    for $e_v \in E_v$ **do**
      Finding a feasible path for e_v *in* G_s'
    putting the feasible solution in to $\mathbb{P}$
    actual_population_size++
  goto *Crossover*
procedure CROSSOVER
Input: $\mathbb{P}$ **output:** child chromosomes $\boldsymbol{c}'$
  Selecting parent chromosomes $(\boldsymbol{c_s}, \boldsymbol{c_r})$ *from* $\mathbb{P}$
  Check node mapping conflicts
  for *each virtual link j* **do**
    Generating child gene $\boldsymbol{g_{(M+1)j}}$ *and* $\boldsymbol{g_{(M+2)j}}$
    goto Mutation
  end for
  goto Updating population
procedure MUTATION
Input: $\boldsymbol{g}$, mutation rate **output:** Modified genes $\boldsymbol{g}'$
  mutating $\boldsymbol{g}$ *with a fixed mutation rate*
procedure UPDATING POPULATION
Input: $\boldsymbol{c}'$ **output:** Updated population $\mathbb{P}$
  Adding $\boldsymbol{c}'$ *into population*
  Updating population and limiting size to M
  if terminating condition triggered **then**
    return the fittest chromosome in $\mathbb{P}$
  else goto Crossover

nodes in the augmented network. Therefore, a gene in the augmented network includes the virtual node mapping solution at the first and last meta links. The intermediate links form a substrate path that indicates a virtual link mapping solution. A gene $\boldsymbol{g_{ij}}$ can be divided into two partial paths as (33): head $\boldsymbol{H_{ijk}}$ and tail $\boldsymbol{T_{ijk}}$, where k is the index of node in the gene:

$$\boldsymbol{c_i} = \left\{\boldsymbol{g_{i1}}, \boldsymbol{g_{i2}}, \ldots, \boldsymbol{g_{ij}}, \ldots, \boldsymbol{g_{i\tilde{e}_v}}\right\} \tag{32}$$

$$\boldsymbol{g_{ij}} = \left[\boldsymbol{H_{ijk}}, \boldsymbol{T_{ijk}}\right], \ \forall k \in \left(1, d_{ij}\right) \tag{33}$$

where,

$$\boldsymbol{H_{ijk}} = n_{ij1}, n_{ij2}, \ldots, n_{ijk}$$
$$\boldsymbol{T_{ijk}} = n_{ij(k+1)}, n_{ij(k+2)}, \ldots, n_{ijd_{ij}}$$

A. Initial Population

In GA, the first step is to initialize the population. The population $\mathbb{P}$ is composed of multiple chromosomes. To get the initial population, we need to find multiple feasible mapping solutions. In this step, we do not consider the performance of these solutions. We only seek feasible ones.

We first randomly select a substrate node from the candidate sets for each virtual node. In this step, all the first and last links of genes are set in a chromosome. Next, we need to find a substrate path to make each gene get connected. We choose the shortest paths based on the hop count factor.

We identify K shortest paths for each source-destination pair in the SN similar to [35]. Each path pool with a source-destination pair is only dependent on the SN topology. Therefore, the path pools can be used for all the online requests as long as the SN topology is not changed. That is to say, the path pools generated before the online requests facilitate our online VNE procedure.

After we find a substrate path for each gene, a chromosome is generated. We have to check if the chromosome is a feasible one before putting the chromosome into the initial population. We define a feasible solution if the allocated resources of the SN can satisfy the requirements of the VNR.

If the chromosome is not feasible, we have to go back to select and check another candidate chromosome again. This process continues until a feasible chromosome is selected. In some special cases, the random initialization process could not find a feasible chromosome due to the exhausted available resources. Instead of rejecting the request directly, we select and stamp some infeasible chromosomes into the population. Therefore, the request still has chances to enter crossover and mutation operations to produce feasible child chromosomes.

B. Selection and Crossover

Before the crossover operation, we have to select two parent chromosomes from the current population. We arrange the selection scheme based on a random selection with replacement same as [35]. Therefore, the parent chromosomes are returned to the population after the crossover.

In the crossover operation, each gene in a parent chromosome crossovers with the corresponding gene in the other parent chromosome. For example, the two parent chromosomes are denoted by $\boldsymbol{c_s}$ and $\boldsymbol{c_r}$. The parent chromosomes generate two child chromosomes, which are denoted by $\boldsymbol{c_{(M+1)}}$ and $\boldsymbol{c_{(M+2)}}$. Each chromosome should exchange partial genes with its counterpart through a crossover point. The crossover point is generally a common node of parent genes. The common node is a node n_{sju} in $\boldsymbol{g_{sj}}$ equivalent to a node n_{rjv} in $\boldsymbol{g_{rj}}$, where u and v are not the indices of source or destination node. If there are more than one common node in parental genes, one common node is selected to become the crossover point. Apparently, such children generated in this pattern are still valid paths. The child genes (34) and (35) are defined as below:

$$\boldsymbol{g_{(M+1)j}} = \boldsymbol{H_{sju}}, \boldsymbol{T_{rjv}} \tag{34}$$

$$\boldsymbol{g_{(M+2)j}} = \boldsymbol{H_{rjv}}, \boldsymbol{T_{sju}} \tag{35}$$

The special case happens when there is no common node between two parent genes. In this case, a link is selected as the crossover point in each parent gene. To make sure the child's gene is still a valid path, a partial path obtained from the shortest path pool connects the child genes. For instance, we randomly select link $(n_{sju}, n_{sj(u+1)})$ as a crossover point

for g_{sj}, and then select $(n_{rjv}, n_{rj(v+1)})$ for g_{rj}. A substrate path between n_{sju} and $n_{rj(v+1)}$ is chosen from the source-destination path set $P^s(n_{sju}, n_{rj(v+1)})$. Similarly, a substrate path is picked from $P^s(n_{rjv}, n_{sj(u+1)})$ to connect the partial child genes. The results of the crossover operator without the common node should be (36) and (37).

$$g_{(M+1)j} = H_{sju}, p_{n_{sju}, n_{rj(v+1)}}, T_{rj(v+1)} \tag{36}$$

$$g_{(M+2)j} = H_{rjv}, p_{n_{rjv}, n_{sj(u+1)}}, T_{sj(u+1)} \tag{37}$$

C. Node Mapping Conflicts

After crossover, the child gene may contain loops, which is an invalid path. Hence, each gene should have a validation check to remove loops. Subsequently, we construct the child chromosomes from child genes. Each parent gene pair can generate two child genes. In this step, the node mapping conflict may happen when a virtual node is mapped into different substrate nodes in the same request. We propose a graph coloring method to detect the node mapping conflict problem.

Definition 1: Under the condition that virtual node mapping is unsplittable, each unique virtual node associated with some virtual links in a chromosome can only be mapped to one substrate node.

We call the chromosomes that satisfy Definition 1 as valid, otherwise invalid.

Inspired by graph coloring theory, we define the specific set of all virtual node mappings in a parent chromosome as having one color. If two parent chromosomes have at least one virtual node mapped to different substrate nodes, we define the set of all virtual nodes in the second parent chromosome that are mapped to different substrate nodes as having the second color. If all the virtual nodes are mapped to different substrate nodes with the two parent chromosomes, we call the two chromosomes complete two-color parent chromosomes. The nodes in a chromosome can have either the first color or the second color. However, each node in a valid chromosome can only have one color.

Lemma 5: If all the virtual nodes in the two parent chromosomes can be represented by one color, the crossover operation will always generate two valid child chromosomes.

Proof: Because all the virtual nodes in the two parent chromosomes have the same color, the crossover operation will not cause any change to the color of each virtual node. Therefore, after crossover, each node in the two child chromosomes will still have one color, which means the two child chromosomes are still valid. ■

Lemma 6: If the virtual nodes in the two parent chromosomes have different colors, the crossover operation may generate invalid children.

Proof: As shown in Fig. 3, complete two-color parent chromosomes are in red and blue, respectively. After the crossover, all neighboring virtual nodes switch colors. Therefore, a virtual node will have different colors from all its neighbors and all its neighbors must have the same color. neighbours with the same colour cause node mapping conflict problems since some virtual nodes may be mapped into two colors shown as virtual node A in Fig. 3. ■

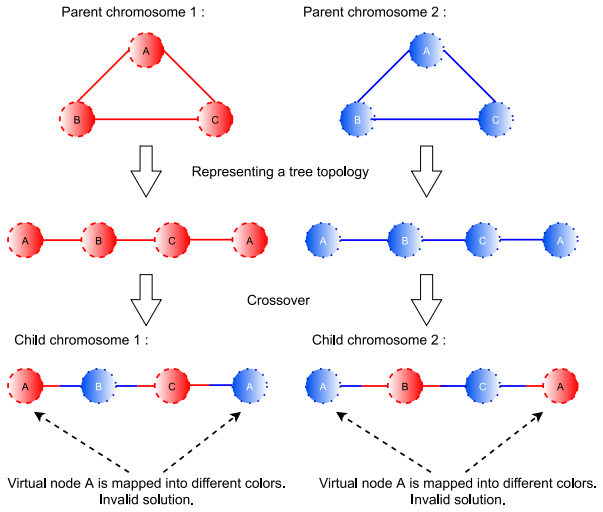

Fig. 3. Invalid child chromosomes generated due to the virtual node mapping conflict.

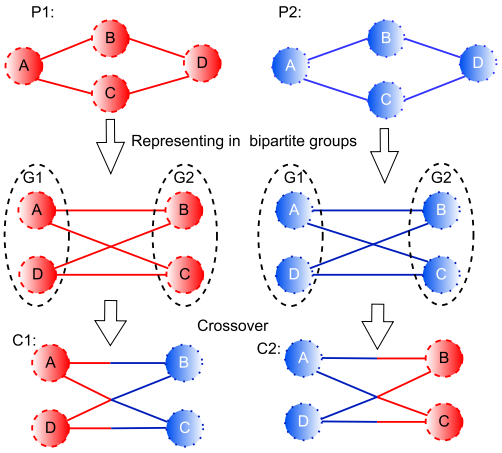

Fig. 4. Child chromosomes are valid if the virtual request is a bipartite graph.

Theorem 5: If the topology of a VNR forms a bipartite graph, then any two valid parent chromosomes will always generate two valid children after crossover operation.

Proof: We divide the virtual nodes in the VNR into the two groups in the bipartite graph as shown in Fig. 4. We call the virtual nodes in one group G1 nodes and the virtual nodes in the other group as G2 nodes. Given Lemma 5, without loss of generality, we assume the two parent chromosomes are complete two-color chromosomes. Suppose all the nodes in the first parent chromosome P1 have the color red and all the nodes in the second parent chromosome P2 have the color blue. Therefore, before the crossover, a red node in G1 will always be connected to a red node in G2 and a blue node in G1 will always be connected to a blue node in G2. After the crossover operation, a red node in G1 will be connected to a blue node in G2 with the first child C1 and a blue node in G1 will be connected to a red node in G2 with the second child C2. Because there is no direct link within each group, the nodes in the same group will also have the same color and

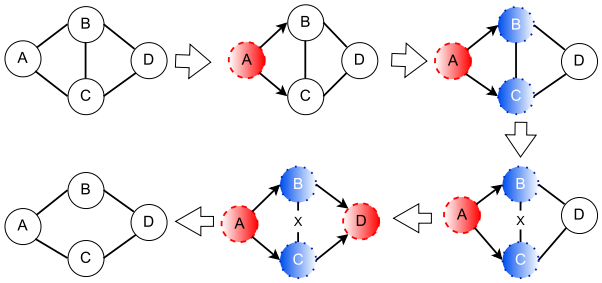

Fig. 5. Finding a bipartite subgraph using breadth-first search.

each node in a child will have one color only, which means both C1 and C2 are valid child chromosomes. ■

Corollary 9: For a VNR with arbitrary topology, there always exists a crossover scheme that will generate two valid child chromosomes by applying the crossover operation to a subset of virtual links in the VNR.

Proof: Apparently, a bipartite subgraph can be generated by removing some links in a VNR with an arbitrary topology. A breadth-first algorithm can be applied to construct a subgraph with bipartite topology [42]. We can then generate two valid child chromosomes based on Theorem 5. For example, we have a VNR with 4 virtual nodes and 5 virtual links as shown in Fig. 5. We first randomly select a virtual node as the source node and color the source node with the red color shown as node A in Fig. 5. Next, we color all the neighbors of the source node with blue color. We remove the link if its two connected nodes are in the same color. This process stops when all the nodes have been colored. Then, we can get a bipartite subgraph of the virtual request. We do not crossover these removed links (the link B-C in Fig. 5). After the crossover operation, we add those links back to the child chromosomes. The results are still valid child chromosomes. ■

D. Mutation

Another core procedure of GA is mutation operation. To avoid the solutions evolving to the local optimum, the mutation operation comes out to jump out of the current searching space. Each child gene has a small probability to be mutated after crossover.

In our GA approach, we have to deal with two scenarios: whether the two mutation points in a gene include a meta node. If the mutation happens at two intermediate substrate nodes in a child gene, we just replace the path between these two intermediate substrate nodes with an alternative path in our path pools. However, if one of the mutation points is a meta node, the virtual node mapping is mutated too. Since the virtual node may be connected by several virtual links, all child genes indicating these virtual links should be mutated consistently. Then, to make sure the child genes are still connected substrate path, we have to find alternative paths to connect the new virtual node mapping solution with previous adjacent nodes for all these child genes.

E. Sorting Population & Synchronization

Now, we generated two child chromosomes. Then we update the population by adding two child chromosomes. We measure chromosomes using the objective function in (6). Only the best M (population size) chromosomes survive in the updated population. The new generation starts again and goes back to the selection and crossover operations. This procedure ultimately stops when the maximum count is reached or there are no different child chromosomes available. The best chromosome becomes the final solution.

F. Convergence Analysis

Convergence analysis is essential for VNE problems, especially as the network scale increases. When the network scales go up, there are more alternative paths available. However, these extra paths tend to be longer. As shown in (19), the blocking probabilities for longer paths go up very fast. Therefore, it is not cost-effective to search these paths for feasible solutions. In our proposed VNE approach, we can leverage the maximum path length E_m to control the convergence time.

In addition, the maximum number of potential link-disjoint paths that have e links is provided in Theorem 1. This theorem also proves that the convergence time of our VNE approach can be bounded.

VI. Numerical Results

In this section, we will compare the results of our analytical model with the simulation results obtained with representative VNE algorithms. The purposes of these comparisons are two folds: 1) We made some assumptions about the independence of certain random variables and some approximation in deriving (17) during our modeling process. We want to use the simulation results to validate whether these assumptions and approximations are reasonable and acceptable. 2) We want to show the performance gaps between our analytical model and existing VNE algorithms. Therefore, our analytical model can serve as a benchmark solution. To fill the gaps, we compare our analytical model with our fully coordinated approach, called GAOne, which achieves a fully coordinated solution as discussed in Section V. In the simulation, GAOne has two strategies: using link-disjoint paths or using non-link-disjoint paths. GAOne using link-disjoint paths is closer to our analytical models since our analytical model uses link-disjoint paths as discussed in Section IV-B.

Our network analytical model as described in Section IV calculates the blocking probability of the node mapping and link mapping separately. However, as (28) shown, our analytical model gets the acceptance probability from the product of the virtual node and virtual link acceptance probability. The product form indicates that (28) includes all possible combinations of different virtual node mapping and virtual link mapping. Therefore, the fully coordinated solution—GAOne is comparable to our analytical model and the results of our GAOne simulation should get converged to our analytical model.

We also compare our analytical model with other existing VNE algorithms as listed in Table I. The compared algorithms we chose are based on the performance-oriented and speed-oriented aspects. In other words, the criteria we used to select

TABLE I
COMPARED VNE ALGORITHMS

GAOne	Our proposed fully coordinated GA mapping solution.
PBGA	Partial coordinated solution with Path Based GA Link Mapping.
SBGA	Partial coordinated solution with Segment Based GA Link Mapping.
G-SP	Uncoordinated greedy solution
R-ViNE	Partial coordinated solution based on MILP with randomized node mapping
D-ViNE	Partial coordinated solution based on MILP with deterministic node mapping
MCTS	Partial coordinated reinforcement learning-based algorithm

algorithms for comparisons are the best in VNR acceptance ratio or the fastest in the execution time. Specifically, we selected G-SP [16] for comparison because it uses the shortest path algorithm for link mapping, which is widely used by other metaheuristic algorithms as mentioned above, and also because G-SP is considered the fastest algorithm due to its simplicity. We selected D-ViNE and R-ViNE [5] for comparisons because they are considered to be a benchmark due to their MILP-based approaches for both node and link mapping. We have demonstrated that the performance of our algorithm is either close or better than D-ViNE and R-ViNE. A reinforcement learning based solution MCTS [43] is chosen for comparisons. MCTS maps node mapping and link mapping in separate stages with partial coordination by introducing the Monte Carlo Tree Search algorithm [44]. We also chose two GA algorithms in [35]: Segment-Based Genetic Algorithm (SBGA) and Path-Based Genetic Algorithm (PBGA). SBGA and PBGA are two-stage solutions. They focus on virtual link mapping solutions. We expect that our GAOne can have better performance than SBGA and PBGA due to the coordinated mapping strategy.

In the simulation, we assume the same topology distributions in our compared algorithms and our analytical model. We start with the description of the scenario setup and then discuss numerical results obtained from both our analytical model and simulation of GAOne.

A. Environment Setups

Our proposed performance analytical model aims to estimate the performance of a VNE algorithm under general environments. To get numerical results, we have to set up a specific environment. In this paper, we use a general environment setup, which is the same as papers in [5], [35].

We chose random topology because most recent studies [5], [11], [15], [33] on VNE have been tested on random topology. Our paper follows the same assumption on random topology as those studies. There are good reasons to use random topology. As it is well-known, one of the core features of SDN is its centralized control plane, which enables more dynamic and sophisticated routing algorithms. However, the performance of a VNE routing algorithm is highly dependent on substrate topology. A star topology, for example, can make all embedding algorithms perform more or less the same due to lack of alternative routes. On the other hand, a full-mesh topology can cause significant differences among different algorithms. A more objective evaluation of a VNE algorithm is to get a statistical average across the performances under random topology. Developing an analytical model for a specific topology is not efficient although it may be easier to develop because it is hard to be generalized to other topologies. For example, suppose we want to show performances for four different topologies, we have to develop four different analytical models, which becomes quite cumbersome, and their results are harder to interpret for getting general conclusions. The selection of those topologies is also subjective and needs lots of justification. Our goal here is to compare different VNE algorithms under all topology scenarios. The statistical average of random topology provides a more objective evaluation for VNE algorithms. Our goal is to provide an analytical benchmark for comparing VNE algorithms under all topology scenarios as stated in Section I.

Instead of generating a random number of substrate nodes for each SN, we generated SNs with a fixed number of nodes 50. To make the SN random, we generated the links using the Waxman model with parameters $\alpha = 0.5$ and $\beta = 0.2$. The capacity of a link $b(e_s)$ is generated with a uniform distribution $P_{b(e_s)}$ ranging from 50 to 100. Each substrate node has a CPU capacity value $c(n_s)$ that follows a uniform distribution $P_{c(n_s)}$ ranging from 50 to 100 and a location (x_{n_s}, y_{n_s}) with $P_{L_{n_s}}(x, y)$ following uniform distribution in a 25×25 grids of a square area W.

In the Waxman model, the probability of any two nodes forming a link depends on their distances. This kind of locality structure is useful if physical locations are important in some deployment applications. We use an average probability in place of the distribution in this paper because we focus on general performance instead of specific deployment scenarios. The average probability of a link between two nodes is shown as follows:

$$P_{s,l} = \mathbb{E}[p(d_s)] = \mathbb{E}\left[\alpha e^{-\frac{d_s}{\beta D_m}}\right] = \frac{1}{X^2 Y^2}\int_W \alpha e^{-\frac{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}{\beta D_m}}\, dx_1\, dx_2\, dy_1\, dy_2, \tag{38}$$

where d_s is the distance between two substrate nodes $(x_1,\ y_1)$ and $(x_2,\ y_2)$, D_m denotes the maximum distance between any two nodes. In our cases, we set $X = Y = D_m = 25$. To introduce randomness, we randomly generated three SNs in our simulation. Therefore, there are $C_2^{50} = 1225$ number of source and destination pairs in one substrate network.

We had to generate a very large number of VNRs for each SN to get steady-state results. On average, we generated around 18,000 VNRs for each SN and each load, which was a very time-consuming process. Specifically, we generated random VNR requests following the Poisson processes with λ_v ranging from 4 to 8 requests per 100-time units. Each request required a holding time, which was exponentially distributed with an average of $\tau_v = 1000$ time units. We conducted simulations for each specific scenario over a period of 50,000 time units.

The distribution of the number of virtual nodes in a VNR P_{N_v} followed a piece-wise uniform distribution between 2 and 9, i.e., from 2 to 4, each happened with the probability 9/32, from 5 to 9, each happened with the probability 1/32. The reason for this specific distribution was to emulate the real scenarios where we tend to have a larger number of smaller requests. *D*, the maximum distance between a virtual node and its associated substrate node, is set to 10. Each virtual node $n_v \in N_v$ is associated with a CPU capacity requirement $c(n_v)$ that follows a uniform distribution $P_{c(n_v)}$ between 0 and 20 and a location (x_{n_v}, y_{n_v}) following the same uniform distribution as $P_{L_{n_s}}(x, y)$ in the same fixed area W. Each virtual link e_v between two virtual nodes has a random bandwidth capacity $b(e_v)$ following the uniform distribution $P_{b(e_v)}$ ranging from 0 to 50. The existence of a virtual link is generated by the Waxman model with $P_{v,l} = P_{s,l}$.

To find $P(\widetilde{n}_{v,s}|\widetilde{n}_s)$ in (27), we need to know the distribution of the substrate nodes. Since the substrate nodes are uniformly distributed in the area *W*. Let *V* be the mapping area for a virtual node, in which all substrate nodes are located within distance constraint *D*. Given the square area *W*, *V* depends on where a virtual node is located. The probability that a substrate node is in area *V* could be calculated as:

$$P\left(\widetilde{n}_{v,s}|\widetilde{n}_s\right) = C_{\widetilde{n}_{v,s}}^{\widetilde{n}_s}\left(\frac{V}{W}\right)^{\widetilde{n}_{v,s}}\left(1-\frac{V}{W}\right)^{\widetilde{n}_s-\widetilde{n}_{v,s}}, \quad (39)$$

where

$$V = \int_{max(x_{n_v}-D,\ 0)}^{min(x_{n_v}+D,\ X)} \int_{max\left(y_{n_v}-\sqrt{D^2-(x_{n_s}-x_{n_v})^2},\ 0\right)}^{min\left(y_{n_v}+\sqrt{D^2-(x_{n_s}-x_{n_v})^2},\ Y\right)} dx_{n_s}\, dy_{n_s}.$$

In our model, the value of E_m is a major factor contributing to the computing complexity. The complexity increases dramatically with increasing E_m due to the number of combinations in $\vec{k}$. In our simulation, we found that there was very little gain when E_m was beyond 6. Therefore, we set E_m to 6 to ensure accuracy while maintaining an acceptable complexity in both simulation and analytical settings.

Similarly, we found the performance improved very little when *K* was larger than 7 with $E_m = 6$. To reduce the computing complexity, we set $K = 7$ for both simulation and analytical settings. In the simulation of GAOne, a dynamic candidate path pool was maintained for each source-destination pair with a maximum number of paths in the pool set to $K = 7$. The purpose of maintaining dynamic path pools was to increase path diversity with a genetic algorithm. This dynamic pool realizes our random topology assumption.

The average number of link-disjoint path lengths is 3.23 (see (22)). In our simulation, the average link-disjoint path lengths for each source-destination pair in the initial shortest path pools are 3.35. The small difference indicates that our analytical model captures the distribution of the topologies well. We further discuss the path lengths that are averaged over all dynamic path pools in Section VI-B2.

B. Simulations on Our Analytical Model

We compare our analytical model with GAOne at three levels: substrate link level, virtual link level and VN level.

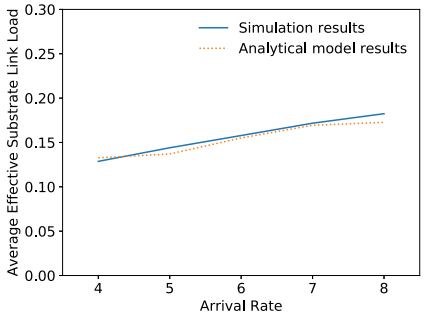

Fig. 6. Average effective link load over arrival rates.

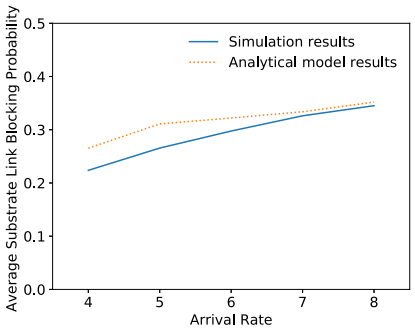

Fig. 7. The substrate link blocking probability over arrival rates.

1) Results at the Substrate Link Level: At the substrate link level, both effective substrate link load and substrate link blocking probability are interesting metrics. The two metrics depend on each other. In our setups, the offered loads only changed in λ_v, the distributions of holding time and bandwidth requirements do not change. Therefore, we use arrival rates as a measurement of loads.

In our analytical model, the effective substrate link load $\lambda_{e,l}$ could be acquired from the offered substrate link load $\lambda_{s,l}$, which is calculated in (30). We have:

$$\lambda_{e,l} = \lambda_{s,l} \times \left(1 - P_{s,l}^{(B)}\right). \quad (40)$$

In our simulation, the effective substrate link load was relatively easy to collect. We averaged the number of requests that have been admitted and carried by substrate links overall substrate links and all SNs. As shown in Fig. 6, the results are very close at different VNR arrival rates and justify that (30) is accurate.

Another metric we measured at the substrate link level was the substrate link blocking probability. In our model, we calculated the substrate link blocking probability $P_{s,l}^{(B)}$ by (10). When it came to the simulation, we followed the PASTA [45] property and averaged the blocking events across all substrate links and all SNs for all virtual link requests at a load level. PASTA stands for Poisson Arrivals See Time Average, which is a well-known property in queuing theory. As shown in Fig. 7, the maximum difference, which happens at arrival rate 4, is around 5%. The simulation results of our proposed

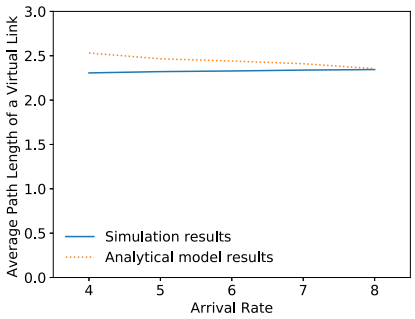

Fig. 8. Average path lengths from analytical model and simulation.

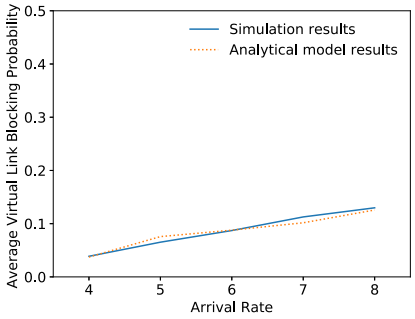

Fig. 9. Virtual link blocking probability over arrival rates.

VNE algorithm approach our analytical model as the substrate link becomes busier. This observation is likely caused by the number of VNRs in the simulation. For each substrate topology, the number of VNRs doubles from 12,000 to 24,000 when the arrival rate rises from 4 to 8. The simulation becomes more accurate as VNRs increase. This result explains why the biggest difference occurs with an arrival rate of 4.

Due to our large substrate network structure, this difference in the substrate link blocking probability actually does not sufficiently reflect the virtual link blocking probability. Specifically, when a virtual link gets blocked, all the potential substrate paths of this virtual link are blocked. The difference in the substrate link blocking probability is cancelled out by the great number of possible paths available in a large substrate network. This claim is supported by Fig. 9.

2) Results at the Virtual Link Level: We first compare the results of average path length for accepted virtual links as shown in Fig. 8. The results in Fig. 8 are consistent with the results shown in Fig. 7. The average path length gets longer when the substrate link gets congested. Furthermore, the average path length goes down a little bit as the load goes up for our analytical model. The simulation results are nearly constant. In our analytical model, the candidate paths did not change with the load as defined in (22). When the load increased, the average path length actually selected for accepted virtual link requests decreased as defined in (25) because longer paths were more likely to be blocked. In our simulation, the average path length of the dynamic candidate path pools increased with increasing load, while the fact that shorter paths were more likely to be selected for a given path pool was still true. The overall effect made the average path length stay roughly the same with increasing load.

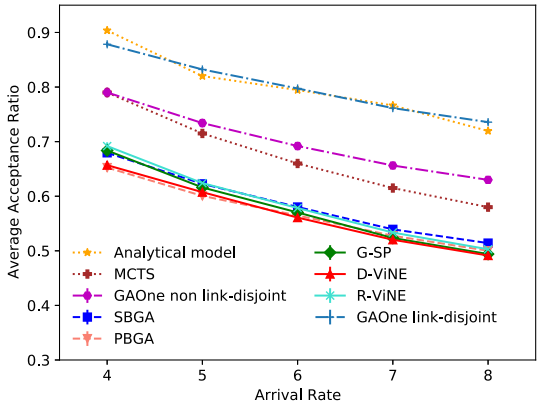

Fig. 10. VNR acceptance ratio over arrival rates.

In our analytical model, the virtual link blocking probability can be obtained by (23). We followed the PASTA property to estimate virtual link blocking probability in simulation. Specifically, we accumulated the number of virtual links that were blocked and the number of virtual links requested across all VNR and all substrate topologies. We then estimated the virtual link blocking probability using the ratios of the two aggregated counts. Fig. 9 shows the results of both the analytical model and simulation, which match each other very closely. The results justify that our DRART model is accurate enough to capture the blocking probability of virtual links.

3) Results at the VN Level: Now, we compare the final VNR acceptance probabilities. As we discussed earlier, selecting a path from link-disjoint paths for mapping a virtual link can achieve better performance compared to selecting paths without considering link-disjoint paths. In Fig. 10, the acceptance ratio of the compared algorithms is the average values over arrival rates from 4 to 8 per 100-time units. The acceptance ratio is measured by the ratio of the number of successfully mapped VNRs and the proposed VNRs.

We calculated our simulation results with a 95% confidence interval using a batch-mean method, which revealed an average range of less than 0.8%. To enhance the clarity of the figure, we have omitted the confidence interval portions in Fig. 10. We first compare simulation results using link-disjoint paths (GAOne link-disjoint) and simulation results using non-link-disjoint paths (GAOne non link-disjoint), with the same $E_m = 6$ and $K = 7$. As shown in Fig. 10, the results for link-disjoint paths are consistently better than the results for non-link-disjoint paths.

Fig. 10 also shows our analytical results in comparison with simulation results. We can see that our analytical results are very close to the GAOne simulation results with link-disjoint

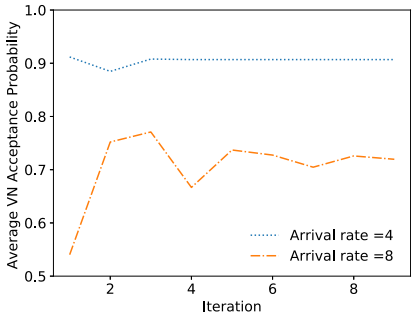

Fig. 11. The VNR acceptance ratio converging process.

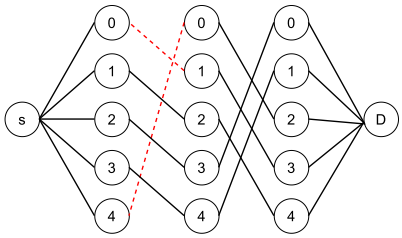

Fig. 12. A feasible scheme for link-disjoint paths between a source-destination pair.

paths. This result confirms that the independence assumptions we made in Section IV-B are acceptable. Another observation is on our GAOne algorithm, our GAOne has a far better performance compared with other related algorithms. This result indicates the efficiency of the GAOne algorithm and also implies that our non-linear objective function can generate more efficient solutions. Moreover, the results verify the benefits of the jointly mapping strategy.

In Section IV-E, we talked about why we require a recursive procedure to estimate the average offered load of a substrate node/link. To verify the correctness of a recursive procedure, we should prove its convergence. In Fig. 11, we show the acceptance ratio of a VNR $P_n^{(A)}(\lambda_{s,l}, \lambda_{s,n})$ in (30) and (31) getting converged after several iterations. We observe that when the arrival rates get higher, the system gets busier. In consequence, more iterations are required before the system gets steady.

VII. Conclusion and Future Work

We have created a novel DRART model for evaluating the blocking probability at the virtual link level. Moreover, we have designed an integrated approach that adopted a recursive procedure among three levels of VNE models. Our numerical results justify that the model we created is accurate and can provide a benchmark for various VNE algorithms. Moreover, our GA approach achieves full coordination by mapping virtual nodes and links jointly. The comparison between our analytical model and our GA approach also validates our assumptions and approximations are reasonable and acceptable.

Our DRART has wide applications and can be used in many scenarios as mentioned earlier. An analytical model for satellite networks is a good extension of our DRART model. A low earth orbit (LEO) satellite network [46] is a dynamic and large-scale network that provides numerous services. The software defined networking (SDN) architecture can help satellite networks to have a centralized view to deploy various routing protocols. An analytical model is required to estimate the performance of different routing strategies and provide a fair comparison. Similar to the scenarios covered in our DRART model, a satellite network also has random topology and dynamic routing due to the constant movements of satellites.

As the network size varies based on the specific application, further research could explore our VNE approach in the context of larger networks.

Appendix A
Proof of Theorems

A. Theorem 1: The Maximum Number of Potential Link-Disjoint Paths Between Any Source and Destination

Since each path has to traverse one of the intermediate nodes for $e \geq 2$ and we only have $\widetilde{n}_s - 2$ intermediate nodes, the maximum number of link-disjoint paths for the first hop and last hop is $\widetilde{n}_s - 2$.

Next, we show a selection scheme that can indeed find $\widetilde{n}_s - 2$ link-disjoint paths for the intermediate hops.

First, we note that the maximum number of different links that connect two of the $\widetilde{n}_s - 2$ intermediate nodes is $L(\widetilde{n}_s - 2) = (\widetilde{n}_s - 2)(\widetilde{n}_s - 3)/2$. Each node can have at most $\widetilde{n}_s - 3$ links with intermediate nodes. Intermediate nodes are indexed from 0 to $\widetilde{n}_s - 3$. We then select links to construct paths using a scheme that is illustrated in Fig. 12. Following the schema, we select links $(i, (i+1) mod (\widetilde{n}_s - 2))$ in the second hop. In the jth hop, where $j \leq e - 1$, we select links $(i, (j - 1 + i) mod (\widetilde{n}_s - 2))$.

Despite the simplicity, the above-mentioned scheme is a feasible scheme, as we show below. We start with node 0 as an example. In the second hop, node 0 is connected to node 1 and node $(\widetilde{n}_s - 3)$ (i.e., node 4 in Fig. 12) because node $(\widetilde{n}_s - 3)$ is connected to node $(\widetilde{n}_s - 3 + 1)\ mod\ (\widetilde{n}_s - 2) = 0$ according to the above selection scheme as shown in red dashed links in Fig. 12. At the jth hop, node 0 is connected to node $(j-1)$ and node $(\widetilde{n}_s - j - 1)$ because node $(\widetilde{n}_s - j - 1)$ is connected to $(\widetilde{n}_s - j - 1 + j - 1) mod\ (\widetilde{n}_s - 2) = 0$ according to the above selection scheme. Due to the cyclical structure of the selection scheme, if $(\widetilde{n}_s - j - 1) > (j - 1)$, each link associated with other intermediate nodes is also used once. If $j \leq e - 1$, where $e < \frac{\widetilde{n}_s}{2} + 1$, each intermediate link is used once. Therefore, the scheme is a feasible scheme.

Under the above scheme, the paths:

$$\{(s,\ i), (i,\ (i+1) mod (\widetilde{n}_s - 2)), \ldots,$$
$$(i + j - 2, (i + j - 2 + j - 1) mod (\widetilde{n}_s - 2)), \ldots,$$
$$(i + e - 1 - 2,\ (i + e - 1 - 2 + e - 1 - 1) mod (\widetilde{n}_s - 2)),$$
$$((i + e - 1 - 2 + e - 1 - 1) mod (\widetilde{n}_s - 2),\ d)\},$$

$0 \leq i \leq \widetilde{n}_s - 2$ form $\widetilde{n}_s - 2$ link-disjoint paths because each path contains unique links as illustrated in Fig. 12.

TABLE II
LIST OF ACRONYMS

DRART	Dynamic Routing And Random Topology
GA	Genetic Algorithm
InP	Infrastructure Provider
NV	Network Virtualization
RL	Reinforcement learning
SN	Substrate Network
UAV	Unmanned Aerial Vehicle
VN	Virtual Network
VNE	Virtual Network Embedding
VNR	Virtual Network Request

TABLE III
LIST OF NOTATIONS

$b(e_s)$	Bandwidth capacity of e_s
B_{e_s}	Maximum value of $b(e_s)$
$b(e_v)$	Bandwidth requirement of e_v
B_{e_v}	Maximum value of $b(e_v)$
b_r	Average request bandwidth of class r
b_u	The average bandwidth capacity of class u
$\boldsymbol{c_i}$	A chromosome
$c(n_s)$	CPU capacity of n_s
C_{n_s}	Maximum value of $c(n_s)$
$c(n_v)$	CPU capacity requirement of n_v
C_{n_v}	Maximum value of $c(n_v)$
D	Maximum acceptable distance of n_v
$dis(loc(n_v), loc(n_s))$	Mapping distance
E_m	A maximum path length
e_s	A substrate link $e_s \in E_s$
E_s	A set of substrate links
$\widetilde{e}_s$	Number of substrate links
$\bar{e}_s$	Average number of $\widetilde{e}_s$
e_{uv}	A substrate link in a virtual link mapping
e_v	A virtual link $e_v \in E_v$
E_v	A set of virtual links in a VNR
$\widetilde{e}_v$	Number of virtual links in a VNR
$\boldsymbol{g_{ij}}$ or $\boldsymbol{g}$	A gene
$G_s(N_s, E_s)$	An SN
$G_v\,(N_v,\ E_v, t_a, t_d,\ D)$	A VNR
$G_{s'} = (N_{s'}, E_{s'})$	An augmented substrate graph
h_r	Number of requests in class r with b_r
$\widetilde{h}$	A set of all feasible combinations of carried requests in a substrate link
$\boldsymbol{H_{ijk}}$	The head part of a gene $\boldsymbol{g_{ij}}$
K	Maximum number of existing link-disjoint shortest paths for any source-destination pair
K_e	Maximum number of potential paths between two nodes with e links
k_e	Number of paths with length e
$\widetilde{k_e}$	Number of paths with length e between two nodes under K shortest Paths.
$K_{I,e}$	Maximum number of potential link-disjoint paths with length e
$\bar{l}$	Average path length
M	Population size
$M_{s,n}$	Maximum value of $\widetilde{n}_s$
$M_{v,n}$	Maximum value of $\widetilde{n}_v$
$\widetilde{n}_{v,s}$	Number of candidate substrate nodes of n_v
$\widetilde{N}_{v,s}$	Maximum value of $\widetilde{n}_{v,s}$
N_s	A set of substrate nodes
n_s, w_s	A substrate node w_s, $n_s \in N_s$
$\widetilde{n}_s$	Number of substrate nodes
$\bar{n}_s$	Average number of $\widetilde{n}_s$
n_v, m_v	A virtual node m_v, $n_v \in N_v$
N_v	A set of virtual nodes in a VNR
$\widetilde{n}_v$	Number of virtual nodes in a VNR
$\bar{n}_v$	Average number of $\widetilde{n}_v$
$\mathbb{N}_c(n_v)$	a Set of all candidate substrate nodes for mapping a virtual node
$\mathbb{P}$	Population
$P_{b(e_s)}$	Distribution of e_s
$P_{b(e_v)}$	Distribution of e_v
$P_{c(n_s)}$	Distribution of $c(n_s)$
$P_{c(n_v)}$	Distribution of $c(n_v)$
$P_{L_{n_s}}(x, y)$	Distribution of (x_{n_s}, y_{n_s})
$P_{L_{n_v}}(x, y)$	Distribution of (x_{n_v}, y_{n_v})
$\bar{P}_n^{(A)}$	Average acceptance probability for a VN
$P^s(m_s, n_s)$	A set of all substrate paths from node m_s to n_s
P_{N_s}	Distribution of $\widetilde{n}_s$
P_{N_v}	Distribution of $\widetilde{n}_v$
$P_{p,\vec{n}_{s,e-1}}$	Probability of given substrate nodes $\vec{n}_{s,e-1}$ forming a path
$P^{(B)}_{p,\ \vec{n}_{s,e-1}}$	Blocking probability of a given path with e links
P_r	Probability of a request with b_r
$P_{s,l}$	Average probability of forming a link between two substrate nodes
$P^{(B)}_{s,l}$	Blocking probability of a substrate link
$P^{(B)}_{s,n}$	Blocking probability of a substrate node
P_u	Probability of a substrate link with the capacity b_u
$P_{v,l}$	Probability of forming a link between two virtual nodes
$P^{(B)}_{v,l}\,(E_m, K)$	Average blocking probability of a virtual link
$P^{(A)}_{v,n}$	Acceptance probability of a virtual node
R	Number of quantized bandwidth request classes
t_a	Arrival time of a VNR
t_d	Duration of the VNR
$\boldsymbol{T_{ijk}}$	The tail part of a gene $\boldsymbol{g_{ij}}$
U	Number of quantized bandwidth capacity classes
W	A fixed area
(x_{n_s}, y_{n_s})	The coordinate of n_s
$\lambda_{e,l}$	Effective substrate link load
$\lambda_{s,l}$	Mean arrive rate at a substrate link
$\lambda_{s,l,r}$	Average arrival rate of class r at a substrate link
λ_v	VNR arrival rate
τ_v	VNR mean holding time
$\mu(n_v)$	A meta-node of n_v
$\rho_{s,l,r}$	Average class r load at a substrate link

B. Corollary 1: The Lower Bound of Link-Disjoint Paths Between Any Substrate Nodes

From Theorem 1, we know that there are at most $K_{I,e} = \widetilde{n}_s - 2$ link-disjoint paths. In the proof of Theorem 1, we identified a specific selection scheme that has exactly $K_{I,e} = \widetilde{n}_s - 2$ potential link-disjoint paths.

Under this specific scheme, the probability that there exist k_e paths is $C_{k_e}^{K_{I,e}}(P_{p,\vec{n}_{s,e-1}})^{k_e}(1 - P_{p,\vec{n}_{s,e-1}})^{K_{I,e}-k_e}$ based on binomial distribution, where $C_{k_e}^{K_{I,e}}$ is the number of combinations.

It should be noted that there are other selection schemes that have link-disjoint paths different from the $K_{I,e}$ paths identified in the proof of Theorem 1. For example, if we switch the link selections of any two hops in the scheme identified in the proof of Theorem 1, we get $K_{I,e}$ link-disjoint paths that are different from but partially overlapped with those in the scheme identified in the proof of Theorem 1. Finding the link-disjoint paths under various selection schemes is a significant challenge because a) the number of selection schemes can be very large and complex; b) paths between different

selection schemes can be partially overlapped making estimation of probability difficult. However, we can have the lower bound in (15).

C. Theorem 2: Estimating the Number of Link-Disjoint Paths Between Any Source-Destination Pair

For all possible schemes, in order to have at least k_e existing link-disjoint paths, the following conditions must be satisfied:

- The first hop must have at least k_e links that actually exist. From binomial distribution, we have: $\widetilde{P}_I(k_e|\ K_{I,e}) = \sum_{i=k_e}^{K_{I,e}} C_i^{K_{I,e}} {P_{s,l}}^i (1-P_{s,l})^{K_{I,e}-i}$.
- Similarly, the last hop must have at least k_e links that actually exist with the same condition as above.
- The intermediate hops must have at least $k_e(e-2)$ link-disjoint links existing out of all potential links $L(\widetilde{n}_s - 2) = (\widetilde{n}_s - 2)(\widetilde{n}_s - 3)/2$ among the $\widetilde{n}_s - 2$ intermediate nodes, from binomial distribution again, we have: $\check{P}_I(k_e|\ K_{I,e}) = \sum_{i=k_e(e-2)}^{L(\widetilde{n}_s-2)} C_i^{L(\widetilde{n}_s-2)} {P_{s,l}}^i (1-P_{s,l})^{L(\widetilde{n}_s-2)-i}$.

Combining the above three results and noting the fact that the existences of all substrate links are independent, we can get (16).

APPENDIX B

See Table II.

APPENDIX C

See Table III.

REFERENCES

[1] C. Wang, S. Shanbhag, and T. Wolf, "Virtual network mapping with traffic matrices," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2012, pp. 2717–2722.

[2] K. Nguyen and C. Huang, "An intelligent parallel algorithm for Online virtual network embedding," in *Proc. Int. Conf. Comput. Inf. Telecommun. Syst. (CITS)*, 2019, pp. 1–5.

[3] M. Zangiabady, A. Garcia-Robledo, J. Gorricho, and J. Serrat, "Self-adaptive online virtual network migration in network virtualization environments," *Trans. Emerg. Telecommun. Technol.*, vol. 30, no. 9, 2019, Art. no. e3692.

[4] C. Aguilar-Fuster, M. Zangiabady, J. Zapata-Lara, and J. Rubio-Loyola, "Online virtual network embedding based on virtual links' rate requirements," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 4, pp. 1630–1644, Dec. 2018.

[5] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEyard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, Feb. 2012.

[6] K. Nguyen and C. Huang, "Toward adaptive joint node and link mapping algorithms for embedding virtual networks: A conciliation strategy," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 3, pp. 3323–3340, Sep. 2022.

[7] N. Metropolis and S. Ulam, "The Monte Carlo method," *J. Amer. Statist. Associat.*, vol. 44, no. 247, pp. 335–341, 1949.

[8] O. O. Sami, M. Atiquzzaman, A. T. Ahamed, and A. Ibrahim, "Softwarization of UAV networks: A survey of applications and future trends," *IEEE Access*, 8, pp. 98073–98125, 2020.

[9] J. P. Gabhane, S. Pathak, and N. M. Thakare, "Metaheuristics algorithms for virtual machine placement in cloud computing environments—A review," *Computer Networks, Big Data and IoT*. Singapore: Springer, 2021, pp. 329–349.

[10] Y. Wang et al., "Joint resource allocation and UAV trajectory optimization for space–air–ground Internet of Remote Things networks," *IEEE Syst. J.*, vol. 15, no. 4, pp. 4745–4755, Dec. 2021.

[11] H. Thakkar, C. Dehury, and P. Sahoo, "MUVINE: Multi-stage virtual network embedding in cloud data centers using reinforcement learning-based predictions," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1058–1074, Jun. 2020.

[12] H. Cao, L. Yang, Z. Liu, and M. Wu, "Exact solutions of VNE: A survey," *China Commun.*, vol. 13, no. 6, pp. 48–62, Jun. 2016.

[13] I. Houidi, W. Louati, and D. Zeghlache, "Exact multi-objective virtual network embedding in cloud environments," *Comput. J.*, vol. 58, no. 3, pp. 403–415, 2015.

[14] I. Houidi, W. Louati, W. B. Ameur, and D. Zeghlache, "Virtual network provisioning across multiple substrate networks," *Comput. Netw.*, vol. 55, no. 4, pp. 1011–1023, 2011.

[15] L. Shen, M. Wu, and M. Zhao, "Secure virtual network embedding algorithms for a software-defined network considering differences in resource value," *Electronics*, vol. 11, no. 10, p. 1662, 2022.

[16] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 17–29, 2008.

[17] P. Zhang, H. Yao, C. Qiu, and Y. Liu, "Virtual network embedding using node multiple metrics based on simplified ELECTRE method," *IEEE Access*, 6, pp. 37314–37327, 2018.

[18] H. Cao, Y. Zhu, L. Yang, and G. Zheng, "A efficient mapping algorithm with novel node-ranking approach for embedding virtual networks," *IEEE Access*, vol. 5, pp. 22054–22066, 2017.

[19] C. Aguilar-Fuster and J. Rubio-Loyola, "A novel evaluation function for higher acceptance rates and more profitable metaheuristic-based Online virtual network embedding," *Comput. Netw.*, vol. 195, Aug. 2021, Art. no. 108191.

[20] M. Diallo, A. Quintero, and S. Pierre, "An efficient approach based on ant colony optimization and tabu search for a resource embedding across multiple cloud providers," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 896–909, Jul.-Sep. 2021.

[21] C. Wang, R. Batth, P. Zhang, G. Aujla, Y. Duan, and L. Ren, "VNE solution for network differentiated QoS and security requirements: From the perspective of deep reinforcement learning," *Computing*, vol. 103, no. 6, pp. 1061–1083, 2021.

[22] P. Zhang, H. Yao, and Y. Liu, "Virtual network embedding based on the degree and clustering coefficient information," *IEEE Access*, vol. 4, pp. 8572–8580, 2016.

[23] S. Prekas, P. Karkazis, V. Nikolakakis, and P. Trakadas, "Comprehensive comparison of VNE solutions based on different coordination approaches," *Telecom*, vol. 2, no. 4, pp. 390–412, 2021.

[24] K. Nguyen, Q. Lu, and C. Huang, "Joint node-link embedding algorithm based on genetic algorithm in virtualization environment," in *Proc. IEEE Veh. Technol. Conf.*, 2021, pp. 1–5.

[25] X. Mi, X. Chang, J. Liu, L. Sun, and B. Xing, "Embedding virtual infrastructure based on genetic algorithm," in *Proc. 13th Int. Conf. PDCAT*, 2012, pp. 239–244.

[26] H. Cao, S. Wu, Y. Guo, H. Zhu, and L. Yang, "Mapping strategy for virtual networks in one stage," *IET Commun.*, vol. 13, no. 14, pp. 2207–2215, 2019.

[27] H. Yu, V. Anand, C. Qiao, D. Hao, and X. Wei, "A cost efficient design of virtual infrastructures with joint node and link mapping," *J. Netw. Syst. Manag.*, vol. 20, no. 1, pp. 97–115, 2012.

[28] P. G. Harrison and N. M. Patel, *Performance Modelling of Communication Networks and Computer Architectures (International Computer S*. Boston, MA, USA: Addison-Wesley Longman Publ. Co., Inc., 1992.

[29] F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios, "Open, closed, and mixed networks of queues with different classes of customers," *J. ACM*, vol. 22, no. 2, pp. 248–260, 1975.

[30] M. Harchol-Balter. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge, U.K.: Cambridge Univ. Press, 2013.

[31] M. Melanie. *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998.

[32] I. Pathak and D. P. Vidyarthi, "A model for virtual network embedding across multiple infrastructure providers using genetic algorithm," *Sci. China Inf. Sci.*, vol. 60, no. 4, Mar. 2017, Art. no. 40308.

[33] P. Zhang, H. Yao, M. Li, and Y. Liu, "Virtual network embedding based on modified genetic algorithm," *Peer-to-Peer Netw. Appl.*, vol. 12, no. 2, pp. 481–492, 2019.

[34] C. W. Huang, C. A. Shen, C. Y. Huang, T. L. Chin, and S. H. Shen, "An efficient joint node and link mapping approach based on genetic algorithm for network virtualization," in *Proc. IEEE 90th Veh. Technol. Conf. (VTC-Fall)*, pp. 1–5, 2019.

[35] Q. Lu, K. Nguyen, and C. Huang, "Distributed parallel algorithms for Online virtual network embedding applications," *Int. J. Commun. Syst.*, vol. 36, no. 1, 2020, Art. no. e4325.
[36] R. A. Barry and P. A. Humblet, "Models of blocking probability in all-optical networks with and without wavelength changers," in *Proc. INFOCOM*, vol. 2, 1995, pp. 402–412.
[37] N. Antunes, C. Fricker, P. Robert, and D. Tibi, "Analysis of loss networks with routing," *Ann. Appl. Probabil.*, vol. 16, no. 4, pp. 2007–2026, 2006.
[38] C. Huang and J. Zhu, "Modeling service applications for optimal parallel embedding," *IEEE Trans. Cloud Comput.*, vol. 6, no. 4, pp. 1067–1079, Oct.–Dec. 2018.
[39] Q. Lu and C. Huang, "Distributed parallel VN embedding based on genetic algorithm," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, 2019, pp. 1–6.
[40] H. Kobayashi and B. L. Mark, "Generalized loss models and queueing-loss networks," *Int. Trans. Oper. Res.*, vol. 9, no. 1, pp. 97–112, 2002.
[41] Z. Drezner and N. Farnum, "A generalized binomial distribution," *Commun. Statist. Theory Methods*, vol. 22, no. 11, pp. 3051–3063, 1993.
[42] A. S. Asratian, T. M. Denley, and R. Häggkvist. *Bipartite Graphs and Their Applications*, vol. 131. Cambridge, U.K.: Cambridge Univ. Press, 1998.
[43] S. Haeri and L. Trajkovic, "Virtual network embedding via Monte Carlo tree search," *IEEE Trans. Cybern.*, vol. 48, no. 2, pp. 510–521, Feb. 2018.
[44] L. Kocsis and C. Szepesvári, "Bandit based Monte-Carlo planning," in *Proc. European Conf. Mach. Learn.*, 2006, pp. 282–293.
[45] E. A. Van Doorn and G. J. K. Regterschot, "Conditional pasta," *Oper. Res. Lett.*, vol. 7, no. 5, pp. 229–232, Oct. 1988.
[46] Z. Xiao et al., "LEO satellite access network (LEO-SAN) towards 6G: Challenges and approaches," *IEEE Wireless Commun.*, early access, Dec. 5, 2022, doi: 10.1109/MWC.011.2200310.