2019-05-02

Probabilistic Virtual Network Embedding under Demand Uncertainty

Hosseini, Fatemeh

UNIVERSITY OF CALGARY

Probabilistic Virtual Network Embedding under Demand Uncertainty

by

Fatemeh Hosseini

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE

CALGARY, ALBERTA

MAY, 2019

Abstract

This thesis investigates the problem of mapping virtual networks onto physical resources where bandwidth demand is uncertain, since, in real world, traffic demands fluctuate significantly over time. Hence, we consider the problem of mapping virtual links to physical paths subject to a constraint on each virtual link congestion probability under the assumption that bandwidth demands of virtual links are uncertain. The problem is formulated as a non-convex optimization problem. Consequently, an approximate formulation is proposed and this results in a second-order cone program that can be solved efficiently for large networks. Also, an existing virtual node embedding algorithm augmented by the proposed link embedding solution is used in simulations and experiments to show the utility and efficiency of our models in various network scenarios. Our results show that both exact and approximate models satisfy the link congestion constraint, and that the approximate model is very close to the exact model.

Preface

This thesis is an original work by the author and has been published [F. Hosseini, A. James and M. Ghaderi, "Congestion-constrained virtual link embedding with uncertain demands", in Proc. IEEE/ACM/IFIP International Conference on Network and Service Management (CNSM), Rome, Italy, November 2018.]

Acknowledgements

I would like to express my very great appreciation to Dr. M. Ghaderi for his valuable and constructive suggestions during the planning and development of this research work. His willingness to give his time so generously has been very much appreciated. I could not have imagined having a better advisor and mentor for my Master study.

Table of Contents

Abstract ii

Preface iii

Acknowledgements iv

Table of Contents v

List of Figures and Illustrations vii

List of Tables viii

List of Symbols, Abbreviations and Nomenclature ix

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Problem Definition . . . 5
- 1.3 Thesis Contribution . . . 6
- 1.4 Thesis Organization . . . 7

2 Related Work 9
- 2.1 The Deterministic VNE . . . 11
  - 2.1.1 Exact Approaches . . . 12
  - 2.1.2 Heuristic Approaches . . . 13
- 2.2 Stochastic VNE . . . 16
  - 2.2.1 Exact Approaches . . . 18
  - 2.2.2 Heuristic Approaches . . . 19
- 2.3 Robust VNE . . . 20
  - 2.3.1 Exact Approaches . . . 20
  - 2.3.2 Heuristic Approaches . . . 21
- 2.4 Summary . . . 23

3 Mathematics Preliminaries 24
- 3.1 Optimization . . . 24
- 3.2 Optimization under Uncertainty . . . 26
- 3.3 Stochastic Optimization . . . 28

3.4 Chance-Constrained Optimization . . . 29
3.5 Robust Optimization . . . 34
3.6 Thesis Approach . . . 37
3.7 Summary . . . 39

4 Virtual Link Embedding **41**
4.1 System Model and Assumptions . . . 41
4.2 Exact and Approximate VLE . . . 44
4.2.1 Congestion on Physical Links . . . 45
4.2.2 Congestion on Physical Paths . . . 45
4.3 Exact VLE Formulation . . . 46
4.4 Approximate VLE Formulation . . . 48
4.5 Performance Analysis of Exact Model vs. Approximate Model . . . 51
4.5.1 Simulation Parameters . . . 52
4.5.2 Performance Measures . . . 53
4.5.3 Exact and Approximate Models . . . 54
4.5.4 Comparison of End-to-End and Link-by-Link Congestion-Constrained Models . . . 56
4.5.5 Effect of Network Parameters . . . 57
4.5.6 Mininet Experiments . . . 61
4.6 Summary . . . 63

5 Virtual Network Embedding **64**
5.1 Virtual Network Embedding Formulation . . . 64
5.2 Virtual Node Embedding Algorithm . . . 66
5.3 Testbed Experiments . . . 71
5.4 Summary . . . 83

6 Conclusions and Future Work **84**
6.1 Summary and Conclusions . . . 84
6.2 Future Research Directions . . . 86

Bibliography **88**

List of Figures and Illustrations

1.1 Network virtualization architecture. 3
1.2 Virtual link to physical path mapping. 6

2.1 VNE Classification. 11

3.1 Perturbation Analysis. 27

4.1 The approximate physical link congestion probability. 50
4.2 Small-scale network topologies. 52
4.3 Average number of admitted virtual links with exact and approximate models for different uncertainty levels. 55
4.4 Congestion probability of exact and approximate models. 55
4.5 Comparison of link-by-link and end-to-end congestion-constrained models. . 57
4.6 Effect of the number of paths (K). 58
4.7 Effect of the physical link capacity (C). 59
4.8 Effect of the number of virtual links on utilization. 59
4.9 Effect of the network topology on admitted links. 59
4.10 Mininet experiments - Average number of admitted virtual links. 60
4.11 Mininet experiments - Average link utilization of physical links. 60
4.12 Mininet experiments - CDF of packet drop probability. 61

5.1 CPU and Bandwidth capacity of nodes. 69
5.2 Ranking nodes in 2-dimension. 70
5.3 Constructing NMT for a VN. 71
5.4 Actual testbed implementation. 72
5.5 Testbed configuration. 75
5.6 First trial - Average number of admitted virtual links. 77
5.7 First trial - Average link utilization of physical links. 78
5.8 First trial - CDF of packet drop probability. 78
5.9 Second trial - Average number of admitted virtual links and requests. 81
5.10 Second trial - Average link utilization of physical links. 81
5.11 Second trial - CDF of packet drop probability. 82
5.12 Third trial - Average link utilization of physical links. 82
5.13 Third trial - CDF of packet drop probability. 83

List of Tables

2.1 Taxonomy of VNE Approaches 12

4.1 Principal Notation Used in the Paper. 42
4.2 Default simulation parameters. 53

5.1 DELL PowerEdge R330 server specification. 73
5.2 Aruba 2930F 24G 4SFP Switch (JL259A) specification. 74

List of Symbols, Abbreviations and Nomenclature

Symbol or abbreviation	Definition
ILP	Integer Linear Programming or Program
ISP	Internet Service Provider
MILP	Mixed-Integer Linear Programming or Program
MIP	Mixed-Integer Programming or Program
LP	Linear Programming or Program
OVS	OpenFlow Virtual Switch
QoS	Quality of Service
RO	Robust Optimization
SO	Stochastic Optimization
SOCP	Second-Order Cone Programming
SDN	Software-Defined Networking
SDP	Semi-Definite Program
VNE	Virtual Network Embedding
VN	Virtual Network
VNR	Virtual Network Request
VLE	Virtual Link Embedding
VM	Virtual Machine

Chapter 1

Introduction

1.1 Motivation

Since 1994, the Internet has developed as a remarkable packet-switched network, extended to billions of users worldwide due to its ease of use. However, the Internet's popularity has become an obstacle to its further growth. Necessary changes have been deployed very slowly, since essential modifications require the coordination and consensus of Internet Service Providers (ISPs), all of which are in competition [25, 53]. For instance, protocols, architectures, and algorithms must be modified to enhance Internet properties, i.e., availability, performance, and QoS guarantee, which is not possible by applying changes only to end users. Rather, hardware and architecture alterations are required [53].

An overlay network is one existing approach to implementing incremental updates and evaluating changes using live traffic. Overlay networks have been typically deployed over IP protocol, and this offers various features such as robust routing architecture, authentication, scalability, and fault tolerance [32]. An overlay network consists of overlay or virtual nodes and links. For example, PlanetLab [5], a geographically distributed computing platform, has leveraged this approach to develop and evaluate planetary-scale network services. This approach, however, cannot resolve Internet ossification problems for two reasons; first, every

overlay network is designed to address a specific architectural issue, and coordination of different overlay networks is not defined, in order to shape the whole architecture; second, overlay networks are mostly implemented in the application layer and the network layer remains untouched [53].

Network virtualization has been introduced as a promising approach to address overlay network deficiencies and support coexistence of multiple virtual networks on the shared physical network by means of determined isolation mechanisms. When virtual networks and end-to-end services are separated from infrastructure, service providers and infrastructure providers replace ISPs. Accordingly, service providers can focus on the quality of service without being concerned about infrastructure management, and this facilitates future innovations and modifications.

Network virtualization is composed of two main components - a **physical network** and a **virtual network**. It is assumed that there is a single **physical network** specified by an undirected graph, where physical nodes and links have fixed capacities. A **virtual network** is also a weighted undirected graph with a given set of nodes and links. The weight of each link indicates the (uncertain) bandwidth demand of that link. Figure 1.1 shows two virtual networks - **VN1** and **VN2** - mapped to the substrate network **SN**.

In the real implementation of network virtualization, an abstraction layer between physical and virtual networks is described. Virtual networks can possess any desired topology; each virtual network can define its own addressing scheme. Virtual networks are isolated; this means virtual nodes from different virtual networks are not accessible, and the required QoS is provided independently.

A virtual machine (VM) on a physical server represents a virtual node with required processing power, and a virtual link connecting a pair of virtual nodes is mapped to the corresponding physical path(s). **Mapping virtual nodes to substrate nodes (servers)** and **mapping virtual links to substrate path(s)** are two main parts of the mapping algorithm, also known as Virtual Network Embedding (VNE) algorithm. These algorithms

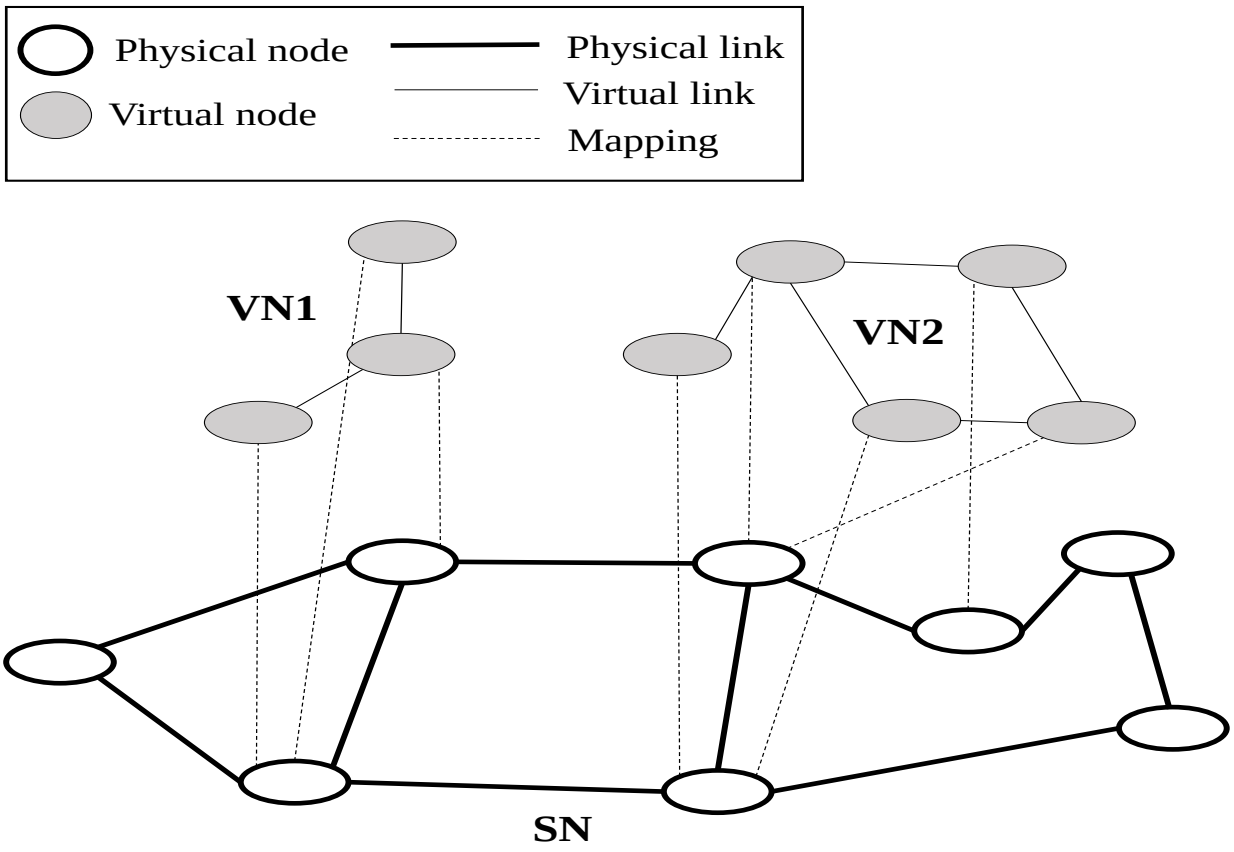


Figure 1.1: Network virtualization architecture.

aim to find the optimal and feasible embedding solution, based on virtual network requests, available physical resources, and QoS constraints.

The ultimate purpose of sharing infrastructure can be divided into the following objectives: 1) flexibility, 2) manageability, 3) scalability, 4) isolation and security, and 5) experimental and deployment facility [25].

Flexibility. Flexibility is the consequence of separating policy from mechanism. Decoupling the mechanism implementations from the policy specifications makes it possible for different services to use the same mechanisms with different policies [6]. Network virtualization abstracts lower level mechanisms so that different tenants can choose a desired network topology, routing protocol, and addressing scheme.

Manageability. By separating infrastructure providers from service providers, end-to-end

services for virtual networks can be managed independently of physical resources. Managing virtual resources is easier than managing hardware devices.

Scalability. Network virtualization is used to facilitate cohabitation of multiple virtual networks on the same substrate network. As many virtual networks as possible must therefore be embedded into the substrate network without violating physical capacities and pre-defined constraints.

Isolation and security. Defining isolation mechanisms helps to keep virtual networks secure. Transmitted information through a virtual network is confidential and must be protected against faults and attacks from other virtual networks. Also, misconfiguration of a virtual network should not hinder the functionality of other virtual networks.

Experimental and Deployment Facility. Without network virtualization, distributed network services are evaluated in small and simple lab testbeds. Conducting experiments on a network of realistic properties and size is expensive and may be impossible in any case. However, deploying virtual networks with specific topologies and characteristics on top of the infrastructure network enables experiments to be run easily and safely.

Furthermore, the critical issue in VNE algorithms is sharing physical resources (processing power and bandwidth) efficiently among virtual networks based on a specific service-relevant metric, for example, minimizing the bounded end-to-end packet loss probability and the cost of mapping [33]. The objective of this thesis is to find optimum mapping with a balanced load between physical end-to-end paths and to increase the acceptance ratio of the VNE algorithm. Different from most existing VNE solutions, where exact knowledge of traffic demands is available, we assume that only partial knowledge is available.

In the real world, bandwidth demand or aggregated traffic of different services is not constant and may change frequently, depending on the type of service and user. Multimedia applications, e.g., Youtube and Netflix, are acclaimed for their high volatile demands. The deviation of flow rates from their nominal values over time can cause significant deviation of aggregate traffic and, consequently, congestion occurs at physical links. Since the embedding

algorithm computes a mapping that probabilistically satisfies virtual network demands, no feasible mapping can be found in specific cases, such as when all flows fluctuate simultaneously, where physical resources are insufficient to meet demands. Congestion is detrimental to the performance of virtual networks (VNs); hence, it is critical to consider them when computing a virtual-to-physical resource mapping under demand uncertainty. Accordingly, the probabilistic VNE problem subject to congestion constraints is investigated in this thesis.

1.2 Problem Definition

The offline version of the VNE problem, where virtual network requests have arrived during the previous time slot and are processed as a batch at the beginning of the current time slot, is considered. Each VN requests a deterministic amount of computing resources for virtual nodes and uncertain amount of bandwidth for virtual links, i.e., only mean and variance of bandwidth demands are given. Hence, we consider a probabilistic embedding problem consisting of virtual-to-physical node-to-node mapping and link-to-path mapping, where bandwidth demands are uncertain and congestion probability for each virtual link is guaranteed to be less than a predefined value.

In this context of virtual link-to-path mapping, guaranteeing congestion probability for a virtual link is reduced to guaranteeing the end-to-end congestion probability on the corresponding path. Existing works, [44], only ensure that the probability of congestion on each physical link is bounded. However, the congestion probability achieved for the corresponding virtual link depends on the structure of the underlying path and, hence, is not guaranteed. Figure 1.2 depicts a simple example, where the virtual link ℓ between virtual nodes v_1 and v_2 is mapped onto $P = \langle e_1, e_2, e_3, e_4 \rangle$, a path in the physical network, and this means that v_1 and v_2 are placed on n_1 and n_2. Suppose then that the pre-specified target congestion probability for the virtual link ℓ is given by ϵ and let ε_k denote the congestion probability on physical link e_k. To satisfy the virtual link congestion requirement ϵ, the following constraint

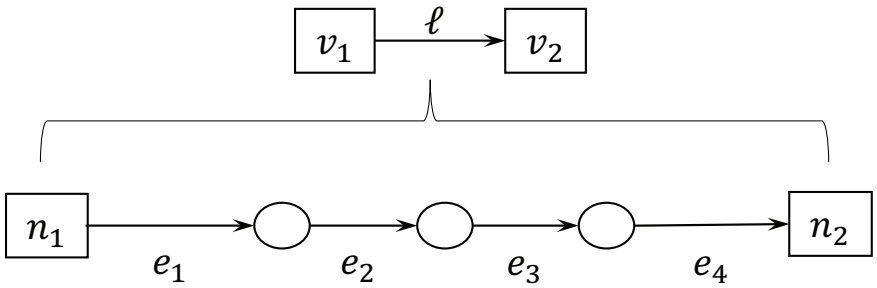


Figure 1.2: Virtual link to physical path mapping.

should be satisfied.

$$1 - \prod_{e_k \in P}(1 - \varepsilon_k) \leq \epsilon\,. \tag{1.1}$$

In this equation, ε_k is a function of the total resource demand on link e_k, and it should be optimally determined by the embedding algorithm.

The complexity of the problem arises as follows: 1) the optimal path(s) connecting the physical nodes n_1 and n_2 is not known in advance, and 2) the bandwidth demand on each link and, consequently, the link congestion probabilities are functions of the unknown path(s). Put differently, the problem expresses a joint optimization of path selection and link congestion allocation, and this optimization problem is non-convex, non-linear, and computationally intractable. Related research assumes that guaranteeing a fixed congestion probability ε for every physical link e_k is sufficient. However, the actual end-to-end congestion probability achieved by this research depends on the length of the physical path and, hence, cannot be guaranteed.

Consequently, the objective is to design a virtual link embedding (VLE) algorithm augmented with an existing node embedding algorithm. As such, VLE considers uncertain bandwidth demands, provides guaranteed end-to-end congestion probability, and can be solved in a reasonable time for large-scale networks.

1.3 Thesis Contribution

Contributions in this work can be summarized as follows:

- Demand uncertainty is considered in a flexible and general model where only limited information about requests, i.e., mean and variance, is needed;
- The link mapping problem is formulated with constrained end-to-end congestion probability as a non-linear optimization problem that can be solved using global optimization solvers for small network instances;
- An approximate link mapping solution is proposed for the problem, which is formulated as a second-order cone program (SOCP) that can be solved efficiently, even for large network instances;
- Both simulation and Mininet [4] experimental results are presented to show the efficiency and utility for solutions in small and large network instances;
- The link embedding solution is conjoined with an existing virtual node embedding approach in order to evaluate the solution in a complete framework using more realistic scenarios;
- The joint link and node embedding solution is implemented on a real testbed, which includes SDN controller and OpenFlow switches, to analyse performance of virtual networks.

1.4 Thesis Organization

This thesis is divided into six chapters, including "Introduction". The remaining chapters are as follows:

- **Chapter 2** (Related Work) In this chapter, a literature review of previous studies in deterministic, stochastic, and robust virtual network embedding is presented.
- **Chapter 3** (Mathematics Preliminaries) Stochastic and robust optimizations, two forms of probabilistic optimization, are explained. It is shown that stochastic and

robust models cannot simply guarantee end-to-end congestion probability for the VNE problem with uncertain demands. To do this, concentration or tail bounds are also explained in this chapter.

- **Chapter 4** (Virtual Link Embedding) Placement of virtual nodes in the substrate network is assumed as known and determined. The link mapping problem is formulated with constrained congestion probability as a non-linear optimization model which can be solved using global optimization solvers for small networks. To tackle the complexity in the non-linear model, an approximate link mapping solution is presented, which is formulated as a second-order cone program (SOCP) and can be solved efficiently even for large-scale networks. Last, simulations and Mininet experimental results are presented to evaluate the efficiency and utility of solutions in both small and large network scenarios.

- **Chapter 5** (Virtual Network Embedding) The general virtual network embedding problem is modelled and shown to be NP-hard. Hence, the VNE problem is broken down into two sub-problems. In the first sub-problem, a node embedding algorithm embeds virtual nodes into physical nodes based on a defined metric. In the second sub-problem, the proposed link mapping solution (in Chapter 4) embeds virtual links with uncertain demands and guarantees end-to-end congestion probability. As the link mapping solution can be augmented by any node embedding algorithm, a node embedding algorithm is selected from related works. Finally, the complete VNE solution to demonstrate the feasibility and efficiency of the proposed solution is examined on a real testbed in realistic scenarios.

- **Chapter 6** (Conclusions and Future Work) This chapter reviews the presented solution for virtual network embedding, concluding remarks, and future directions in the final chapter.

Chapter 2

Related Work

This chapter provides a survey of virtual network embedding algorithms. VNE problems fall into different categories, depending on the type of demand (deterministic/uncertain), coordination of virtual network embedding phases (coordinated/uncoordinated), and the type of solution (exact/heuristic). For deterministic demands, the exact amount of physical resource request (processing power and bandwidth) is known. This category shapes a large body of the related work. However, virtual network demands are uncertain in most real scenarios, that is, mean and variance or range of demand is given. In this case, deterministic solutions allocate physical resources based on the worst-case or average demands for each VN. However, both of these approaches lead to significant over or under-utilization of physical resources. This has motivated significant studies on probabilistic virtual network embedding algorithms. This research assumes either the distribution of resource demand is known (e.g., it follows a Normal distribution) or only limited information about demands (e.g., the range of demands) is given.

Hence, VNE solutions with uncertain demands can be classified into two groups: 1) VNE algorithms that allocate resources stochastically, where the probability distribution of requests is known, and 2) VNE algorithms that leverage robust optimization, where more uncertainty is engaged. In either case, physical resources are allocated probabilistically,

and congestion can happen on the physical nodes and links, where the real traffic rate and requested processing power deviate significantly from their nominal values. As mentioned in Chapter 1, congestion degrades network performance and quality of service, and it is critical to consider it in any VNE solution. Consequently, related studies are mainly investigated with regard to the type of demand and how congestion is handled in probabilistic solutions.

The next classification is performed based on the coordination of VNE phases. As a VNE algorithm consists of two steps - virtual node and link mapping - we can categorize them into two types of algorithms: first, uncoordinated algorithms where the virtual node and link mapping are handled separately in two steps; and second, coordinated algorithms where the virtual node and link mapping are joint in one or two stages with coordination. As inefficient as uncoordinated algorithms are, in order to allocate physical resources, most recent studies have addressed coordinated algorithms. Thus, this thesis focuses on coordinated VNE algorithms.

The proposed problems can be solved using exact and heuristic models. Exact VNE solutions refer to linear or quadratic programs that provide a reference for evaluating the performance of their corresponding heuristic solution [18], but they are generally computationally very expensive and can be applied only to very small networks. Heuristic approaches can find feasible and close-to-optimal solutions in a reasonable time when facing realistic network scenarios. Some research has adopted both exact and heuristic approaches. They model the exact VNE problem which is NP-hard, and then propose heuristic algorithms to solve the problem efficiently for large networks. But, they are classified as heuristic solutions. Figure 2.1 depicts the classification of VNE algorithms discussed in this thesis.

As well, virtual network embedding algorithms can be classified as static/dynamic, concise/redundant, and centralized/distributed solutions. Static embedding considers a set of virtual network requests (VNRs) at one time in an offline manner, where reconfiguration of one or more embedded VNRs is not possible. However, dynamic embedding maps VNRs on the fly as they arrive and contemplates changes in virtual networks. For instance, topologies

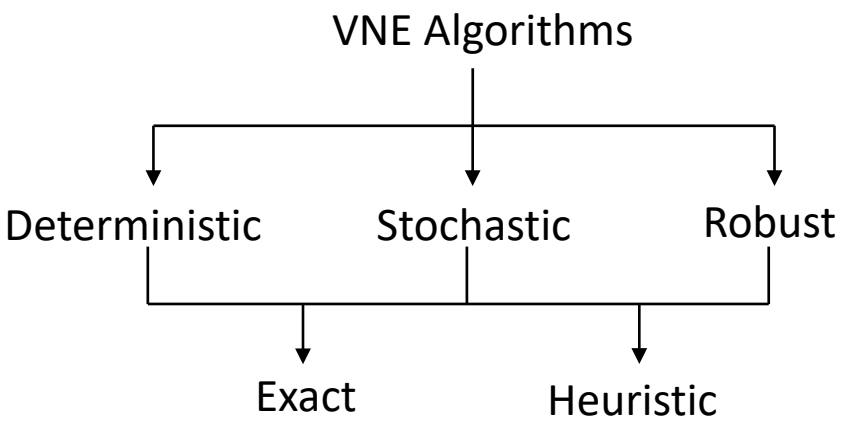


Figure 2.1: VNE Classification.

and demands may change from time to time and here VNRs need to be remapped because of substrate network updates.

Concise and redundant are another set of VNE algorithms. The former aims to minimize substrate resource usage, but the latter combines multiple substrate resources to satisfy one virtual resource. The last category refers to centralized and distributed embedding. In centralized embedding, one entity or controller has global knowledge and conducts mapping to maximize network performance. However, it poses several problems, such as single point of failure and scalability issues in large networks, both of which can be addressed by distributed solutions. For a comprehensive review of the virtual network embedding, the reader is referred to [33]. This thesis investigates VNE algorithms based on types of VNRs and corresponding deterministic/probabilistic solutions. A complete classification of investigated studies is presented in Table 2.1 below.

2.1 The Deterministic VNE

The virtual network embedding problem is NP-hard; it is reduced to the multiway separator problem that tries to map logical nodes to physical nodes by considering the requested bandwidth [9]. Mapping virtual nodes independently is reduced to a simple Integer Linear Program (ILP). As well, a virtual link embedding phase can be modelled as an ILP in the

Table 2.1: Taxonomy of VNE Approaches

Category	Reference	Exact/ Heuristic	Static/ Dynamic	Centralized/ Distributed
Deterministic	[40, 50]	Exact	Dynamic	Centralized
	[16, 57, 58]	Exact	Static	Centralized
	[20, 61–63]	Heuristic	Dynamic	Centralized
	[21, 23, 24, 45, 47, 48]	Heuristic	Static	Centralized
	[39]	Heuristic	Dynamic	Distributed
Stochastic	[46, 55, 60]	Exact	Static	Centralized
	[52]	Heuristic	Dynamic	Centralized
Robust	[44]	Exact	Static	Centralized
	[27, 28]	Heuristic	Static	Centralized

case of single path routing and as an LP model in the case of multipath routing. Virtual node embedding is NP-hard, even without taking bandwidth constraints into consideration. But, the virtual link embedding stage can have polynomial time complexity by allowing flows to split among paths. Hence, most related studies tackled the VNE problem with heuristic solutions that are computationally tractable for large networks.

2.1.1 Exact Approaches

Authors in [50] proposed a Mixed-Integer Linear Program (MILP) algorithm that jointly optimizes node placements and link embeddings integrated with cost-aware migrations. This algorithm considers dynamic VN requests that (1) may arrive over time and may leave at any time, (2) they are dynamic, and (3) origins of requests may change where user mobility or time-zone effects are defined.

Houidi et al. [40] embedded VNs across multiple substrate networks. As a first step, VN provider split VN request graphs into sub-graphs, and each VN sub-graph is sent to the appropriate infrastructure provider. Then, the dynamic VNE algorithm is performed separately for each VN. The problem was formulated as a Mixed Integer Program (MIP), with the objective of decreasing embedding costs for infrastructure providers and increasing the acceptance ratio. Evaluations were conducted over small- and medium-sized networks

with around 50 nodes due to the unscalable MIP model.

Botero et al. [16] proposed an MIP to model the energy-aware VNE problem with given bandwidth and processing power demands required by virtual nodes. The proposed model is NP-complete and unscalable when applies to large scenarios, due to integer variables, but it defines the optimal bound for heuristic approaches. In [57, 58], an overlay mapping model was developed by formulating the VNE problem as an Integer Linear Program. Furthermore, the proposed model was enhanced by considering substrate topology information as a way to provide effective resilience. In the enhanced model, two aspects were considered. First, the working and backup paths have been chosen in such a manner that they avoid common links in the substrate; second, the overlap of two overlay links have been minimized so as to reduce the effect of a single substrate link failure on the mapped overlay links.

2.1.2 Heuristic Approaches

Zhu et al. [63] presented heuristic on-demand or dynamic VNE algorithms to achieve a balanced load on both substrate nodes and links. They defined neighbourhood resource availability as a metric for balancing the workload. The metric was measured as the product of (deterministic) residual capacity of the physical node and available bandwidth on directly connected physical links. Accordingly, a basic algorithm was developed that considers this metric during the virtual node embedding stage. Virtual links were then mapped to substrate paths with the shortest distances. In addition, they provided an improved version of the basic algorithm. As the VN topology becomes larger and more sophisticated, node assignment and load balancing become more difficult when the whole VN is considered at once. As a result, the VN was broken up into a number of connected sub-virtual networks (subVN) to utilize flexibility of small topologies. Each subVN was then mapped onto the substrate network, based on the basic algorithm.

Similarly, Yu et al. [61] introduced a heuristic online algorithm to map VNs onto a subset of the physical network that satisfies VNs' deterministic demands, for example, bandwidth,

CPU capacity, and constraints on the location of virtual nodes. The proposed online algorithm gathers a group of incoming requests during a timeslot, different from [63], where requests are serviced one by one. Some requests might be deferred and returned to the request queue due to lack of bandwidth and CPU capacity, and other requests in the queue are dropped if they cannot be served within a specified deadline. Last, all VN requests arriving in the request queue within the timeslot are processed in decreasing value of their revenues. Similar to [63], the coordination of node and link embedding algorithms is achieved by considering available bandwidth on adjacent physical links at the node embedding stage. After node placement, virtual links are embedded based on the k-shortest path algorithm.

Chowdhury et al. [23, 24] decomposed the static VNE problem to two steps: virtual node and virtual link mapping. Since embedding virtual nodes without considering link constraints reduces the solution space, they presented a VNE solution with coordinated steps, using virtual links' bandwidth constraints in the virtual node mapping phase. The virtual node embedding problem was modelled as MIP and then relaxed to a linear program with rounding techniques, i.e., randomized and deterministic rounding techniques. Therefore, the model can be applied to large network instances. Once all virtual nodes have been embedded, the multi-commodity flow algorithm is used to map virtual links. The second phase can be solved in polynomial time because there are no integer variables in multi-path routing.

Houidi et al. [39] proposed a decentralized fault-tolerant VN embedding algorithm to cope with dynamic changes in service demands. They conducted VN provisioning in four steps: 1) resource description and advertisement, 2) resource discovery and matching, 3) VN embedding, and 4) VN binding. In the first step, infrastructure providers advertise available physical resources. The resource discovery and matching step consists of searching and finding resource candidates that must meet requirements specified by the VN request. Once candidate resources are identified, the infrastructure provider assigns virtual nodes and links to a specific set of substrate nodes and paths. Virtual networks are maintained adaptively when substrate nodes/links fail and VN demands change over time, such that new candidate

resources are selected to replace those that are no-longer available and usable. Likewise, Chen et al. [20] and Yu et al. [62] presented VNE algorithms with resiliency guarantees, in which disjointed primary and restoration physical paths are allocated to each virtual link.

Cheng et al. [21] introduced two new heuristics that increase the acceptance ratio and the overall revenue of the physical substrate. This paper is distinguished from works defining a node embedding metric, e.g., [47, 61, 63], by applying the successful PageRank algorithm to VNE problems. In the NodeRank algorithm, both virtual and physical nodes are ranked based on the availability of physical resources and topological characteristics of the substrate network.

Two VNE algorithms were devised - RWMaxMatch and RW-BFS - based on NodeRank. RWMaxMatch is a two-stage VN embedding algorithm in which virtual nodes and links are mapped in two stages. This results in higher substrate resource consumption and, thus, RW-BFS was proposed to map virtual nodes and links during the same stage. A Breadth-first search (BFS) tree is constructed, where the root node is the virtual node with the largest NodeRank value and nodes are sorted in a non-increasing order at each level of the search tree. An ordered candidate substrate node list is provided for each virtual node, consisting of the nodes whose available physical resources are at least as large as those of the virtual node.

Finally, every virtual node is embedded into the substrate node that meets the CPU resource requirement, and directly connected virtual links are mapped onto the substrate paths, satisfying bandwidth and connectivity requirements, using the shortest path algorithm. If there is no suitable mapping for a virtual node in its candidate substrate node list, the algorithm backtracks to the previous virtual node, re-maps it to another substrate node, and continues mapping. Additionally, a hop bound was defined, i.e., the maximum number of hops between two directly connected virtual nodes on the substrate network, to reduce the BFS search space and avoid placing virtual nodes too far away from those already mapped and saving substrate link resources.

Li et al. [45] proposed a heuristic algorithm that allocates datacenter resources to workloads. Existing heuristic mapping algorithms, such as one proposed by [21], evaluates the substrate resources according to one resource factor or the product of resource factors; however, this leads to an imbalance resource allocation and inefficient resource utilization. Therefore, the presented algorithm in [45], called TK-Match, employs the top-k dominating model to rank nodes based on the effects of both CPU and bandwidth requirements. This algorithm is used to complete our VNE framework in this thesis and, thus, a detailed description of TK-Match is provided in Chapter 5.

2.2 Stochastic VNE

Roots of the stochastic resource allocation can be traced back to the literature on effective capacity concept [35]. Networks must support different bandwidth requirements, i.e., traffic characteristics and their dynamic nature and, thus, efficient bandwidth allocation is difficult. It is important to characterize the aggregate effective bandwidth utilization of connections based on the desired grade or level of service.

In [35], an approximate equivalent capacity of a single connection and multiplexed connections were computed based on their statistical characteristics. Two complementary approaches have been used to capture the main aspects of the connections' behaviour, one fluid-flow model, estimating the equivalent capacity for a single connection and, second, an approximation of the stationary bit rate distribution, representing bandwidth requirements of multiplexed connections.

The two-state fluid-flow model represents a source with idle and burst states. Then, the distribution of idle and peak rates, as well as the peak rate of a connection identify traffic characteristics of the corresponding connection. Hence, equivalent capacity of a single connection ($\hat{c}_i$) can be calculated based on the peak rate, the mean burst period, and the utilization. If the aggregate traffic (without considering the effect of multiplexed connections

- $\hat{C}_{(F)}$) is approximated by the summation of equivalent capacities of single connections, then we have:

$$\hat{C}_{(F)} = \sum_{i=1}^{N} \hat{c}_i \tag{2.1}$$

The second approach ensures that the aggregate bit rate of multiplexed connections (B) exceeds equivalent capacity ($\hat{C}_{(S)}$) with a probability less than ϵ as follows:

$$Pr(B > \hat{C}_{(S)}) < \epsilon \tag{2.2}$$

Based on the literature review of approximation of the stationary bit rate distribution [41], bit rates can be rather accurately approximated by Gaussian distribution when flows are not bursty for a long period. Hence, the mentioned probability can be calculated by Gaussian tail probability as follows:

$$\hat{C}_{(S)} \approx m + \sigma\sqrt{-2\ln(\epsilon) - \ln(2\pi)}, \tag{2.3}$$

where m is the mean and σ is the standard deviation of the aggregate bit rate given by:

$$m = \sum_{i=1}^{N} m_i \tag{2.4}$$

$$\sigma^2 = \sum_{i=1}^{N} (\sigma_i)^2 \tag{2.5}$$

Consequently, the equivalent capacity is the minimum of $\hat{C}_{(F)}$ and $\hat{C}_{(S)}$, since both approaches overestimate the actual value:

$$\hat{C} = min\{m + \sigma\sqrt{-2\ln(\epsilon) - \ln(2\pi)}, \sum_{i=1}^{N} \hat{c}_i\} \tag{2.6}$$

2.2.1 Exact Approaches

Wang et al. [55] mapped end-to-end connections from different types of classes to available virtual paths with the objective of maximizing revenue. The proposed bandwidth allocation scheme considers traffic fluctuation, and it is based on the minimum required bandwidth, residual capacity, and budget constraint. Let B denotes the total budget, c_j the average cost of one unit bandwidth for class j, b_j^{min} the minimum required bandwidth for class j, and K_j the number of available virtual paths for class j. Then, the available bandwidth range for connection i is defined as follows:

$$b_i^{min} < x_i < \frac{B - \sum_{j \neq i} K_j c_j b_j^{min}}{K_i c_i} \tag{2.7}$$

In this way, there is a guaranteed minimum required bandwidth while being flexible to volatile traffic and keeping costs under budget.

Yu et al. [60] specifically addressed the VM allocation problem in datacenters, with the purpose of guaranteeing network performance where bandwidth demand between each pair of VMs is uncertain. The competition of applications over scarce network resources causes unpredictable performance, i.e., variable data transmission latency and poor datacenter throughput. Related studies providing performance guarantee for datacenters are typically based on the deterministic estimate of bandwidth demands, while measurement studies show that network traffic is highly volatile [13]. To address this issue, Yu et al. [60] proposed a stochastic framework to model uncertain bandwidth demands of cloud tenants and guarantee bandwidth on the shared multi-tenant datacenter. Finally, the dynamic programming approach is employed to implement VM placement while minimizing the maximum bandwidth occupancy ratio in the network.

The work presented in [46] addresses problems of cloud service brokerages regarding how to reserve servers and distribute tenant demands among them, such that the total reservation cost is minimized while satisfying all demands based on tenants' service level agreement

(SLA) (i.e., satisfying all the demands with a given probability). Hence, a Probabilistic Demand Allocation (PDA) system is proposed that predicts demands and prediction errors using a stochastic model.

Tenants' demand is estimated based on historical data and the seasonal autocorrelated moving average (SARMA) model following normal distribution with zero mean. Variance of prediction errors is estimated by the maximum likelihood estimation (MLE). As a result, the estimated demand and prediction error vectors are used to solve the stochastic problem. The model aims to minimize the reservation cost such that each demand is satisfied with the probability no smaller than $1 - \epsilon$. However, the presented model is non-linear because of integer variables and, thus, they relax and decompose the model by the Lagrangian dual decomposition technique to find the final solution.

2.2.2 Heuristic Approaches

Sun et al. [52] proposed an online VNE algorithm that allocates physical nodes and links jointly, with the objective of minimizing mapping cost. Also, it addresses the dynamic nature of network resource demands and explains that traditional works, i.e., deterministic VNE solutions, solved the problem by over-provisioning. However, network resources are becoming scarce and need to be allocated more efficiently. Hence, a stochastic VNE mapping (StoVNM) was presented that considers bandwidth demands following normal distribution. They formulated the virtual link embedding as a stochastic MILP link packing problem.

Similar to the aggregate stationary bit rate approximation in [35], Sun et al. [52] ensured that aggregate bandwidth demand exceeds the physical link capacity with a small probability $Pr[x > b(e_s)] \leq \alpha$, where x represents the aggregate bandwidth demand, α is the capacity violation probability, and $b(e_s)$ is the available bandwidth on substrate link e_s. And this inequality holds if and only if $\mu + \Phi^{-1}(1-\alpha)\sigma \leq b(e_s)$, where bandwidth demand follows normal distribution.

As can be seen, the coefficient of standard deviation in [35]'s inequality ($\sqrt{-2\ln(\epsilon) - \ln(2\pi)}$)

is just replaced by $\Phi^{-1}(1-\alpha)$, which has the same value for a given α $(=\epsilon)$. As a result, they guarantee that capacity violation probability for each physical link is less than α, and it is held until input traffic follows normal distribution. However, the presented model that considers both node and link constraints is NP-hard and, thus, they proposed a heuristic window-based algorithm to solve the StoVNM problem. The window-based algorithm considers VN requests that arrive in a specific period of time and maps them onto the substrate network at once.

2.3 Robust VNE

As mentioned earlier, different levels of uncertainty for bandwidth demand are defined [37]. In addition to stochastic uncertainty, there is a robust strategy to satisfy users with high probability in most scenarios, and this means that the solution is often valid in reality. Moreover, there is total uncertainty where nothing is known, and the provider can allocate a flexible amount based on previous requests. In the following, the focus is on robust VNE solutions where limited demand information is available.

2.3.1 Exact Approaches

Lee et al. [44] formulated the virtual link embedding problem as a robust optimization model, where demand uncertainty is described by an uncertainty set, i.e., the so-called Ellipsoidal uncertainty set [12]. It means that they guarantee the minimum required bandwidth to the user, and more bandwidth up to the upper limit can be allocated with the packet loss probability less than a pre-defined value (ϵ). To do so, they consider the following features in the network architecture:

1. Each user can specify their own VN topology and a bandwidth descriptor, (u_l, g_l), for each virtual link l (VL_l), where u_l and g_l stands for upper limit bandwidth and guaranteed bandwidth requirements,

2. Bandwidth g_l is hard guaranteed no matter what the usage of other users in the network is, and the input traffic in excess of its upper limit bandwidth u_l is dropped,

3. A VN is identified as adhering to its bandwidth agreement on VL_l, if the average data rate of VL_l is no more than $r = (u_l + g_l)/2$ over a window of time T. For a VN user adhering to its bandwidth agreement, the input traffic on VL_l is soft guaranteed. Soft guarantee means that in a short period of time the VN can transmit data on VL_l with a rate up to its upper limit u_l with successful probability $1 - \epsilon$,

4. Also, each user can use OpenFlow commands to control and manage their own VN, and they can use VLAN or MPLS to partition sub-networks inside their VN. The virtual network service should be hitless when a new virtual network user is admitted into the network.

The proposed RO model was then approximated by a SOCP program, which is convex and can be solved efficiently. Exact mathematical formulation of this approach and the approximate second-order cone program are presented in the next chapter.

2.3.2 Heuristic Approaches

Coniglio et al. [27, 28] addressed the offline virtual network embedding problem, where a physical network and a collection of virtual networks are given. First, they modelled VNE as a chance-constrained Mixed-Integer Linear Program where the uncertain demands are assumed to be random variables. Then different from [44], they proposed the Γ-robust optimization approach to approximate the original chance-constrained formulation. This is capable of yielding solutions that are feasible for almost all the possible realizations of the uncertain demands.

The Γ-robustness declares that not all the random variables (VN requests) deviate from their nominal values at the same time, but a subset of cardinality Γ does. This approach not only guarantees a feasible solution when there are, at most, Γ deviations, but also

sustains feasibility for more than Γ deviations with a probability corresponding to a monotone increasing function of Γ, when random variables are independent and their intervals are symmetric [27]. The challenge is to find optimal value for Γ, since the set of feasible solutions gets smaller with increasing Γ value, while more traffic configurations can be covered. Based on conducted simulations, they chose Γ equal to 3, which covers almost all possible traffic configurations with high probability.

Although the proposed Γ-robust formulation is much more tractable than its original chance-constrained counterpart, the Γ-robust version is still very hard to solve for large instances within a reasonable computing time. Hence, they propose two MILP-based heuristics to produce good-quality robust solutions at a smaller computational effort. Both approaches rely on splitting the VNE problem into robust counterparts, including two sub-problems which are then solved sequentially within a given time limit.

The first heuristic solution implements an uncoordinated VNE algorithm, where they conduct admission control and node embedding without considering link constraints in the first phase. A link embedding algorithm that is a counterpart to the multicommodity flow problem with single path routing is performed in the second phase. Based on the conducted experiments, they claim that more than 50% of admitted VNs in the node embedding stage is discarded in the link embedding stage, which is the consequence of not considering routing aspects in the node embedding step [27,28]. To avoid this, they propose the adaptive heuristic algorithm, in which the mapping of a pair of virtual nodes that share traffic demand is restricted to pair of physical nodes that are as close as possible in the substrate network. This decreases the length of physical paths and, accordingly, reduces the consumption of substrate links.

2.4 Summary

In summary, most VNE solutions do not consider uncertainty of demands, which is an undeniable constituent of realistic VNE problems. Also, none of the presented probabilistic VNE approaches, namely stochastic VNE and robust VNE, can guarantee the congestion probability experienced by a virtual link, as they consider congestion on each physical link independent from other links on the end-to-end path that maps to the virtual link.

Moreover, the exact formulation of the VNE problem, considering both node and link constraints, is NP-hard and computationally demanding. But, it provides a reference for heuristic solutions. Heuristic approaches are presented to reduce computation time by dividing the VNE problem into two sub-problems, i.e., virtual node and link embedding. Although the two sub-problems are addressed separately, coordination between node and link mapping phases is necessary in order to use physical resources efficiently and admit as many VNs as possible to increase revenue.

Accordingly, this thesis addresses a virtual link embedding problem with uncertain demands that guarantees end-to-end congestion probability to be less than a pre-defined value and employs one of the existing coordinated node embedding algorithms to complete the solution. The next chapter demonstrates fundamental statistics concepts and theorems regarding optimization problems.

Chapter 3

Mathematics Preliminaries

3.1 Optimization

The word "optimum" is Latin and means "the ultimate ideal" [10]. Therefore, optimization refers to bringing something towards its ultimate possible state. Optimization has a very long history from Greek mathematicians such as Euclid, who explored the minimum distance between a point and a line to the 19th century, when the first optimization algorithms were presented. Generally, optimization problems are formulated in the form of maximizing or minimizing some objective subject to some constraints.

There are three main steps to characterizing an optimization problem. First, one must identify, in the decision problem, activities which can be controlled and influenced. Each activity is associated with a variable whose value is to be decided upon. The remaining quantities are constants in the problem. Second, one must define a real-valued function of the variables to measure how good a vector of variables can be. This quantity is to be given a highest or lowest value, i.e., minimize or maximize the so-called "objective function". Last, restrictions on activities that form constraints on the possible choices of the variable values must be determined, since an activity is often associated with utilization of resources such as time, money, and raw materials which are limited [10].

General form of an optimization problem can be formulated as

$$min_x\ f(x) \tag{3.1}$$

$$\text{subject to } g_i(x) \leq b_i,\ i \in \mathcal{I} \tag{3.2}$$

$$g_i(x) = d_i,\ i \in \mathcal{E} \tag{3.3}$$

$$x \in X \tag{3.4}$$

including inequality and equality constraints. The problem type depends on the nature of functions f and g_i, and the set X, as described in the following [10].

Nonlinear programming. Some functions f and g_i ($i \in \mathcal{I} \cup \mathcal{E}$) are nonlinear.

Linear programming. Objective function is linear and constraint functions are affine (affine function is composed of a linear function and a constant):

$$min_x\ f(x) = c^T x = \sum_{j=1}^{n} c_j x_j \qquad c \in \mathbb{R}^n \tag{3.5}$$

$$\text{subject to } g_i(x) = a_i^T x - b_i \leq 0 \qquad i \in \mathcal{I} \cup \mathcal{E}, a_i \in \mathbb{R}^n, b_i \in \mathbb{R} \tag{3.6}$$

$$x \in X \qquad X = \{x \in \mathbb{R}^n | x_j \geq 0, j = 1, 2, \cdots, n\} \tag{3.7}$$

Continuous Optimization. Functions f and g_i ($i \in \mathcal{I} \cup \mathcal{E}$) are continuous on an open set containing X, where X is closed and convex.

Integer programming. $X \subset \{0,1\}^n$ (binary) or $X \subset \mathbb{Z}^n$ (integer).

Convex programming. f is convex, g_i ($i \in \mathcal{I}$) are concave, g_i ($i \in \mathcal{E}$) are affine; and X is closed and convex.

Quadratic Programming. Objective function f is quadratic in the form of $\frac{1}{2}x^T Q x + c^T x$, and constraints ($g_i$) are all linear functions of the variables.

Second-order Cone Programming. A second-order cone program (SOCP) is a convex problem, where inequality constraints are in the form of $\|C_i x + d_i\|_2 \leq e_i^T x + h_i$.

However, the presented formulation (3.1) does not cover optimization under uncertainty, where some of f and g_i functions are only known probabilistically or with limited information, which is discussed in the following.

3.2 Optimization under Uncertainty

As discussed in the previous chapters, real-world data is often uncertain and cannot be accurately provided, since it is impossible to measure system/environment characteristics with high accuracy or to implement a solution exactly as it is computed [12]. In other words, if nominal data are used to solve the optimization model, optimal solutions can become infeasible when the true data deviate from the nominal. Even small perturbations (e.g., 0.1%) from nominal values can cause violated constraints and infeasible solutions. Therefore, the perturbation analysis is employed as a traditional approach to find the "critical" region, i.e., the set of deviated values for which the model remains feasible and optimal. To put it differently, the goal is to determine allowable changes in the coefficients that maintain the feasibility and optimality of the basic model.

Perturbation analysis considers a large set of any simultaneous changes in the coefficients, which narrows down to more specific analyses as depicted in Fig. 3.1: tolerance, symmetric tolerance, parametric, and sensitivity analysis [36]. The tolerance analysis deals with percentage change in the values, whereas the symmetric tolerance analysis studies equal percentage change over all coefficients. The parametric analysis considers simultaneous increasing/decreasing changes, and the sensitivity analysis specifically examines allowable change in one coefficient value.

More recently, data uncertainty in optimization is addressed by stochastic, chance-constrained, and robust optimization that has emerged as a new paradigm. These approaches tend to answer a slightly different question, which is: "How to find the feasible (optimal) solution for an optimization model when input data are uncertain" [12]. Roots of optimization un-

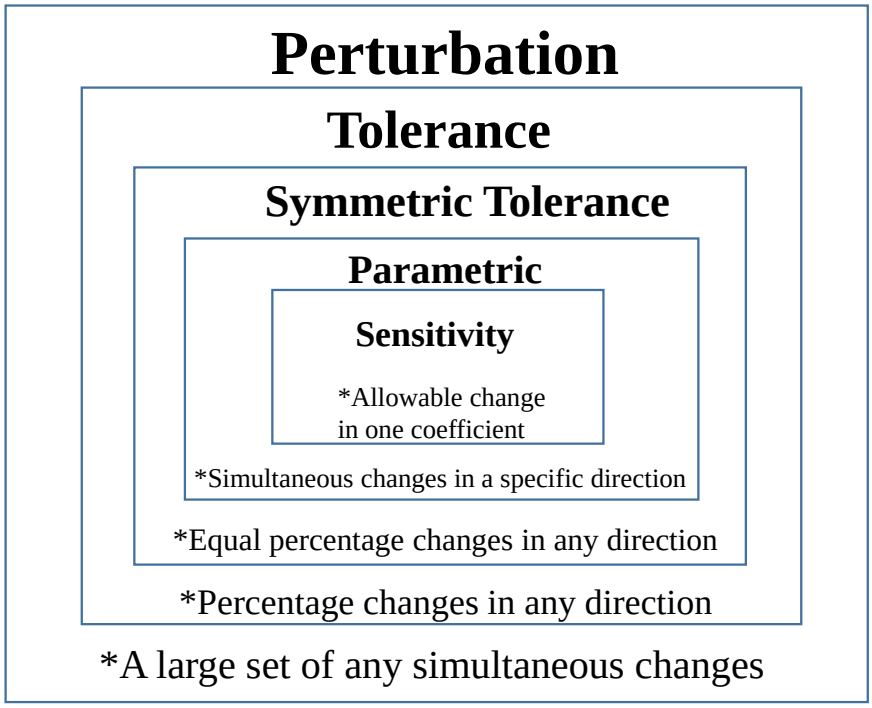


Figure 3.1: Perturbation Analysis.

der uncertainty and stochastic optimization (SO) can be found in the original papers of Dantzig [31] and Soyster [51], where the uncertain data are random and, in the simplest case, it is assumed to follow a priori known probability distribution.

The SO method is comprehensively described in [15,42] textbooks. However, it is difficult to characterize the underlying probability distribution, since proper identification requires a extremely large number of observations and, thus, SO is used often with oversimplified assumptions [12]. Nonetheless, an advanced setting such as the chance-constrained method is also presented where the distribution can be partially known. Finally, a more recent approach to optimization under uncertainty is robust optimization (RO), which finds feasible solutions under any realization of the uncertainty defined as a set, i.e., the so-called "uncertainty set". In the following, a more detailed description and examples of three approaches are provided.

3.3 Stochastic Optimization

Stochastic programs are programs with uncertain data or, more precisely, random variables under the form of probability distributions, densities, or probability measures [15]. The standard form of a deterministic Linear Program (LP) is as follows,

$$\begin{aligned} &\min\ c^T x \\ &\quad Ax \leq b \end{aligned} \tag{3.8}$$

that can be rewritten in another form like,

$$\begin{aligned} &\min\ t \\ &\quad c^T x \leq t \\ &\quad Ax \leq b \end{aligned} \tag{3.9}$$

This deterministic LP can be written as a stochastic problem as follows,

$$\begin{aligned} &\min\ t \\ &\quad \tilde{c}^T x \leq t \\ &\quad \tilde{A}x \leq \tilde{b} \end{aligned} \tag{3.10}$$

where $\tilde{A}$, $\tilde{b}$, and $\tilde{c}$ are random vectors and each component of a vector may follow a different probability distribution such as normal, uniform, and gamma. Moreover, random variables are often restricted to specific confidence intervals like 99%, since they are unbounded. Deterministic form of the above model (3.10) can be achieved by replacing random variables with their expected values such that

$$\min\ t \tag{3.11}$$

$$\sum_{i=1}^{n}(E[c_{1i}]) \cdot x_{i1} \leq t$$
$$\sum_{j=1}^{n}(E[a_{ij}]) \cdot x_{j1} \leq E[b_{i1}], \forall i \in \{1, \cdots, m\}$$

In the case of discrete distribution, expected value can be calculated as

$$\begin{aligned} &\min t \\ &\sum_{i=1}^{n}(\sum_{k} k \cdot P_i^3[c_{1i}]) \cdot x_{i1} \leq t \\ &\sum_{j=1}^{n}(\sum_{k} k \cdot P_i^1[a_{ij}]) \cdot x_{j1} \leq \sum_{k} k \cdot P_i^2[b_{i1}], \forall i \in \{1, \cdots, m\} \end{aligned} \tag{3.12}$$

where $\tilde{A}$, $\tilde{b}$, $\tilde{c}$, and x are $m \times n$, $m \times 1$, $1 \times n$ and $n \times 1$ matrices, respectively. Each row of $\tilde{A}$ matrix follows the corresponding P_i^1 discrete probability, whereas each component of $\tilde{b}$ and $\tilde{c}$ vectors follow P_i^2 and P_i^3 discrete distributions. In the case of continuous random variables, the above constraints will be replaced by integral ones as follows,

$$\begin{aligned} &\min t \\ &\sum_{i=1}^{n}(\int k \cdot f_i^3(c_{1i})d_k) \cdot x_{i1} \leq t \\ &\sum_{j=1}^{n}(\int k \cdot f_i^1(a_{ij})d_k) \cdot x_{j1} \leq \int_{k} k \cdot f_i^2(b_{i1}), \forall i \in \{1, \cdots, m\} \end{aligned} \tag{3.13}$$

where f_i^1, f_i^2, and f_i^3 are probability density functions of $\tilde{A}$, $\tilde{b}$, and $\tilde{c}$, accordingly.

3.4 Chance-Constrained Optimization

Chance-constrained optimization is one of the other approaches to solve optimization problems under uncertainty, acting as a middle phase in the transition of a stochastic model to a robust model. The chance-constrained method is less conservative compared to the robust

optimization (RO) due to the given distribution or other parameters, and it is suitable for systems providing long serve services for many users and required to *guarantee* specific constraints probabilistically [12]. It defines probabilistic constraints and makes sure that the probability of satisfying those constraints is above a certain level. In other words, it reduces the number of feasible solutions so that the confidence level of the solution is high [12]. Again, consider the linear program (3.9) that can be written as a chance-constrained problem as follows [12],

$$min_{x,t} = \{t : Prob_{(c,A,b)\sim P}\{c^T x \leq t,\, Ax \leq b\} \geq 1 - \epsilon\}, \tag{3.14}$$

where $\epsilon \ll 1$ is the given tolerance and P is the distribution of data. In the advanced setting, the distribution can be chosen from a family of probability distributions ($\mathcal{P}$). The more general form of the presented chance-constrained problem is [12],

$$min_{x,t} = \{t : Prob_{(c,A,b)\sim P}\{c^T x \leq t,\, Ax \leq b\} \geq 1 - \epsilon, \forall P \in \mathcal{P}\} \tag{3.15}$$

However, chance-constrained problems are often computationally intractable, and it is difficult to verify whether a given candidate solution is feasible, since these kinds of problems often result in non-convex feasible sets.

Therefore, it is important to find the distributional condition under which the problem is convex. For instance, it is well known that under multivariate normal distribution, a chance-constrained problem becomes a second-order cone optimization model, which is computationally tractable in theory and in practice [19]. More generally, it is shown that the chance constraint can be converted to second-order cone constraints when the random variables are under radial distributions [17]. Radial distributions refer to a general family including normal, truncated normal, as well as uniform distributions on ellipsoidal support.

Consider a single generic chance constraint of the form [17],

$$Prob\{a^T x + b \leq 0\} \geq 1 - \epsilon,\ 0 < \epsilon \ll 1 \tag{3.16}$$

and define the random vector

$$d = [a^T b]^T \in \mathbb{R}^{n+1}, \text{ with } a \in \mathbb{R}^n, b \in \mathbb{R} \tag{3.17}$$

with given mean, $\hat{d}^T$, and covariance, Σ. Then, equivalent second-order cone convex constraint is

$$\{x : k\sqrt{(\tilde{x}^T \Sigma \tilde{x})} + \hat{d}^T \tilde{x} \leq 0\} \tag{3.18}$$

for suitable k value dependent on ϵ and $\tilde{x} = [x^T 1]^T \in \mathbb{R}^{n+1}$. For symmetrical radial distributions around the mean, we can apply tail inequalities or concentration bounds such as Chebychev [54] and its extensions to estimate the chance constraint more precisely.

Concentration Bounds. There are several forms of concentration bounds. They differ from each other in the amount of information required from the distribution of a random variable.

Theorem 1 (Markov's Inequality)**.** The most elementary tail bound is Markov's inequality with a given non-negative random variable X and finite mean [54],

$$\mathbb{P}[X \geq t] \leq \frac{\mathbb{E}[X]}{t},\ \forall t > 0 \tag{3.19}$$

For monotonically increasing function f,

$$\mathbb{P}[f(X) \geq f(t)] \leq \frac{\mathbb{E}[f(X)]}{f(t)} \tag{3.20}$$

Theorem 2 (Chebyshev's Inequality)**.** When random variable X also has a finite variance, Chebyshev's inequality can be computed as follows,

$$\mathbb{P}[|X - \mathbb{E}[X]| \geq t] \leq \frac{Var(X)}{t^2}, \forall t > 0 \tag{3.21}$$

Assuming $X = X_1 + \cdots + X_n$ for independent X_i's and $\mathbb{E}[X_i] = \mu_i$, we have

$$\mathbb{P}[\sum_i |X_i - \mu_i| \geq t] \leq \frac{\sum_i Var(X_i)}{t^2}, \forall t > 0 \tag{3.22}$$

Theorem 3 (Chernoff's Inequality)**.** Chernoff's inequality [22] provides a tighter bound for a given mean and variance. If an exponential moment generating function is applied to independent X_i's variables, then we have

$$\mathbb{P}\{X \geq t\} \leq \inf_{\theta \geq 0} \frac{\mathbb{E}\left[e^{\theta X}\right]}{e^{\theta t}} = \inf_{\theta \geq 0} e^{-t\theta} \prod_i \mathbb{E}\left[e^{\theta X_i}\right] . \tag{3.23}$$

If $\mathrm{Var}[X_i]$ is bounded, i.e., $\mathrm{Var}[X_i] \leq \sigma_i^2$, then we have [54],

$$\mathbb{E}\left[e^{\theta X_i}\right] \leq e^{\mu_i \theta + \frac{1}{2}\sigma_i^2 \theta^2} . \tag{3.24}$$

By taking derivatives of the terms inside the exponent with respect to θ, it is obtained that $\theta = \frac{t-\mu_i}{\sigma_i^2} \geq 0$ minimizes the expression and, hence, we have,

$$\mathbb{P}\left\{X = \sum_i X_i \geq t\right\} \leq \exp\left(-\frac{(t - \sum_i \mu_i)^2}{2\sum_i \sigma_i^2}\right) . \tag{3.25}$$

A variety of other tail bounds can be obtained as particular cases of Chernoff's inequality (3.25), such as **Gaussian** and **sub-Gaussian**. A random variable X with mean $\mu = \mathbb{E}[X]$ is sub-Gaussian [54] if there is a positive number σ such that

$$\mathbb{E}\left[e^{\lambda(X-\mu)}\right] \leq e^{\sigma^2\lambda^2/2}, \forall \lambda \in \mathbb{R} \tag{3.26}$$

The constant σ is referred to as the sub-Gaussian parameter; for instance, X is sub-Gaussian with parameter σ when the condition (3.26) holds. So, in the case of given mean and bounded $Var(X_i)$, i.e., $Var(X_i) < \sigma_i^2$, an inequality similar to Chernoff's inequality (3.25) can be obtained. On the other hand, Gaussian tail, i.e., $X_i = N(\mu_i, \sigma_i^2)$, can be obtained with the same formulation but tighter bound compared to Chernoff's inequality, when we have exact values for variance. Therefore, any Gaussian variable with variance σ^2 is sub-Gaussian with parameter σ. More generally, any bounded random variable ($X_i \in [a_i, b_i]$) with given mean is sub-Gaussian with parameter $\sigma = \frac{b-a}{2}$ which is formulated as Hoeffding's inequality.

Theorem 4 (Hoeffding's inequality)**.** Let $X_1, X_2, \cdots, X_n$ be random variables such that $a_i \leq X_i \leq b_i$. Let $X = X_1 + \cdots + X_n$ and set $\mathbb{E}[X_i] = \mu_i$, then we have,

$$Var(X_i) \leq (b_i - a_i)^2/4\,. \tag{3.27}$$

This happens when the distribution is concentrated at end points a_i and b_i. Applying Chernoff's inequality for sub-Gaussian tails results in the following bound,

$$\mathbb{P}\left\{\sum_i X_i - \mu_i \geq t\right\} \leq e^{-\frac{2t^2}{\sum_i (b_i - a_i)^2}}\,. \tag{3.28}$$

Theorem 5 (Bernstein's inequality)**.** For independent X_i's, if $E[X_i^2] \leq V_i$, $X_i < M$, and $\mathbb{E}[X_i] = \mu_i$, it is obtained that,

$$\mathbb{P}\left\{\sum_i X_i - \mu_i \geq t\right\} \leq e^{-\frac{t^2}{2(W + Mt/3)}},\,. \tag{3.29}$$

where $W = \sum_i V_i$.

Consequently, we can choose the suitable concentration bound based on given information to estimate the chance constraint.

3.5 Robust Optimization

RO methodology is applicable for general optimization classes that are convex, such as LP, SOCP, and Semi-Definite Program (SDP). The robust counterparts of most uncertain problems are provided with the underlying uncertainty sets $\mathcal{U}$, satisfying mild convexity and tractable computability [12] and it can be formulated as,

$$min_{x,t} = \{t : c^T x \leq t,\ Ax \leq b,\ \forall (c, A, b) \in \mathcal{U}\} \tag{3.30}$$

Tractability of the RO models is an important issue in problems with a large set of uncertain coefficients. RO counterparts are not necessarily tractable, since the uncertain set makes the problem semi-infinite and difficult to solve. Therefore, the uncertainty set $\mathcal{U}$ must be structured in specific types, such as ellipsoidal (or quadratic), box, and tail sets so that the size of the set can be controlled and a tractable equivalent reformulation or safe approximation of the problem can be computed, i.e., every feasible solution to the approximation is feasible for the main model [12].

Uncertainty Set. A quadratic or ellipsoidal uncertainty set considers all values within a radius of Ω from the nominal vector r, where the ellipsoid is tilted by the covariance (Σ) [29]:

$$\mathcal{R}^Q(\Omega) = \{\tilde{r} : (\tilde{r} - r)'\Sigma^{-1}(\tilde{r} - r) \leq \Omega^2\} \tag{3.31}$$

The box model considers all values such that each component of the corresponding vector is in the interval $[r_i - \Delta_i, r_i + \Delta_i]$, where Δ_i shows the maximum deviation and the total weight of deviation from the nominal value is restricted to Γ [29]:

$$\mathcal{R}^B(\Gamma) = \{\tilde{r} : \exists u \in \mathbb{R}^n \text{ s.t. } \tilde{r}_i = r_i + \Delta_i \cdot u_i, |u_i| \leq 1, \sum_{i=1}^{n} |u_i| \leq \Gamma\} \tag{3.32}$$

The tail uncertainty set considers the convex hull of all possible $N(1 - \alpha)$ point averages of

the N solutions. When $\alpha = 0$, this set is the singleton set of the nominal vector, and when $\alpha = (N-1)/N$, this set is the convex hull of all N solutions [29]:

$$\mathcal{R}^T(\alpha) = \{\tilde{r} : \exists q \in \mathbb{R}^n_+ \text{ s.t. } \tilde{r} = \sum_{i=1}^{N} q_i \tilde{r}_i, 1'q = 1, q_i \leq \frac{1}{N(1-\alpha)}, i = 1, \cdots, N\} \tag{3.33}$$

Most well-known classes of optimization problems have tractable RO formulations and can be solved efficiently by interior point methods, as shown in [12, 29]. However, this depends on the class of optimization problem, coupled with the type of uncertainty set. For example, the robust counterpart of an LP becomes an SOCP under ellipsoidal uncertainty sets; an SOCP becomes an SDP, while the robust counterpart of an SDP is NP-hard to solve [30]. As a result, the difficulty of the robust problem increases by defining an ellipsoidal uncertainty set, whereas box or tail uncertainty sets may keep the degree of difficulty at the same level.

Here, the RO model with ellipsoidal uncertainty set focuses on the first two moments of decision variables. The model with box uncertainty set focuses on the worst-case scenarios, and the tail approach focuses on averages over the lower tail of the distribution. Based on conducted simulations by [29], the box uncertainty set is more robust and conservative when perturbations occur, compared to the others. As a result, choosing a suitable uncertainty set is a trade-off between complexity and conservatism of the resulting model.

Probability Guarantee. Another important issue in RO is probability guarantee, which is referred to as chance constraint. One of the early attempts was made by Ben-Tal and Nemirovski [12], who propose a robust linear model with an ellipsoidal set of radius Ω and symmetric support, followed by the presented RO model in [44] for the virtual link embedding problem. Specifically, they define a linear constraint like [29],

$$\sum_j \tilde{a}_{ij} x_j \leq b_i, \tag{3.34}$$

where $\tilde{a}_{ij}$ are uncertain coefficients given by $\tilde{a}_{ij} = (1 + \rho\zeta_{ij})a_{ij}$. a_{ij} are nominal values, ζ_{ij}

are random variables with zero mean and range within $[-1, +1]$, and ρ is a constant between 0 and 1. Then the robust counterpart of the form

$$\sum_j a_{ij}x_j + \rho\Omega\sqrt{(\sum_j a_{ij}^2 x_j^2)} \leq b_i^+ \tag{3.35}$$

implies the robust feasible solution satisfies the constraint with high probability of $1 - e^{-\Omega^2/2}$ [29]. Bertsimas and Sim [14] proposed a new RO approach, where a parameter Γ is introduced. This approach has been used by [28] to provide a Γ-Robust framework for virtual network embedding problem. Γ shows the maximum number of coefficients that are allowed to deviate from their nominal values. Also, they assume that simultaneous changes in all coefficients is unlikely. Consider the uncertainty set defined as [29],

$$\mathcal{U}_\Gamma = \{\bar{A} + \sum_{j\in J} z_j\hat{a}_j \big| \, \|z\|_\infty < 1, \sum_{j\in J} 1(z_j) \leq \Gamma\} \tag{3.36}$$

Here, $\bar{A}$ represents a vector of nominal values, $1 : \mathbb{R} \to \mathbb{R}$ denotes the indicator function of y, i.e., $1(y) = 0$ if and only if $y = 0$, and $J \subset \{1, 2, \cdots, n\}$ is an index set of uncertain coefficients, where $\Gamma \leq |J|$. $\|z\|_\infty < 1$ refers to the infinity norm of z forming the surface of a hypercube with edge length 2, i.e., $\|z\|_\infty := maximum(|z_1|, \cdots, |z_n|)$. So, the uncertainty set described by infinity norm can be formulated as an LP robust counterpart.

If we assume that the random vector $\tilde{a}$ has independent components distributed symmetrically on $[\bar{a}_j - \hat{a}_j, \bar{a}_j + \hat{a}_j]$, then the probability of violating constraint (3.34) is proved to be less than $e^{-\frac{\Gamma^2}{|J|}}$. On the other hand, if the uncertainty set is described by Euclidean norm and coefficients have an arbitrary distribution, but with known expected value and covariance matrix, then the violation probability of the constraint can be computed as $\frac{1}{1+\Delta^2}$, where Δ represents the distance from the nominal vector [30].

This method provides better probability guarantee as the number of simultaneous changes increases, which means that Γ increases, whereas it does not heavily penalize the objective function value in order to protect the model against constraint violation [30]. Overall, both

approaches control the degree of conservatism vs performance by adjusting the parameters Ω and Γ [14]. The advantage of the second approach is LP formulation for the uncertainty set.

3.6 Thesis Approach

As discussed in our paper [38], we propose exact formulation of the optimization model under uncertainty, following a chance-constrained framework, since we need to guarantee link congestion probability. Let W_{ik} denote the bandwidth demand of virtual link $\ell_i \in \mathcal{L}$ on physical link $e_k \in \mathcal{E}$, where $\mathbb{E}[W_{ik}] = \mu_{ik}$ and $\mathrm{Var}[W_{ik}] = \sigma_{ik}^2$. Note that W_{ik} must be determined by our model based on bandwidth demands of all virtual links and capacities of all physical links (i.e., C_k's). The congestion probability on physical link e_k is then given by,

$$\mathbb{P}\{\text{congestion on } e_k\} = \mathbb{P}\left\{\sum_{\ell_i \in \mathcal{L}} W_{ik} \geq C_k\right\}, \forall e_k \in \mathcal{E}\,. \tag{3.37}$$

As explained earlier, we have three choices to convert the above chance constraint to a deterministic one:

1. Approximate the chance-constrained model by tail inequalities
2. Apply the RO approach presented by Ben-Tal and Nemirovski [12]
3. Apply the Γ-Robust approach [14]

For simplicity of notation, in the following derivation, the summation index is abbreviated, using i in place of $\ell_i \in \mathcal{L}$. In the first approach, we use a *concentration bound* to estimate (3.37) without any assumptions about the distribution of the uncertain bandwidth demands. As mentioned earlier, there are several forms of concentration bounds that can be used, based on how much information about the distribution of the uncertain demands

is available. In this work, we use the Chernoff bound [22], as follows:

$$\mathbb{P}\left\{\sum_i W_{ik} \geq C_k\right\} \leq \inf_{\theta \geq 0} \frac{\mathbb{E}\left[e^{\theta \sum_i W_{ik}}\right]}{e^{\theta C_k}}\,. \tag{3.38}$$

and $\mathrm{Var}\left[W_{ik}\right]$ is bounded, i.e., $\mathrm{Var}\left[W_{ik}\right] \leq \sigma_{ik}^2$, then we have [54],

$$\mathbb{E}\left[e^{\theta W_{ik}}\right] \leq e^{\mu_{ik}\theta + \frac{1}{2}\sigma_{ik}^2\theta^2}\,. \tag{3.39}$$

Noting that W_{ik}s are independent from each other and taking derivatives of the terms inside the exponent with respect to θ, $\theta = \frac{C_k - \sum_i \mu_{ik}}{\sum_i \sigma_{ik}^2} \geq 0$ minimizes the expression. Hence it is obtained that,

$$\mathbb{P}\left\{\sum_i W_{ik} \geq C_k\right\} \leq \exp\left(-\frac{(\sum_i C_k - \sum_i \mu_{ik})^2}{2\sum_i \sigma_{ik}^2}\right)\,. \tag{3.40}$$

To restrict congestion probability on the physical link e_k by ε_k, the following inequality should be satisfied,

$$\exp\left(-\frac{(\alpha C_k - \sum_{\ell_i} y_{ik}\mu_i)^2}{2\sum_{\ell_i} y_{ik}^2\sigma_i^2}\right) \leq \varepsilon_k, \tag{3.41}$$

that leads to the following inequality,

$$\left(2\ln\frac{1}{\varepsilon_k}\right)\sum_{\ell_i \in \mathcal{L}} \sigma_i^2 y_{ik}^2 \leq \left(\alpha C_k - \sum_{\ell_i \in \mathcal{L}} \mu_i y_{ik}\right)^2. \tag{3.42}$$

This is still a non-linear non-convex constraint, when ε_k is variable. The second approach, i.e., defining an ellipsoidal uncertainty set, results in the same form of constraint. The main problem is posed by the logarithmic term and can be solved only by estimating ε_k values. The last method provides an LP robust formulation, but in order to restrict physical link congestion probability to be less than ε_k, we are required to add exponential constraints in

the form of

$$e^{-\frac{\Gamma_k^2}{|J|}} \leq \varepsilon_k \tag{3.43}$$

Again, this is a non-linear non-convex constraint in the case of unknown ε_k variable and Γ_k parameter. Therefore, the resulting model presented by the Γ-robust approach becomes computationally intractable. We have three options to approximate the (3.43) constraint. First, we can compute the most significant points subject to a given error tolerance and approximate the exponential term with a piecewise linear function [56]. However, this method causes more complications and computations in the model which are not preferable. Second, we can fix Γ_k parameter for all physical links to a specific value. This works only when we can estimate how many flows deviate from their nominal values simultaneously on a specific physical link and at a specific point of time. Last, we can approximate ε_ks by suitable values. Since ε_k values are needed to be approximated in either the first (i.e., Approximate the chance-constrained model by tail inequalities) or third approach (i.e., the Γ-Robust approach), we chose the first as our solution. With this, we do not need to make any assumptions about the number of flow rates that can fluctuate at the same time on a physical link. Approximation of ε_ks values is described in the next chapter.

3.7 Summary

In this chapter, different existing optimization approaches for uncertain data are investigated, such as stochastic, chance-constrained, and robust optimization. Use cases, as well as pros and cons of each optimization framework, are explained. In summary, the stochastic approach requires the exact probability distribution of uncertain data to be known, which is unrealistic in most scenarios. Chance-constrained optimization can provide probability guarantee by limited information, i.e., mean and variance. On the other hand, robust approaches provide optimization under uncertainty by defining the so-called "uncertainty sets"

that can be structured in specific types, such as ellipsoidal and box. By moving from the stochastic approach to robust counterparts, we get more conservative solutions in general, while RO counterparts can be reformulated to convex and tractable models. The next chapter elaborates on the selected optimization approach to model the virtual link embedding problem under uncertainty and the corresponding results.

Chapter 4

Virtual Link Embedding

To cope with the complexity of the problem, most existing works on VNE decompose the problem into two sub-problems, where node and link mappings are performed sequentially [28, 45]. First, the focus is on link mapping in VNE to avoid cluttering the model and algorithms. A complete VNE framework is then proposed, based on an existing node embedding algorithm. Thus, it is assumed that the placement of virtual nodes in the substrate network is already determined and focus on mapping virtual links to physical paths, which we refer to as the *virtual link embedding (VLE)* problem. Table 4.1 lists the principal notation used throughout this chapter.

4.1 System Model and Assumptions

In the following, the primary assumptions used in our VLE models is described.

Physical Network. There is a single physical network specified by an undirected graph $G = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ denotes the set of physical nodes and $\mathcal{E}$ denotes the set of physical links (or edges) between the nodes. Each physical link e_k has a fixed capacity denoted by $C_k > 0$.

Virtual Network Requests. Setting up virtual networks takes time. Therefore, we assume

Table 4.1: Principal Notation Used in the Paper.

Symbol	Definition
Input parameters	
$\mathcal{N}$	Set of physical nodes in the substrate network
$\mathcal{E}$	Set of physical links in the substrate network
$\mathcal{L}$	Set of virtual links from all virtual networks
ℓ_i	Virtual link i ($\ell_i \in \mathcal{L}$)
e_k	Physical link k ($e_k \in \mathcal{E}$)
$O(\ell_i)$	Physical origin node of ℓ_i
$D(\ell_i)$	Physical destination node of ℓ_i
C_k	Capacity of physical link e_k
B_i	Bandwidth demand of virtual link ℓ_i
$\mathcal{P}_i$	Set of candidate physical paths for $\ell_i \in \mathcal{L}$
$\mathcal{P}$	Set of all candidate paths in substrate network
$\lvert P_j \rvert$	Length of path $P_j \in \mathcal{P}$
ϵ	Required congestion probability on virtual links
Decision variables	
$0 \leq x_{ij} \leq 1$	Fraction of demand B_i on path $P_j \in \mathcal{P}_i$
$0 \leq y_{ik} \leq 1$	Total fraction of demand B_i on link $e_k \in \mathcal{E}$
$0 \leq \varepsilon_k \leq 1$	Congestion probability on link $e_k \in \mathcal{E}$
$\alpha \geq 0$	Utilization of the most congested physical link

that virtual network (VN) requests are processed in batches. Each batch corresponds to requests that have arrived during the previous time interval. Our link mapping algorithm is run over each batch in an offline manner. Each virtual network request is represented as a weighted undirected graph with a given set of nodes and links. The weight of each link indicates the (uncertain) bandwidth demand of that link. Each virtual link requires its congestion probability to be bounded by some target ϵ, for $0 \leq \epsilon \leq 1$.

Virtual Link Embedding. Let $\mathcal{L}$ denote the set of the virtual links of all VNs in a batch. Let $O(\ell_i) \in \mathcal{N}$ and $D(\ell_i) \in \mathcal{N}$, for virtual link ℓ_i, denote the physical nodes embedding the origin and destination of virtual link ℓ_i, respectively. The VLE problem is to map every virtual link $\ell_i \in \mathcal{L}$ to a set of physical paths connecting $O(\ell_i)$ to $D(\ell_i)$ in the substrate network (i.e., we allow multi-path routing), so that the virtual link ℓ_i satisfies the target congestion probability ϵ.

Bandwidth Demand Uncertainty Model. Let B_i denote the (uncertain) bandwidth

demand of virtual link $\ell_i \in \mathcal{L}$. We assume that only limited statistical information about B_i is available. Specifically, we assume that the mean and variance of B_i, denoted by $\mathbb{E}[B_i] = \mu_i$ and $\mathrm{Var}[B_i] = \sigma_i^2$, are known. It is relatively straightforward to estimate the mean and variance based on historical traffic data [49]. In fact, as long as σ_i^2 provides an upper bound on the actual variance of B_i, our formulation holds.

We emphasize that the considered uncertainty model is quite general. For example, in the literature on robust optimization (RO), a common uncertainty model is the so-called box uncertainty model [12]. In this model, each uncertain variable B_i is allowed to deviate from its nominal value by a maximum deviation Δ_i. That is, $B_i \in [\mu_i - \Delta_i, \mu_i + \Delta_i]$. Our uncertainty model can easily accommodate the box uncertainty model by computing an upper bound on the variance of an arbitrary random variable that is confined to interval $[\mu_i - \Delta_i, \mu_i + \Delta_i]$. In this case, it can be shown that $\sigma_i^2 = \Delta_i^2$, which is attained when all the probability mass is assigned to the extreme points of the interval. Alternatively, a less conservative approach is to assume B_i is uniformly distributed over the interval, which leads to $\sigma_i^2 = \Delta_i^2/3$.

Congestion Probability. Let W_{ik} denote the bandwidth demand of virtual link $\ell_i \in \mathcal{L}$ on physical link $e_k \in \mathcal{E}$, where $\mathbb{E}[W_{ik}] = \mu_{ik}$ and $\mathrm{Var}[W_{ik}] = \sigma_{ik}^2$. Note that W_{ik} must be determined by our embedding algorithm based on bandwidth demands of all virtual links (i.e., B_i's) and capacities of all physical links (i.e., C_k's). The congestion probability on physical link e_k is then given by,

$$\mathbb{P}\{\text{congestion on } e_k\} = \mathbb{P}\left\{\sum_{\ell_i \in \mathcal{L}} W_{ik} \geq C_k\right\}, \forall e_k \in \mathcal{E}\,. \tag{4.1}$$

For simplicity of notation, in the following derivation, we abbreviate the summation index and use i in place of $\ell_i \in \mathcal{L}$. To avoid making any assumptions about the distribution of the uncertain bandwidth demands, we use a *concentration bound* to estimate (4.1). There are several forms of concentration bounds which can be used, based on how much information

about the distribution of the uncertain demands is available. In this work, we use the Chernoff bound [22], as follows:

$$\mathbb{P}\left\{\sum_i W_{ik} \geq C_k\right\} \leq \inf_{\theta \geq 0} \frac{\mathbb{E}\left[e^{\theta \sum_i W_{ik}}\right]}{e^{\theta C_k}} . \tag{4.2}$$

If $\mathrm{Var}\left[W_{ik}\right]$ is bounded, i.e., $\mathrm{Var}\left[W_{ik}\right] \leq \sigma_{ik}^2$, then we have [54],

$$\mathbb{E}\left[e^{\theta W_{ik}}\right] \leq e^{\mu_{ik}\theta + \frac{1}{2}\sigma_{ik}^2\theta^2} . \tag{4.3}$$

Noting that W_{ik}'s are independent from each other and taking derivatives of the terms inside the exponent with respect to θ, we see that $\theta = \frac{C_k - \sum_i \mu_{ik}}{\sum_i \sigma_{ik}^2} \geq 0$ minimizes the expression and, hence, it is obtained that,

$$\mathbb{P}\left\{\sum_i W_{ik} \geq C_k\right\} \leq \exp\left(-\frac{(\sum_i C_k - \sum_i \mu_{ik})^2}{2\sum_i \sigma_{ik}^2}\right) . \tag{4.4}$$

4.2 Exact and Approximate VLE

We assume that each virtual link $\ell_i \in \mathcal{L}$ can be mapped to multiple paths from the set of candidate physical paths $\mathcal{P}_i$. In other words, $\mathcal{P}_i$ consists of multiple paths, e.g., first K shortest paths, between origin and destination nodes $O(\ell_i)$ and $D(\ell_i)$. Denote the set of all candidate paths in the substrate network by $\mathcal{P}$, that is, $\mathcal{P} = \cup_{\ell_i \in \mathcal{L}} \mathcal{P}_i$. Let x_{ij}, for $0 \leq x_{ij} \leq 1$, denote the fraction of bandwidth demand B_i that is allocated on path $P_j \in \mathcal{P}_i$ for virtual link ℓ_i. Our goal is to find the routing variables x_{ij} that minimize the utilization of the most congested link in the substrate network, subject to a constraint on the congestion probability on ℓ_i. Let α denote the utilization of the most congested link. The virtual link embedding problem is feasible only if $\alpha \leq 1$.

4.2.1 Congestion on Physical Links

Let y_{ik}, for $0 \leq y_{ik} \leq 1$, denote the total fraction of bandwidth demand B_i for virtual link $\ell_i \in \mathcal{L}$ that is allocated on physical link $e_k \in \mathcal{E}$. We have,

$$y_{ik} = \textstyle\sum_{P_j \in \mathcal{P}_i} x_{ij} \cdot \mathbf{I}_{e_k \in P_j}, \tag{4.5}$$

where $\mathbf{I}_{e_k \in P_j}$ denotes the indicator function, which is 1 if $e_k \in P_j$, and 0 otherwise. Notice that $W_{ik} = y_{ik} B_i$, and thus, $\mathbb{E}[W_{ik}] = y_{ik}\mu_i$ and $\mathrm{Var}[W_{ik}] = y_{ik}^2 \sigma_i^2$. Next, applying (4.4), for a desired link utilization α, it is obtained that,

$$\mathbb{P}\left\{\sum_{\ell_i \in \mathcal{L}} y_{ik} B_i \geq \alpha C_k\right\} \leq \exp\left(-\frac{(\alpha C_k - \sum_{\ell_i \in \mathcal{L}} y_{ik}\mu_i)^2}{2\sum_{\ell_i \in \mathcal{L}} y_{ik}^2 \sigma_i^2}\right). \tag{4.6}$$

Therefore, to restrict the congestion probability on physical link e_k by ε_k, the following inequality should be satisfied,

$$\exp\left(-\frac{(\alpha C_k - \sum_{\ell_i} y_{ik}\mu_i)^2}{2\sum_{\ell_i} y_{ik}^2 \sigma_i^2}\right) \leq \varepsilon_k, \tag{4.7}$$

which leads to the following inequality,

$$\left(2\ln\frac{1}{\varepsilon_k}\right)\sum_{\ell_i \in \mathcal{L}} \sigma_i^2 y_{ik}^2 \leq \left(\alpha C_k - \sum_{\ell_i \in \mathcal{L}} \mu_i y_{ik}\right)^2. \tag{4.8}$$

4.2.2 Congestion on Physical Paths

A path is a sequence of links; thus, we have,

$$\mathbb{P}\{\text{congestion on path } P_j\} = 1 - \textstyle\prod_{e_k \in P_j}(1 - \varepsilon_k). \tag{4.9}$$

In practice, we have $\varepsilon_k \ll 1$ and, thus, using the union bound, we obtain the following approximation for the path congestion probability,

$$\mathbb{P}\{\text{congestion on path } P_j\} \approx \textstyle\sum_{e_k \in P_j} \varepsilon_k \,. \tag{4.10}$$

Therefore, to restrict congestion probability on path P_j by ϵ, the following inequality should be satisfied,

$$\textstyle\sum_{e_k \in P_j} \varepsilon_k \leq \epsilon \,. \tag{4.11}$$

4.3 Exact VLE Formulation

The VLE problem can be formulated as a non-linear optimization problem, as presented in Problem 1, where,

- Constraint (4.12a) enforces bandwidth embedding over all possible candidate paths.
- Constraint (4.12c) enforces end-to-end congestion probability ϵ on every path.
- Constraint (4.12d) enforces link congestion probability ε_k on link e_k, so as to satisfy Constraint (4.12c).

All the constraints in Problem 1 are linear except (4.12d), which is non-linear and non-convex. To show the non-convexity of this constraint, we transform it to an equivalent system of inequalities, as follows. Let $\theta_k = 2 \ln \frac{1}{\varepsilon_k}$. Define the following auxiliary constraint,

$$\theta_k y_{ik}^2 \leq z_{ik}^2 \,. \tag{4.13}$$

The Constraint (4.12d) is then equivalent to the following constraints,

$$\textstyle\sum_{\ell_i \in \mathcal{L}} \sigma_i^2 z_{ik}^2 \leq u_{ik}^2, \tag{4.14a}$$

$$u_{ik} = \alpha C_k - \textstyle\sum_{\ell_i \in \mathcal{L}} \mu_i y_{ik}, \tag{4.14b}$$

Problem 1 Exact VLE.

$$\text{minimize} \quad \alpha$$

subject to:

$$\sum_{P_j \in \mathcal{P}_i} x_{ij} = 1, \qquad \forall \ell_i \in \mathcal{L} \tag{4.12a}$$

$$y_{ik} = \sum_{P_j \in \mathcal{P}_i} x_{ij} \mathbf{I}_{e_k \in P_j}, \qquad \forall \ell_i \in \mathcal{L}, \forall e_k \in \mathcal{E} \tag{4.12b}$$

$$\sum_{e_k \in P_j} \varepsilon_k \leq \epsilon, \qquad \forall P_j \in \mathcal{P} \tag{4.12c}$$

$$\big(2 \ln \frac{1}{\varepsilon_k}\big) \sum_{\ell_i \in \mathcal{L}} \sigma_i^2 y_{ik}^2 \leq \big(\alpha C_k - \sum_{\ell_i \in \mathcal{L}} \mu_i y_{ik}\big)^2, \forall e_k \in \mathcal{E} \tag{4.12d}$$

$$x_{ij}, y_{ij} \in [0, 1], \tag{4.12e}$$

$$\alpha \geq 0\,. \tag{4.12f}$$

$$\theta_k y_{ik}^2 \leq z_{ik}^2, \tag{4.14c}$$

$$\theta_k \geq 0\,. \tag{4.14d}$$

It is clear that (4.14a) is a second-order cone (SOC) constraint, (4.14b) and (4.14d) are linear, but (4.14c) is non-convex.

To solve Problem 1, one approach is to use a global nonlinear solver such as Knitro [2]. We have implemented this approach and experimented with various network configurations. While it is possible to solve the problem for small network instances, it takes a prohibitively long time to solve the problem for any realistic network size. Moreover, since the problem is not convex, the computed solutions may not even be globally optimal. Therefore, to solve the problem for large network instances, we design an approximate solution, as presented in the next sub-section. We show that the approximation results in a second-order cone program (SOCP) [7] that can be solved efficiently (in polynomial time) using conventional solvers such as Gurobi [3].

4.4 Approximate VLE Formulation

The complication in Constraint (4.12d) is due to the fact that the program tries to *optimally* assign congestion probabilities to each link on a given path. If we could pre-compute ε_k for each link e_k, then Problem 1 could be converted to a SOCP, as demonstrated by (4.14a)-(4.14d). Specifically, Constraint (4.12d) is equivalent to the following constraints,

$$\sum_{\ell_i} \sigma_i^2 z_{ik}^2 \leq u_{ik}^2 \tag{4.15a}$$

$$u_{ik} = \frac{1}{\sqrt{-2\ln \varepsilon_k}} \Big(\alpha C_k - \sum_{\ell_i} \mu_i y_{ik} \Big) \tag{4.15b}$$

which are SOC for a given ε_k.

Our approximate formulation is based on the simplification that all links in a path achieve the same congestion probability. Clearly, this results in a sub-optimal solution because the optimal solution may assign different congestion probabilities to different links on the same path. Let ε_j denote the link congestion probability for each link in path P_j, that is, $\varepsilon_k = \varepsilon_j$, for all $e_k \in P_j$. Let $|P_j|$ denote the length of path P_j, we have,

$$\mathbb{P}\left\{\text{congestion on path } P_j\right\} = 1 - (1-\varepsilon_j)^{|P_j|}\,. \tag{4.16}$$

Therefore, to satisfy the end-to-end path congestion probability, we obtain that,

$$1 - (1-\varepsilon_j)^{|P_j|} \leq \epsilon \Leftrightarrow \varepsilon_j \leq 1 - \sqrt[|P_j|]{1-\epsilon}\,. \tag{4.17}$$

One problem arising from the above congestion probability allocation policy is that a link may be common among multiple paths of different lengths. In such cases, the longest path determines the required congestion probability on the common links, as it requires lower congestion on each link. As a result, the congestion probability assignment for the uncommon links must be updated (i.e., increased) to satisfy the end-to-end path congestion constraint, as described next. Consider some path P_j. Let $\hat{P}_j$ denote the set of links in this

path whose congestion probabilities are not assigned yet. For the remaining links in the path, i.e., $e_k \in P_j \setminus \hat{P}_j$, their congestion probabilities have already been assigned, as they are common with some longer paths. Let $\hat{\varepsilon}_j$ denote the congestion probability that should be assigned to every link in $\hat{P}_j$. The following relation should be satisfied:

$$1 - (1 - \hat{\varepsilon}_j)^{|\hat{P}_j|} \prod_{e_k \in P_j \setminus \hat{P}_j} (1 - \varepsilon_k) \leq \epsilon, \tag{4.18}$$

which yields the following relation,

$$\hat{\varepsilon}_j \leq 1 - \sqrt[|\hat{P}_j|]{\frac{1-\epsilon}{\prod_{e_k \in P_j \setminus \hat{P}_j} (1 - \varepsilon_k)}}. \tag{4.19}$$

The congestion assignment process is described in Algorithm 1.

Algorithm 1 Link Congestion Assignment.

$\vec{\mathcal{P}} \leftarrow$ sort $\mathcal{P}$ from longest to shortest path
$\varepsilon_k \leftarrow 0, \quad \forall e_k \in \mathcal{E}$
for $j = 1$ **to** $|\vec{\mathcal{P}}|$ **do**
  $\pi \leftarrow 1$
  $P_j \leftarrow \vec{\mathcal{P}}[j]$
  $\hat{P}_j \leftarrow \{e_k \in P_j | \varepsilon_k = 0\}$
  for all $e_k \in P_j \setminus \hat{P}_j$ **do**
    $\pi \leftarrow \pi \times (1 - \varepsilon_k)$
  end for
  for all $e_k \in \hat{P}_j$ **do**
    $\varepsilon_k = 1 - \sqrt[|\hat{P}_j|]{\frac{1-\epsilon}{\pi}}$
  end for
end for

Figure 4.1 is an illustrative example of the above process. We assume that virtual nodes v_1 and v_2 are placed on physical nodes n_1 and n_2. Accordingly, the virtual link that connects the virtual nodes is mapped onto multiple paths, P_1 and P_2, that connect these two physical nodes. As it is shown in Figure 4.1, the required end-to-end congestion probability is ϵ.

First, physical paths are sorted from longest to shortest. Here, we only have two paths: P_1 and P_2 containing three and four links, respectively. Therefore, path P_2 is selected as

the longest path. At first, none of the links have assigned congestion probability, and thus $P_j \setminus \hat{P}_j$ set is empty. This means that ϵ is divided between four links and corresponding congestion probabilities are computed as,

$$\varepsilon_{3,4,5,6} = 1 - \sqrt[4]{1-\epsilon}\,. \tag{4.20}$$

Now, path P_1 is selected and ε_6 as congestion probability of the common link between P_1 and P_2 is known. Hence, $\hat{P}_j$ includes only physical links e_1 and e_2 with undetermined congestion probabilities of ε_1 and ε_2. Consequently, the remaining congestion probabilities are computed as,

$$\varepsilon_{1,2} = 1 - \sqrt[2]{\frac{1-\epsilon}{(1-\varepsilon_6)}}\,. \tag{4.21}$$

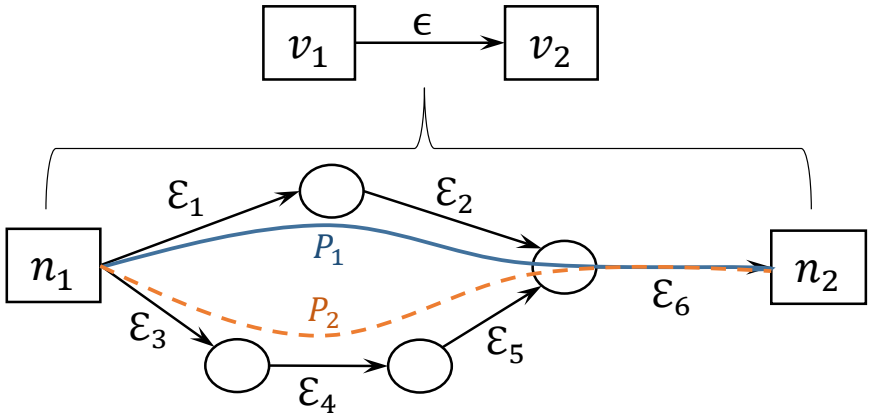


Figure 4.1: The approximate physical link congestion probability.

The optimization problem is reduced to the SOCP problem presented in Problem 2. Once the routing variables x_{ij} are computed, we may find that some paths are not used by any virtual network. Thus, we can adjust link congestion probabilities accordingly. To adjust link congestion probabilities, we simply remove the unused paths, re-assign link congestion probabilities and solve the optimization problem again.

Problem 2 Approximate VLE.

$$\text{minimize} \quad \alpha$$

subject to:

$$\sum_{P_j \in \mathcal{P}_i} x_{ij} = 1, \qquad \forall \ell_i \in \mathcal{L} \tag{4.22a}$$

$$y_{ik} = \sum_{P_j \in \mathcal{P}_i} x_{ij} \mathbf{I}_{e_k \in P_j}, \qquad \forall \ell_i \in \mathcal{L}, \forall e_k \in \mathcal{E} \tag{4.22b}$$

$$\big(2 \ln \frac{1}{\varepsilon_k}\big) \sum_{\ell_i \in \mathcal{L}} \sigma_i^2 y_{ik}^2 \leq \big(\alpha C_k - \sum_{\ell_i \in \mathcal{L}} \mu_i y_{ik}\big)^2, \forall e_k \in \mathcal{E} \tag{4.22c}$$

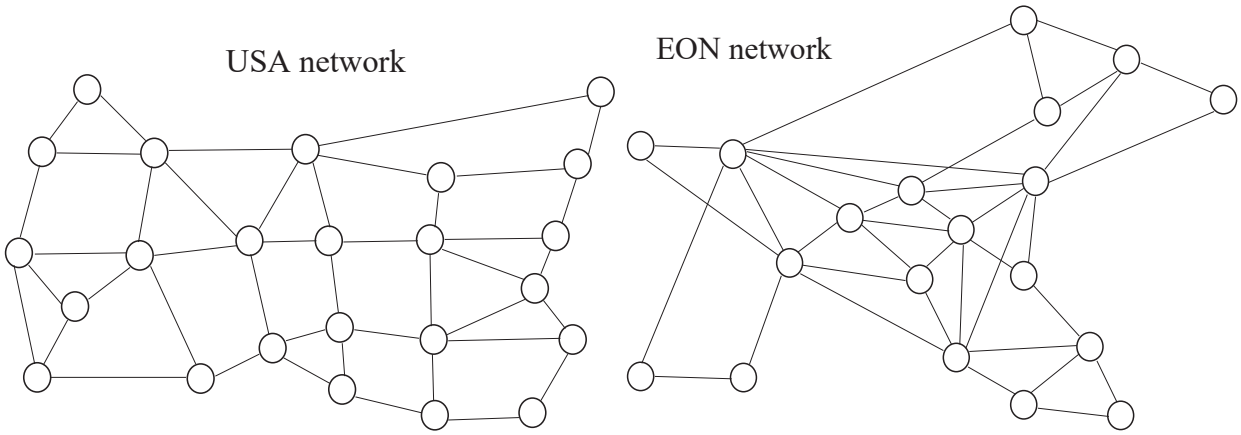
$$x_{ij}, y_{ik} \in [0, 1], \tag{4.22d}$$

$$\alpha \geq 0\,. \tag{4.22e}$$

4.5 Performance Analysis of Exact Model vs. Approximate Model

We evaluate exact and approximate models through simulations, Mininet and real OpenFlow testbed experiments. Knitro [2], an advanced solver for nonlinear optimization, is used to solve the exact VLE problem. Moreover, AMPL [1] modelling language is employed to enhance the accuracy of the local optimal solution. The approximate VLE model is a SOCP, which is solved using Gurobi [3] version 7.5.2. Simulations are carried out on a single machine with an Intel(R) Core(TM) i7-4770 CPU@3.40 GHz, 4 Cores and 8 Logical Processors, with 16 GB RAM.

First, exact and approximate solutions are compared on small-scale network topologies in Section 4.5.3 since solving the exact problem for large networks is computationally intractable. Second, a comparison is conducted between the approximate model and an existing virtual link embedding algorithm with physical link congestion constraint [44] in Section 4.5.4. Finally, large-scale evaluations considering randomly generated large network topologies are used to study the effect of various system parameters on the utility and scalability of the approximate VLE solution in Section 4.5.5. Moreover, more realistic results are



(a) USA network topology. (b) EON network topology.

Figure 4.2: Small-scale network topologies.

illustrated through Mininet and real testbed experiments. Real testbed experimental results for the approximate VLE are illustrated in Chapter 5, compared with a complete framework which includes a virtual node embedding algorithm.

4.5.1 Simulation Parameters

For small-scale simulations, the USA (24 nodes and 43 links) and European Optical Network (EON) (19 nodes and 37 links) topologies are used (see Figure 4.2). These topologies are widely used in the literature for similar evaluation purposes (e.g., see [8,44]). For large-scale simulations, random topologies are generated, as discussed in Section 4.5.5.

For the sake of simplicity, we assume uniform bandwidth demand of μ for all virtual links. Physical link capacities are also assumed to be equal and given by C, where C is scaled with respect to the mean bandwidth demand μ (i.e., $C = \rho$ means that the link capacity is $\rho\mu$ in bps). In our simulations, C varies between 10 and 100, but is set to 20 by default. We note that using unscaled values for link capacities (e.g., 10 Gbps) actually slows down solving a quadratic model considerably, since it leads to large coefficient range and numerical issues. We also set the variance of bandwidth demand to σ^2 for all virtual links. In this section, we use the Coefficient of Variation, denoted by CoV and defined as $\text{CoV} = \sigma/\mu$, to describe the variability of bandwidth demands. A high CoV indicates high uncertainty in bandwidth

demands, and vice versa.

Table 4.2: Default simulation parameters.

Parameter	Value
C	20
μ	1
K	3
CoV	1.0
ϵ	0.1
Network Topology	USA

Generally, optimization programs take as input the set of virtual network requests that need to be embedded in the network. Moreover, virtual nodes can be embedded by a desired algorithm before the link mapping stage. In simulations, we assume that a virtual network simply consists of an origin-destination pair connected with a virtual link to simplify the node embedding phase and saturate the substrate network as much as possible. Hence, each virtual link can be easily assigned to a randomly chosen origin-destination pair in the physical network. However, more general and realistic VN requests embedded with an existing virtual node embedding algorithm are considered in real testbed experiments.

Later, a K-shortest-path algorithm [59] is run to compute K candidate paths between the chosen origin-destination nodes for each virtual link. Next, the optimization models are solved based on the computed candidate paths and network parameters. The default values for the parameters are presented in Table 4.2. The value of a parameter changes only when its impact is investigated. Each point in the plots (for simulations) is the average of four simulation runs. The error bars (showing the min and max values) are not presented in cases where the deviation from the average was very small.

4.5.2 Performance Measures

We use the following measures to compare the performance of the models: 1) the number of admitted virtual links, 2) the achieved congestion probability, and 3) the utilization of the most congested link (denoted by α). To compute the number of admitted virtual links, we

solve the problems starting with a small set of virtual link requests. Then, we iteratively increase the number of virtual link requests until the problems become infeasible. While this linear search can be improved, e.g., by a binary search, it takes only seconds to find the maximum number of admitted links for the approximate model. For the exact model, however, it is a time-consuming process and the starting point matters significantly due to local optimality of solutions returned by the nonlinear solver. Therefore, we have used starting points based on the approximate model solution to speedup the search process.

4.5.3 Exact and Approximate Models

In this subsection, performance of the exact and approximate models is compared in terms of the average number of admitted virtual links and achieved congestion probability. Results show that the approximate model produces results that are very close to those of the exact model.

Number of Admitted Links. Figure 4.3 shows the number of admitted virtual links by each model for different coefficients of variation (CoV $= 0, 0.5, 1, 1.5$). By increasing CoV, demand uncertainty increases and more bandwidth is reserved per virtual link, which causes a sharp decline in the number of admitted virtual link requests. We observe that the results achieved by the approximate and exact model are very close to each other. The reason for the slightly higher number of admitted links under the approximate model can be explained by looking at Fig 4.4. We can see that the approximate model generally achieves higher congestion probabilities, which translates to more admitted virtual link requests.

Congestion Probability. Figure 4.4 depicts the achieved end-to-end congestion probabilities for exact and approximate algorithms. For both models, the objective is to satisfy a maximum congestion probability of $\epsilon = 0.1$. We observe that: 1) both models generally satisfy the target congestion probability, and 2) the exact model generally achieves lower congestion probabilities compared to the approximate model. This means that, in some cases, the approximate model may admit more virtual links. The reason is that the approximate

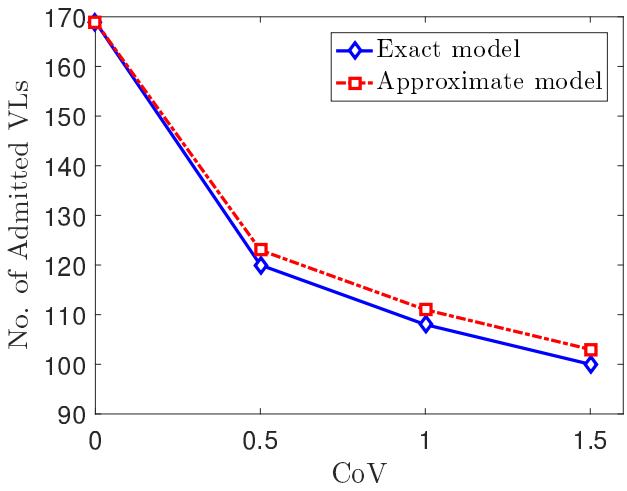
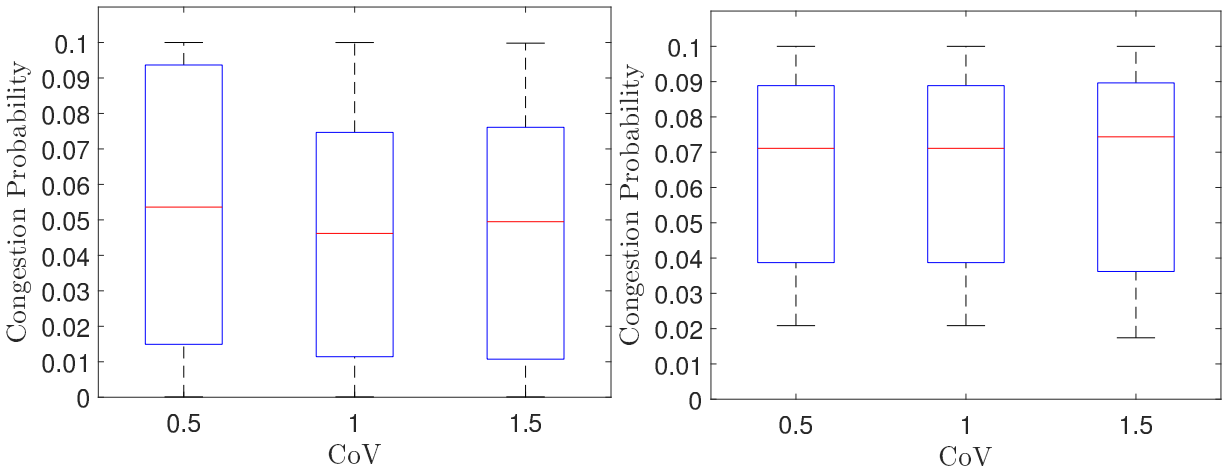


Figure 4.3: Average number of admitted virtual links with exact and approximate models for different uncertainty levels.

model is forced to achieve a specific congestion level at each physical link (as discussed in subsection 4.4), which may not be optimal, but the exact model is free to decide about the optimal level of congestion for each link.



(a) Exact model.

(b) Approximate model.

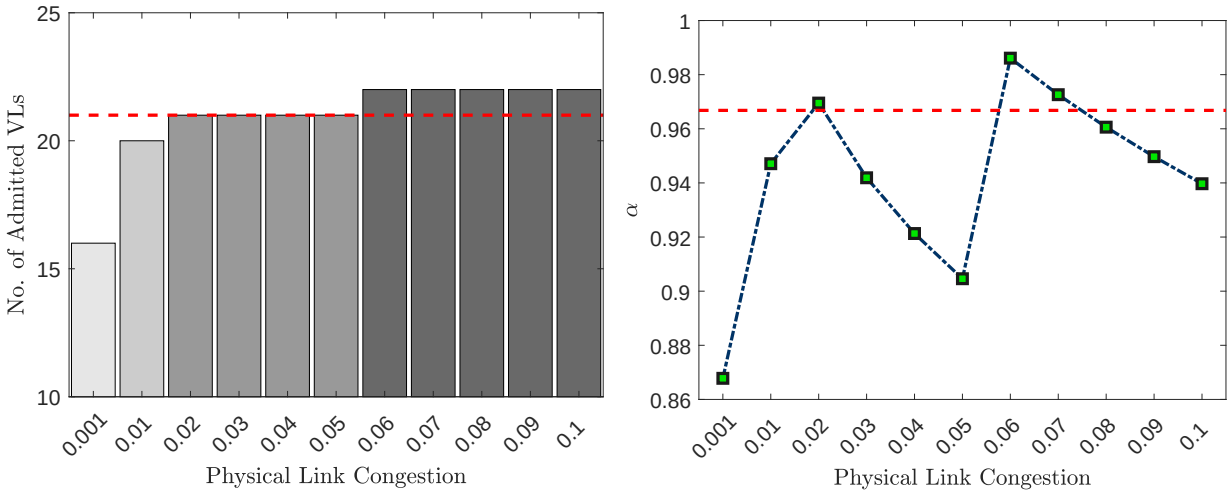
Figure 4.4: Congestion probability of exact and approximate models.

4.5.4 Comparison of End-to-End and Link-by-Link Congestion-Constrained Models

In this section, the approximate model (Approx. VLE) is compared with an existing VLE approach, which defines congestion constraints link-by-link [44] instead of end-to-end paths, i.e., congestion constraint is applied on every physical link without considering end-to-end path. Physical link congestion probability varies between 0.001 to 0.1 in this simulation. 0.001 or 0.1% congestion is selected as a close value to zero, since congestion cannot be set to 0% as such we have $ln(\frac{1}{\epsilon_k})$ term in the optimization model. 0.1 or 10% is the maximum acceptable end-to-end congestion. In the case of the link-by-link model, there is no guarantee that end-to-end congestion probability remains less than a specified value ($\epsilon = 10\%$) for any of these numbers, since it depends entirely on the number of physical links on substrate paths. For instance, when link congestion is set to 10%, and path lengths are more than one, end-to-end congestion is approximately equal to the summation of physical link congestions along the path which exceeds 10%.

Figure 4.5 shows utilization of the most congested link (α) and the number of admitted virtual links (origin-destinations) for the link-by-link approach in comparison with a reference indicated by a red line. The red line reference shows the result for our end-to-end approach. Generally, utilization of the most congested link must increase monotonically by assigning a higher congestion probability to physical links, since more bandwidth can be allocated. However, as depicted in Figure 4.5(b), two drops occurred at link congestions of 2% and 6%. This is due to the fact that the number of admitted virtual links stands still when physical link congestion is scaled up from 2 to 5% or from 6 to 10% which is shown in Figure 4.5(a). Recall that the objective of the bandwidth allocation model is to minimize α by balancing load among paths. Therefore, α decreases when we have the same number of admitted virtual links and the model has more freedom for load balancing because of higher congestion probability.

The average physical link congestion probability assigned by our approximate model is



(a) Number of admitted virtual links.

(b) Utilization of the most congested link.

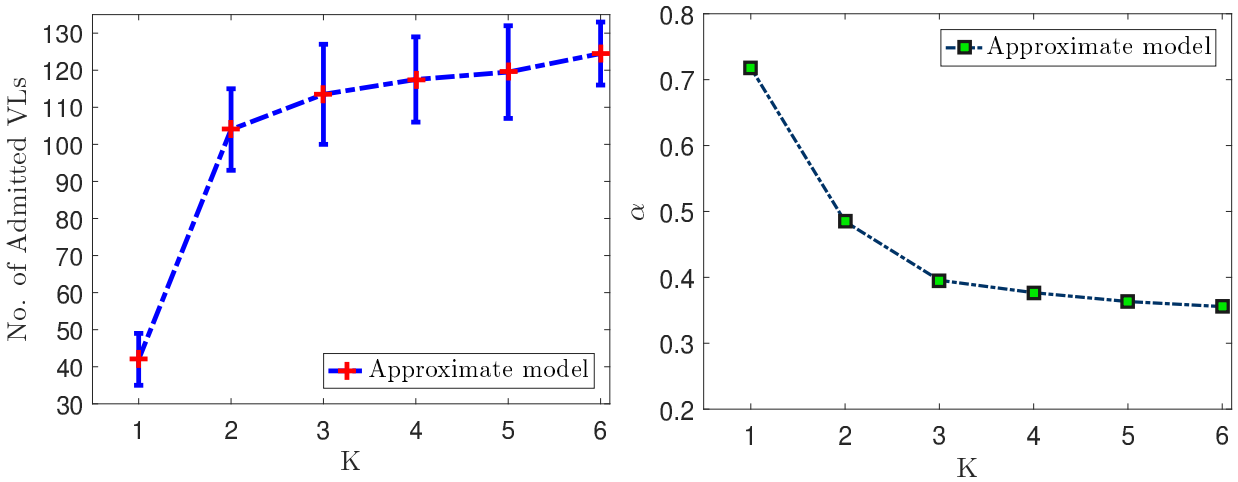
Figure 4.5: Comparison of link-by-link and end-to-end congestion-constrained models.

around 2%, based on the average path length of 5 between origin-destination. So that link utilization of the end-to-end model is close to the link-by-link model where link congestion probability is set to 2%. Likewise, the number of admitted virtual links in the end-to-end model indicated by the red line is 21, which is equal to link-by-link results with congestion probability within $2-5\%$. The main issue in the proposed link-by-link model [44] is that finding a suitable value for the physical link congestion probability is not straightforward, as it depends on the node embedding scheme and the order of VN requests arriving to the system. Furthermore, setting all probabilities to a specific value cannot guarantee end-to-end congestion.

4.5.5 Effect of Network Parameters

In this section, we focus on the approximate model, as it is computationally fast and is reasonably accurate compared to the exact model.

Number of Candidate Paths. Figure 4.6 shows the effect of increasing the number of candidate paths K on performance of the approximate model. As can be seen in Figure 4.6(a), the maximum number of admitted links increases drastically by increasing the number of candidate paths from 1 to 3. However, the gain diminishes as the number of paths increases



(a) Number of admitted virtual links.

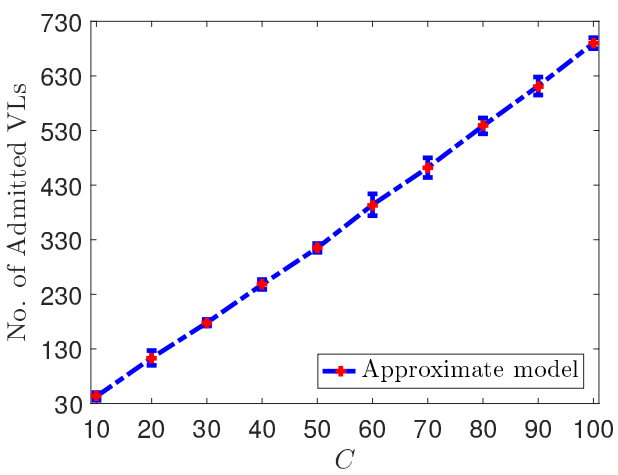
(b) Utilization of the most congested link.

Figure 4.6: Effect of the number of paths (K).

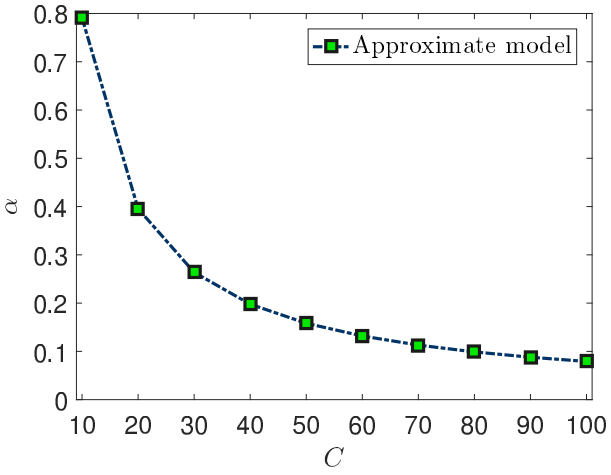
beyond 3. Therefore, we set the default number of candidate paths to 3 in the rest of the simulations in this section. Figure 4.6(b) illustrates the impact of K on the most congested link, i.e., the link with the highest utilization α. In this figure, the number of virtual links is fixed at $|\mathcal{L}| = 30$. As expected, the highest link utilization drops as we increase the number of candidate paths. Again, there is no considerable reduction after $K = 3$.

Physical Link Capacity. Figs. 4.7(a) and 4.7(b) show the effect of the physical link capacity on the number of admitted virtual links and most congested link utilization. In this scenario, CoV and μ are set to their default values, and the scaled link capacity C varies between $C = 10$ to $C = 100$. As expected, the maximum number of accepted links increases linearly with respect to link capacity. In Figure 4.7(b), the number of virtual links is fixed at $|\mathcal{L}| = 30$. We observe that the utilization α drops by almost 87% when C increases from 10 to 100. Also, the reduction in link utilization diminishes as the link capacity increases beyond $C = 30$.

Number of Virtual Link Requests. Figure 4.8 shows the CDF of the maximum link utilization when the number of virtual link requests is set to 50 and 100. All other parameters are set to their default values. For 50 virtual link requests, the network achieves, at most, 50% utilization, which increases to 90% by doubling the number of virtual link requests to



(a) Number of admitted virtual links.



(b) Utilization of the most congested link.

Figure 4.7: Effect of the physical link capacity (C).

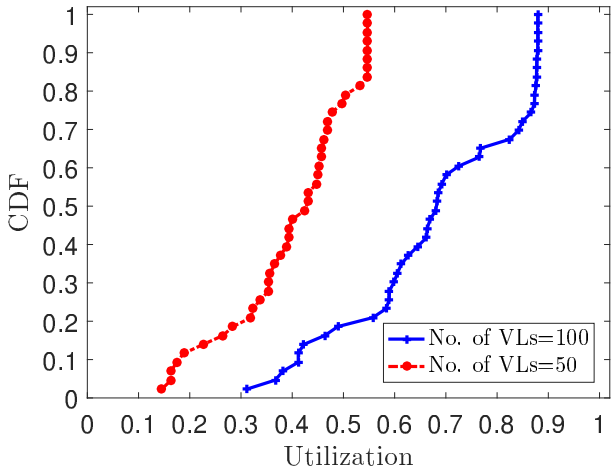


Figure 4.8: Effect of the number of virtual links on utilization.

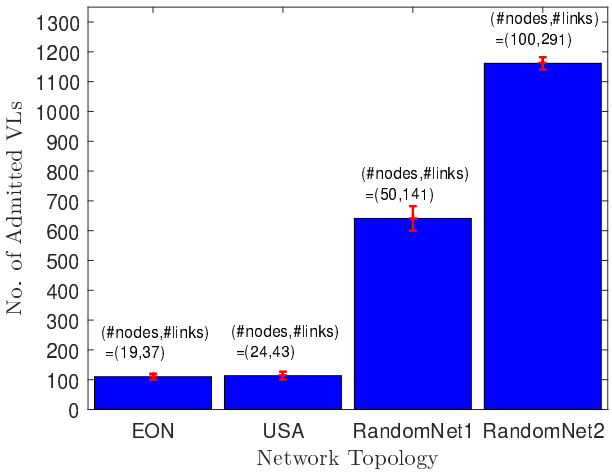


Figure 4.9: Effect of the network topology on admitted links.

100.

Network Topology. Figure 4.9 shows the number of admitted links for different network topologies. Besides the USA and EON topologies, we have generated two large random networks with 50 (141 links) and 100 (291 links) nodes. The Barabasi-Albert model [11] is used to generate random scale-free networks. The objective of this experiment is to show that the approximate model can be used to handle large networks with ease. Four different random sets of origin-destination pairs are generated for each topology. The average number of admitted virtual links, along with bars showing the minimum and maximum values are

plotted in the figure. As expected, the number of admitted links increases by increasing the size of the network.

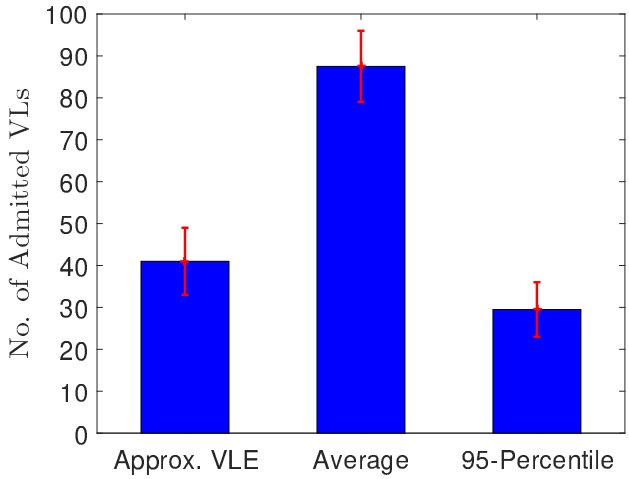


Figure 4.10: Mininet experiments - Average number of admitted virtual links.

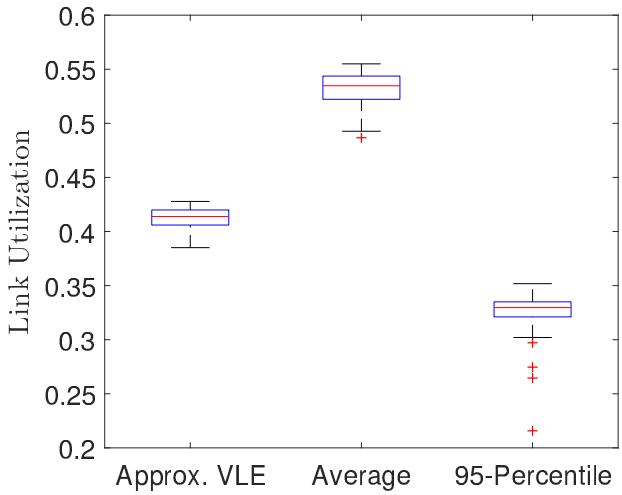


Figure 4.11: Mininet experiments - Average link utilization of physical links.

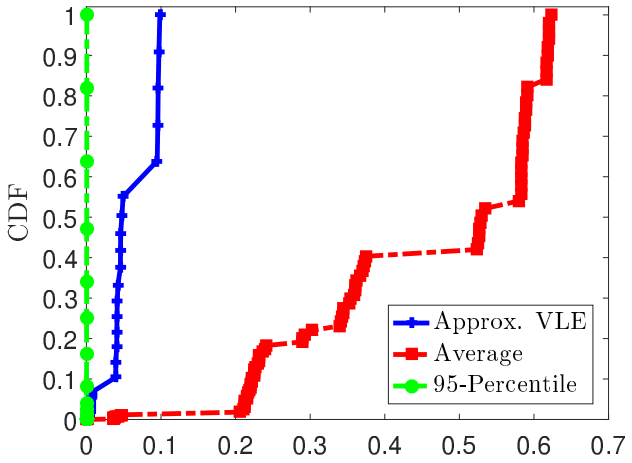


Figure 4.12: Mininet experiments - CDF of packet drop probability.

4.5.6 Mininet Experiments

In this subsection, our goal is to determine whether the results obtained via simulations can qualitatively match the results measured on Mininet. In these experiments, we use the USA topology; origin-destinations corresponding to virtual links are randomly picked from physical nodes, and candidate paths between origin-destination pairs are based on K-shortest paths, as in the simulations. To implement multi-path routing in Mininet, we use the routing variables x_{ij} computed in the optimization problem to split bandwidth demands statically among the candidate paths.

Implemented Algorithms. We implemented three VLE algorithms, as described below:

- **Approx VLE:** This is the approximate VLE model developed in this paper (see Problem 2).
- **Average:** This algorithm ignores demand variability and bases its bandwidth allocation decisions on the mean bandwidth demand only. Specifically, it allocates μ bps bandwidth to each virtual link.
- 95**-Percentile:** This algorithm actually considers demand variability when mapping virtual links. Specifically, it assumes that the bandwidth demands follow a normal

distribution with mean μ and variance σ^2 (which are assumed to be known). It then computes an *effective demand* for each virtual link, which is equal to the 95-percentile of bandwidth demand given by $\mu + 1.65\sigma$. We chose the 95-percentile so that if a virtual link is mapped to a path of length 2, and no other link uses that path, then the end-to-end congestion probability of the link remains below $\epsilon = 0.1$.

Notice that **Average** and 95-**Percentile** are deterministic mapping algorithms, which were solved exactly by modifying our exact VLE formulation in Problem 1.

Experiment Setup. In Mininet, we generate traffic for each virtual link at a rate that is distributed normally with parameters μ and σ. We set μ to 1 Mbps, CoV to 1, and C to 10 (i.e., link capacity is 10 Mbps). We note that CoV = 1 represents a scenario with a high level of uncertainty. However, it is chosen here to show that even in such an extreme scenario, the proposed model achieves reasonable performance. Even though 1 Gbps network links are very common, these small values are used to decrease the Mininet experimentation time, as every operation (e.g., switching with Open vSwitch) is performed in software. To emulate demand variability, the traffic rates are changed over time. Specifically, for each virtual link, its rate of traffic is sampled from the normal distribution every second during the course of the experiment. Each experiment is run for 15 minutes to achieve stable results.

Measurements. In each experiment, we measure the utilization of every physical link and compute the packet drop probability for each virtual link. To compute the packet drop probability, we count the number of packets transmitted and received at the origin and destination nodes of a link. The ratio of dropped packets to transmitted packets gives us the packet drop probability. To compute link utilization, we keep track of every packet transmitted over each link throughout the experiment by creating a trace file. Given the size of packets (1066 bytes including link layer headers), we then compute the utilization of each link over 10- second intervals and use them to compute the overall average utilization of each link. Congestion events are the primary cause of packet drops in our experiments. However, during each congestion event, multiple packets could be dropped. Thus, while there is a

strong correlation between packet drop probability and congestion probability, their values do not necessarily match.

Results and Discussion. Mininet results are summarized in Figures 4.10, 4.11, and 4.12. Specifically, Figure 4.10 depicts the average number of admitted virtual links with each algorithm. As expected, **Approx VLE** admits significantly fewer links compared to the very optimistic **Average**. It, however manages to admit more links compared to 95-**Percentile** as it takes advantage of statistical multiplexing of multiple demands on each link. We see a similar relation between average physical link utilizations in Figure 4.11, where **Average** achieves much higher utilization compared to the other algorithms. However, the price to be paid for such a high utilization is the unacceptably high packet drop probability achieved by **Average**, as depicted in Figure 4.12. As can be seen, while **Approx VLE** and 95-**Percentile** achieve less than 10% drop probability, the drop probability of **Average** can go as high as 60%, which would render the network unusable for any network services relying on TCP. Another interesting observation is that, 95-**Percentile** is very conservative and achieves almost 0% drop probability, at the cost of admitting fewer virtual link requests.

4.6 Summary

In this chapter, we propose exact and approximate formulations for the virtual link embedding problem (VLE). The approximate model solves the problem in a reasonable time and has close results to those of the exact model shown by simulations. Moreover, Mininet experiments have been conducted and the corresponding results confirm simulation results qualitatively. The next chapter presents a complete framework for virtual network embedding problem by replacing random node placement with an existing virtual node embedding algorithm. In addition, real testbed experiment results are provided in the following chapter.

Chapter 5

Virtual Network Embedding

5.1 Virtual Network Embedding Formulation

A congestion-constrained virtual network embedding model, subject to physical node and link capacity constraints, is proposed. We assume that origin-destination of each virtual link is given, virtual network graphs are directed, and virtual nodes of the same virtual network are hosted on different physical nodes. There is a single physical network specified by a directed graph $G(\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ denotes the set of physical nodes and $\mathcal{E}$ denotes the set of physical links. Each physical link is indicated by (i, j) tuple that has fixed capacity denoted by $C_{i,j}$.

Let $\mathcal{R}$ denotes a set of virtual network requests and each request r contains virtual nodes ($v_r \in \mathcal{V}$) which are connected by virtual links ($\ell_r \in \mathcal{L}$), where $\mathcal{V}$ and $\mathcal{L}$ are sets of all virtual nodes and links, respectively. Virtual node v_r demands ω_{v_r} compute capacity and virtual link ℓ_r demands B_{ℓ_r} uncertain bandwidth with known mean and variance $(\mu_{\ell_r}, \sigma^2_{\ell_r})$. Let $\delta_{v_r,n}$ denotes a binary variable that indicates whether or not virtual node v_r is mapped to physical node $n \in \mathcal{N}$, and $y^{\ell_r}_{i,j}$ denotes the total fraction of bandwidth demand B_{ℓ_r} allocated on physical link $(i, j) \in \mathcal{E}$. The VNE problem can be formulated as a non-linear optimization problem, as presented in Problem 3.

Problem 3 Exact VNE

minimize α

subject to:

$$\sum_{n\in\mathcal{N}} \delta_{v_r,n} = 1 \qquad \forall v_r \in \mathcal{V}, r \in \mathcal{R} \qquad (5.1a)$$

$$\sum_{j:(i,j)\in\mathcal{E}} y_{i,j}^{\ell_r} - \sum_{j:(j,i)\in\mathcal{E}} y_{j,i}^{\ell_r} = \delta_{O(\ell_r),i} - \delta_{D(\ell_r),i} \qquad \forall \ell_r \in \mathcal{L}, r \in \mathcal{R}, i \in \mathcal{N} \qquad (5.1b)$$

$$\sum_{v_r\in\mathcal{V},r\in\mathcal{R}} \omega_{v_r}\delta_{v_r,n} \leq C_n \qquad \forall n \in \mathcal{N} \qquad (5.1c)$$

$$\left(2\ln\frac{1}{\varepsilon_{i,j}}\right) \sum_{\ell_r\in\mathcal{L},r\in\mathcal{R}} \sigma_{\ell_r}^2 (y_{i,j}^{\ell_r})^2 \leq \left(\alpha C_{i,j} - \sum_{\ell_r\in\mathcal{L},r\in\mathcal{R}} \mu_{\ell_r} y_{i,j}^{\ell_r}\right)^2 \qquad \forall (i,j) \in \mathcal{E} \qquad (5.1d)$$

$$Z_i^{\ell_r} \geq \lceil y_{j,i}^{\ell_r} \rceil . [Z_j^{\ell_r} + \varepsilon_{j,i}] \qquad \forall \ell_r \in \mathcal{L}, i \in \mathcal{N}, (j,i) \in \mathcal{E} \qquad (5.1e)$$

$$\delta_{D(\ell_r),i} . Z_i^{\ell_r} \leq \varepsilon \qquad \forall \ell_r \in \mathcal{L}, i \in \mathcal{N} \qquad (5.1f)$$

$$\delta_{v_r,n} \in \{0,1\} \qquad (5.1g)$$

$$0 \leq y_{i,j}^{\ell_r}, Z_i^{\ell_r}, \varepsilon_{i,j} \leq 1 \qquad (5.1h)$$

$$\alpha \geq 0\,. \qquad (5.1i)$$

The first constraint ensures that each virtual node is mapped only to one physical node. So when virtual node v_r is mapped to physical node n, $\delta_{v_r,n}$ becomes 1. The second constraint is a flow conservative constraint, i.e., if physical node i is allocated to the origin node of virtual link ℓ_r ($O(\ell_r)$), then summation of the allocated bandwidth to virtual link ℓ_r on physical links connected to node i is equal to 1. But, if physical node i is allocated to the destination node of virtual link ℓ_r ($D(\ell_r)$), then summation of the allocated bandwidth to virtual link ℓ_r on physical links connected to node i equals to -1. Otherwise, physical node i is neither origin nor destination of virtual link ℓ_r, and input traffic is equal to output traffic and, thus, the equation is equal to 0. Constraint 5.1c checks that the aggregate compute capacities of allocated virtual nodes (on a physical node) does not exceed the node capacity.

Constraint 5.1d determines the probabilistic bandwidth allocation to each physical link (i,j) similar to what we have in the VLE formulation (Chapter 4). However, guaranteeing end-to-end congestion probability in the VNE problem is not as straightforward as end-to-

end congestion constraint in the VLE model, since we do not have any information about the routing and paths between origin-destinations in the VNE problem. Hence, we define a new variable $Z_i^{\ell_r}$ that shows the maximum congestion from the origin of virtual link ℓ_r ($O(\ell_r)$) to physical node i. Then, constraint 5.1e computes $Z_i^{\ell_r}$ to be the maximum congestion probability of all paths allocated to virtual link ℓ_r and enter physical node i. If i is the origin of virtual link ℓ_r, then we have $Z_i^{\ell_r} \geq 0$. Constraint 5.1f makes sure that the maximum end-to-end congestion probability of virtual link ℓ_r at $D(\ell_r)$, i.e.,$\delta_{D(\ell_r),i}.Z_i^{\ell_r}$, is less than a predefined value (ε).

The goal is to find routing variables $y_{i,j}^{\ell_r}$ that minimize the utilization of the most congested link in the network while congestion probability for each physical link ($\varepsilon_{i,j}$) is unknown. Let α denotes the utilization of the most congested link. The embedding problem is feasible only if $\alpha \leq 1$. The proposed model is nonlinear and non-convex because of constraint 5.1d, as explained in the previous chapter. Also, it includes binary variables ($\delta_{D(\ell_r),i}$). Therefore, solving this problem is time consuming even for small networks and the solution is sub-optimal.

As discussed in the literature review, we can decompose the VNE problem into two sub-problems (virtual node and link embedding) to tackle the complexity. We use our approximate VLE model for the virtual link embedding stage and an existing heuristic virtual node embedding algorithm (TK-Match) for the node embedding stage. The next section explains the TK-Match algorithm in more details.

5.2 Virtual Node Embedding Algorithm

The TK-Match node embedding algorithm is conducted in three steps: 1) ranking virtual and physical nodes, 2) constructing the node mapping tree (NMT) for each VN request, and 3) mapping nodes based on the selected metric. For ranking the nodes, Li et al. [45] adopt the Top-k dominating model [43] inspired by the query technology used in databases, instead

of using one resource factor (bandwidth, CPU) or the product of resource factors, which eventually leads to low resource utilization. In the next step, a mapping tree is constructed for each VN request according to the virtual node ranking values and the VN's topology. Last, when we have obtained the constructed NMT, TK-Match adopts the BFS strategy to map the virtual nodes. The exact corresponding algorithm for each step can be found in [45]. However, the process of the main algorithm is described in Algorithm 2.

Algorithm 2 Mapping Virtual Nodes

```
1: Rank virtual nodes of each VN request
2: Rank substrate nodes and sort them in decreasing order
3: Construct an NMT for each VN request
4: for all VN request r_i do
5:    n_i ← substrate node with the highest rank
6:    if n_i has enough capacity for r_i.root then
7:       map r_i.root to n_i
8:    else
9:       reject r_i
10:   end if
11:   for all v_j ∈ NMT(r_i) do
12:      n_j ← substrate node with the highest NF
13:      if n_j has enough capacity for v_j then
14:         map v_j to n_j
15:      else
16:         reject r_i
17:      end if
18:   end for
19: end for
```

Adaptation to Uncertain Demands. The TK-Match algorithm ranks virtual nodes based on the bandwidth demands of virtual links connected to them. In the ranking process proposed in [45], bandwidth demands are assumed to be *deterministic*. However, we only know the mean and variance of the bandwidth demand of each virtual link (denoted by μ and σ^2, respectively). Thus, we define a general metric as $\mu + \sigma$ to compare uncertain bandwidth demands. In the case of deterministic bandwidth demands, μ represents the total bandwidth requirement of a virtual link, whereas σ is zero. On the other hand, we have $\sigma > 0$ for uncertain bandwidth demands. CPU capacity demands are compared by

their exact values, as they are assumed to be deterministic.

Node Ranking. The TK-Match algorithm relies on ranking physical and virtual nodes. The Top-k Dominating model [43] is adopted to rank the nodes as opposed to ranking them based on one resource, i.e., bandwidth or CPU, which may lead to an imbalanced resource allocation. The concept of *domination* is defined such that object $t_i = (t_i^1, ..., t_i^k)$ dominates object $t_j = (t_j^1, ..., t_j^k)$ if t_i has higher or equal values in all k dimensions. Also, t_i *directly dominates* t_j, if no other object t_x exists that satisfies $t_i \prec t_x$ and $t_x \prec t_j$. As an illustrative example, consider the partial network depicted in Fig. 5.1. In this network, nodes v and w are at the same level, i.e., there is no domination relationship between them. This means that either they have exactly equal bandwidth and CPU capacities or, if v has higher bandwidth capacity, then w has higher CPU capacity and vice versa. In this example, node v has 10 units of CPU capacity, which is lower than the CPU capacity of node w, whereas the bandwidth capacity of v (i.e., the summation of bandwidth on links connected to v) is 500 units, which is higher than the bandwidth capacity of node w.

After finding domination relationships between each pair of nodes, a Ranking Score (RS) is computed for each node v as follows,

$$RS(v) = \sum_{j=1}^{k} RS(v_j) + 1/n_j, \tag{5.2}$$

where v_j is a node directly dominated by v, and n_j is the number of nodes that directly dominate v_j. From this, we can rank nodes in different levels, like the example in Figure 5.2 taken from [45].

Node Mapping Tree. Once all nodes are ranked, a so-called Node Mapping Tree (NMT) is created for each VN request such that the root of the NMT is the virtual node with the highest rank. Then, the virtual nodes directly connected to the root node are selected and added to the NMT as children of the root node from left to right according to the descending order of their ranks. This process is repeated for other virtual nodes until all virtual nodes of

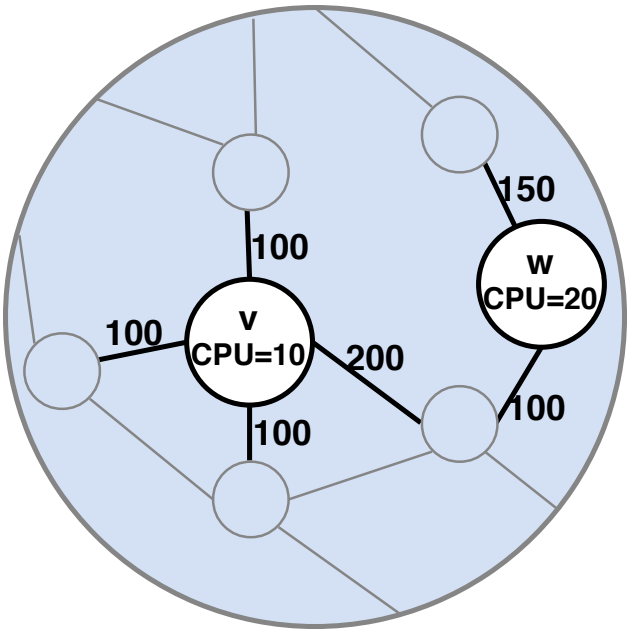


Figure 5.1: CPU and Bandwidth capacity of nodes.

the VN are successfully included in the NMT. In this way, topological attributes of network are considered in the node embedding stage.

Figure 5.3 shows an example of constructing an NMT for virtual networks. The numbers in rectangles show the required CPU and the paired numbers over the virtual links represent the required mean and standard deviation (μ_i, σ_i) of flow rates in Figure 5.3(a). The numbers in Figure 5.3(b) represent the RS values of the virtual nodes. Also, the parent–child pair of nodes refer to the two nodes that are connected by edges in the NMT; for example, nodes E and C are a parent–child pair of nodes in Figure 5.3(b).

Node Embedding. To embed a VN, the root of its NMT is first mapped to the substrate node with the highest rank. Other nodes in the corresponding NMT of the VN are mapped to substrate nodes, starting from the children of the root node, following a Breadth-First approach, in which the children of a node are mapped from left to right. Each child node is mapped to a substrate node with the highest NF value. The NF is defined to be proportional to the ranking scores of the substrate nodes and inversely proportional to the length of the substrate paths between the parent-child pair of virtual nodes [45]. When we map the virtual

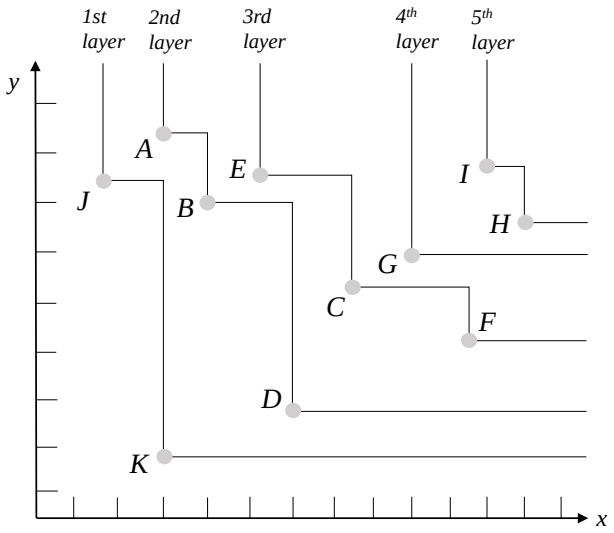


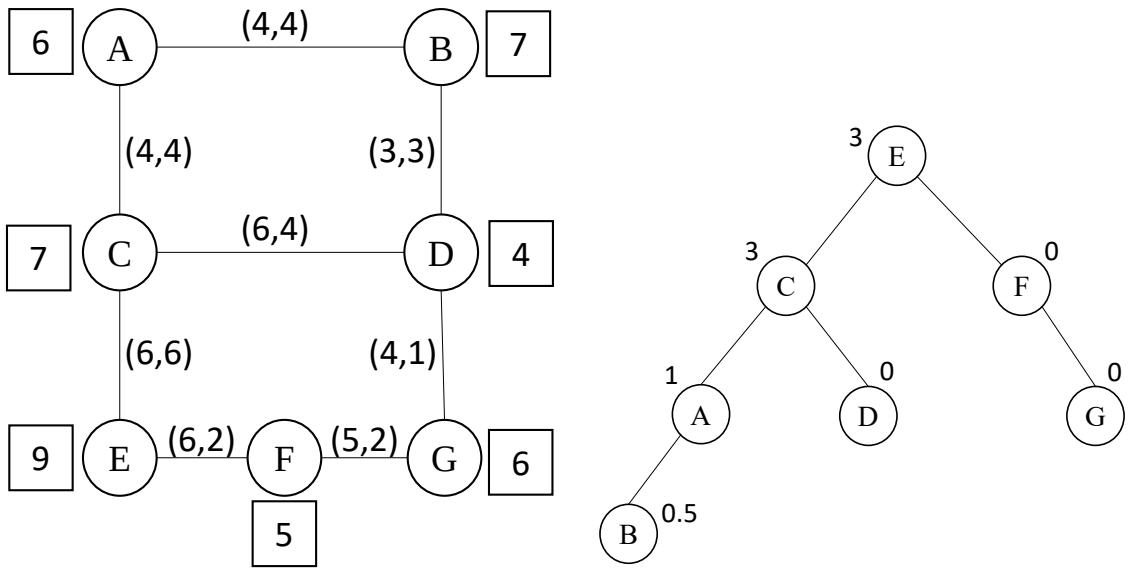
Figure 5.2: Ranking nodes in 2-dimension.

node v_w to the substrate node v_s, the NF of v_s is defined as follows:

$$NF(v_s) = RS(v_s).\frac{1}{hops(M^V(f(v_w)), v_s)}, \tag{5.3}$$

where $f(v_w)$ refers to the father of v_W in NMT, from which we know that $f(v_w)$ and v_w are a parent–child pair of nodes. $M^V(f(v_w))$ refers to the substrate node that the virtual node $f(v_w)$ is mapped onto, and $hops(M^V(f(v_w)), v_s)$ refers to the substrate path with the least hops between node v_s and node $M^V(f(v_w))$ in the substrate network [45].

The set of VNs in a batch are embedded in a first-come first-served manner by applying the procedure described above. Once the node embedding algorithm terminates, we have a set of VN requests that are accepted and another set that are rejected. At this point, the Approximate VLE algorithm is used to embed the virtual links belonging to the set of accepted VNs. The outcome of the algorithm is the set of optimal routing variables x_{ij} and the maximum link utilization α.



(a) VN. (b) Node Mapping Tree.

Figure 5.3: Constructing NMT for a VN.

5.3 Testbed Experiments

For this final set of experiments, we have deployed our algorithm on a Software-Defined Networking (SDN) testbed to explore its behavior and performance in a realistic network setting.

Physical Topology. The testbed consists of four Aruba 2930F JL259A OpenFlow switches and five DELL PowerEdge R330 servers equipped with Intel Xeon E3-1240 v6 processors and 16 GB RAM. Detailed specifications of OpenFlow switches and servers can be found in Table 5.2 and 5.1, respectively. Each physical server runs several virtual machines (VMs) corresponding to the nodes of the substrate network. Also, each VM is connected to a different OpenFlow Virtual Switch (OVS) [34]. Every two or three OpenFlow Virtual Switches are deployed on a separate physical switch. This grouping and configuration is conducted in a way that minimizes the number of physical links between physical switches. Each VM-to-switch or switch-to-switch physical link has 1 Gbps bandwidth. Using this setup, the network topology used for the experiments includes 11 OVSs and 11 VMs. Fig. 5.4 shows the actual implementation of the SDN testbed and Fig. 5.5 depicts the configured physical

network. We replicate the Abilene topology in the testbed, which is smaller than the USA topology in terms of the number of nodes and links. This was due to the number of available Ethernet ports on our switches. As can be seen, one server is dedicated to control and service management tasks and can be administered directly by a XenCenter client. An SDN Ryu controller [26] is used to send the required information to virtual machines on other servers, communicate information down to the switches, and collect a set of OpenFlow statistics through its REST interface.

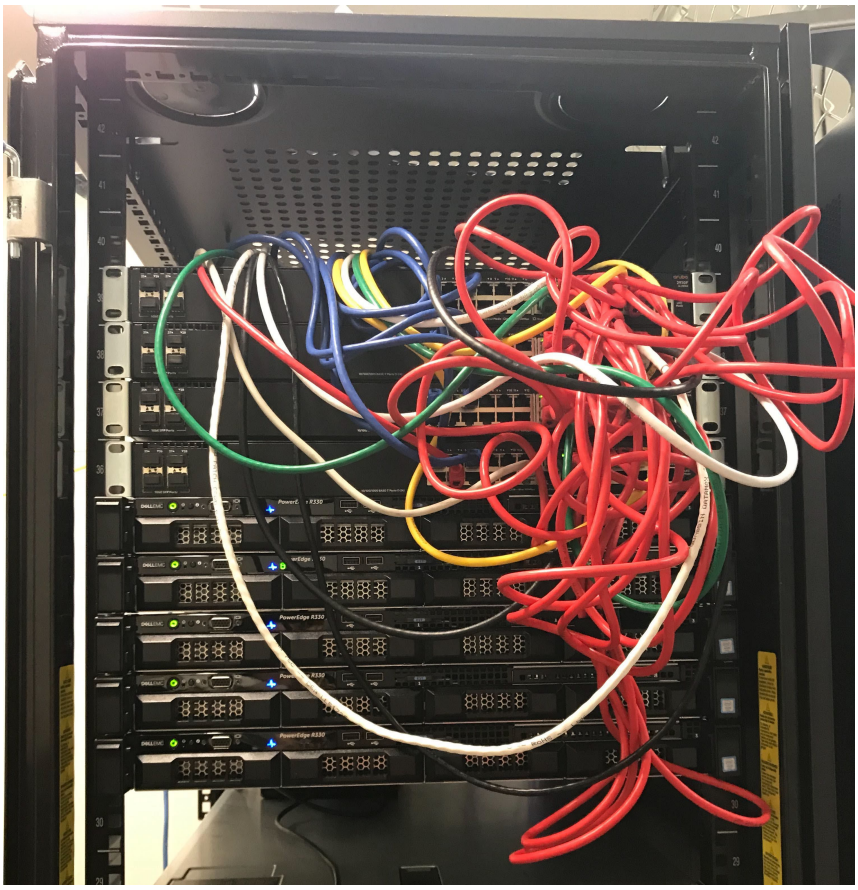


Figure 5.4: Actual testbed implementation.

Physical Switch. Each physical switch supporting OpenFlow v1.3 can have up to 16 OpenFlow Virtual Switches, known as OpenFlow instances, running concurrently. Each OpenFlow instance has its own member VLAN (list of ports), controller (IP address, port number), and flow tables. We control all instances with one controller. By default each OpenFlow instance has three flow tables: 1) Table 0: Basically a dummy table that redirects

Table 5.1: DELL PowerEdge R330 server specification.

CPU	Architecture: x86_64 CPU(s): 4 Thread(s) per core: 2 Core(s) per socket: 2 Socket(s): 1 Model Name: Inter(R) Xeon(R) CPU E3-1240 v6 3.70GHz L1d cache: 32K L1i cache: 32K L2 cache: 256K L3 cache: 8192K
Network Adapter	3 Standards-compliant dual-port 1GBASE-T PCIe 2.1 Ethernet Controllers Energy Efficient Ethernet compliant with IEEE 802.3az-2010 Dual 10/100/1000BASE-T full-duplex/half-duplex MACs Virtualization support with VMware NetQueue and Microsoft VMQ Wake-on-LAN support
RAM	16GB
Disk	1TB

all traffic to table 100, 2) Table 100: The flow table that is implemented in hardware (HW) as TCAM, where OpenFlow rules are added, and 3) Table 200: The software (SW) flow table that can be used if we exceed the capacity of Table 100; however, switching rates when using this table will be very slow (less than 1 Mbps). We only used HW flow tables, as there was no issue with having enough capacity for our set of rules.

Virtual Network Generation. Virtual networks are generated randomly such that the number of virtual nodes and links for a VN are uniformly chosen from the range $[2, 3, 4]$. Then the Barabasi-Albert model [11] is used to generate the corresponding virtual network topology. The next step is about assigning CPU and uncertain bandwidth demands to virtual nodes and links, respectively. Since the focus of our work is on link embedding, CPU demands are considered small enough compared to actual processing power capacity of servers so that they do not force any capacity restrictions in the node embedding process.

Table 5.2: Aruba 2930F 24G 4SFP Switch (JL259A) specification.

I/O ports and slots	Additional ports and slots	Physical characteristics
24 RJ-45 autosensing 10/100/1000 ports (IEEE 802.3 Type 10BASE-T, IEEE 802.3u Type 100BASE-TX, IEEE 802.3ab Type 1000BASE-T); Duplex: 10BASE-T/100BASE-TX: half or full; 1000BASE-T: full only; 4 SFP.	1 dual-personality (RJ-45 or USB micro-B) serial console port.	17.42 (w) x 7.88 (d) x 1.73 (h) in (44.25 x 20.02 x 4.39 cm) (1U height), 5.31 lb (2.41 kg)
Memory and processor	**Performance**	**Electrical characteristics**
Dual Core ARM Coretex A9 1016 MHz, 1 GB DDR3 SDRAM; Packet buffer size: 12.38 MB 4.5MB Ingress/7.875MB Egress, 4 GB eMMC.	IPv6 Ready Certified; 1,000 Mb Latency $< 3.8\ \mu s$ (64-byte packets); Throughput up to 41.7 Mpps; Switching capacity 56 Gbps; Routing table size 2,000 IPv4, 1,000 IPv6 in hardware, 200 OSPF, 256 Static, 10,000 RIP; MAC address table size 32,768 entries.	Frequency 50/60 Hz; Maximum heat dissipation 100 BTU/hr (105.5 kj/hr); Voltage 100 - 127 / 200 - 240 VAC, rated; Current 0.6/0.4 A; Maximum power rating 29.3 W; Idle power 19.5 W.
Emissions	**Management**	**Environment**
EN 55032:2012/CISPR 32 Class A; FCC CFR 47 Part 15 Class A; VCCI Class A; ICES-003 Class A; CNS 13438.	Aruba Central; Aruba AirWave Network Management; IMC – Intelligent Management Center; Command-line interface; Web browser; Configuration menu; SNMP manager; Telnet; RMON1; FTP; Out-of-band management (serial RS-232C or micro USB).	Operating temperature 32°F to 113°F (0°C to 45°C); up to 5,000 Feet, 0°C to 40°C (32°F to 104°F) up to 10,000 Feet; Non-operating/Storage temperature -40°F to 158°F (-40°C to 70°C); up to 15,000 Feet; Acoustic Power: 49.7 dB, Pressure: 37.1 dB; Airflow direction Side-to-side.

On the other hand, the uncertain bandwidth demand for each virtual link ℓ is defined by its mean and standard deviation, i.e., (μ_ℓ, σ_ℓ). In testbed experiments, we have two sets of input traffic: (1) one set with homogeneous bandwidth demands, where μ_ℓ and σ_ℓ are fixed and set to 100 Mbps, and (2) another set with randomly chosen means and coefficient of variations from $\{100, 200, 300\}$ and $\{0, 0.5, 1\}$, respectively.

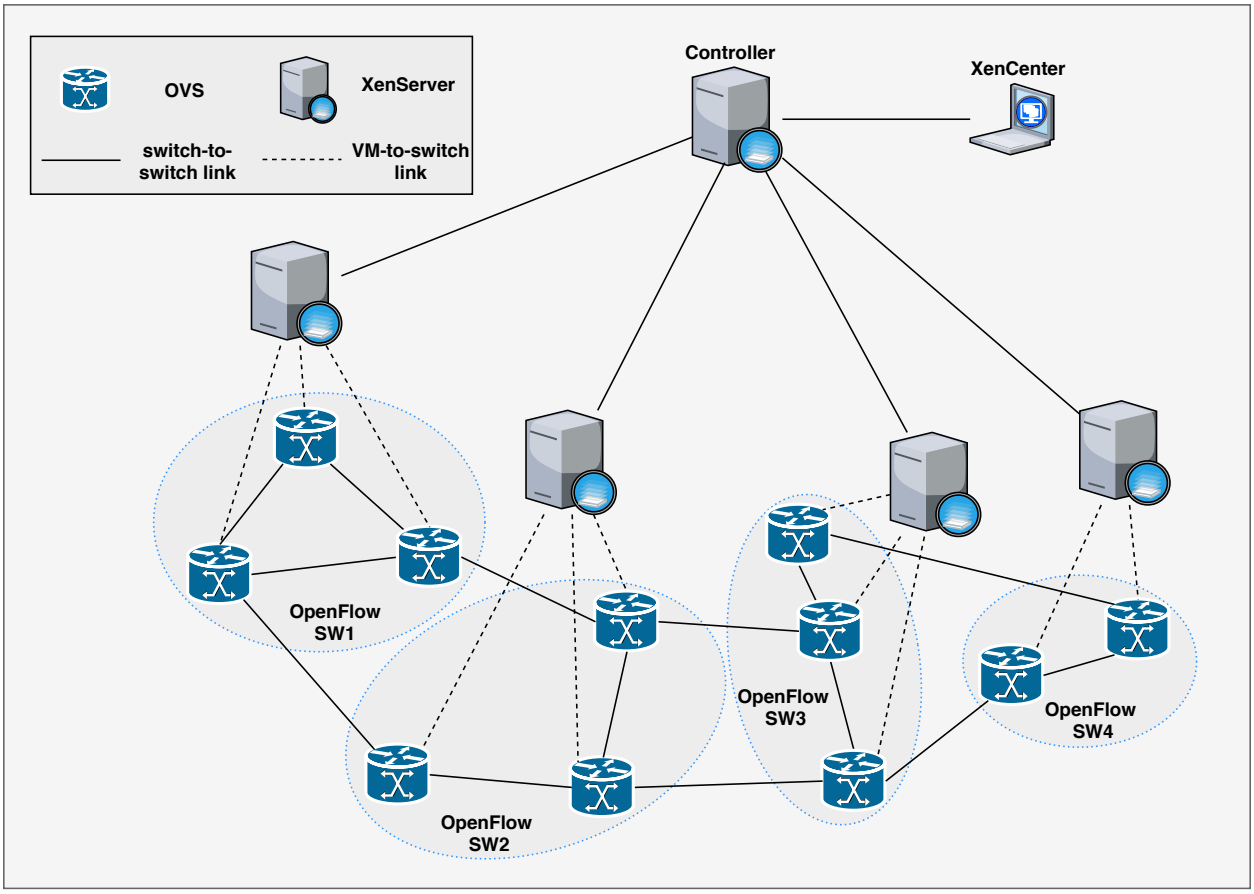


Figure 5.5: Testbed configuration.

Multipath Routing. A flow is denoted as a unidirectional series of packets sent from one end host to another via the substrate network. The experiments presented in this section were conducted while maintaining the assumption that there are flows with identical source and destination end hosts, since each flow or virtual link may utilize multiple distinct physical paths in the substrate network. In fact, the presented model assumes that it is possible to route the packets of a particular flow over multiple paths in the substrate network while

maintaining a precise splitting ratio for the traffic that comprises the flow. From the perspective of practical implementation, a problem arises only if the source-destination addresses are used for routing, as the OpenFlow protocol mandates that the mapping between header fields of a packet and the resulting forwarding action taken by the switch be deterministic. As such, the values of the Layer 3 network header are needed to uniquely identify and route an end-to-end flow.

More specifically, consider two flows F_i and F_j denoted by the 2-tuples (S_i, D_i) and (S_j, D_j). If $S_i = S_j$ and $D_i = D_j$ then the forwarding action taken at a particular switch at a specific point in time for these two packets must be identical. While this does not pose a problem when it comes to discriminating between multiple constituent flows on the network, it implies that all traffic for a particular flow must be routed over an identical path in the physical network. This is resultant of the fact that any given switch in the network will not be able to distinguish between different packets belonging to a single flow that were intended to be routed over distinct physical paths in the substrate network.

In order to subvert this problem, each flow (over a virtual link) is segmented into K subflows where K is the number of distinct physical paths in the substrate network over which the flow's traffic will be routed. An additional IP header field is used to discriminate between each of these subflows. In this case, the Differentiated Services Codepoint (DSCP) value is set so that it indicates which subflow a particular packet belongs to. Thus the j_{th} subflow of the i_{th} flow can be uniquely identified by the 3-tuple (S_i, D_i, C_{ij}), where C_{ij} is the corresponding DSCP value, allowing for the correct implementation of the multipath routing described in Section 4.2.

Suppose that VN requests are embedded, then information regarding placement of virtual nodes on the substrate network, their corresponding bandwidth demands, K-shortest paths between each pair of connected virtual nodes, and ratio of traffic on each end-to-end physical path is given to the SDN controller for the purpose of establishing origin-destinations and routes between them. Then, each trial runs for a period of 15 minutes, during which time,

each origin VM, representing a virtual node generates traffic following Gamma distribution with given mean and standard deviation, and traffic rates are changed every second in order to induce variability. We decided to use Gamma distribution over Normal, since all randomly generated rates are positive and there is no need to ignore negative values.

Traffic Generation. Because it was necessary to implement the DSCP tagging scheme described previously, we were not able to utilize any existing traffic generation tools. As such, we implemented a basic traffic generation application that allows packets to be generated and transmitted in compliance with a number of common statistical distributions. In addition, the traffic generation application is also responsible for tagging generated packets according to the path splitting ratios, i.e., x_{ij} variables. The traffic generator is implemented in Python using the Python sockets API.

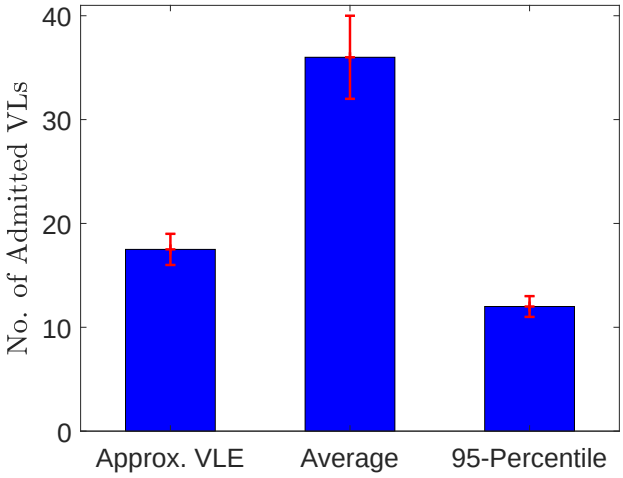


Figure 5.6: First trial - Average number of admitted virtual links.

Measurements and Statistics Collection. The testbed statistics are collected from two distinct sources: per port and per flow statistics. The term *port* describes a single physical port contained in one of the OpenFlow switches that implements the substrate network. The statistics collected for each of these ports include the following:

1. The number of packets received on the port.

2. The number of packets transmitted on the port.

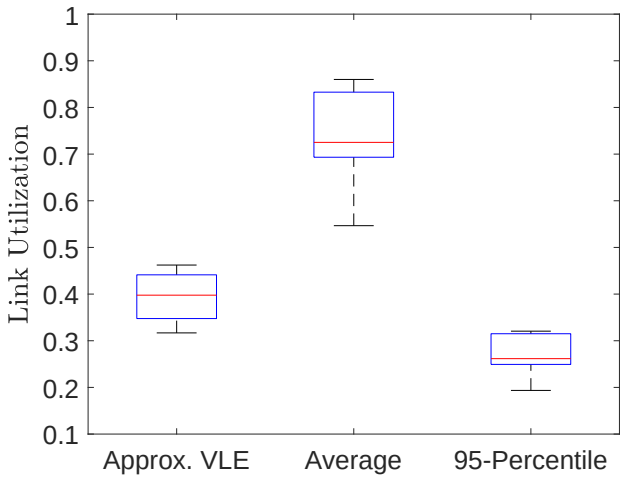


Figure 5.7: First trial - Average link utilization of physical links.

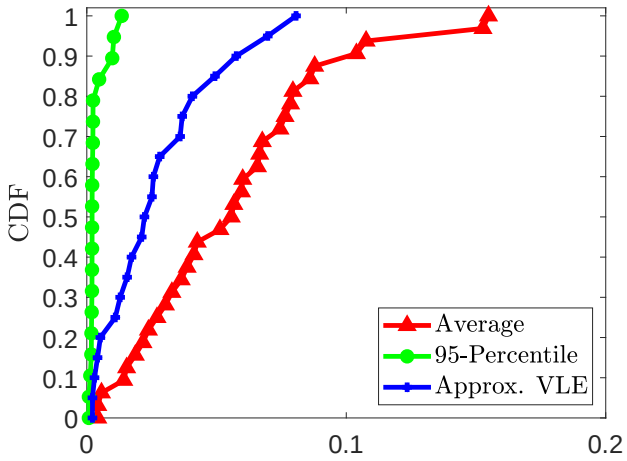


Figure 5.8: First trial - CDF of packet drop probability.

3. The number of bytes received on the port.

4. The number of bytes transmitted on the port.

To obtain this information, the application responsible for orchestrating the activities of the testbed issues REST calls to the Ryu controller, causing the transmission of `OFPMP_PORT` requests to all of the switches in the substrate network at some fixed time interval θ. Thus, for a test that is conducted over a period of T seconds, $\lfloor \frac{T}{\theta} \rfloor$ measurements will be sampled from each of the switches in the substrate network. This set of measurements is then used to compute the port utilization statistics for each of the sampled time intervals. The only statistic collected for each of the flows in the network is the end-to-end packet drop rate,

which is computed using statistics about the total number of packets transmitted and received at the origin and destination of a flow. To collect this information, the orchestrator application monitors both the traffic generation and traffic reception applications running on the source and destination end hosts of the flows in order to determine the number of packets sent and received respectively. More formally, the end-to-end packet drop rate, denoted as p, can be defined as follows,

$$p = \frac{PacketsReceived - PacketsSent}{PacketsSent} \tag{5.4}$$

Experiment Setup. Three types of trials are performed as follows:

1. Random node embedding followed by the **Approx VLE**: In this case, (μ_{VL}, σ_{VL}) is equal to $(100, 100)$ Mbps for all virtual links.

2. Node embedding with TK-Match followed by the**Approx VLE** algorithm: In this set of trials, we explore more realistic scenarios where virtual links can have a different mean rate and coefficient of variation. Specifically, mean rate is randomly chosen from $\{100, 200, 300\}$ Mbps, while CoV can be in $\{0, 0.5, 1\}$. Based on given mean and variance, virtual nodes generate traffic following Gamma distribution.

3. Similar to the second trial, the **Approx VLE** algorithm is joint with the TK-Match algorithm. Also, mean rate and CoV are randomly selected from the same sets, but virtual nodes generate traffic following Uniform distribution.

For each type of trial, three VLE algorithms are implemented, as in the Mininet experiments. We use fixed mean and standard deviation for flows in the first trial to eliminate the effect of other parameters when comparing different algorithms in terms of link utilization and packet drop probability. In the second trial, experimental results in a more realistic setting are depicted, where mean and variance of flows are random. And in the third trial, we apply settings of the second trial and use Uniform distribution for real traffic generation.

Discussion. Testbed results for the first trial are depicted in Figures 5.6, 5.7, and 5.8. As can be seen, these results closely match the results obtained in Mininet experiments. We observe that **Approx VLE** is able to keep the packet drop rate below 10%, while embedding as many virtual links as possible in the network.

Results for the second trial are shown in Figures 5.9, 5.10, and 5.11. In this trial, more uncertainty exists in bandwidth demands compared to the first trial, such that mean rates can be 100, 200, or 300 Mbps, whereas coefficient of variation can be 0, 0.5, or 1. As a result, we see that the packet drop probability for all three algorithms has increased more than 10% compared to the first trial. Virtual networks can have any desired topology and include 2-4 virtual nodes, and they are placed based on the TK-Match algorithm. Figure 5.9 illustrates the number of admitted VN requests and their corresponding virtual links for all three algorithms. We observer that, compared to the first trial, more virtual links are embedded in the network on average. This is due to the fact that the employed node embedding algorithm (i.e., TK-Match) is more efficient than the random embedding considered in previous trials.

An interesting behavior is observed in Fig. 5.10 with respect to link utilization under 95-**Percentile** and **Approx VLE**. They have closer range and average. The *difference* between the number of admitted VN requests by these two algorithms decreased because of the heuristic node embedding process. The reason is that the order and type of VNs become important as we consider heterogeneous VNs in this experiment. This means that a different set of VNs is admitted under each algorithm. Therefore, we can have situations where the 95-**Percentile** algorithm rejects some large VNs with high bandwidth demand while the **Approx VLE** algorithm admits some of the large VNs. As a result, while both algorithms admit roughly the same number of virtual networks, the link utilization under **Approx VLE** is higher.

Finally, as shown in Fig. 5.11, there is a probability that both algorithms **Approx VLE** and 95-**Percentile** result in packet drop probabilities higher than 10%. As discussed earlier, this does not necessarily mean that the achieved congestion probabilities are higher than

10%. Recall that during a single congestion event, multiple packets could be dropped. As such, in general the packet drop probability provides an upper bound on the actual congestion probability.

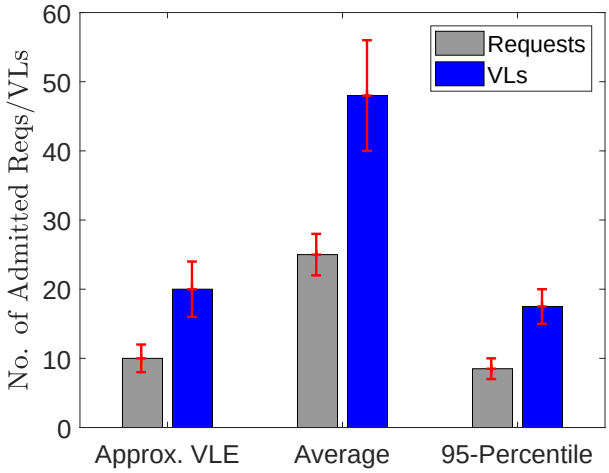


Figure 5.9: Second trial - Average number of admitted virtual links and requests.

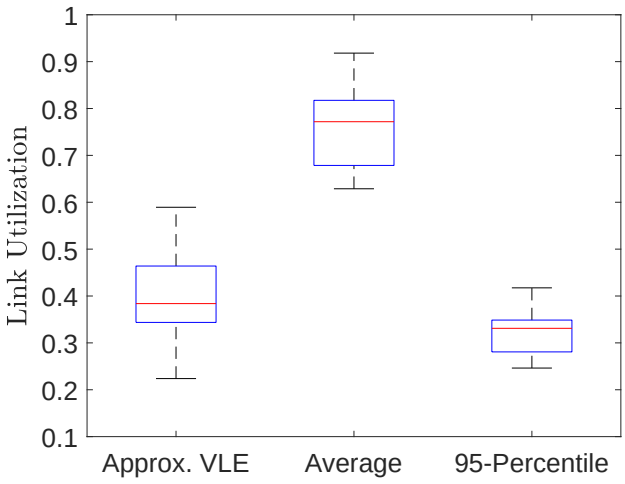


Figure 5.10: Second trial - Average link utilization of physical links.

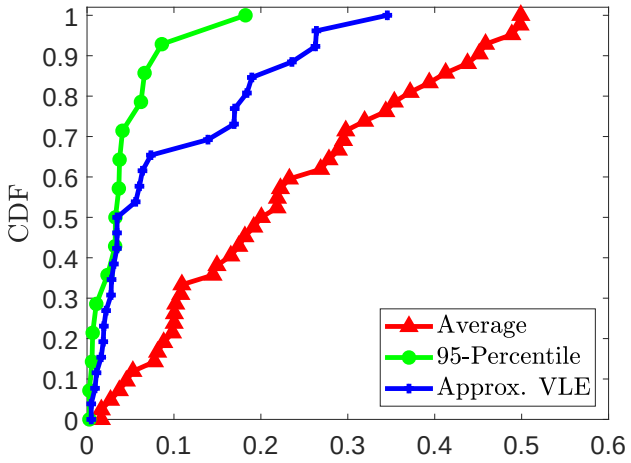


Figure 5.11: Second trial - CDF of packet drop probability.

Figures 5.12 and 5.13 depict the results for the third trial. In this trial, virtual nodes (VMs) generate traffic following Uniform distribution. We examine the same set of VN requests as shown in Figure 5.9. However, we see more than 10% higher link utilization and packet drop probability in 5.12 and 5.13, respectively. This is due to the fact that high flow rates are as probable as flow rates around the mean rate in uniformly distributed rates.

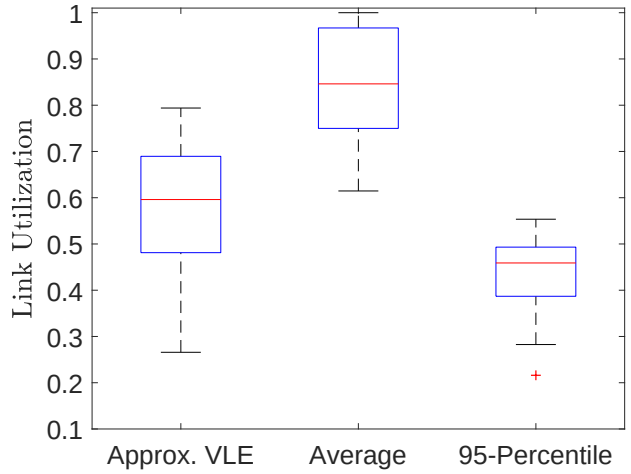


Figure 5.12: Third trial - Average link utilization of physical links.

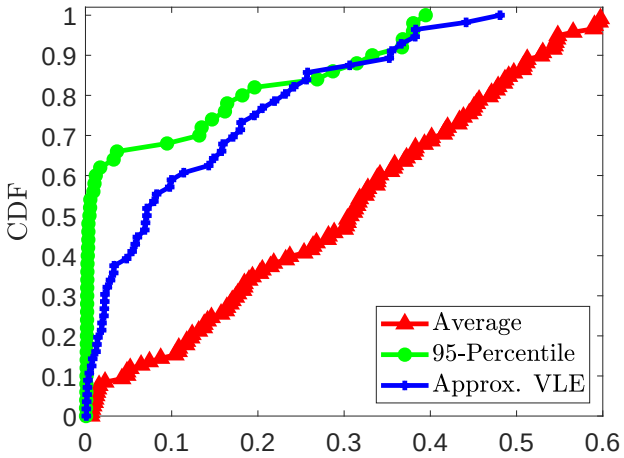


Figure 5.13: Third trial - CDF of packet drop probability.

5.4 Summary

In summary, three types of trials are examined in this chapter. One implements the Approximate VLE model, augmented by the random placement of virtual nodes on the substrate network, second implements the Approximate VLE model, coordinated with an existing virtual node embedding algorithm (TK-Match), and third uses the same setting as the second trial but with different distribution (Uniform distribution) for the traffic generation. Based on the provided plots, testbed experimental results match the Mininet results presented in the previous chapter qualitatively. Moreover, the second trial shows 10% more packet drop probability compared to the first trial, since more uncertainty is considered in bandwidth demands of virtual links in the second trial. Also, the number of embedded virtual links increased on average, because the TK-Match algorithm efficiently embeds VN requests by considering both bandwidth demands and topological attributes in the node embedding stage. The third trial shows higher link utilization and packet drop probability compared to the second one as Uniform distribution makes high rates more probable.

Chapter 6

Conclusions and Future Work

This chapter presents a summary of the possible future directions and conclusions of the thesis. Section 6.1 restates the work's main contributions and concluding remarks about the research whereas Section 6.2 presents some areas for future work.

6.1 Summary and Conclusions

The following conclusions can be drawn from this research:

- Demand uncertainty is considered in the modelling of the virtual link embedding problem, where only limited information about virtual network requests, i.e.,mean and variance, is needed.
- Different optimization strategies, i.e., stochastic and robust optimizations, are explored to identify a suitable approach for probability guarantee under demand uncertainty.
- The congestion-constrained virtual link embedding problem under demand uncertainty can be formulated as a chance-constrained or robust optimization problem. The pros and cons of these approaches are investigated, demonstrating that both the chance-constrained problem approximated with tail bounds and the robust optimization problem with ellipsoidal uncertainty set result in the same nonlinear model.

- The Γ-Robust framework provides a linear formulation for a general virtual link embedding problem under demand uncertainty as opposed to chance-constrained optimization and robust optimization with ellipsoidal uncertainty set; however, it transforms to a nonlinear optimization problem by considering congestion constraints on virtual links.

- The link mapping problem is formulated with constrained end-to-end congestion probability as a non-linear optimization problem that can be solved using global optimization solvers for small networks. It provides a reference for our approximate solution.

- An approximate link mapping solution is proposed for the problem, which is formulated as a second-order cone program (SOCP) and can be solved efficiently for large networks in real scenarios.

- Simulation and Mininet [4] experiments are conducted to show the efficiency and utility of our link mapping solutions in small and large network instances.

- The congestion-constrained virtual network embedding problem is formulated as a complete solution, and it is shown to be NP-hard. So, we are required to break down the VNE problem into two sub-problems (virtual node and link embedding).

- Our approximate link mapping solution coordinates with an existing virtual node embedding approach to implement a complete framework for the virtual network embedding problem.

- The joint link and node embedding solution is implemented and evaluated on a real testbed, which includes an SDN controller and OpenFlow switches, in order to analyse the acceptance ratio of the framework and performance of virtual networks.

- Simulations, Mininet, and OpenFlow testbed experiments indicate that considering uncertainty in demands leads to more efficient usage of network resources, and the proposed design can successfully tolerate the traffic fluctuation in OpenFlow networks.

6.2 Future Research Directions

This section highlights interesting future research directions of VNE algorithms. We can categorize them to four main fields:

- Dynamic Distributed VNE: Recent studies were explored in Chapter 2, and the only dynamic and distributed approach was proposed by Houidi et al. [39]. It considers online virtual network embedding and addresses the weaknesses of centralized approaches. But, it suffers from two main problems: an excessive number of coordination messages and suboptimal embedding cost. Hence, a good direction for the future research is to propose distributed algorithms, trying to reduce the message overhead by, for instance, using clustering techniques that confine the embedding tasks to a limited subset of the substrate network, thereby reducing embedding costs.

- Investigating new VNE objectives: Other important metrics in a virtual network embedding solution are energy consumption and security. We need VNE algorithms that aim to minimize energy consumption without compromising network performance, while finding near-optimal solutions within a reasonable time. Additionally, there are security requirements for infrastructure/service providers and users. Therefore, security-aware VNE algorithms are needed to minimize risk exposure and threats for all involved partners.

- Applying VNE algorithms to new environments: Most virtual network embedding algorithms are presented in the context of wired networks and are discussed less specifically in the context of mobile computing and wireless networks. It is a challenging field, since characteristics of wireless environments such as mobility, distribution, and the broadcast nature of wireless links must be considered in the VNE problem. Therefore, dynamic distributed VNE approaches should be explored for wireless networks as well.

- Improving node ranking algorithms: We suggest improving the multi-factor node ranking algorithm in two directions: first, providing a better metric to compare uncertain bandwidth demands instead of simply adding mean and standard deviation of demands; second, considering uncertain CPU demands in the ranking process to generalize the problem. Together, these result in more efficient resource allocation.

Bibliography

[1] Ampl, https://ampl.com/, 2018.

[2] Artelys knitro, https://www.artelys.com/en/optimization-tools/knitro, 2018.

[3] Gurobi optimization, http://www.gurobi.com/, 2018.

[4] Mininet, http://mininet.org/, 2018.

[5] Planetlab, https://planet-lab.org/, 2018.

[6] Separation of mechanism and policy, https://en.wikipedia.org/wiki/separation_of_-mechanism_and_policy, 2018.

[7] F. Alizadeh and D. Goldfarb. Second-order cone programming. *Math. Program.*, 95(1):3–51, December 2001.

[8] A. Altin et al. Provisioning virtual private networks under traffic uncertainty. *Netw.*, 49(1):100–115, January 2007.

[9] D. G. Andersen. Theoretical approaches to node assignment. `https://pdfs.semanticscholar.org/7f19/a844e2dcdbdb03f5dcb6fa842e19118ca80f.pdf`, 2002. unpublished Manuscript.

[10] N. Andreasson, A. Evgrafov, and M. Patriksson. *An Introduction to Optimization: Foundations and Fundamental Algorithms.* Chalmers University of Technology Press, 2005.

[11] A. Barabas and Reka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, October 1999.

[12] A. Ben-Tal, L. E. Ghaoui, and A. Nemirovski. *Robust optimization*. Princeton University Press, 2009.

[13] T. Benson et al. Understanding data center traffic characteristics. *SIGCOMM Comput. Commun. Rev.*, 40(1):92–99, 2010.

[14] D. Bertsimas and M. Sim. The price of robustness. *Oper. Res.*, 52(1):35–53, February 2004.

[15] J. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer-Verlag, New York, 1997.

[16] J.F. Botero et al. Energy efficient virtual network embedding. *Comput. Netw.*, 16(5):756–759, May 2012.

[17] G. C. Calafiore and L. El Ghaoui. Distributionally robust chance-constrained linear programs with applications. *Optim. Theory Appl.*, 130(1):1–22, 2006.

[18] H. Cao et al. Exact solutions of vne: A survey. *China Commun.*, 13(6):48–62, June 2016.

[19] W. Chen et al. From cvar to uncertainty set: Implications in joint chance-constrained optimization. *Operations Research*, 58(2):470–485, 2010.

[20] Y. Chen et al. Resilient virtual network service provision in network virtualization environments. In *International Conference on Parallel and Dist. Sys.*, Shanghai, China, December 2010.

[21] X. Cheng et al. Virtual network embedding through topology-aware node ranking. *ACM SIGCOMM Comput. Commun. Review*, 41(2), April 2011.

[22] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Math. Statist.*, 23(4):493–507, December 1952.

[23] N. M. Chowdhury, M. R. Rahman, and R. Boutaba. Virtual network embedding with coordinated node and link mapping. In *Proc. IEEE INFOCOM*, Rio de Janeiro, Brazil, April 2009.

[24] N. M. Chowdhury, M. R. Rahman, and R. Boutaba. Vineyard: virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Trans. Netw.*, 20(1):206–219, February 2012.

[25] N.M. Chowdhury and R. Boutaba. A survey of network virtualization. *Comput. Netw.*, 54(5):862–876, 2010.

[26] Ryu SDN Framework Community. Component-based software-defined networking framework, https://osrg.github.io/ryu/, 2017.

[27] S. Coniglio, A. Koster, and M. Tieves. Virtual network embedding under uncertainty: exact and heuristic approaches. In *Proc. IEEE Design of Reliable Commun. Netw.*, Kansas City, USA, March 2015.

[28] S. Coniglio, A. Koster, and M. Tieves. Data uncertainty in virtual network embedding: robust optimization and protection levels. *Netw. Syst. Manage.*, 24(3):681–710, May 2016.

[29] D. B. Brown D. Bertsimas and C. Caramanis. Theory and applications of robust optimization. *Society for Industrial and Applied Math.*, 53(3):464–501, 2011.

[30] D. Pachamanova D. Bertsimas and M. Sim. Robust linear optimization under general norms. *Oper. Res. Lett.*, 32(6):510–516, November 2004.

[31] G. Dantzig. Linear programming under uncertainty. *Management Sci.*, 1(3/4):197–206, 1955.

[32] E. K. Lua et al. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Commun. Surveys and Tutorials*, 7(2):72–93, 2005.

[33] A. Fischer et al. Virtual network embedding: a survey. 15(4):1888–1906, October 2013.

[34] Linux Foundation. Production quality, multilayer open virtual switch, https://www.openvswitch.org, 2016.

[35] R. Guerin, H. Ahmadi, and M. Naghshineh. Equivalent capacity and its application to bandwidth allocation in high-speed networks. *IEEE J. Sel. Areas Commun.*, 9(7):968–981, September 1991.

[36] H.Arsham and M.Oblak. Perturbation analysis of general lp models: A unified approach to sensitivity, parametric, tolerance, and more-for-less analysis. *Math. Comput. Modelling*, 13(8):79–102, 1990.

[37] O. Heckmann, J. Schmitt, and R. Steinmetz. Robust bandwidth allocation strategies. In *IEEE IWQoS*, Miami Beach, USA, May 2002.

[38] F. Hosseini, A. James, and M. Ghaderi. Congestion-constrained virtual link embedding with uncertain demands. In *Proc. IEEE/ACM/IFIP International Conference on Netw. and Service Manage.*, Rome, Italy, November 2018.

[39] I. Houidi et al. Adaptive virtual network provisioning. In *ACM SIGCOMM workshop on Virtualized infrastructure sys. and architectures*, New York, NY, USA, September 2010.

[40] I. Houidi et al. Virtual network provisioning across multiple substrate networks. *Comput. Netw.*, 55(4):1011–1023, March 2011.

[41] P. Joos and W. Verbiest. A statistical bandwidth allocation and usage monitoring algorithm for atm networks. In *Proc. ICC*, Massachusetts, USA, December 1989.

[42] P. Kall and S. Wallace. *Stochastic Programming.* John Wiley, Chichester, UK, 1994.

[43] S. S. W. Lee et al. Probabilistic skylines on uncertain data. In *Proc. VLDB*, Vienna, Austria, September 2007.

[44] S. S. W. Lee et al. Design of bandwidth guaranteed openflow virtual networks using robust optimization. In *Proc. IEEE GLOBECOM*, Austin, USA, December 2014.

[45] X. Li et al. Resource allocation with multi-factor node ranking in data center networks. *Future Generation Comput. Sys.*, 32, March 2014.

[46] C. Qiu, H. Shen, and L. Chen. Probabilistic demand allocation for cloud service brokerage. In *Proc. IEEE INFOCOM*, San Francisco, USA, April 2016.

[47] A. Razzaq et al. Minimizing bottleneck nodes of a substrate in virtual network embedding. In *Netw. of the Future*, Paris, France, September 2011.

[48] A. Razzaq and M. Rathore. An approach towards resource efficient virtual network embedding. In *Evolving Internet*, Valcencia, Spain, September 2010.

[49] M. Roughanx et al. Class-of-service mapping for qos: a statistical signature-based approach to ip traffic classification. In *Proc. ACM IMC*, Taormina, Italy, October 2004.

[50] G. Schaffrath et al. Generalized and resource-efficient vnet embeddings with migrations. In *Arxiv preprint arXiv10124066*, December 2010.

[51] A. L. Soyster. Convex Programming with Set-Inclusive Constraints and Applications to Inexact Linear Programming. Technical Report 21-05, Operations Research, 1973.

[52] G. Sun et al. Exploring online virtual networks mapping with stochastic bandwidth demand in multi-datacenter. *Photon. Netw. Commun.*, 23(2), April 2012.

[53] et al. T. Anderson. Overcoming the internet impasse through virtualization. *Computer*, 38(4):34–41, 2005.

[54] M. Wainwright. Basic tail and concentration bounds. `https://www.stat.berkeley.edu/~mjwain/stat210b/Chap2_TailBounds_Jan22_2015.pdf`, 2015. Online; accessed 28 May 2018.

[55] C. Wang and H. P. Luh. Analysis of bandwidth allocation on end-to-end qos networks under budget control. *Computers and Mathematics with Applications*, 62(1):419–439, July 2011.

[56] Wikipedia. Piecewise linear function, https://en.wikipedia.org/wiki/piecewise_linear_-function, 2018.

[57] C. Phillips X. Zhang and X. Chen. An overlay mapping model for achieving enhanced qos and resilience performance. In *Ultra Modern Telecommun. and Control Sys. and Workshops*, Budapest, Hungary, October 2011.

[58] X. Chen X. Zhang and C. Phillips. Achieving effective resilience for qos-aware application mapping. *Computer Networks*, 56(14):3179 – 3191, 2012.

[59] J. Y. Yen. Finding the k shortest loopless paths in a network. *Manag. Science*, 17(11):712–716, July 1971.

[60] L. Yu and H. Shen. Bandwidth guarantee under demand uncertainty in multi-tenant clouds. In *Proc. IEEE ICDCS*, Madrid, Spain, June 2014.

[61] M. Yu et al. Rethinking virtual network embedding: Substrate support for path splitting and migration. *ACM SIGCOMM CCR*, 38(2), April 2008.

[62] Y. Yu et al. Rmap: An algorithm of virtual network resilience mapping. In *International Conference on Netw. and Mobile Comput.*, Wuhan, China, 2011.

[63] Y. Zhu and M. Ammar. Algorithms for assigning substrate network resources to virtual network components. In *Proc. IEEE INFOCOM*, Barcelona, Spain, April 2006.