UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

LEONARDO F.S. MOURA

Branch & Price for the Virtual Network Embedding Problem

Thesis presented in partial fulfillment of the requirements for the degree of Master of Computer Science

Advisor: Prof. Dr. Luciana Buriol

Porto Alegre
2015

CIP – CATALOGING-IN-PUBLICATION

Moura, Leonardo F.S.

Branch & Price for the Virtual Network Embedding Problem / Leonardo F.S. Moura. – Porto Alegre: PPGC da UFRGS, 2015.

69 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2015. Advisor: Luciana Buriol.

1. Branch & Price. 2. Virtual Network Embedding. 3. Network Virtualization. 4. Exact Algorithms. I. Buriol, Luciana. II. Título.

ACKNOWLEDGMENTS

First I would like to thank my advisor Luciana Buriol for accompanying me since the inception of my academic career. With her help I have grown so much as a person and as a professional. An equal share of thanks goes to my former advisors Marcus Ritt, Aline Villavicencio, and Van-Dat Cung. Their guidance was essential in my academic success.

Without my parents, Raquel and Vitor, I would not have made it this far. A sincere thanks to all my friends for all their support and for putting up with me during difficult times.

A special thanks is due to my research group colleagues — Leonardo, Paulo, Marcelo, Tadeu, André, Germano, Renato, Alex, Fernando, and Arton — for all their help, either by discussing technical, theoretical or personal issues. Many thanks to all the organizers of ELAVIO 2014, there I learned a lot and made good friends for life.

This work was supported by CNPq (Brazilian National Research Council) and by Instituto de Informática da UFRGS.

ABSTRACT

Virtualization allows one or more virtual networks to share physical infrastructures. The Virtual Network Embedding problem (VNEP) is one of the main challenges in the virtualization of physical networks. This problem consists in mapping a virtual network into a physical network while respecting capacity constraints. This work shows that finding a feasible solution for this problem is NP-Hard. However, many instances can be solved up to optimality in practice by exploiting the problem structure. We present a Branch & Price algorithm applied to instances of different topologies and sizes. The experimental results suggest that the proposed algorithm is superior to the Integer Linear Programming model solved by CPLEX.

Keywords: Branch & Price. Virtual Network Embedding. Network Virtualization. Exact Algorithms.

Branch & Price para o Problema de Mapeamento de Redes Virtuais

RESUMO

Virtualização permite o compartilhamento de uma rede física entre uma ou mais redes virtuais. O Problema de Mapeamento de Redes Virtuais é um dos principais desafios na virtualização de redes. Esse problema consiste em mapear uma rede virtual em uma rede física, respeitando restrições de capacidade. O presente trabalho mostra que encontrar uma solução factível para esse problema é NP-Difícil. Mesmo assim, muitas instâncias podem ser pode ser resolvidas na prática através da exploração de sua estrutura. Nós apresentamos um algoritmo de Branch & Price aplicado a instâncias de diferentes topologias e tamanhos. Os experimentos realizados sugerem que o algoritmo proposto é superior ao modelo de programação linear resolvido com CPLEX.

Palavras-chave: Problema de Mapeamento de Redes Virtuais. Programação Linear Inteira. Geração de Colunas. Algoritmos Exatos.

LIST OF FIGURES

3.1 An input instance for the VNEP. 27
3.2 Optimal solution for instance of Figure 3.1. 27
3.3 Virtual Graph resulted from transformation ϕ. 30
3.4 Substrate Graph resulted from transformation ϕ. 31

4.1 Auxiliary graph for the input instance of Figure 3.1. 35
4.2 Subproblem graph 36

6.1 Topology examples 43
6.2 Average time in seconds. 47
6.3 Average cost of feasible solutions. 48
6.4 Number of instances solved. 49
6.5 Time to obtain the first integer solution. 51

LIST OF TABLES

2.1 Overview of selected related works 16
2.2 Benchmark characteristics of selected related works 17

6.1 Node Selection Results 44
6.2 Results for sparse random instances. 46
6.3 Results for dense random instances. 50
6.4 Results for hierarchical instances. 52
6.5 Results for transit-stub instances. 53
6.6 Column Generation Components Running Time - Sparse Instances 54
6.7 Column Generation Components Running Time - Dense Instances 55
6.8 Column Generation Components Running Time - Hierarchical Instances 56
6.9 Column Generation Components Running Time - Transit-stub Instances 57

A.1 Sparse Random Instance - Low demands 62
A.2 Sparse Random Instances - High Demands 63
A.3 Dense Random Instances - Low Demands 64
A.4 Dense Random Instances - High Demands 65
A.5 Hierarchical Instances - Low Demands 66
A.6 Hierarchical Instances - High Demands 67
A.7 Transit-Stub Instances - Low Demands 68
A.8 Transit-Stub Instances - High Demands 69

LIST OF ALGORITHMS

1 Column Generation Algorithm for VNEP. 33
2 Modified Dijkstra's Algorithm 37
3 Branch & Price Algorithm 40

LIST OF ABBREVIATIONS AND ACRONYMS

VNEP	Virtual Network Embedding Problem
UFP	Unsplittable Flow Problem
ILP	Integer Linear Program
B&P	Branch & Price
B&B	Branch & Bound
RMP	Restricted Master Problem
CG	Column Generation

CONTENTS

LIST OF FIGURES 6
LIST OF TABLES 7
LIST OF ALGORITHMS 8
1 INTRODUCTION 11
2 RELATED WORK 14
2.1 Overview 14
2.2 Related Problems 15
2.3 Virtual Network Embedding Problem 16
2.4 Branch & Price 23
3 VIRTUAL NETWORK EMBEDDING PROBLEM 25
3.1 Problem Definition 26
3.2 Models 26
3.3 Complexity 29
4 COLUMN GENERATION 33
4.1 Obtaining an initial set of columns 34
4.2 The Pricing Problem 34
4.3 Proposed Modified Dijkstra Algorithm 36
5 BRANCH & PRICE 39
5.1 Branching 39
5.1.1 Variable Selection 39
5.1.2 Node mapping branching 40
5.1.3 Path branching 40
5.2 Node Selection 41
5.3 Heuristic Solution 41
6 EXPERIMENTAL RESULTS 42
6.1 Datasets 42
6.2 Parameter Testing 44
6.3 Branch & Price Evaluation 45
6.4 Running time of column generation components 52
7 CONCLUDING REMARKS 58
REFERENCES 59
APPENDIX A FULL RESULTS 62

1 INTRODUCTION

The current Internet architecture supports a large array of applications and technologies. However due to its decentralized and heterogeneous nature, it has become difficult to address new requirements. Any architectural change in the Internet has to be agreed upon by Internet Service Providers, and hardware and software vendors. This problem has been called the "ossification of the Internet."

Network Virtualization has been seen as the solution for the problem of ossification of the Internet, a way to facilitate the evolution of Internet protocols (ANDERSON et al., 2005). By creating a new layer of abstraction over the physical networks, multiple networks can simultaneously use the same physical structure in a transparent way. In this way, new protocols can be tested on heterogeneous experimental architectures (ANDERSON; REITER, 2006).

Likewise, Service Providers can take advantage of Network Virtualization to offer customized services, like customized protocols or co-location from expanded network presence, by leasing resources from infrastructures providers (FEAMSTER; GAO; REXFORD, 2007).

Virtualization is already being used in practice for supporting experimental facilities, such as GENI (ANDERSON; REITER, 2006) and the Planet Lab architecture (CHUN et al., 2003). It has being seen as an enabler of Cloud Computing and Software Defined Networks (SDN) (GUERZONI et al., 2014).

Several technical problems arise in the implementation of virtual network environments. Among the main operational challenges are resource discovery, resource allocation and resource configuration (CHOWDHURY; BOUTABA, 2010). The current work focusses on resource allocation in network virtualization.

The Virtual Network Embedding Problem (VNEP) — also known as Network Testbed Mapping (RICCI; ALFELD; LEPREAU, 2003), Virtual Network Assignment (ZHU; AMMAR, 2006), and Virtual Network Mapping (BELBEKKOUCHE; HASAN; KARMOUCH, 2012) — is central to achieve virtualization. It consists in allocating physical resources to virtual networks. Virtual nodes are mapped into physical nodes; and virtual links into physical paths that link the physical nodes that host their endpoints. Those physical resources have processing limitations that need to be taken into account. Additionally, applications usually need to select the best mapping according to some metric, such as cost, revenue, or power usage.

As network virtualization is not yet a mature field, problems are still being defined and classified. Different classifications and more information about different constraints of virtualization problems are found in (FISCHER et al., 2011; CHOWDHURY; BOUTABA, 2010; FISCHER et al., 2013). There are several variations of the problem such as mapping multiple networks simultaneously (HOUIDI et al., 2011), instead of a single one at a time (CHOWDHURY; BOUTABA, 2010). Some works map virtual networks in an online fashion (requests arrive dynamically, lasting for a period of time), using and releasing physical resources dynamically (YU et al., 2008). Physical nodes and links can be restricted to host a limited number of virtual nodes and links or certain virtual links can have a location restriction. Some applications have security restrictions, that limit the subset of physical nodes and links that can be used for mapping (BAYS et al., 2012). Additional constraints can also be present such as delay (INFüHR; RAIDL, 2011), efficient use of energy (BOTERO et al., 2012), and redundancy (SHAMSI; BROCKMEYER, 2008). This work captures the core of those constraints into a VNEP definition that is both hard and generic.

Due to the difficulty of solving the VNEP, few exact algorithms are proposed in the literature. For most practical purposes, heuristic algorithms can provide good suboptimal solutions for the problem. Still, exact algorithms serve as a comparison or baseline to evaluate heuristic algorithms. Moreover, they can be practical for specific cases, such as for small virtual virtual networks or physical networks with a large amount of resources.

Only recently exact algorithms for the VNEP have been explored. This work proposes a Branch & Price algorithm for the VNEP that is able to solve optimally larger instances than those that were previously solved through other exact algorithms presented in the literature. This algorithm is compared experimentally with randomly generated instances with four different types of topologies, ranging from twenty to two hundred nodes.

This work contributions are the following:

- a concise VNEP version is presented. That problem is both generic and hard to solve, preserving the main components of all VNEP definitions;
- the problem is further classified by providing a complexity proof;
- a new extensive model is presented for the Single-Path VNEP;
- this model is solved with a column generation algorithm;
- an efficient algorithm for solving the pricing problem of column generation algorithm is provided;
- the column generation algorithm is extended to a Branch & Price algorithm to provide optimal integer solutions for the problem;

An overview of related works with a focus on exact solutions for the VNEP is presented in Chapter 2. Chapter 3 describes the Virtual Network Embedding problem, presents available models and an analysis of its complexity. Chapter 4 presents the Column Generation algorithm. The Branch & Price algorithm is detailed in Chapter 5. Experimental results are presented and analysed in Chapter 6. Finally, this work is concluded in Chapter 7 with a summary of this study and future research directions.

2 RELATED WORK

Network virtualization is still a recent field, problems are still being defined and classified. Different classifications and more information about different constraints of virtualization problems are found in (FISCHER et al., 2011; CHOWDHURY; BOUTABA, 2010; FISCHER et al., 2013).

This chapter gives an overview of related works with a focus on exact algorithms. It is divided in four sections. Section 2.1 presents a historical background and overview of some of the most important works on the VNEP. Related problems are presented in Section 2.2. Section 2.3 summarizes related works, mainly those that contain exact algorithms for the VNEP. Section 2.4 refers to important publications related to Branch & Price and Column Generation.

2.1 Overview

Virtual Network Embedding is the main problem involved in network virtualization. Due to the difficulty of finding optimal solutions for large instances of the VNEP, early works, such as (ZHU; AMMAR, 2006), solve VNEP using heuristic algorithms. Others relax constraints, such as node and link capacity constraints (FAN; AMMAR, 2006). Many works, starting with (YU et al., 2008), propose the splitting of virtual links into multiple paths in the physical network. Although those relaxations can be more easily solved, they cannot be always applied in practice. Moreover, heuristic algorithms are not guaranteed to find feasible solutions.

The first exact algorithm proposed for VNE was a backtrack algorithm presented in (LISCHKA; KARL, 2009). Another algorithm based on an integer linear model was proposed in (CHOWDHURY; RAHMAN; BOUTABA, 2012). However, both works use heuristic versions of their algorithms in their evaluation, as exact algorithms are deemed too expensive to be used in practice.

The first article to propose and test an exact algorithm based on an Integer Linear Program for VNE was (HOUIDI et al., 2011). Since then, more authors are proposing exact algorithms for the problem

A diverse set of constraints is modeled with Integer models, such as optical virtual networks (PAGES et al., 2012), security constraints (BAYS et al., 2012), virtual networks sharing physical links availability (TRINH; ESAKI; ASWAKUL, 2011), and optimizing energy usage (BOTERO et al., 2012).

Two column generation models were presented in the literature for the VNEP. In (JARRAY; KARMOUCH, 2012), column generation is used to deal with multiple requests and in (HU;

WANG; CAO, 2013), individual requests are mapped with a path-based formulation.

2.2 Related Problems

Virtual private networks (VPNs) are virtual “sub-networks” in larger physical networks (GUPTA et al., 2001). A group of nodes of the physical networks reserves a part of the bandwidth capacity of physical links in order to communicate with each other. The VPN Design problem consists in providing a routing between the nodes that support their communication. As the resources are limited, the reservation of links has a cost. The total cost of the final design ought to be minimized. Moreover, some applications desire that the final sub-network has a simple structure like a tree to ease routing and communication.

The VNEP is considerably harder than the VPN Design Problem (ZHU; AMMAR, 2006). Virtual nodes are not fixed in physical nodes, increasing the solution space. Moreover, in the VNEP, links cannot generally share bandwidth, hence demanding more coordination between the routing of virtual links and node mapping.

Some works, such as (CHOWDHURY; RAHMAN; BOUTABA, 2012), reduce VNEP to another known problem, the minimum-cost multicommodity flow problem (MMFP). This problem consists in finding a flow that ships multiple commodities through a single network without violating the capacity constraints of the edges and has a minimum cost (GOLDBERG et al., 1998). Given an undirected graph $G = (V, E)$ with capacity c_e for each edge $e \in E$, and a set of terminal pairs T, and a demand ρ_i for each terminal pair i, the objective of this problem is to find a flow through G that fulfils the demand without violating the constraints. For networks that allow path splitting, each mapping of virtual nodes in an instance of the VNEP yields an instance of the multicommodity flow problem.

Since this work focus on the Single-Path VNEP, another important problem to consider is the Unsplittable Flow Problem (UFP). It consists in finding a set of valid paths P such that the demand ρ_i of each terminal pair flows through the paths in P without violating the capacities c_e. VNEP is related to a problem identified by Kleinberg (1996): the problem of finding a subset of terminals that maximizes the total demand fulfilled. Each possible combination of a possible mapping of the VNEP gives rise to an instance of the Unsplittable Flow Problem.

2.3 Virtual Network Embedding Problem

This section summarizes some of the most relevant papers related to the exact solution of the Virtual Network Embedding Problem. Table 2.1 summarizes a selected list of related works. Columns **Exact** and **Heuristic** show whether the work contains an exact or heuristic algorithm. Column **SP**, if the work presents a single-path version of the VNEP and, **MP**, a multiple-path version. Works that support the embedding of multiple requests simultaneously are marked in Column **MR** . Finally, column **Objective** shows what feature is optimized in the presented algorithm.

Table 2.1: Overview of selected related works

Reference	Exact	Heuristic	SP	MP	MR	Objective
Yu et al. (2008)		x		x		Revenue and Cost
Lischka and Karl (2009)	x		x	x		Cost
Houidi et al. (2011)	x			x	x	Cost
Trinh, Esaki and Aswakul (2011)	x		x			Cost
Inführ and Raidl (2011)	x			x		Cost
Chowdhury, Rahman and Boutaba (2012)	x	x		x		Cost and load balancing
Bays et al. (2012)	x		x			Resources Utilization
Pages et al. (2012)	x	x		x		Requests served
Botero et al. (2012)	x			x		Inactive resources
Jarray and Karmouch (2012)	x		x			Revenue
Alkmim, Batista and Fonseca (2013)	x	x	x			Resources Utilization
Hu, Wang and Cao (2013)	x			x		Cost
Guerzoni et al. (2014)	x		x		x	Total revenue

Source: from author (2015).

Table 2.2 summarizes the main characteristics of benchmark instances used in related works. Due to the heterogeneity of units of measure presented in the literature, values are presented in absolute units. Columns $|S|$ and $|V|$ show the number of nodes of the physical and virtual graphs, respectively. Column V_{cap} shows the range or average of the capacity of nodes, and Column V_{dem} shows the range or average of virtual node demands. Column E_{cap} shows the bandwidth capacities of physical nodes while Column E_{dem}, the bandwidth demands of virtual links.

The contribution of Yu et al. (2008) is twofold: It proposes a virtual network embedding algorithm that allows one virtual edge to be mapped to multiple substrate paths (*path splitting*) and migration of these paths to optimize the physical substrate usage. Additionally it proposes specialized virtual network embedding algorithms for special classes of virtual network requests. The objective function treated is multi-objective, a maximization of the revenue and

Table 2.2: Benchmark characteristics of selected related works

Reference	$\|S\|$	$\|V\|$	V_{cap}	V_{dem}	E_{cap}	E_{dem}
Lischka and Karl (2009)	100	$[10, 40]$	$[0, 100]$	$[0, 90]$	$[0, 100]$	$[0, 90]$
Houidi et al. (2011)	50	$[2, 10]$	$[50, 100]$	$[0, 20]$	$[50, 100]$	$[0, 50]$
Inführ and Raidl (2011)	$[20, 100]$	$[2, 20]$	$[25, 100]$	$[1, 5]$	$[25, 100]$	$[1, 7]$
Chowdhury, Rahman and Boutaba (2012)	50	$[2, 10]$	$[50, 100]$	$[0, 20]$	$[50, 100]$	$[0, 50]$
Bays et al. (2012)	$[50, 100]$	$[17, 66]$	100	$[1, 50]$	$[1, 10]$	$[0.1, 5]$
Alkmim, Batista and Fonseca (2011)	$[5, 25]$	$[5, 10]$	256	128	$[1, 10]$	$[0.1, 0.4]$
Hu, Wang and Cao (2013)	$[10, 50]$	$[2, 10]$	$[1, 50]$	$[1, 20]$	$[1, 50]$	$[1, 20]$
Guerzoni et al. (2014)	$[50, 250]$	$[2, 5]$	$[50, 100]$	$[1, 10]$	$[50, 100]$	$[1, 25]$

Source: from author (2015).

a minimization of the total cost.

In the proposed approach, the virtual network requests are put in priority queue. Each virtual network request has a lifetime that can range from a few minutes to several days. Periodically, requests in the queue are processed in order of decreasing revenue. If the heuristic algorithm fails to find a valid mapping, the network request is rejected. The mapping algorithm works in two sequential phases: The node mapping and the link mapping.

Two algorithms are used for node mapping: a specialized version for hub-and-spoke topologies and a general algorithm. The general algorithm maps the nodes greedily using information of the substrate node resources and the available bandwidth of the adjacent physical links. Hub-and-spoke are hierarchical topologies, commonly found in centralized databases/servers. The specialized algorithm maps greedily the hubs and uses a shortest-path algorithm to map the spoke nodes to the closest substrate nodes.

After the virtual nodes are mapped, virtual links need to be mapped to physical paths between the endpoints of those virtual links. If path splitting is not allowed, those paths are calculated using k-shortest paths algorithm iteratively. For networks that allow path splitting, a multicommodity flow algorithm is used. If after the flow is calculated, there is a substrate edge that is congested, i.e. an edge that holds more flow than its capacity, the endpoint of this edge is mapped to another substrate node and another multicommodity flow algorithm is run. The process is repeated until either a valid mapping is found or a limit number of iterations is reached.

Results show that the use of specialized algorithms of node mapping for some topologies has a clear benefit in the cost and revenue. Path splitting is shown to better use the physical resources, specially when resources are more limited in relation to the demands.

Lischka and Karl (2009) presents a one-stage method for the VNEP based on *vnmFlib*, a backtrack algorithm for subgraph isomorphism. Nodes and links are mapped simultaneously, allowing unfeasible node mappings to be detected early. Moreover, a short NP-Completeness proof of the VNEP is presented.

The proposed algorithm works as follows: Suppose one part of the virtual network is already mapped, the next virtual node u to be mapped is selected. All substrate nodes that do not host virtual nodes in the current solution are candidates to host v. The first node in this list is selected. Then, all virtual links that join u and another nodes already mapped are mapped. If no path is found, the algorithm backtracks to the last valid solution and maps u to another substrate node. When all nodes and links of the virtual network are mapped or there are no more candidates, the algorithm stops.

As the objective of this work is to find valid mappings of multiple virtual networks in a window of time, some steps are taken to limit the search space. The first is to limit the maximum size of a path in the substrate graph to a parameter ϵ, what can improve the running time of the algorithm, but can fail to find a valid solution in instances where a valid mapping exist. Likewise, the number of mappings (nodes in the backtrack tree) is limited by a parameter ω.

Experiments show that the proposed algorithm results in better mappings and is faster than the traditional two stage approach taken in (YU et al., 2008) for requests with large demands.

Houidi et al. (2011) treats two problems related to virtualization: Splitting virtual network requests across multiple physical infrastructure providers and embedding those requests on the available physical resources. An exact algorithm based on an Integer Linear Program is presented for the Virtual Network Embedding problem. This program is solved with a branch and bound algorithm. The presented model supports the mapping of multiple virtual network requests simultaneously. This work is the first to use an exact algorithm to solve VNEP as previous works deemed the problem intractable.

In its evaluation section, it is shown that exact embedding is useful for small scale virtual networks. It is also shown that the proposed algorithm makes a better use of physical resources, accepting more virtual network requests and resulting in lower embedding costs overall.

Trinh, Esaki and Aswakul (2011) presents a nonlinear model for the VNEP. In certain applications — such as e-mail, fax, SMS —, the allocated resources do not need to be exclusive, but can be offered within a certain guaranteed level of quality. Substrate link resources are divided between the allocated virtual networks. The model is solved with MATLAB. The authors claim that the sharing of resources brings forth an economy of roughly 26%. However, the experimental results are limited to only one physical substrate network.

Inführ and Raidl (2011) introduces new constraints of delay, routing and location. Each physical link used by a virtual link adds delay to communication. Virtual links have a limit on the total delay of their route. Physical nodes have a capacity on the amount of bandwidth they can route. Virtual nodes have a limited set of physical nodes they can be mapped to. Those constraints are modeled with an ILP and solved with CPLEX. Multigraphs are used to model physical networks because substrate nodes can be connected with an heterogeneous set of physical links.

That work presents a hybrid algorithm for generating realistic instances for the VNEP. Substrate networks are generated by modifying real Internet topologies available online. The tool nem-0.9.6 was used to randomly remove nodes in order to reduce those topologies to proper sizes while maintaining their original properties. Capacities of nodes and links are assigned according node connectivity: heavily connected nodes have more capacity, likewise for their adjacent links. Virtual network requests were constructed from prototypical graphs called slices. Each slice has special requirements of delay and bandwidth based on their characteristics. Four kinds of slices were used: Web Slices, Stream Slices, P2P Slices, and VoIP Slices. Those slices are assembled randomly to generate graphs with 50% to 90% of the size of the substrate graph. An initial slice is solved with CPLEX, if it is able to solve the generated instance, more slices are aggregated to the network request. This process is repeated until CPLEX fails to find an integer solution for the instance in less than 300 seconds in five tries.

Instances are solved optimally in more than 74% of instances in less than an hour. According to the authors, the main factor predicting the hardness of instances was the chosen topology and not the size of the instance.

Chowdhury, Rahman and Boutaba (2012) presents a different approach for the VNEP. Instead of mapping nodes and links separately, both mappings are done simultaneously by transforming the problem into a variation of the multicommodity problem. On top of the substrate graph, one meta node is added to each virtual node. Each of the meta nodes is linked with substrate nodes with enough capacity to hold them. Each virtual link is then converted in a commodity with terminals in the meta nodes of its endpoints. To assure that each virtual node is mapped to a single substrate node, all virtual links adjacent to a virtual node must pass through a single meta edge.

This transformed problem is modeled as a Mixed Integer Programming (MIP) model. As solving MIP models is computationally expensive, two heuristic algorithms are presented. Those algorithms use the decision variables values obtained from the linear relaxation of the MIP model to produce a rounded solution. Each virtual node v is associated with a set of binary decision variables x relative to each substrate node s. The value of $x_{v,s}$ is one if and only if the virtual node v is mapped to s. Two rounding schemes are presented in order to obtain integer solutions from solutions of the linear relaxation of the MIP model: In the deterministic version, for each virtual node v, the variable with the largest value $x_{v,s}$ is set to one, and all others to zero. In the randomized version, the variables are selected randomly with probability corresponding to their values. After virtual nodes are mapped, virtual links are mapped with the multicommodity flow algorithm.

Results show that the proposed methods result in a better acceptance ratio and larger revenue than greedy algorithms. In terms of execution times, the proposed algorithms perform worse than the greedy algorithm due to the use of the multicommodity flow algorithm two consecutive times. However, the overhead in computation can lead to more revenue and better resource utilization.

Bays et al. (2012) proposes an integer linear program to solve optimally the VNEP with security requirements, such as data isolation and encryption. The presented model encompasses location constraints, and three levels of security. In the first level, data is encrypted with end-to-end cryptography. The second level provides header protection in order to secure routing information. A third level security protects information by not allowing physical node sharing between specific virtual networks. In most of the experiments, the optimal solution is found in less than three hours, but some instances take more than 24 hours to solve. Reported results show that after 20 minutes the optimality gap was 10.72%. Authors suggest that suboptimal solutions could be used by infrastructure providers by using a gap or time threshold.

Pages et al. (2012) proposes both an integer linear programming model and a metaheuristic for the Virtual Optical Network Allocation (VONA) problem. Optical networks differ from normal networks in the sense that each physical edge has a limited number of wavelengths, and the substrate nodes cannot change the wavelength of the virtual link passing through it, so each virtual link has to be mapped to a path in the physical network were all edges use the same wavelength. The objective is to find any valid solution, with no restrictions on its quality.

Two variations of the VONA problem are presented: The transparent VONA, that requires the exact set of wavelengths for every virtual link, and the opaque VONA, in which there is no need to allocated the same wavelengths for each virtual link.

The article presents two ILP models, one for the transparent case and one for the opaque VONA. Since solving an ILP model is computationally expensive, a heuristic algorithm is presented based on the metaheuristic GRASP.

The algorithms presented are tested in a real network topology with 16 nodes. The virtual optical network requests are randomly generated. The running times of the GRASP algorithm are better than the exact algorithm with at most one more blocked request. The behavior of the algorithms in larger graphs is not tested.

Botero et al. (2012) proposes an algorithm to reduce energy expenditure. Low average link utilization in backbones of large ISPs yield unnecessary energy consumption. In order to achieve an efficient energy consumption, the number of active nodes and links is minimized. By doing so, parts of the physical networks can be switched off in periods of low traffic demands.

Results show that under low traffic loads, the proposed model can save up to 35% of nodes and 25% of links used in the physical network with no serious decrease in the acceptance ratio.

Jarray and Karmouch (2012) develops a column generation algorithm for the single-path VNEP with multiple requests. The objective function is the maximization of the total revenue. This paper is an extension of a previous ILP Formulation called *Join Node-Link Embedding*. The authors propose a column generation model based on Independent Embedding Configurations (IEC) to improve the scalability of the previous formulation. Each IEC is a mapping of one virtual network request. The master problem selects at most n IECs to be served, while the pricing problem generates IECs. The model is solved with a Branch & Price algorithm that uses the commercial solver CPLEX in the relaxed master problem as well as in the integer pricing problem. The set of all paths has to be generated prior to the execution of the algorithm, this can be an issue if larger physical substrate graphs are used. No details were given as how columns are selected and branched in the Branch & Price algorithm.

The proposed algorithm is compared with two greedy algorithms, the tests were run on a physical substrate graph extracted from the US metro backbone with 30 edges. The virtual networks were generated with a randomly selected size ranging from 2 to 20. The results evaluated revenue, amount of requests blocked, and resource utilization of the served requests. The proposed column generation performed the best in those three criteria for the selected set of instances.

That work differs from the current work in that it uses column generation to address the mapping of multiple requests and it uses CPLEX to solve the pricing problem.

Alkmim, Batista and Fonseca (2013) presented a formulation that takes into account the transportation and installation of software images necessary to the virtual routers. To solve this issue, the problem is broken in two parts: The VNEP and the routing of the images through the network. Six algorithms are proposed to solve the former, one exact and five heuristic algorithms. The *optimal algorithm* solves two ILP models optimally by applying Branch & Cut through CPLEX. The *root algorithm* uses the linear relaxation of the model. In the iterative rounding scheme the relaxed decision variable with the largest value is set to one, and all others to zero. In the random scheme, values of the relaxed decision variables are used as the probability of that variable being rounded to one. The heuristic algorithms can be either iterative or not. Iterative algorithms fix fractional variable one by one, solving a new relaxed problem each time a variable is fixed. Non-iterative algorithms fix variables all in one step.

Hu, Wang and Cao (2013) formulates a path-based integer linear programming model for the VNE Problem and solves it using column generation. Each possible path is represented by a binary variable in the model, since the number of paths in an arbitrary graph grows exponentially, there is a large number of variables. To circumvent this problem, the column generation algorithm is used. In this technique, the linear relaxation of the problem with a few initial columns is solved. New columns are generated the pricing problem. When no new columns are generated, the relaxation is optimal.

The pricing problem is the generation of new paths and the master problem is the selection of paths. The relaxed version of the master problem is solved and Branch & Bound is used to obtain an integer solution.

Although the authors claim that their algorithm has a better running time than the one presented in (CHOWDHURY; RAHMAN; BOUTABA, 2009), no data is presented to support it. Only a comparison of the solution quality of heuristic variations of the presented algorithm is presented.

Guerzoni et al. (2014) presents a MIP formulation called *Edge-wise Node mapping* (EdWiN) to optimally map several networks simultaneously. This formulation is more flexible and is capable of embedding more networks simultaneously than previous ILP Models. It is flexible because it allows network administrators to embed VN requests to be mapped according to predefined policies. It is more efficient in mapping multiple requests because it allows partial embeddings. In the model presented in (HOUIDI et al., 2011), a number of requests has to be mapped completely, whereas in EdWiN, only part of the virtual networks can be embedded. The model embed a predefined number of requests simultaneously. Those requests are broken into subgraphs with 2 nodes. Nodes can be limited to specific physical locations or specific “colors” of physical nodes. The colors represent characteristics of physical nodes, such as which Operating Systems can be installed on the node. The presented model is compared with the exact method presented in (HOUIDI et al., 2011) and the greedy method presented in (YU et al., 2008). Although both algorithms support multiple paths, a single-path version of those algorithm is used for comparison. The proposed algorithm has shown a better resource utilisation, a larger embedding rate and better running time performance. The improved performance over previous formulation is explained by the model having fewer variables.

2.4 Branch & Price

Branch & Price (B&P) is type of Branch & Bound (B&B) algorithm where a column generation algorithm is used at each node. Column generation (CG) is a mathematical programming technique proposed by Dantzig and Wolfe (DANTZIG; WOLFE, 1960) which solves a relaxation of a model. CG is used to solve models with a large number of variables in comparison to the number of constraints. These models are generally impossible to solve with the Simplex Algorithm due to memory constraints. However, since most variables in these models have a value zero, they can be treated implicitly.

One of the advantages of modeling a problem with a large number of variables is that these models often have relaxations that yield better lower bounds than that of compact models (BARNHART et al., 1998).

These techniques have a number of implementations issues that have a direct impact on the performance of the algorithm, starting by the way the problem is modeled in its extensive form, the way in which variables are decomposed, which constraints are delegated to the pricing problem, how generated columns are managed, how to select variables to be branched and how to branch on variables, wow branched variables affect pricing, which cuts can be added to improve relaxations, whether the pricing problem should be always solved to optimality and how to deal with the tailing off effect, and how to select nodes in the BP tree. These, among other issues, were widely studied in the literature in the last fifty years.

Barnhart et al. (1998) presents a general methodology for B&P. Two example applications are given to illustrate general concepts. Computational issues are considered such as that of obtaining an initial solution through the introduction of artificial variables. These variables are not removed during BP execution because an initial solution is necessary at each node of the BP tree. The pricing problem does not need to be solved to optimality and approximation algorithms can be used. Additionally, more than one column can be generated at each iteration of CG. Trade-offs are discussed such as that of having expensive, tighter lower bounds and smaller search trees or loose lower bounds with larger trees. Algorithms for solving the master problem and the generation of rows is also discussed.

Lubbecke and Desrosiers (2005) surveys recent contributions and explores the dual point of view of CG. That paper begins by showing a list of applications solved through CG and presents Dantzig-Wolfe decomposition. Algorithms for solving the restricted master problem are discussed, as well as alternative pricing rules. The tailing off effect is also discussed: in Simple-based CG, near optimal solutions are obtained quickly, but convergence to the optimal is slow. Authors recommend dual and primal stabilization strategies to attenuate such effect. As for branching decisions, this work suggest the introduction of meaningful cuts, i.e., cuts based on the compact model.

Barnhart, Hane and Vance (2000) uses a Branch & Price & Cut to solve a related problem, the origin-destination integer multicommodity flow problem. In this problem, similarly to the UFP, only one path can be used for each commodity. In this paper, lifted cover inequalities are used to strengthen linear relaxations of the linear program and to overcome symmetry problems. Additionally, a branching rule is introduced that do not destroy the structure of the problem. This branching rule is inspired on the compact model: instead of fixing paths, arcs are forbidden to server certain commodities. Such rule is easily enforced in the pricing problem by increasing the cost of forbidden arcs to a large value.

3 VIRTUAL NETWORK EMBEDDING PROBLEM

As seem in Chapter 2, several variation of VNEP are presented in the literature. This work captures the essential components of the problem that make the problem hard to solve.

Virtual nodes consume resources from physical nodes. Generally CPU processing capacity and memory space are considered. To simplify the modeling, just CPU processing is considered in this work. Due to load balancing considerations (HOUIDI et al., 2011), each virtual node has to be mapped to a different physical node.

Some applications have geographical restrictions due to the demand of connecting different geographical locations or security reasons (BAYS et al., 2012). Hence some formulations allow virtual nodes to be restricted to a subset of physical nodes. As the formulation adopted in this work aims to be generic, no location constraints are considered.

Virtual links have to be mapped into physical paths between physical nodes that host their endpoints. Multiple virtual links can share the same physical link. A part of the limited bandwidth capacity of physical edges is reserved for each virtual link.

Finding a feasible link mapping is NP-Hard even if node mapping is fixed. In order to render the problem easier to solve, Yu et al. (2008) proposes a relaxation of the constraint that each virtual link has to be mapped into a single physical path. By allowing virtual links to be mapped to multiple paths in the physical network, link mapping can be done in polynomial time. Several works in the literature have taken this approach. However, some applications, such as teleconferencing, do not allow the splitting of data (BARNHART; HANE; VANCE, 2000) and current network technology do not generally support path splitting (GUERZONI et al., 2014). Therefore in this work path splitting is not allowed.

Other link mapping constraints found in the literature include routing capacities, delay limits, and limits on the sizes of paths (INFüHR; RAIDL, 2011).

All variations of VNEP aim for feasible mappings of virtual network nodes and links into physical resources. But different objective functions are considered such as minimizing the total cost of the mapping in networks where physical nodes and links have heterogeneous associated costs. Others try to maximize the total profit when virtual networks have different revenues. In this work, the objective function minimizes the amount of resources used by the obtained mapping. Since all virtual nodes have to be mapped, a solution is evaluated by the used amount of bandwidth capacity of physical links. Thus, for example, if a path with four links is used instead of one with two, the amount of bandwidth used is doubled.

Subsection 3.1 presents a formal definition of the problem. It is followed by a presentation of the Integer Linear Models for VNEP in Subsection 3.2. The chapter ends with an analysis of the complexity of the problem in Subsection 3.3.

3.1 Problem Definition

A VNEP instance has as input a virtual network and physical substrate network. The physical substrate is represented by an undirected graph $G^S = (V^S, E^S)$ with a CPU capacity of C_s for each physical node $s \in V^S$ and a bandwidth capacity B_e for each edge $e \in E^S$. The virtual network is represented by a undirected graph $G^V = (V^V, E^V)$ along with a demand C_v for each virtual node $v \in V^V$, and a bandwidth demand of B_k for each virtual link $k \in E^V$.

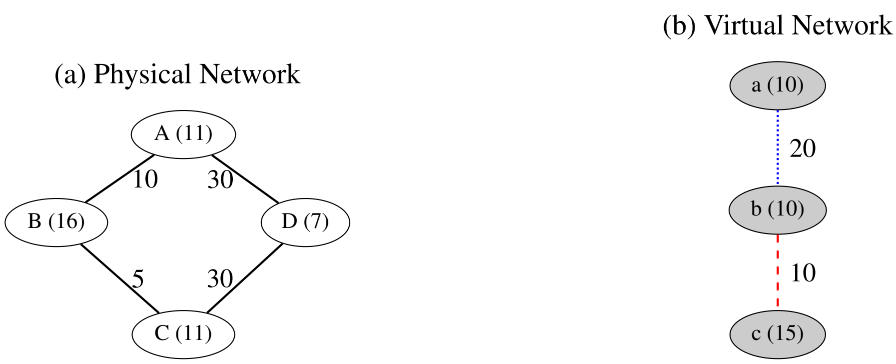
The objective of the proposed problem is to find a feasible mapping of the virtual nodes and links onto the physical network with minimal cost. A feasible mapping is a pair of functions (f_v, f_e): A mapping of nodes $f_v : V^V \to V^S$ and a mapping of links $f_{e=(w,u)} : E^V \to P$, where P is the set of paths in the substrate graph with endpoints $f_v(w)$ and $f_v(u)$. Each virtual node has to be mapped into a single substrate node with enough CPU capacity to host it. A substrate node can host at most one virtual node. Each virtual link $(w, u) \in E^V$ has to be mapped to a path in the physical graph between the nodes $f_v(w)$ and $f_v(u)$. An edge can host several virtual links, but the sum of their demands could not surpass the capacity of the edge. The cost of a mapping is the amount of bandwidth used in the physical network by the mapping.

Figure 3.1 presents an instance of the problem composed of a physical network with four nodes and a virtual network with three nodes. Edges and nodes are labelled with their capacities or demands. The optimal mapping is shown in Figure 3.2. The optimal solution is to map node a to C, b to A, and c to B; the virtual link (a, b) is mapped to $C - B - A$, and the virtual link (b, c) is mapped to $A - B$. The cost of this solution is 50.

3.2 Models

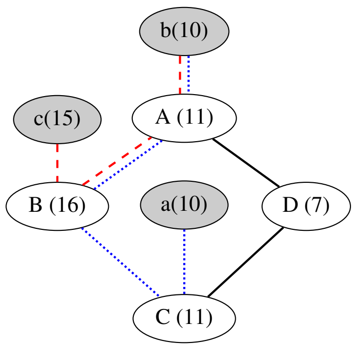
Most exact algorithms in the literature for the VNEP are based on ILP models. The following ILP model is based on (ALKMIM; BATISTA; FONSECA, 2013). It is adapted to the set of constraints presented in this work: let the decision variables $x_{v,s} = 1$ iff the substrate node s hosts the virtual node v. And let $y_{v,w,s,j} = 1$ iff the physical link (s, j) hosts the virtual link (v, w).

Figure 3.1: An input instance for the VNEP.



Source: from author (2015).

Figure 3.2: Optimal solution for instance of Figure 3.1.



Source: from author (2015).

$$\min \sum_{(s,j)\in E^S} \sum_{(v,w)\in E^V} B_{v,w} y_{v,w,s,j} \tag{3.1}$$

$$s.t. \sum_{v\in V^V} C_v x_{v,s} \leq C_s \qquad \forall s \in V^S \tag{3.2}$$

$$\sum_{s\in V^S} x_{v,s} = 1 \qquad \forall v \in V^V \tag{3.3}$$

$$\sum_{v\in V^V} x_{v,s} \leq 1 \qquad \forall s \in V^S \tag{3.4}$$

$$\sum_{j\in V^S} y_{v,w,s,j} - \sum_{j\in V^S} y_{v,w,j,s} = x_{v,s} - x_{w,s} \qquad \forall (v,w) \in E^V, s \in V^S \tag{3.5}$$

$$\sum_{(v,w)\in E^V} B_{v,w} y_{v,w,s,j} \leq B_{s,j} \qquad \forall (s,j) \in E^S \tag{3.6}$$

$$x_{v,s} \in \{0,1\} \qquad \forall v \in V^V, s \in V^S \tag{3.7}$$

$$y_{k,l,m,n} \in \{0,1\} \qquad \forall (k,l) \in E^V, (m,n) \in E^S \tag{3.8}$$

The objective function (3.1) minimizes the amount of bandwidth used by the virtual network. Constraints (3.2) ensure that the capacities of substrate nodes are not surpassed. Constraints (3.3) and (3.4) enforce, respectively, that every virtual node is mapped to a different substrate node and every substrate node hosts at most one virtual node. Constraints (3.5) are path constraints, they ensure that every virtual link is mapped to a path in the substrate graph. Finally, Equations (3.6) guarantee that the bandwidth capacities of physical edges are not violated.

Let us call this model **Compact Model** in contrast with the large model that will be presented later in this section. As the VNEP is NP-Complete (a proof is given in Section 3.3), solving this ILP model is NP-Complete. Branch & Bound algorithms use linear relaxations to obtain bounds on optimal integer solutions. The first problem with using the linear relaxation of the Compact Model is that there is no efficient algorithm to obtain integer solutions from that relaxation (proof in Section 3.3). The second problem is that this relaxation provides a poor lower bound for the optimal solution, what has a negative impact on the performance of a Branch & Bound algorithm. It is always possible to obtain a solution with cost zero for the relaxation of this model.

Proposition 1. *There is always a solution with cost zero to the compact model.*

Proof. If every x_{vs} is set to $1/|V^S|$, all constraints are respected and the left hand side of Constraints (3.5) is zero. Therefore all variables y_{vwsj} can be set to zero. Hence, the cost of the optimal solution for the relaxed problem is always zero, resulting in a trivial lower bound. □

In order to improve the lower bound given by the linear relaxation of the compact model, VNEP can be modeled in terms of paths in the auxiliary graph. Two sets of decision variables are used. For each $v \in V^V$ and $s \in V^S$, the variable $x_{vs} \in \{0, 1\}$ is set to one iff the substrate node s hosts the virtual node v. Likewise, for each path p in the set of all paths P, the variable $z_p \in \{0, 1\}$ is set to one if the path is used in the VNEP. For each path p, the input data $\delta_{e,p}$ is one if the physical edge e is in the path p, and zero otherwise. For each virtual link $k = (v, w) \in V^V$, P^k is the set of all simple paths in the substrate graph whose endpoints have enough CPU capacity to host v and w, and whose links have enough capacity to host k. The set of Constraints (3.2) of the compact model can be omitted because only paths that attend this constraint are part of the model. The objective function is the minimization of the total bandwidth used by the virtual network. If a path p serves a virtual link k, the cost of this path c_p is defined as the number of physical edges in the path. The **Flow-based Model** for the single-path VNEP is presented below:

$$\min \sum_{k \in E^V} \sum_{p \in P^k} c_p B_k z_p \tag{3.9}$$

$$\sum_{s \in V^S} x_{v,s} = 1 \qquad \forall v \in V^V \tag{3.10}$$

$$\sum_{v \in V^V} x_{v,s} \leq 1 \qquad \forall s \in V^S \tag{3.11}$$

$$\sum_{k \in E^V} \sum_{p \in P^k} \delta_{e,p} B_k z_p \leq B_e \qquad \forall e \in E^S \tag{3.12}$$

$$\sum_{k \in E^V} \sum_{p \in P^k : (v,s) \in p} z_p \leq M x_{v,s} \qquad \forall v \in V^V, s \in V^S \tag{3.13}$$

$$\sum_{p \in P^k} z_p = 1 \qquad \forall k \in E^V \tag{3.14}$$

$$x_{v,s} \in \{0,1\} \qquad \forall v \in V^V, s \in V^S$$

$$z_p \in \{0,1\} \qquad \forall p \in P$$

Constraints (3.10) ensure that all virtual nodes are mapped. Each substrate node can host at most one virtual node (3.11). Constraints (3.14) state that all virtual links have to be mapped to a single path in the substrate graph. Constraints (3.12) ensure that the bandwidth constraints are not violated. Finally, let M be the greatest degree of the virtual network, Constraints (3.13) enforce that only one auxiliary edge is used for each virtual node.

Solving the problem using decomposition introduces a large number of new variables. However, the relaxation of the flow-based Model provides better lower bounds and, since it uses the structure of the problem directly, we can obtain more information about the problem with relaxed solutions.

3.3 Complexity

We show in this section that the VNEP problem is NP-Hard by a reduction from the Bin Packing Problem (BPP), which is a classical NP-Complete problem. The VNEP was previously shown to be NP-Hard by a reduction from the Unsplittable Flow Problem (YU et al., 2008), but by the reduction from the BPP we further show it does not exist an algorithm that is guaranteed to generate a feasible solution unless P=NP, and then VNEP cannot be approximated.

The Bin Packing Problem has as input a set N of items and a bin size of capacity B. Each item i has a weight $w_i \leq B$. The goal is to fit all items using the minimum number of bins, respecting their capacities. The decision version asks if it is possible to fit all items in at most k bins.

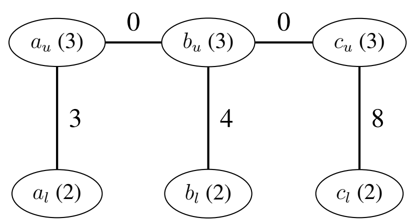
Any instance I of the BPP can be transformed into an instance of the VNEP through the procedure ϕ, which generates a virtual and physical graphs as described next.

Virtual Network: For each item $i \in N$, two nodes are created: one with demand three and another with demand two. Those nodes are linked with virtual links with bandwidth w_i. Moreover, nodes corresponding to items i and $i+1$ for every $i < |N|$ are linked to assure the connectivity of the network. Their demand is not important and are set to zero.

Physical Substrate Network: For each item $i \in N$ two nodes are created, one with capacity three and another with capacity two. Furthermore, $2k$ nodes are created with capacity one. For convenience, let us call nodes with capacity three the upper nodes, nodes with capacity two the lower nodes, and nodes with capacity one the middle nodes, in the physical and virtual networks. Each upper node is linked with k middle nodes, the rest of the k middle nodes is linked with all lower nodes. Each of the k middle nodes linked with the upper nodes are linked with one single middle node which is linked to the lower nodes. Capacities of all edges are set to B, the bin size.

An example of this transformation is show in Figures 3.3 and 3.4. The original BPP consists of three items of weights three, four, and eight, and $B = 8$. The value of k in the decision version is set to two. A solution is given by mapping the virtual links with weights three and four on physical edges on the left-central, and the other virtual link of weight eight into the edge on the right-central.

Figure 3.3: Virtual Graph resulted from transformation ϕ.

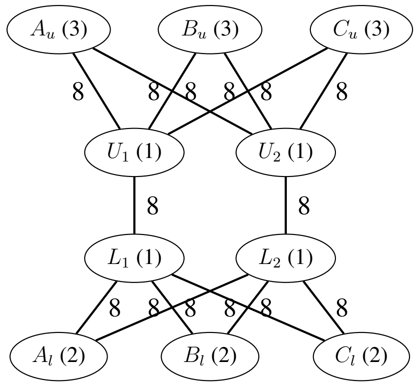


Source: from author (2015).

Lemma 3.3.1. *Any Bin Packing instance can be reduced to an instance of the Virtual Network Embedding Problem through procedure ϕ.*

Proof. There are n virtual nodes with demand three and n physical nodes with capacity three, therefore any feasible solution would map all virtual upper nodes to physical upper nodes.

Figure 3.4: Substrate Graph resulted from transformation ϕ.



Source: from author (2015).

Likewise for physical and virtual nodes with capacity and demand two. Hence, in the instance $\phi(I)$ there is always a feasible mapping of the nodes. Moreover, each of the n edges has to be mapped to a single path in the substrate graph between the upper nodes and the lower nodes. Additionally, every path has to contain at least one edge between the middle nodes. Those edges represent bins. Mapping virtual links to those edges is tantamount to fit items in a bin. □

Lemma 3.3.2. *The transformation $\phi(I)$ is polynomial in $|N|$.*

Proof. The transformation ϕ creates two graphs, one with $2|N|+2k$ nodes and $2|N|k+k$ edges and another with $2|N|$ nodes and $2n-1$ edges. As $k < |N|$, and every item fits into a single bin, the size of the VNEP instance is limited polynomially by $|N|$, the number of items of I. □

Lemma 3.3.3. *If there exists a polynomial time algorithm that finds a feasible solution for $\phi(I)$, there is a polynomial time algorithm that finds a feasible solution for I.*

Proof. All virtual nodes with demand three have to be mapped to substrate nodes with capacity three. If one of the nodes of demand two is mapped to a substrate node of capacity three, there will be an unmapped virtual node of capacity three. Likewise, for nodes of capacity two. Therefore there is always a feasible node mapping for $\phi(I)$. Note that the mapping order is not important: any virtual node with demand x can be mapped to any physical node with capacity x.

If there is a feasible mapping for the virtual network of $\phi(I)$, the answer to I is YES: Given a feasible mapping M, all virtual edges are mapped to a path in the substrate graph. Let p_i be the path for which the virtual link i, corresponding to the item i, was mapped to. Let j be the first substrate edge in p_i that links the middle nodes. The edge j corresponds to the bin j. Thus a solution for I can be constructed if item i is allocated into bin j. Since the capacity of all edges are not surpassed, the k bins can hold the $|N|$ items.

If the answer for I is YES, there is a feasible mapping for $\phi(I)$: Suppose that there is a configuration of the items into k bins. A solution for $\phi(I)$ can be constructed from a solution for I. Suppose any feasible mapping M of nodes for the virtual network (as it was shown before, there always exists a feasible mapping of nodes). Let $f_M(u)$ be the substrate node for which the virtual node u is mapped. Every virtual link (u, v) corresponding to the item i is mapped to the path $(f_M(u), x)$, (x, y), $(y, f_M(v))$, where (x, y) is the substrate edge corresponding to the bin j for which the item j was allocated in the solution for I. Since the sum of the items in a bin j does not exceed the capacity B, every virtual link can be mapped in the physical graph. $\square$

Lemma 3.3.4. *VNEP is an NP problem*

Proof. A solution for a VNEP instance I can be verified in polynomial time in the size of the instance. For every virtual node u it suffices to check if the physical node $f_v(u)$ has enough capacity to host it, for every virtual link (u, w) it suffices to check the path $p = f_e(u, w)$ is indeed a path (every adjacent edge shares one endpoint), and for every physical edge e, if all the links that are mapped in e do not surpass its capacity. $\square$

Theorem 3.3.1. *VNEP is an NP-Complete problem.*

Proof. The hardness of the VNEP problem follows from Lemmas 3.3.1-3.3.4. $\square$

Corollary 1. *There is no polynomial time algorithm that finds a feasible solution for VNEP, unless $P = NP$.*

Proof. Suppose there is a polynomial time algorithm that finds a feasible solution (not necessarily optimal) for VNEP. From 3.3.3, it follow that there is a polynomial algorithm for solving BPP. But this is only possible if $P = NP$. $\square$

Corollary 1 implies that there is probably no time efficient approximation algorithm for VNEP, since such an algorithm would provide a polynomial time algorithm to solve BPP.

4 COLUMN GENERATION

Column generation is a mathematical programming technique proposed by Dantzig and Wolfe (DANTZIG; WOLFE, 1960) to solve models with a large number of variables. These "large" models usually provide better bounds on the optimal integer solutions than compact models, and this is the case for the VNEP.

Although the number of variables in a large model grows exponentially with the size of the instance, the number of nonzero variables in a solution is small. The column generation algorithm takes advantage of that. A Restricted Master Problem (RMP) comprised of a limited set of columns is solved. More columns are generated and added to the RMP by solving a subproblem (pricing), until no more columns improving the solution.

The proposed column generation algorithm is summarized in Algorithm 17. Initially, in Line 2, artificial variables are added to the RMP so that there is always a feasible solution. That process is explained in Section 4.1. The RMP is solved in Line 4. The pricing problem is solved in lines 5 through 11, and it is further explained in Section 4.2 The pricing problem generates columns to be added to the RMP, and when no more columns are generated, the algorithm stops.

Algorithm 1: Column Generation Algorithm for VNEP.

1 $P' = \emptyset$;
2 Add artificial variables to the model such that model is feasible;
3 **repeat**
4   Solve model with limited set of columns P';
5   **foreach** *Virtual link* $(u, v) \in E^V$ **do**
6   Construct auxiliary graph A using dual variables;
7   Find minimum cost u-v-path p in A with reduced cost r_p;
8   **if** $r_p < 0$ **then**
9   Add path to P';
10   **end**
11   **end**
12 **until** *No paths were added to* P';
13 **if** *All artificial variables are equal to zero* **then**
14   The problem is solved optimally;
15 **else**
16   The problem is infeasible;
17 **end**

This algorithm is based on a column generation algorithm introduced in (HU; WANG; CAO, 2013). However, that approach allows virtual links to be mapped to multiple paths. Next we describe how to adapt those approach to the single-path VNEP.

4.1 Obtaining an initial set of columns

With no columns, the RMP is initially infeasible. An approach similar to the simplex two-phase method is used to add columns comprised of a set X_0 of artificial decision variables to have a feasible RMP. Variables X_0 are added to Constraints (3.14) and (3.12) of the flow-model presented in Section 3.2, and impose a high penalty to the objective function. Any feasible solution to the original RMP have a better objective value than a solution that uses any variable of X_0. When the column generation algorithm stops, if some X_0 variable has a value different than zero, the original RMP is infeasible. Otherwise X_0 variables are discarded.

4.2 The Pricing Problem

At each iteration, new columns are generated implicitly and added to the RMP. Columns are obtained solving a pricing problem that consists in finding the column with the minimum reduced cost. If the column with the minimum reduced cost has a negative value, the column is added to the RMP. Otherwise, no column can improve the current solution, and thus the current solution is optimal.

Let $\lambda \geq 0$, $\eta \geq 0$, and π unrestrict be the dual variables associated to constraints (3.14), (3.12), and (3.13), respectively. The reduced cost r_p of each variable z_p that covers the virtual link k is:

$$r_p = c_p B_k - \lambda_k - \sum_{e \in p : e \in E^S} B_k \eta_e - \sum_{(v,s) \in p : (v,s) \notin E^S} \pi_{v,s}$$

Since c_p is the number of edges on the path p, the equation above is equivalent to:

$$r_p = \sum_{e \in p : e \in E^S} B_k (1 - \eta_e) - \sum_{e \in p : e \notin E^S} \pi_e - \lambda_k$$

Thus, the minimum reduced cost for the VNEP is obtained by solving the following problem:

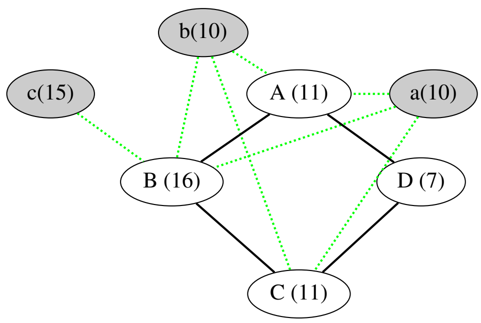
$$\min_{\forall k \in E^V, \forall p \in P^k} \quad \sum_{e \in p : e \in E^S} B_k (1 - \eta_e) - \sum_{e \in p : e \notin E^S} \pi_e - \lambda_k$$

If we construct an auxiliary graph in which the cost of each path corresponds to its reduced cost, the pricing problem consists in finding a path with the minimum cost, which can be solved polynomially. To this end, the dual variables obtained from the RMP optimal dictionary are used to construct an auxiliary graph. For each virtual link $k = (u, v) \in E^V$, we construct an auxiliary graph in which auxiliary edges (v, s) have a cost of $-\pi_{v,s}$ and substrate edges $e \in E^S$ have a cost of $B_k(1 - \eta_e)$. That way each path p in that auxiliary graph between two auxiliary nodes u and v has a cost exactly equal to its reduced cost r_p. Thus to obtain the path with smallest reduced cost, we can use a polynomial algorithm to find the shortest path in the constructed auxiliary graph.

The auxiliary graph is constructed in the following way: On top of the physical substrate graph, for each virtual node $v \in |V^V|$ an auxiliary node is added. Those auxiliary nodes are connected through auxiliary edges with all physical nodes that have enough capacity to host them.

An auxiliary graph built based on the instance from Figure 3.1 is shown in Figure 4.1. Auxiliary nodes a, b and c are presented in gray. The node a is linked to all nodes but node D, which does not have enough capacity to host it. Two commodities are created: (a, b), with demand 20, and (b, c), with demand 10.

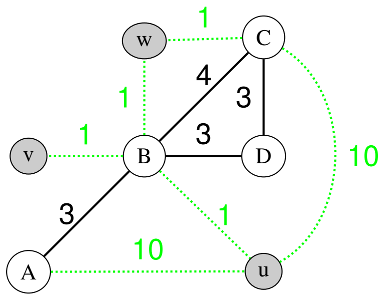
Figure 4.1: Auxiliary graph for the input instance of Figure 3.1.



Source: from author (2015).

Paths in the auxiliary graph yield a valid column to the RMP if they respect some constraints. Suppose we have to find a path to map the virtual link (u, v). First, the path has to contain exactly two auxiliary edges, that are the ones with endpoints in u and v. This is easily enforced by setting a cost ∞ to all auxiliary edges that connect auxiliary nodes other than u or v. Second, a path in the auxiliary graph has to contain at least one substrate edge, and the node adjacent to the source has to differ from the node adjacent to the target node. If one of these two conditions are not satisfied it means that both auxiliary nodes are mapped to the same substrate node, which is a restriction of the problem. Hence, Dijkstra's algorithm cannot be directly applied to solve the pricing problem. Figure 4.2 shows an example of a pricing problem. Auxiliary nodes are filled with gray and auxiliary edges are dotted.

Figure 4.2: Subproblem graph



Source: from author (2015).

Suppose that we want to find a path between u and v in the auxiliary graph. The minimum-cost path in the graph $u - B - v$ does not contain any substrate edge and then is invalid. The path $u - B - D - C - B - v$ is also invalid, because both u and v are mapped to the same substrate node B. The path $u - C - w - B - v$ is invalid because it uses auxiliary edges not connected to u or v. Therefore the minimum-cost valid path in the graph is $u - A - B - v$, where node u is mapped to A and v to B, and virtual link (u, v) is mapped to (A, B). Note that the edge (w, B) has a cost of ∞, so no path could use this auxiliary edge.

4.3 Proposed Modified Dijkstra Algorithm

We propose a modified Dijkstra's algorithm (Algorithm 2) to find a valid path with the minimum reduced cost. It works as multiple Dijkstra's algorithms running in parallel. Each node to be visited is associated with an origin, i.e. the first physical node in the path. Initially all neighbors of the source node s are added to the priority queue Q, with an origin equal to itself. At each iteration the node in the queue with the minimum distance is selected to be visited. When the node v with origin w is visited, its neighbors are added to the queue with an origin w if it the cost of reaching the neighbor through v is smaller than its current distance. Hence the same node can be added multiple times if it is reached from different origins. But the algorithm is still polynomial since the number of times each node will be visited is bound by the degree of s. The target t can only be added when the path contains at least one physical edge, this is detected by checking if the origin of the node is equal to itself. The algorithm stops when the target node t is visited.

Algorithm 2: Modified Dijkstra's Algorithm

Input: Let s be the source node, and $G^A = (V^A, E^A)$ be the auxiliary graph, and $c(u, v)$ the cost of the edge (u, v).

Output: $d[v, w]$ is the cost of the minimum cost path from s to v with origin w.

initialize every d as ∞;
foreach *edge* $(s, v) \in E^A$ **do**
    $d[v, v] := c(s, v)$;
    $Q := Q \cup \{(v, v)\}$;
end
while Q *is not empty* **do**
    $(u, origin) := extract\_min(Q)$;
    if *u* = *t* **then**
        return best path with cost $d[u, origin]$;
    end
    foreach *edge* $(u, v) \in E^A$ **do**
        if $v = t$ *and* $u = origin$ **then**
            continue;
        end
        if $d[v, origin] < d[u, origin] + c(u, v)$ **then**
            $d[v, origin] := d[u, origin] + c(u, v)$;
            if $(v, origin) \in Q$ **then**
                decrease cost of $(v, origin)$ in Q;
            else
                $Q := Q \cup \{(v, origin)\}$;
            end
        end
    end
end

Proposition 2. *The Modified Dijkstra's Algorithm finds a valid path to the problem with the smallest cost in the auxiliary graph in polynomial time.*

Proof. At each moment the set of nodes that were already visited have the shortest distance from the source node. A node is not visited more than its indegree, i.e., the number of arcs incoming that node. Suppose a node u with cost d that was already visited is adjacent to a node v being visited with cost $d' < d$. This is an impossible situation because if u has distance $d' < d$ it would be visited before u. So once a node is visited it has the smallest possible distance from the source. Thus once the target node t is visited it has the smallest possible distance from the source.

Moreover, the algorithm has to find the shortest distance taking into account the aforementioned constraints. With this aim, the target node t can only be added to the queue if its predecessor is not equal to its origin. In that way the path obtained by this algorithm always contains at least one physical edge. Additionally, since there are no edges with negative costs, the path obtained does not contain a loop. This algorithm is polynomial and corresponds to run a Dijkstra algorithm for each node adjacent to the source node of s. Theoretically, this algorithm has a time complexity of $O(|E^S||V^S| + |V^S|^2 log|V^S|)$ with Fibonacci heaps, and $O(|V^S||E^S|log|V^S|)$ with binary heaps. $\square$

5 BRANCH & PRICE

The column generation (CG) algorithm solves only the linear relaxation of the problem. To obtain integer solutions for the VNEP, CG is embedded in a Branch & Bound algorithm (B&B). A B&B algorithm where each node is solved through a column generation is called Branch & Price (B&P).

We implemented a classic B&P procedure which is summarized in Algorithm 13. Initially, the relaxation of the problem is solved. If the optimal solution is not integral, the problem is split into two subproblems. This process is called branching (Section 5.1). These subproblems are inserted in a priority queue with the key set to their relaxation value. At each iteration, a subproblem is selected (Section 5.2) from the queue and solved through a CG algorithm. To speed up the algorithm by improving upper bounds, a constructive solution is build at each iteration (Section 5.3) aiming to obtain an integral solution. In case the incumbent integral solution has a better objective value than a subproblem relaxation, that branch is not expanded anymore since the relaxation of the problem is a lower bound on the integral solution.

The problems are branched with the introduction of cuts. Those cuts can turn the problem infeasible. New columns have to be inserted in order to make the problem feasible again. The two-phase approach explained in Chapter 4 is used to overcome this problem.

Next we describe decisions taken by the B&B for the VNEP we have implemented.

5.1 Branching

When expanding a node of the B&B tree, if the solution is fractional, a variable is selected to be branched (Section 5.1.1). The model solved at each node has two sets of decision variables: node mapping variables $x_{v,s}$ and path mapping variables z_p. The former has a fixed size, while the latter grows during B&P execution. Therefore each one is treated differently.

5.1.1 Variable Selection

Variables $x_{v,s}$ are intregralized first starting with variable with values closest to 0.5. Once all variables x are integral, path variables z_p are verified in order of the origin node label. Paths with the same origin are verified in the order they were generated.

Algorithm 3: Branch & Price Algorithm

Q = relaxation of IP
while *Q is not empty* **do**
    subproblem = select_node(Q);
    x^* = solve subproblem using column generation;
    if *feasible(x^*) and $\phi(x^*) < UB$* **then**
        if *x^* is integral* **then**
            $UB = \phi(x^*)$;
        else
            build heuristic solution;
            split subproblem into subproblems and add them to Q;
        end
    end
end

5.1.2 Node mapping branching

We tested two methods of branching for variables $x_{v,s}$. In the first, two subproblems are created, one for which the substrate node s is forbidden of hosting v ($x_{v,s} \leq 0$) and another for which the node s is forced to host v ($v_{v,s} \geq 1$). The second is called Generalized Upper Bound (GUB): since $\sum_{s' \in V^S} x_{v,s'} = 1$, one can cut more than one variable at a time. Two problems are created, one of which half the variables x_v are forbidden by adding the cut $\sum_{s < |V^S|/2} x_{v,s} \leq 0$, and another for which the other half of variables are forbidden. In both methods, the addition of cuts affect the pricing problem. The auxiliary edge (v, s) is either forbidden or fixed accordingly.

5.1.3 Path branching

Branching of path variables is not as straightforward as branching node variables. Individually branching on path variables would yield a large number of nodes to explore, since the number of paths grows exponentially with the size of the instance. To avoid this problem, paths are branched by introducing constraints that simulate restrictions on variables y from the compact model. Suppose that for a given node, there exists one path variable z_p that has a fractional value. Then there must be at least another variable $z_{p'}$ that is also fractional, since the sum of the variables that cover a virtual link is 1. The paths p and p' associated to variables z_p and $z_{p'}$ must have at least two different substrate edges $e \in p, e \notin p'$ and $f \in p', f \notin p$, otherwise they would be the same path. Then, two branches are generated, one with e forbidden to belong to the path, and another for which f is forbidden. Edges are not fixed, because it is easier to find a path that does not use multiple edges (it suffices to set their weights to a large value) than it is to fix edges. An edge e is forbidden to be used to cover the virtual link k by adding the cut $c(k, e) = \sum_{p \in P^k} \delta_{e,p} z_p \leq 0$.

Adding these cuts affects the pricing problem. For each $k \in E^V$ and $e \in E^S$ a dual variable $\psi_{k,e}$ associated with the cut $c(k, e)$ is created. The values of these variables are added to the cost of each edge in the auxiliary graph. So each physical edge in the auxiliary graph has the cost $B_k(1 - y_e - \psi_{k,e})$.

5.2 Node Selection

The order in which active nodes are visited affects the performance of the algorithm. A balance has to be found between finding good upper bounds and visiting promising nodes. We tested three approaches: Depth First Search (DFS), Best First Search (BFS), and Best Projection (BPJ). DFS visits nodes in the order they are created. BFS visits first the most promising nodes, i.e., nodes with the lowest dual bound. BPJ visits nodes with fewer fractional variables first, since they are possibly closer to an integral solution.

5.3 Heuristic Solution

It was seen in Section 3.3, that to obtain a feasible solution for the VNEP is NP-Hard. Nevertheless, a greedy algorithm can be applied and in many situations is able to find a feasible solution. Moreover, if part of the solution is already fixed, only the remaining part has to be constructed. This happens when a greedy solution is applied on a subproblem, which uses all variables set on that branch of the B&B tree since the root node. Constructed feasible solutions can improve upper bounds allowing to prune entire branches of the B&P search tree.

A solution obtained by the column generation may contain several fractional variables. The values of those variables contain valuable information about the problem. Each virtual node v is mapped to the free physical node s for which the value of $x_{v,s}$ is the largest. After all nodes are mapped, a breadth first search is used to map virtual links to paths in the physical substrate graph. The mapping of edges can fail if no path with enough bandwidth is found.

This algorithm is applied at each node of the B&B tree. If a feasible solution is successfully built by the constructive heuristic algorithm, and the obtained solution is better than the current upper bound, the incumbent solution is updated.

6 EXPERIMENTAL RESULTS

This section presents an experimental evaluation of the proposed B&P algorithm. Results from B&B are compared with results from the compact model presented in Section 3.2 solved with CPLEX. Four aspects of the algorithm are tested: the time spent to find the first integral solution, the running time, the number of instances solved, and the quality of solutions that were not proved to be optimal due to a time limit constraint. All tests were performed on a processor Intel Core i7 930 with 12 Gb of memory using a single thread. The commercial solver CPLEX 12.5 was used to solve the relaxed and integer models. All running times are in seconds, and a time limit of one hour was used in all runs.

6.1 Datasets

As virtualization is a fairly recent field, there is no established set of benchmark instances available for the VNEP. The properties of physical substrate networks and virtual network embedding requests are not well understood (CHOWDHURY; RAHMAN; BOUTABA, 2012). Hence most works use synthetic networks (FISCHER et al., 2013). Following this approach, all instances used in this experimental evaluation are generated with the GTI-ITM tool (ZEGURA; CALVERT; BHATTACHARJEE, 1996). This tool is capable of generating graphs with three different topologies: random, hierarchical and transit-stub. Figure 6.1 presents an illustrative example of each topology.

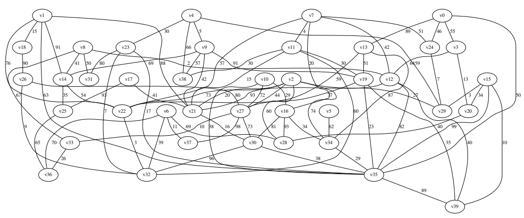
Random instances are generated by spreading the nodes randomly in a 100 by 100 plane. These nodes are randomly linked using Waxman model (ZEGURA; CALVERT; BHATTACHARJEE, 1996). In this model, the probability of connecting two edges is $\alpha e^{-d/(100\beta)}$, where d is the Euclidean distance between the nodes, and α and β are parameters. Thus, a larger α means a denser graph, and a larger β means longer edges in the graph. Two types of random instances are generated: **Dense Random**, using $\alpha = 0.8$ and **Sparse Random**, using $\alpha = 0.5$. Both sets use $\beta = 0.2$.

Hierarchical instances are generated recursively. Initially, a random graph is created in a 100 by 100 plane. Then each node is replaced by a random graph with an average of five nodes. To generate those graphs the parameters $\alpha = 0.5$ and $\beta = 0.2$ were used.

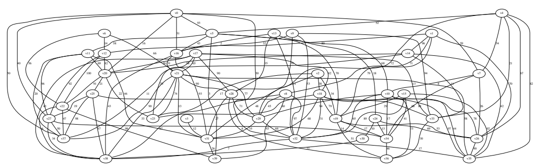
Transit-stub is an hierarchical topology that best model the Internet (ZEGURA; CALVERT; BHATTACHARJEE, 1996). Each node in a randomly generated graph is replaced by a subgraph called transit domain with an average of four nodes. To these domains are attached an average of three stub domains. Each stub domain has an average of two nodes. The

Figure 6.1: Topology examples

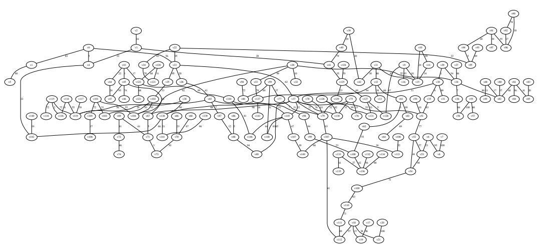
(a) Sparse Random



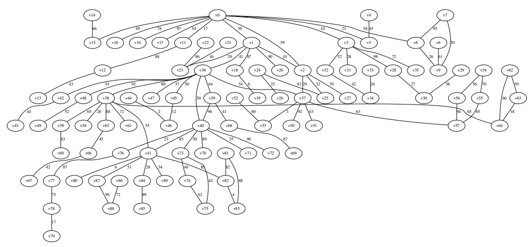
(b) Dense Random



(c) Hierarchical



(d) Transit-Stub



Source: from author (2015).

transit domains as well as the stub domains are connected randomly with a probability $\alpha = 0.5$.

Each instance is composed of one physical substrate graph of one of the four topologies (Dense Random, Sparse Random, Hierarchical, Transit-stub) and one Sparse Random virtual network. Virtual networks were generated with connectivity 50% as in (CHOWDHURY; RAHMAN; BOUTABA, 2012). Capacities of physical nodes and edges are randomly generated integers in the interval $[1, 100]$. Considering the demands of virtual networks, two sets of instances were generated: one with *low demand* with demands in the interval $[1, 25]$, and another with *high demand* with demands in the interval $[25, 100]$.

As one of the purposes of this study is to test the limits of exact algorithms, large physical networks were generated. Instances were generated from 20 to 200 nodes for random sparse and random dense topologies (6 sizes of substrate networks each). For hierarchical instances, from 125 to 250 nodes (6 sizes of substrate networks). For Transit-stub, graphs from 90 to 270 nodes (5 sizes of substrate networks). Virtual networks are in general small, and were generated from 2 to 18 nodes considering all even sizes in this interval. Input data for the problem was build considering all combinations of substrate and virtual networks, and for each combination two instances are build considering a low and high demands of the substrate network. Thus, in overall 414 instances were tested.

In Section 6.2 we describe the parameter setting, while that in Section 6.3 we present the B&P results, and compare against the ones found by the compact model solved with CPLEX. The amount of time spent by each component of column generation is analysed in Section 6.4

6.2 Parameter Testing

As explained in the previous section, the node selection and branching methods influence the overall performance of the algorithm. The use of the heuristic constructive method can also influence the performance of Branch & Price. The following experiments compare different options for these components considering one instance for each size in a total of 414 instances. The main metric used is the running time to obtain the optimal solution and the number of instances solved optimally in the available time limit. Initially we tested the performance of the B&B algorithm selecting nodes using DFS, BFS, and BPJ and results are summarized in Table 6.1. The table shows for each of the four sets of instances, the number of tested instances in column **#inst**, and for each node selection method, the number of optimal solutions found within the time limit in column **#opt**, and the average running time in seconds in column **time(s)**.

Table 6.1: Node Selection Results

		DFS		BFS		BPJ			
set	#inst	#opt	time(s)	#opt	time(s)	#opt	time(s)	DFS $\sim$ BFS	BPJ $\sim$ BFS
Random-Sparse	108	76	1086.20	85	851.90	83	863.66	$< 10^{-3}$	0.991
Random-Dense	108	78	961.50	88	792.10	87	691.44	$< 10^{-3}$	0.994
Hierarchical	108	62	1503.62	64	1375.61	59	1542.29	$< 10^{-3}$	0.035
Transit-stub	90	59	1393.76	66	1138.92	62	1294.34	$< 10^{-3}$	0.003

Source: from author (2015).

By comparing the number of optimal solutions and running time, BFS is clearly the best method. Using Wilcox Signed-rank test with a confidence interval of 95%, our results show that BFS has a better running time performance than DFS for all instances. With the same test, BFS is significantly better for Hierarchical and Transit-stub instances. Results for the Wilcox Signed-rank test performed are shown in Table 6.1.

Further tests were performed to compare normal branching of variables and GUB. Branching on variables is also significantly better than GUB with a confidence interval of 95%.

A final test was made to evaluate the use of the constructive heuristic method at each node of the Branch & Price search tree. No conclusions could be draw from these tests with a confidence interval of 95%. However, the average time with B&P with a constructive heuristic method was 6 seconds better than the average time of the algorithm without this method.

6.3 Branch & Price Evaluation

Results for the proposed B&P algorithm with normal branching of variables, BFS strategy for selecting nodes, and using the constructive method at each node are presented in this section. Moreover, we present results for the compact ILP Model from Section 3.2 for the sake of comparison.

For each instance size, three substrate networks and three virtual networks were randomly generated and combined to form nine different VNEP instances. The values in the results are the mean for the nine instances. In total 3726 instances compose the dataset.

All results are shown in Tables 6.2-6.5. For summarizing results, each line corresponds to the mean values obtained by all instances run with the virtual network size $|V|$. We grouped instances in this fashion since time changes considerable with the size of virtual networks, and not much with the size of substrate networks. Column **#NF** is the mean number of instances that did not finish withing the time limit, i.e., either they did not prove solution optimality, or they did not prove solution infeasibility in less than an hour. Column **time_int** shows the average time to obtain the first integer solution, when one was found. Columns **time(s)** and **cost** show the average running time and solution cost, respectively.

Full results are in Tables A.1-A.8 of the Appendix. Each line in those tables is the average of nine instances of the same size, generated as explained in Section 6.1.

Figures 6.2, 6.3, 6.4, and 6.5 demonstrate the behavior of the algorithms for all instances. All graphs compare the results for ILP solved with CPLEX in dashed lines, and B&P with a solid line. Each figure contains four different graphs, one for each topology. Graphs show results for instances sorted alongside the x-axis first by the size of virtual nodes and second by the number of physical nodes. So for example the point corresponding to a random sparse instance with 4 virtual nodes and 20 physical nodes is right to left of the result of the instance with 4 virtual nodes and 40 physical substrate nodes. In that way it can be seen that the number of virtual nodes is a better predictor of the time complexity of each instance. Hence even instances with a large number of physical nodes can be easily solved by B&P when there is at most 8 virtual nodes.

Table 6.2: Results for sparse random instances.

	Low Demand							
	Branch & Price				CPLEX			
$\|V\|$	#NF	time_int	time(s)	cost				
2	0.00	0.04	0.04	16.33	0.00	0.06	0.13	16.33
4	0.00	0.17	3.05	25.04	0.00	1.27	41.57	25.04
6	0.00	0.43	54.79	63.17	3.00	14.94	1580.90	63.17
8	0.00	0.98	136.27	77.86	6.50	123.32	2733.78	77.91
10	0.50	1.00	842.67	124.43	7.50	362.98	3071.16	132.59
12	0.83	1.84	694.94	129.28	7.83	713.39	3163.46	154.21
14	1.83	2.53	1468.01	154.61	7.67	1063.17	3123.54	175.57
16	4.33	15.09	2211.49	234.78	7.67	1152.81	3117.38	298.26
18	4.17	73.52	2212.18	246.19	7.50	1378.41	3012.03	326.40
	High Demand							
	Branch & Price				CPLEX			
$\|V\|$	#NF	time_int	time(s)	cost	#NF	time_int	time(s)	cost
2	0.00	0.03	0.03	19.61	0.00	0.06	0.13	19.61
4	0.00	0.11	4.00	59.45	0.00	1.12	17.54	59.45
6	0.00	0.20	159.68	137.92	2.17	10.61	1210.49	137.92
8	0.00	0.42	44.15	170.51	4.50	62.90	2031.45	170.51
10	0.00	0.58	167.44	203.39	6.00	349.79	2473.09	219.81
12	2.50	0.74	1478.81	284.16	6.33	826.02	2596.43	374.19
14	2.83	1.68	1557.97	426.72	6.50	1062.09	2719.31	536.23
16	5.67	1.96	2501.02	585.75	6.83	1570.38	2852.49	788.15
18	5.50	3.01	2490.59	639.29	7.33	1930.45	2963.74	973.25

Source: from author (2015).

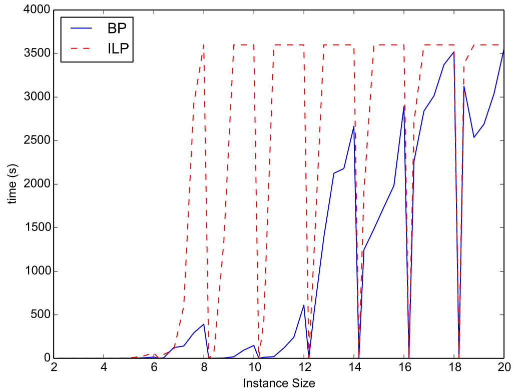
Figure 6.2 compares average running times. B&P had a better running time for 80 of the 90 instance sizes of the Transit-stub set, 97 out of 108 for the hierarchical set, and for all but 3 and 4 instances for the dense and sparse sets, respectively. Even though solving the large model is in general more expensive than solving the linear relaxation of the compact model, its better lower bounds and the better exploitation of the problem structure compensate for its longer running times.

Figure 6.3 shows the average cost obtained considering the best solution value of instances in which the algorithm was able to find at least one feasible solution within the time limit. Those graphs show the cost of the best obtained solution on the time limit. These graphs show some discontinuities since in some cases algorithms were not able to find any feasible solution, or to prove infeasibility.

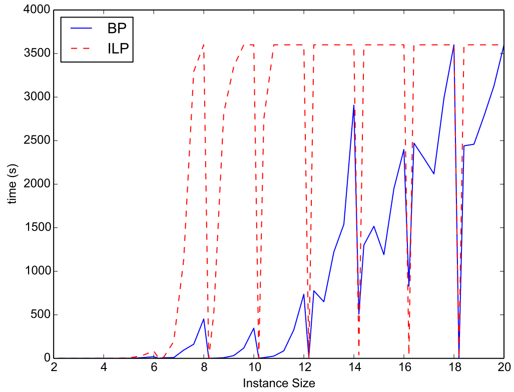
The number of instances solved can be seen in Figure 6.4. It shows the number of instances out of 9 that were either proved optimal or infeasible.

Figure 6.2: Average time in seconds.

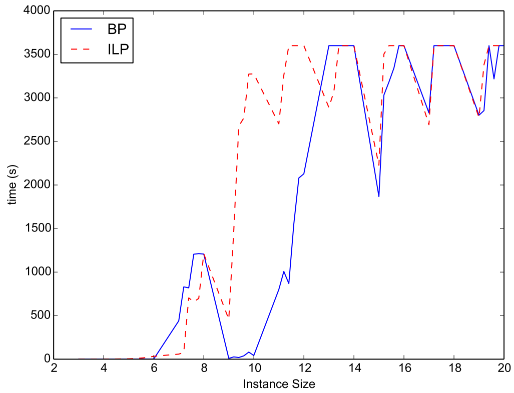
(a) Sparse Random



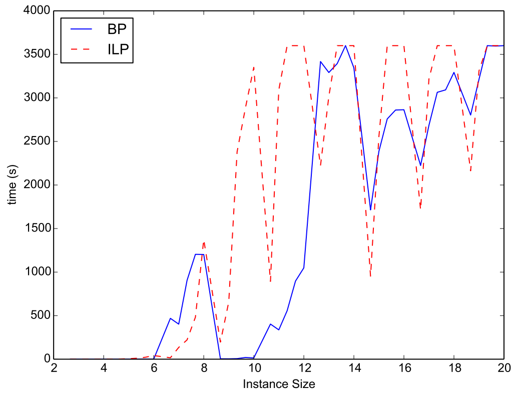
(b) Dense Random



(c) Hierarchical



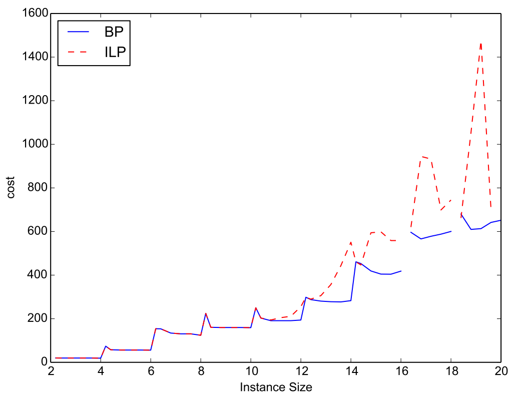
(d) Transit-stub



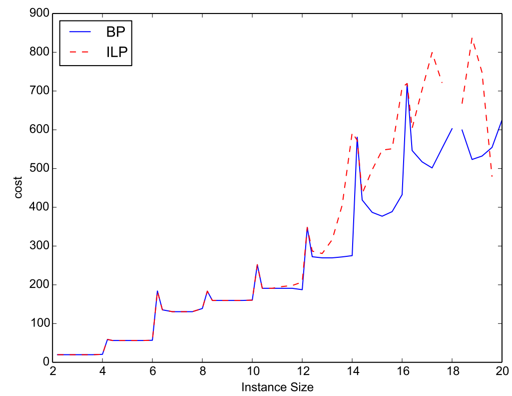
Source: from author (2015).

Figure 6.3: Average cost of feasible solutions.

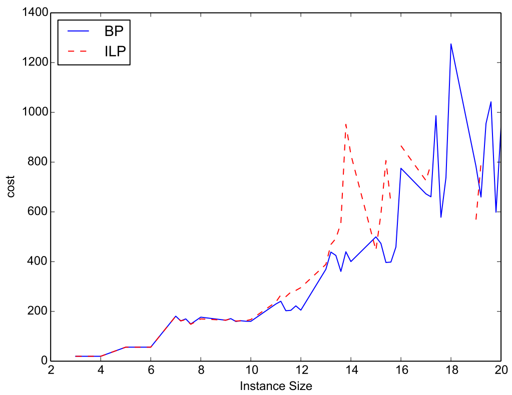
(a) Sparse Random



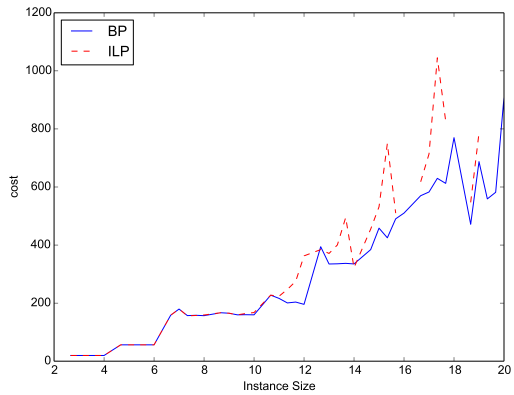
(b) Dense Random



(c) Hierarchical



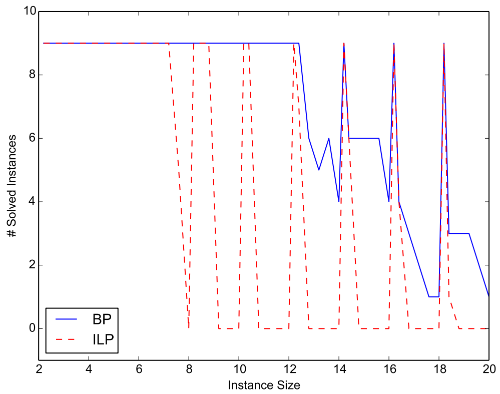
(d) Transit-stub



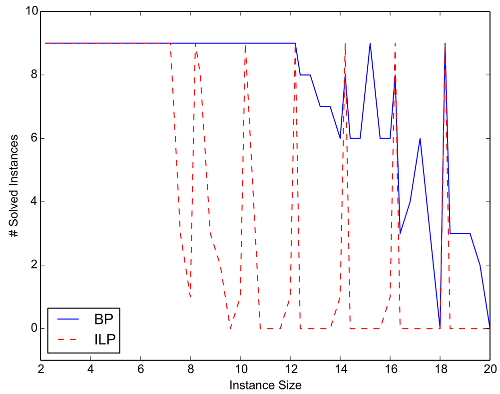
Source: from author (2015).

Figure 6.4: Number of instances solved.

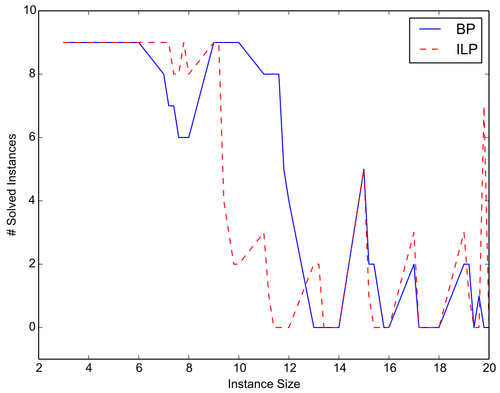
(a) Sparse Random



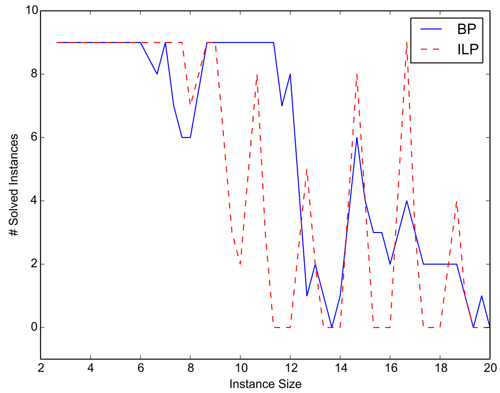
(b) Dense Random



(c) Hierarchical



(d) Transit-stub



Source: from author (2015).

Table 6.3: Results for dense random instances.

	Low Demand							
	Branch & Price				CPLEX			
$\|V\|$	#NF	time_int	time(s)	cost				
2	0.00	0.08	0.08	16.33	0.00	0.07	0.14	16.33
4	0.00	0.33	4.17	25.00	0.00	1.34	96.69	25.00
6	0.00	0.70	77.48	63.06	3.67	14.75	1790.96	63.06
8	0.00	1.14	204.96	77.50	7.00	142.51	2872.77	77.50
10	0.67	3.81	752.48	122.58	7.33	463.34	3064.37	127.93
12	0.67	2.85	841.00	129.69	8.00	739.44	3293.93	143.25
14	2.00	12.13	1350.52	153.78	8.00	1096.43	3246.88	166.88
16	3.67	8.14	2029.86	227.89	7.83	1155.70	3292.66	253.93
18	3.17	21.96	2020.03	237.44	8.50	893.13	3400.04	292.07
	High Demand							
	Branch & Price				CPLEX			
$\|V\|$	#NF	time_int	time(s)	cost	#NF	time_int	time(s)	cost
2	0.00	0.04	0.04	19.82	0.00	0.06	0.14	19.82
4	0.00	0.17	5.18	56.87	0.00	1.23	23.55	56.87
6	0.00	0.33	121.04	141.74	2.33	9.44	1372.85	141.74
8	0.00	0.68	84.33	163.81	5.17	54.82	2313.42	163.81
10	0.00	0.79	198.03	200.56	6.33	429.71	2859.08	205.82
12	1.50	1.19	1184.64	284.49	7.33	850.40	3001.72	371.76
14	2.17	4.95	1477.19	430.88	7.33	1041.58	3006.72	552.10
16	5.00	28.19	2383.60	573.33	7.50	906.56	3005.92	709.45
18	5.67	5.27	2401.59	566.95	7.50	1650.41	3000.13	682.96

Source: from author (2015).

Figure 6.5 shows the time to obtain the first integer solution. In those graphs it is clear that B&P is able to find initial integer solutions quicker than CPLEX due to the constructive heuristic used in each node.

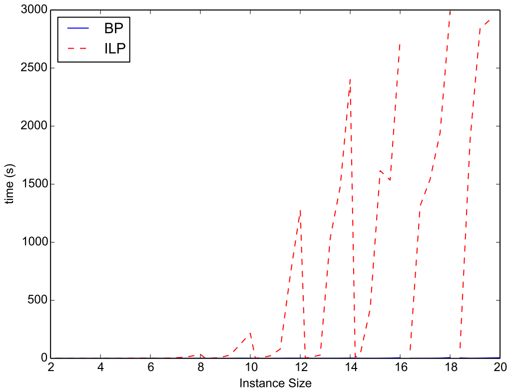
B&P runs in significantly less time than CPLEX for most instances.Than, it is able to solve larger instances than CPLEX. For some hierarchical instances CPLEX consume all memory space and was not able to successfully find any solution.

The difference in performance is specially noticeable for larger instances. For example, for a substrate graph with a dense random topology and composed of 200 nodes, CPLEX was only able to solve instances with up to 4 virtual nodes, while B&P could solve all instances with up to 10 nodes optimally in less than 10 minutes on average. Moreover, it was able to solve some of the larger instances in less than an hour.

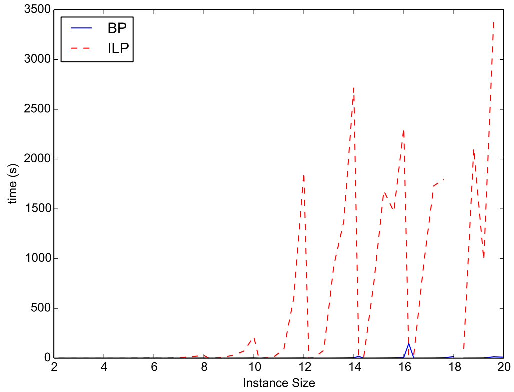
CPLEX has a better performance for sets of instances with high demands, which contain more infeasible instances. However, both algorithms have a harder time proving infeasibility.

Figure 6.5: Time to obtain the first integer solution.

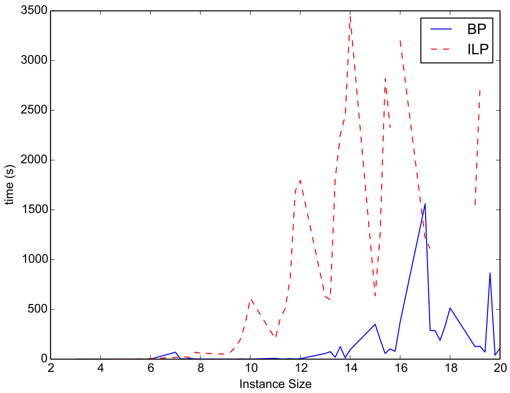
(a) Sparse Random



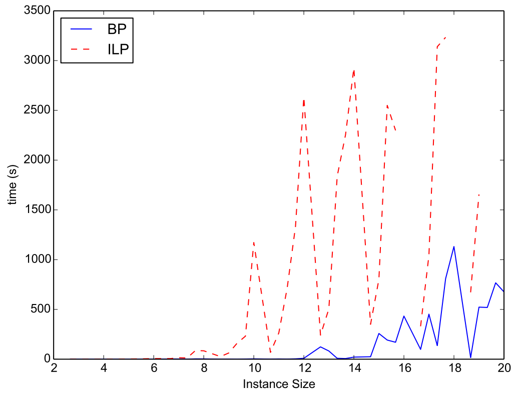
(b) Dense Random



(c) Hierarchical



(d) Transit-stub



Source: from author (2015).

Table 6.4: Results for hierarchical instances.

	Low Demand							
	Branch & Price				CPLEX			
$\|V\|$	#NF	time_int	time(s)	cost				
2	0.00	0.01	0.01	16.33	0.00	0.11	0.26	16.33
4	0.00	0.05	0.95	25.00	0.00	3.78	30.41	25.00
6	0.00	0.33	12.43	62.67	1.00	32.77	1149.86	62.67
8	0.00	0.67	65.63	76.67	8.17	249.94	3499.51	80.00
10	4.33	4.69	2156.26	131.79	9.00	770.83	3600.00	167.09
12	3.50	15.29	1712.39	138.04	9.00	1221.98	3600.00	214.18
14	7.17	43.64	2963.53	217.70	8.67	1943.51	3600.00	276.11
16	9.00	164.24	3600.00	391.06	8.00	2360.40	3600.00	442.60
18	9.00	196.71	3600.00	532.74	8.67	3201.44	3600.00	573.00
	High Demand							
	Branch & Price				CPLEX			
$\|V\|$	#NF	time_int	time(s)	cost	#NF	time_int	time(s)	cost
2	0.00	0.01	0.01	19.67	0.00	0.11	0.25	19.67
4	0.00	0.07	0.99	56.33	0.00	3.08	15.72	56.33
6	2.33	14.97	952.28	167.03	0.50	34.73	567.71	164.50
8	0.00	1.45	36.06	163.09	4.17	234.69	2321.97	164.91
10	2.17	4.36	1406.51	217.52	8.33	905.15	3393.83	270.13
12	9.00	64.83	3600.00	405.57	8.33	1863.78	3393.66	614.59
14	7.33	193.54	3104.03	500.35	8.00	2048.64	3355.05	668.70
16	8.67	528.56	3471.32	818.49	8.50	1162.94	3448.68	759.00
18	8.17	223.88	3279.01	829.86	7.17	2148.84	3391.17	682.50

Source: from author (2015).

Hence further study ought to be made investigating methods to detect infeasibility with the Branch & Price algorithm.

6.4 Running time of column generation components

At each node of the B&P tree, a column generation model is solved. This section analyse the amount of time spent by each component of the column generation algorithm. Tables 6.6, 6.7, 6.8, and 6.9 show the percentage of the total running time (including B&P) spent on each part of the algorithm. Each line is the mean of nine instances of the same size, Column **master** is the time to solve the restricted master problem with CPLEX, Column **add** is the time to add columns to the model, Column **pricing** is the modified Dijkstra's algorithm time, and **aux** is the time to construct the auxiliary graph used in the pricing problem. In the last line, the mean of all tests is shown. Due to measuring inaccuracies, results with small

Table 6.5: Results for transit-stub instances.

	Low Demand							
	Branch & Price				CPLEX			
$\lvert V \rvert$	#NF	time_int	time(s)	cost				
2	0.00	0.01	0.01	16.33	0.00	0.11	0.26	16.33
4	0.00	0.06	0.30	25.00	0.00	3.75	30.41	25.00
6	0.00	0.38	5.86	62.67	0.20	38.56	951.26	62.67
8	0.00	0.72	40.04	76.67	8.20	253.12	3492.03	82.36
10	1.40	3.75	925.04	123.62	9.00	845.33	3600.00	166.20
12	3.40	7.11	1616.76	134.35	9.00	1355.77	3600.00	198.94
14	6.40	17.38	2655.33	169.02	9.00	1774.96	3600.00	253.97
16	8.20	86.86	3401.50	263.98	9.00	1766.10	3600.00	390.89
18	7.20	68.49	3163.51	290.49	9.00	2519.80	3600.00	413.33
	High Demand							
	Branch & Price				CPLEX			
$\lvert V \rvert$	#NF	time_int	time(s)	cost	#NF	time_int	time(s)	cost
2	0.00	0.01	0.01	19.67	0.00	0.11	0.24	19.67
4	0.00	0.14	0.55	56.33	0.00	3.30	16.54	56.33
6	1.80	2.16	836.59	161.99	0.40	40.81	445.63	162.17
8	0.00	1.28	9.82	162.31	3.20	330.52	1894.55	164.71
10	0.60	3.40	648.30	208.78	6.80	1001.89	2957.12	267.28
12	8.00	48.45	3409.63	347.29	7.60	1553.02	3208.40	394.55
14	5.40	215.75	2516.70	453.62	6.60	1499.95	2861.61	561.64
16	6.40	525.69	2871.73	633.02	6.60	1936.91	3146.33	801.06
18	8.20	500.51	3362.14	641.82	8.00	1163.51	3254.62	666.50

Source: from author (2015).

instances are omitted. Times in those columns do not sum up to 100% because they do not subsume the time to construct the model and to select nodes of the Branch & Price.

The majority of the B&P running time is spent solving the pricing problem (constructing the auxiliary graph and finding the path with the minimum reduced cost). In Dense Random Instances, the pricing problem takes up most of the running time. Dijkstra's modified algorithm occupies more time in those instances because of the large number of links in dense graphs.

These results suggest that the algorithm bottleneck is the pricing problem. Two strategies could be taken to improve the performance of B&P: a single column could be generated at each iteration of CG. Thus reducing the number of independent pricing problems solved. Another strategy would be to use a dynamic algorithm for searching minimum cost paths, such algorithms use information of previously found paths to find new paths instead of recalculating them from scratch.

Table 6.6: Column Generation Components Running Time - Sparse Instances

		Low Demand				High Demand			
$\|S\|$	$\|V\|$	master(%)	add(%)	pricing(%)	aux(%)	master(%)	add(%)	pricing(%)	aux(%)
40	4	0.00	0.00	0.00	14.00	0.00	0.00	0.00	18.20
40	6	0.00	0.00	0.00	53.20	0.82	0.00	0.00	41.00
40	8	2.72	0.00	0.78	56.30	5.73	0.00	14.30	39.60
40	10	50.20	0.00	19.20	26.00	4.18	0.00	0.00	51.90
40	12	6.02	0.00	0.49	71.50	45.10	0.00	4.19	38.20
40	14	5.40	0.00	1.02	71.40	68.20	0.00	5.99	22.30
40	16	39.90	0.00	21.70	35.20	50.60	0.00	10.50	32.90
40	18	35.40	0.00	3.29	53.80	66.90	0.00	5.84	25.10
80	4	0.89	0.00	66.10	15.20	3.45	0.00	0.00	58.60
80	6	1.08	0.00	64.90	17.90	2.12	0.00	55.80	22.40
80	8	8.26	0.00	56.80	27.40	14.50	0.00	31.00	44.40
80	10	24.40	0.00	60.50	12.20	11.00	0.00	33.10	45.00
80	12	4.86	0.00	67.40	21.90	15.90	0.00	50.60	28.90
80	14	6.71	0.00	62.60	26.20	29.10	0.00	43.40	25.20
80	16	20.70	0.00	63.20	14.30	29.00	0.00	40.40	27.60
80	18	9.15	0.00	68.30	19.80	29.10	0.00	49.60	19.40
120	4	7.50	0.00	74.20	7.74	10.60	0.00	69.20	10.60
120	6	5.43	0.00	81.90	7.78	6.31	0.00	73.80	12.50
120	8	5.62	0.00	80.50	9.72	8.89	0.00	75.20	11.20
120	10	10.90	0.03	79.60	7.37	7.66	0.00	73.70	13.60
120	12	4.63	0.00	85.60	7.58	7.57	0.00	81.70	8.76
120	14	5.17	0.00	83.60	8.83	14.50	0.00	76.00	8.19
120	16	6.56	0.00	85.00	7.09	10.80	0.00	75.30	11.70
120	18	4.98	0.14	86.20	7.43	13.50	0.04	77.00	8.25
160	4	6.65	3.40	80.90	5.39	5.95	0.00	77.90	7.86
160	6	4.63	1.61	86.40	4.73	5.73	0.00	83.20	6.28
160	8	5.18	0.51	85.80	5.78	8.82	0.00	78.80	8.83
160	10	5.59	0.53	88.40	4.27	7.07	0.00	82.30	7.62
160	12	3.56	0.32	90.50	4.40	5.64	0.01	85.80	6.62
160	14	3.72	0.43	90.40	4.31	6.39	0.13	87.40	5.23
160	16	4.64	0.13	91.00	3.72	8.65	0.00	82.80	6.99
160	18	3.64	0.20	91.60	3.97	9.80	0.20	83.00	6.23
200	4	3.24	3.42	89.10	2.14	8.15	0.00	81.60	4.49
200	6	3.36	2.25	90.30	2.63	5.27	0.40	88.40	3.96
200	8	4.01	1.54	90.00	3.14	7.31	0.00	84.80	5.27
200	10	3.07	0.74	93.30	2.45	6.73	0.75	85.90	4.92
200	12	2.64	0.58	93.90	2.25	4.14	0.12	91.80	3.20
200	14	2.68	0.87	93.80	2.27	5.44	0.15	90.10	3.77
200	16	2.35	0.26	95.00	2.08	5.33	0.11	89.90	3.97
200	18	2.59	0.34	94.60	2.21	6.41	0.30	88.80	3.95
mean		6.07	0.32	49.22	12.14	10.41	0.04	42.76	13.23

Source: from author (2015).

Table 6.7: Column Generation Components Running Time - Dense Instances

		Low Demand				High Demand			
$\|S\|$	$\|V\|$	master(%)	add(%)	pricing(%)	aux(%)	master(%)	add(%)	pricing(%)	aux(%)
40	4	0.00	0.00	0.00	57.10	0.00	0.00	0.00	0.00
40	6	0.25	0.00	0.38	67.70	0.60	0.00	0.20	61.40
40	8	1.78	0.00	0.32	69.90	2.06	0.00	0.26	52.30
40	10	16.50	0.00	3.50	68.90	12.70	0.00	0.64	56.40
40	12	3.45	0.00	0.58	79.60	20.40	0.00	10.30	59.10
40	14	8.02	0.00	0.35	79.50	52.90	0.00	10.40	33.10
40	16	23.10	0.00	8.65	63.50	44.10	0.00	11.40	38.60
40	18	12.00	0.00	4.18	77.10	58.00	0.00	9.43	30.00
80	4	0.94	0.00	77.80	11.30	0.99	0.00	40.60	27.70
80	6	5.47	0.00	75.80	12.60	8.66	0.00	69.30	15.40
80	8	4.33	0.00	75.90	14.50	9.90	0.00	57.50	22.00
80	10	7.41	0.00	77.00	12.60	9.44	0.00	63.60	19.90
80	12	4.96	0.00	80.00	12.20	8.91	0.00	66.30	20.80
80	14	4.89	0.00	79.50	12.80	20.80	0.00	64.80	12.60
80	16	11.50	0.00	77.10	9.88	13.20	0.00	62.80	20.90
80	18	6.08	0.00	79.50	12.50	22.20	0.00	62.10	13.80
120	4	3.95	0.00	82.90	5.76	6.25	0.00	74.00	8.88
120	6	4.40	0.09	86.80	5.20	5.40	0.00	83.90	7.11
120	8	5.25	0.00	84.90	6.39	6.90	0.00	77.20	10.90
120	10	5.92	0.33	87.10	5.28	6.63	0.00	81.20	8.73
120	12	4.08	0.00	89.30	4.98	6.37	0.00	85.20	6.64
120	14	3.88	0.05	89.60	4.95	7.05	0.01	85.40	6.29
120	16	4.36	0.16	89.70	4.81	6.75	0.00	84.00	7.61
120	18	4.25	0.10	90.00	4.74	8.70	0.07	83.60	6.85
160	4	2.67	1.29	91.80	1.82	7.75	0.00	83.30	4.60
160	6	2.87	0.43	93.00	2.22	5.12	0.13	88.90	3.48
160	8	3.39	0.70	91.80	2.75	6.98	0.00	85.50	4.91
160	10	3.92	0.49	92.60	2.46	4.99	0.49	89.40	3.78
160	12	2.89	0.30	93.80	2.30	4.89	0.01	90.50	3.76
160	14	3.16	0.51	93.50	2.44	5.33	0.15	90.20	3.65
160	16	2.92	0.23	94.20	2.31	5.33	0.03	89.70	4.16
160	18	2.95	0.19	94.20	2.26	6.33	0.26	88.90	3.97
200	4	2.39	1.68	93.00	1.42	6.97	0.00	86.20	3.31
200	6	3.59	1.58	91.60	2.08	3.60	0.51	92.40	2.28
200	8	2.48	1.10	94.00	1.69	5.71	0.38	89.00	3.41
200	10	2.68	0.62	94.60	1.81	4.72	0.84	90.80	3.01
200	12	2.32	0.57	95.30	1.50	3.63	0.17	93.40	2.22
200	14	2.29	0.67	95.20	1.54	3.73	0.20	93.30	2.43
200	16	2.14	0.27	95.90	1.51	4.26	0.13	92.50	2.62
200	18	2.36	0.09	95.80	1.55	4.83	0.37	91.60	2.79
mean		3.55	0.21	52.61	13.62	7.83	0.07	48.51	11.14

Source: from author (2015).

Table 6.8: Column Generation Components Running Time - Hierarchical Instances

		Low Demand				High Demand			
$\|S\|$	$\|V\|$	master(%)	add(%)	pricing(%)	aux(%)	master(%)	add(%)	pricing(%)	aux(%)
125	4	0.00	0.00	1.28	32.10	1.55	0.00	17.10	30.20
125	6	3.08	0.00	46.90	28.00	3.49	0.00	16.30	39.10
125	8	24.10	0.00	18.90	38.40	15.90	0.00	35.70	30.70
125	10	46.90	0.00	39.70	11.50	25.70	0.00	28.90	31.40
125	12	23.40	0.00	25.40	42.80	20.70	0.00	61.30	15.20
125	14	25.20	0.00	61.20	11.80	26.80	0.00	37.80	30.40
125	16	24.30	0.00	56.70	17.40	47.80	0.00	30.80	19.00
125	18	38.50	0.00	42.30	17.40	31.50	0.00	14.10	43.60
150	4	0.69	0.00	0.00	43.20	3.78	0.00	27.00	31.60
150	6	8.74	0.00	29.20	40.90	23.10	0.00	8.28	39.00
150	8	18.90	0.00	16.60	50.30	22.70	0.00	37.10	30.20
150	10	56.50	0.01	28.70	12.90	22.20	0.00	7.27	52.00
150	12	12.60	0.00	16.00	60.50	18.00	0.00	67.20	12.70
150	14	36.40	0.01	38.00	22.70	32.70	0.00	50.60	15.00
150	16	41.40	0.00	34.30	22.50	28.00	0.00	56.60	13.50
150	18	39.60	0.02	39.20	19.50	40.70	0.00	26.80	28.80
175	4	0.97	0.00	0.00	52.40	1.56	0.00	60.30	21.40
175	6	15.30	0.00	23.30	46.50	12.40	0.00	21.40	44.60
175	8	15.10	0.00	8.26	63.60	19.60	0.00	28.90	36.20
175	10	58.70	0.00	24.70	14.50	32.50	0.00	11.00	43.90
175	12	18.30	0.00	44.60	31.50	12.70	0.00	73.30	12.30
175	14	20.30	0.03	25.20	48.10	24.00	0.00	59.80	14.70
175	16	16.30	0.01	65.80	16.70	20.70	0.00	66.80	11.10
175	18	30.70	0.01	52.70	15.00	36.30	0.00	46.80	15.30
200	4	1.33	0.00	13.40	41.70	2.02	0.00	48.00	27.00
200	6	18.20	0.00	16.90	49.80	15.20	0.00	35.00	36.90
200	8	7.62	0.00	54.00	31.00	16.80	0.00	53.60	23.40
200	10	51.20	0.02	33.30	13.50	26.90	0.00	30.20	34.30
200	12	19.50	0.00	44.70	30.40	14.70	0.00	71.40	12.10
200	14	17.20	0.01	40.30	37.20	24.60	0.00	58.80	15.20
200	16	31.10	0.02	48.00	19.20	25.00	0.07	60.40	13.00
200	18	47.50	0.01	35.80	15.00	23.60	0.00	38.20	34.20
225	4	0.81	0.00	11.40	56.90	3.81	0.00	39.40	28.80
225	6	18.00	0.00	37.90	30.60	9.28	0.00	59.30	23.70
225	8	10.70	0.00	47.10	32.30	12.30	0.00	62.50	19.00
225	10	51.80	0.03	36.00	10.20	18.80	0.15	27.10	44.90
225	12	19.00	0.00	54.90	21.50	13.00	0.00	73.10	12.30
225	14	22.80	0.00	53.10	20.10	25.60	0.00	56.10	16.90
225	16	14.50	0.00	74.90	9.37	20.60	0.08	66.30	11.60
225	18	31.90	0.04	54.70	11.80	24.80	0.00	58.30	15.50
250	4	9.83	0.00	33.40	29.40	2.56	0.00	0.00	62.40
250	6	12.80	0.00	48.90	27.70	11.00	0.00	46.30	32.70
250	8	14.10	0.00	40.20	36.90	14.20	0.00	48.70	29.90
250	10	41.80	0.00	43.20	13.10	18.80	0.00	28.90	43.10
250	12	10.70	0.00	61.10	24.60	7.18	0.00	83.40	8.27
250	14	11.20	0.02	78.80	8.66	24.70	0.00	58.30	15.60
250	16	28.00	0.00	50.00	20.30	22.40	0.05	62.70	13.40
250	18	32.00	0.02	48.60	17.70	22.00	0.00	61.80	14.90
mean		20.36	0.00	33.32	25.35	17.12	0.01	39.24	23.17

Source: from author (2015).

Table 6.9: Column Generation Components Running Time - Transit-stub Instances

		Low Demand				High Demand			
$\|S\|$	$\|V\|$	master(%)	add(%)	pricing(%)	aux(%)	master(%)	add(%)	pricing(%)	aux(%)
90	4	0.00	0.00	0.00	38.40	0.53	0.00	38.50	27.30
90	6	2.64	0.00	30.70	37.40	0.92	0.00	20.60	39.60
90	8	1.81	0.00	6.60	59.50	6.49	0.00	18.20	41.70
90	10	41.70	0.00	27.30	26.20	7.57	0.00	13.60	47.50
90	12	7.06	0.00	6.97	66.20	22.30	0.00	50.10	23.80
90	14	17.10	0.00	35.90	39.50	48.10	0.00	25.50	22.90
90	16	26.90	0.00	49.20	21.80	28.50	0.00	41.70	26.00
90	18	37.50	0.00	35.90	23.90	48.40	0.00	27.80	21.60
135	4	6.25	0.00	12.50	37.50	0.56	0.00	57.90	19.10
135	6	1.91	0.00	22.10	48.50	2.10	0.00	1.20	49.40
135	8	15.00	0.00	40.70	33.00	10.70	0.00	47.50	29.00
135	10	31.50	0.00	52.20	14.30	18.10	0.00	24.80	42.40
135	12	12.80	0.00	41.70	39.30	16.80	0.00	65.20	15.60
135	14	19.50	0.09	39.60	35.90	34.70	0.00	40.60	22.10
135	16	23.10	0.00	53.80	21.40	21.80	0.00	55.80	19.80
135	18	25.40	0.00	50.60	22.00	37.50	0.00	40.30	20.40
180	4	5.41	0.00	10.80	48.60	1.82	0.00	41.00	29.60
180	6	8.24	0.00	4.42	64.30	3.56	0.00	36.20	40.80
180	8	10.90	0.00	38.30	41.00	18.70	0.00	26.50	39.80
180	10	19.00	0.00	24.20	50.40	16.20	0.00	10.40	58.10
180	12	9.83	0.00	39.00	42.30	11.60	0.00	71.50	15.10
180	14	25.50	0.00	44.30	25.90	23.90	0.00	54.10	20.00
180	16	30.10	0.03	47.20	20.90	17.70	0.00	67.60	13.20
180	18	16.40	0.00	34.80	44.60	25.50	0.00	58.20	14.90
225	4	4.88	0.00	58.50	19.50	3.12	0.00	0.00	62.50
225	6	9.43	0.00	42.50	36.80	14.90	0.00	6.38	68.10
225	8	11.40	0.00	45.30	31.80	13.60	0.00	25.40	48.00
225	10	11.60	0.00	53.40	28.50	17.80	0.00	15.00	54.70
225	12	10.50	0.00	50.60	32.30	9.10	0.00	77.20	12.20
225	14	12.60	0.00	55.60	26.80	25.20	0.00	49.70	23.20
225	16	22.90	0.04	61.20	14.30	18.70	0.09	61.30	18.00
225	18	11.70	0.02	62.40	22.70	23.60	0.00	59.70	15.40
270	4	8.82	0.00	47.10	29.40	3.00	0.00	71.20	13.90
270	6	11.80	0.00	42.00	35.00	13.20	0.00	28.50	45.90
270	8	10.20	0.00	38.10	40.90	6.21	0.00	71.50	17.10
270	10	31.70	0.00	54.90	11.70	15.60	0.00	34.60	41.10
270	12	8.27	0.00	59.40	28.50	6.84	0.00	82.70	9.30
270	14	31.00	0.00	53.30	13.90	16.30	0.00	65.80	16.50
270	16	10.00	0.15	75.30	13.50	12.60	0.06	74.90	11.40
270	18	12.50	0.03	68.40	17.60	14.90	0.00	72.10	11.80
mean		13.66	0.01	35.93	29.02	14.19	0.00	38.46	25.97

Source: from author (2015).

7 CONCLUDING REMARKS

This work presented a new Branch & Price algorithm for the Single-Path Virtual Network Embedding Problem. It characterized the complexity of this problem by showing that finding a feasible solution for the VNEP is NP-Hard. The master and pricing problem of a column generation algorithm were described in details and an algorithm to solve the pricing problem was thoroughly presented. Implementation details were presented with alternative algorithms for node selection and branching. Different implementations of B&P were tested and the best version was compared with a compact model solved with CPLEX. Algorithms were extensively tested by using four different topologies of different sizes, each with two settings of demands.

Results have shown that B&P has a better performance and is able to solve larger instances than the standard model solved with CPLEX. The presented B&P is able to find optimal solutions for most instances with virtual networks of up to 8 nodes. For example, for dense random instances with high demands, B&P is able to solve six of the nine instances of 14 virtual nodes and 200 physical nodes, while CPLEX is able to solve only one instance of this size. B&P can also find integer solutions faster than CPLEX when one exists. Thus the presented algorithm can be used in practice for small instances, can obtain good solutions for large instances, and can be used to evaluate heuristic algorithms since it is able to solve large instances in a reasonable time.

By exploiting the problem structure, B&P is able to obtain good heuristic solutions quickly. Those integer solutions provide high quality upper bounds that can prune large portions of the search tree, improving running time and reducing memory usage. However, there is still room for improvement in the B&P implementation. The algorithm can be extended to a Branch & Price & Cut by adapting cover inequalities presented in (BARNHART; HANE; VANCE, 2000) for the Multicommodity Flow Problem. Such cuts could reduce the number of nodes to explore, improving both the performance of the algorithm and the size of instances that the algorithm is able to solve. Additionally, an adaption of the model to map multiple virtual networks simultaneously could improve resource utilization or other metrics such as revenue. To implement this, a possible approach is that of (GUERZONI et al., 2014) of mapping virtual networks arriving in a window of time and allowing partial solutions to these networks.

REFERENCES

ALKMIM, G.; BATISTA, D.; FONSECA, N. da. Mapping virtual networks onto substrate networks. **Journal of Internet Services and Applications**, Heidelberg, v. 4, n. 3, p. 1–15, jan. 2013.

ALKMIM, G. P.; BATISTA, D. M.; FONSECA, N. L. S. da. Optimal Mapping of Virtual Networks. In: **Global Telecommunications Conference**. Houston: [s.n.], 2011. p. 1–6.

ANDERSON, T. et al. Overcoming the Internet impasse through virtualization. **Computer**, Los Alamitos, v. 38, n. 4, p. 34–41, abr. 2005.

ANDERSON, T.; REITER, M. K. **GENI: Global Environment for Network Innovations Distributed Services Working Group**. 2006. Available from Internet: <http://www.geni.net/>. Accessed in: 5 Jan. 2015.

BARNHART, C.; HANE, C. A.; VANCE, P. H. Using Branch-and-Price-and-Cut to Solve Origin-Destination Integer Multicommodity Flow Problems. **Operations Research**, Maryland, v. 48, n. 2, p. 318–326, mar. 2000.

BARNHART, C. et al. Branch-and-Price: Column Generation for Solving Huge Integer Programs. **Operations Research**, Maryland, v. 46, n. 3, p. 316–329, may 1998.

BAYS, L. R. et al. Security-aware optimal resource allocation for virtual network embedding. In: **Network and service management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualiztion management (svm)**. Las Vegas: [s.n.], 2012. p. 378–384.

BELBEKKOUCHE, A.; HASAN, M. M.; KARMOUCH, A. Resource Discovery and Allocation in Network Virtualization. **Communications Surveys Tutorials, IEEE**, Winnipeg, v. 14, n. 4, p. 1114–1128, feb. 2012.

BOTERO, J. F. et al. Energy Efficient Virtual Network Embedding. **Communications Letters, IEEE**, Thessaloniki, v. 16, n. 5, p. 756–759, may 2012.

CHOWDHURY, M.; RAHMAN, M. R.; BOUTABA, R. ViNEYard: Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping. **IEEE/ACM Transactions on Networking**, Champaign, v. 20, n. 1, p. 206–219, feb. 2012.

CHOWDHURY, N. M. M. K.; BOUTABA, R. A survey of network virtualization. **Computer Networks**, New York, NY, USA, v. 54, n. 5, p. 862–876, abr. 2010.

CHOWDHURY, N. M. M. K.; RAHMAN, M. R.; BOUTABA, R. Virtual Network Embedding with Coordinated Node and Link Mapping. In: **IEEE International Conference on Computer Communications.** Rio de Janeiro: IEEE, 2009. p. 783–791.

CHUN, B. et al. PlanetLab: An Overlay Testbed for Broad-coverage Services. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 33, n. 3, p. 3–12, jul. 2003.

DANTZIG, G. B.; WOLFE, P. Decomposition Principle for Linear Programs. **Operations Research**, Maryland, v. 8, n. 1, p. 101–111, jan. 1960.

FAN, J.; AMMAR, M. H. Dynamic Topology Configuration in Service Overlay Networks: A Study of Reconfiguration Policies. In: **IEEE International Conference on Computer Communications.** Barcelona, Spain: [s.n.], 2006. p. 1–12.

FEAMSTER, N.; GAO, L.; REXFORD, J. How to lease the internet in your spare time. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 37, n. 1, p. 61–64, jan. 2007.

FISCHER, A. et al. ALEVIN - A Framework to Develop, Compare, and Analyze Virtual Network Embedding Algorithms Andreas. **Electronic Communications of the EASST**, Berlin, v. 37, p. 1–12, mar. 2011.

FISCHER, A. et al. Virtual Network Embedding: A Survey. **Communications Surveys Tutorials, IEEE**, Winnipeg, v. 15, n. 4, p. 1888–1906, feb. 2013.

GOLDBERG, A. et al. An Implementation of a Combinatorial Approximation Algorithm for Minimum-Cost Multicommodity Flow. In: BIXBY, R.; BOYD, E.; RÍOS-MERCADO, R. (Ed.). **Integer Programming and Combinatorial Optimization**. Verlang: Springer Berlin Heidelberg, 1998, (Lecture Notes in Computer Science, v. 1412). p. 338–352.

GUERZONI, R. et al. A novel approach to virtual networks embedding for SDN management and orchestration. In: **Network Operations and Management Symposium**. Krakow: [s.n.], 2014. p. 1–7.

GUPTA, A. et al. Provisioning a Virtual Private Network: A Network Design Problem for Multicommodity Flow. In: **Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing**. New York, NY, USA: ACM, 2001. p. 389–398.

HOUIDI, I. et al. Virtual network provisioning across multiple substrate networks. **Computer Networks**, New York, NY, USA, v. 55, n. 4, p. 1011–1023, mar. 2011.

HU, Q.; WANG, Y.; CAO, X. Resolve the virtual network embedding problem: A column generation approach. In: **INFOCOM, 2013 Proceedings IEEE**. Turin: [s.n.], 2013. p. 410–414.

INFüHR, J.; RAIDL, G. Introducing the Virtual Network Mapping Problem with Delay, Routing and Location Constraints. In: PAHL, J.; REINERS, T.; VOSS, S. (Ed.). **Network Optimization**. Verlang: Springer Berlin Heidelberg, 2011, (Lecture Notes in Computer Science, v. 6701). p. 105–117.

JARRAY, A.; KARMOUCH, A. Column generation approach for one-shot virtual network embedding. In: **Globecom Workshops**. Anaheim, CA: [s.n.], 2012. p. 863–868.

KLEINBERG, J. M. Single-Source Unsplittable Flow. In: **37th Annual Symposium on Foundations of Computer Science**. Washington, DC, USA: [s.n.], 1996. p. 68–77.

LISCHKA, J.; KARL, H. A virtual network mapping algorithm based on subgraph isomorphism detection. In: **1st ACM workshop on Virtualized infrastructure systems and architectures**. New York, NY, USA: ACM, 2009. p. 81–88.

LUBBECKE, M.; DESROSIERS, J. Selected Topics in Column Generation. **Operations Research**, Maryland, v. 53, p. 1007–1023, dec. 2005.

PAGES, A. et al. Strategies for Virtual Optical Network Allocation. **Communications Letters, IEEE**, Thessaloniki, v. 16, n. 2, p. 268–271, feb. 2012.

RICCI, R.; ALFELD, C.; LEPREAU, J. A Solver for the Network Testbed Mapping Problem. **ACM SIGCOMM Computer Communication Review**, ACM, New York, NY, USA, v. 33, n. 2, p. 65–81, 2003.

SHAMSI, J.; BROCKMEYER, M. Efficient and dependable overlay networks. In: **IEEE International Symposium on Parallel and Distributed Processing**. Miami, FL: [s.n.], 2008. p. 1–8.

TRINH, T.; ESAKI, H.; ASWAKUL, C. Quality of service using careful overbooking for optimal virtual network resource allocation. In: **8th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology**. Khon Kaen: [s.n.], 2011. p. 296–299.

YU, M. et al. Rethinking virtual network embedding: substrate support for path splitting and migration. **ACM SIGCOMM Computer Communication Review**, Acm, New York, NY, USA, v. 38, n. 2, p. 17–29, abr. 2008.

ZEGURA, E. W.; CALVERT, K. L.; BHATTACHARJEE, S. How to model an internetwork. In: **15th Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation.** San Francisco, CA: [s.n.], 1996. v. 2, p. 594–602 vol.2.

ZHU, Y.; AMMAR, M. Algorithms for Assigning Substrate Network Resources to Virtual Network Components. In: **INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings**. Barcelona, Spain: [s.n.], 2006. p. 1–12.

AppendixA FULL RESULTS

Table A.1: Sparse Random Instance - Low demands

		Branch & Price				CPLEX			
$\|S\|$	$\|V\|$	#NF	time_int	time(s)	cost	#NF	time_int	time(s)	cost
20	2	0	0.00	0.00	16.33	0	0.01	0.01	16.33
20	4	0	0.01	0.02	25.22	0	0.06	0.29	25.22
20	6	0	0.02	0.13	65.67	0	0.20	1.99	65.67
20	8	0	0.04	0.82	83.78	0	0.43	9.42	83.78
20	10	0	0.11	6.13	140.78	1	3.94	565.91	140.78
20	12	0	0.42	32.24	142.00	2	6.06	980.77	142.33
20	14	0	2.70	197.09	176.33	1	16.83	741.24	177.11
20	16	3	69.37	1465.91	303.00	1	22.56	704.30	304.88
20	18	3	397.88	1200.07	298	0	67.416	72.19	290
40	2	0	0.01	0.01	16.33	0	0.02	0.04	16.33
40	4	0	0.01	0.06	25.00	0	0.20	0.70	25.00
40	6	0	0.02	0.39	62.67	0	0.85	15.87	62.67
40	8	0	0.05	1.47	76.67	3	2.37	1993.26	76.67
40	10	0	0.06	198.00	121.11	8	4.61	3461.04	122.78
40	12	0	0.20	78.63	126.00	9	6.79	3600.00	127.67
40	14	1	0.11	701.76	146.56	9	11.39	3600.00	153.44
40	16	4	0.16	1837.95	215.22	9	417.85	3600.00	227.56
40	18	1	0.18	1120.02	221.11	9	371.33	3600.00	283.89
80	2	0	0.01	0.01	16.33	0	0.03	0.07	16.33
80	4	0	0.04	0.39	25.00	0	0.57	4.69	25.00
80	6	0	0.09	4.20	62.67	0	4.97	288.35	62.67
80	8	0	0.12	11.64	76.67	9	15.43	3600.00	76.67
80	10	0	0.28	255.95	120.67	9	32.48	3600.00	126.22
80	12	1	0.60	442.74	126.56	9	91.15	3600.00	134.56
80	14	0	0.53	1015.49	146.00	9	920.69	3600.00	185.00
80	16	4	0.40	1990.83	212.11	9	1182.38	3600.00	304.22
80	18	4	2.53	2092.46	223.67	9	943.55	3600.00	309.50
120	2	0	0.03	0.03	16.33	0	0.05	0.12	16.33
120	4	0	0.12	1.97	25.00	0	1.19	16.48	25.00
120	6	0	0.21	27.30	62.67	0	9.85	1979.18	62.67
120	8	0	0.36	55.61	76.67	9	49.50	3600.00	76.67
120	10	0	0.61	696.93	120.67	9	259.51	3600.00	127.33
120	12	0	1.49	372.39	126.00	9	739.56	3600.00	141.56
120	14	2	1.91	1735.27	148.33	9	784.18	3600.00	170.62
120	16	2	3.10	1706.44	212.44	9	1553.34	3600.00	320.14
120	18	2	4.27	2026.57	222.11	9	2180.70	3600.00	290.62
160	2	0	0.07	0.07	16.33	0	0.09	0.20	16.33
160	4	0	0.26	8.99	25.00	0	2.16	71.25	25.00
160	6	0	0.45	78.36	62.67	9	24.79	3600.00	62.67
160	8	0	0.79	191.68	76.67	9	116.07	3600.00	76.67
160	10	1	1.39	1527.90	121.56	9	811.94	3600.00	134.56
160	12	1	2.53	891.74	126.56	9	1431.75	3600.00	194.78
160	14	1	3.04	1841.81	148.00	9	1655.31	3600.00	177.75
160	16	5	5.17	2724.22	229.44	9	2587.91	3600.00	334.50
160	18	6	27.26	3233.96	239.78	9	3329.04	3600.00	458
200	2	0	0.15	0.15	16.33	0	0.15	0.32	16.33
200	4	0	0.56	6.88	25.00	0	3.43	156.03	25.00
200	6	0	1.77	218.34	62.67	9	48.95	3600.00	62.67
200	8	0	4.53	556.39	76.67	9	556.12	3600.00	77.00
200	10	2	3.55	2371.14	121.78	9	1065.41	3600.00	143.89
200	12	3	5.79	2351.89	128.56	9	2005.06	3600.00	184.38
200	14	7	6.91	3316.64	162.44	9	2990.62	3600.00	189.50
200	16	8	12.35	3543.59	236.44	9	-	3600.00	-
200	18	9	8.99	3600.00	272.44	9	-	3600.00	-

Source: from author (2015).

Table A.2: Sparse Random Instances - High Demands

		Branch & Price				CPLEX			
$\|S\|$	$\|V\|$	#NF	time_int	time(s)	cost	#NF	time_int	time(s)	cost
20	2	0	0.00	0.00	19.89	0	0.01	0.01	19.89
20	4	0	0.01	0.01	74.11	0	0.05	0.23	74.11
20	6	0	0.01	0.06	154.17	0	0.14	0.57	154.17
20	8	0	0.02	0.11	224.50	0	0.52	0.75	224.50
20	10	0	0.06	0.57	250.20	0	1.68	2.19	250.20
20	12	0	0.38	1.88	298.50	0	2.86	1.56	298.50
20	14	0	2.045	2.86	461	0	11.604	1.69	461
20	16	0	-	0.04	-	0	-	0.37	-
20	18	0	-	0.04	-	0	-	0.49	-
40	2	0	0.01	0.01	19.67	0	0.01	0.03	19.67
40	4	0	0.01	0.07	57.67	0	0.18	0.50	57.67
40	6	0	0.01	7.06	153.67	0	0.58	45.72	153.67
40	8	0	0.02	0.75	160.56	0	0.97	53.59	160.56
40	10	0	0.04	12.66	203.22	0	5.18	436.33	203.22
40	12	0	0.05	508.49	287.67	2	9.25	1177.05	289.67
40	14	3	0.27	1245.66	453.00	3	34.96	1914.15	446.56
40	16	5	0.42	2259.53	596.33	5	70.18	2714.59	620.44
40	18	6	3.90	3123.82	680.22	8	87.59	3381.96	661.89
80	2	0	0.01	0.01	19.67	0	0.03	0.08	19.67
80	4	0	0.05	0.34	56.33	0	0.46	2.42	56.33
80	6	0	0.05	124.12	134.22	0	2.91	98.17	134.22
80	8	0	0.07	3.79	159.67	0	8.02	1334.36	159.67
80	10	0	0.16	19.63	191.00	9	21.66	3600.00	194.22
80	12	3	0.25	1393.46	280.78	9	28.56	3600.00	305.89
80	14	3	0.28	1485.60	419.11	9	425.16	3600.00	594.22
80	16	6	0.40	2840.44	566.11	9	1312.95	3600.00	944.44
80	18	6	0.68	2537.17	609.67	9	1878.77	3600.00	1053.78
120	2	0	0.02	0.02	19.67	0	0.05	0.12	19.67
120	4	0	0.07	1.89	56.33	0	1.15	10.75	56.33
120	6	0	0.13	141.60	130.67	0	7.55	596.01	130.67
120	8	0	0.19	18.34	159.67	9	29.01	3600.00	159.67
120	10	0	0.33	118.74	191.00	9	79.92	3600.00	204.89
120	12	4	0.51	2124.60	278.00	9	1035.93	3600.00	356.67
120	14	3	0.64	1738.19	404.67	9	1616.20	3600.00	598.50
120	16	7	0.75	3014.10	578.11	9	1541.42	3600.00	932.57
120	18	6	1.59	2689.84	613.22	9	2838.64	3600.00	1475.67
160	2	0	0.04	0.04	19.67	0	0.09	0.21	19.67
160	4	0	0.17	6.49	56.33	0	1.99	28.45	56.33
160	6	0	0.36	292.52	130.67	4	17.57	2922.97	130.67
160	8	0	1.33	94.32	159.67	9	122.95	3600.00	159.67
160	10	0	1.60	242.19	191.00	9	716.96	3600.00	211.11
160	12	3	0.98	2180.04	277.11	9	1478.29	3600.00	443.50
160	14	3	2.18	1983.75	404.22	9	1535.28	3600.00	558.50
160	16	8	1.92	3371.85	587.89	9	1942.46	3600.00	698.80
160	18	7	3.78	3038.77	641.67	9	2916.82	3600.00	701.67
200	2	0	0.07	0.07	19.09	0	0.16	0.35	19.09
200	4	0	0.32	15.20	55.90	0	2.92	62.91	55.90
200	6	0	0.63	392.74	124.10	9	34.91	3599.52	124.10
200	8	0	0.87	147.57	159.00	9	215.92	3600.00	159.00
200	10	0	1.30	610.86	193.90	9	1273.32	3600.00	255.20
200	12	5	2.28	2664.41	282.90	9	2401.21	3600.00	550.90
200	14	5	4.64	2891.79	418.30	9	2749.34	3600.00	558.60
200	16	8	6.32	3520.17	600.33	9	2984.89	3600.00	744.50
200	18	8	5.12	3553.89	651.67	9	-	3600.00	-

Source: from author (2015).

Table A.3: Dense Random Instances - Low Demands

		Branch & Price				CPLEX			
$\|S\|$	$\|V\|$	#NF	time_int	time(s)	cost	#NF	time_int	time(s)	cost
20	2	0	0.01	0.01	16.33	0	0.01	0.02	16.33
20	4	0	0.01	0.02	25.00	0	0.06	0.31	25.00
20	6	0	0.01	0.17	65.00	0	0.21	1.71	65.00
20	8	0	0.01	0.79	81.67	0	0.32	55.90	81.67
20	10	0	0.03	5.04	130.56	0	0.74	565.44	130.56
20	12	0	0.04	44.13	141.00	3	2.08	1763.58	141.33
20	14	0	0.04	110.10	164.22	3	3.55	1481.26	166.11
20	16	2	0.11	1112.67	253.33	2	12.04	1755.99	253.22
20	18	2	0.26	1323.39	242.00	6	5.61	2400.26	258.50
40	2	0	0.01	0.01	16.33	0	0.03	0.05	16.33
40	4	0	0.02	0.06	25.00	0	0.18	0.90	25.00
40	6	0	0.03	0.41	62.67	0	0.90	21.43	62.67
40	8	0	0.11	1.81	76.67	6	2.36	2780.74	76.67
40	10	0	0.10	18.07	120.67	8	5.88	3420.77	121.78
40	12	0	0.10	109.11	126.00	9	6.79	3600.00	126.33
40	14	0	0.20	47.50	146.00	9	32.44	3600.00	151.33
40	16	2	0.21	1178.12	211.78	9	93.98	3600.00	229.56
40	18	0	0.33	675.74	219.33	9	307.87	3600.00	272.11
80	2	0	0.02	0.02	16.33	0	0.03	0.06	16.33
80	4	0	0.09	0.67	25.00	0	0.50	8.83	25.00
80	6	0	0.15	7.34	62.67	0	5.13	508.01	62.67
80	8	0	0.27	21.62	76.67	9	16.14	3600.00	76.67
80	10	0	0.53	238.55	120.67	9	33.05	3600.00	121.56
80	12	0	0.56	93.00	126.00	9	195.71	3600.00	130.78
80	14	0	3.03	208.06	146.00	9	769.71	3600.00	170.22
80	16	2	1.76	1115.46	210.22	9	832.80	3600.00	253.56
80	18	0	1.15	1110.37	219.33	9	1114.13	3600.00	309.67
120	2	0	0.06	0.06	16.33	0	0.06	0.13	16.33
120	4	0	0.26	2.31	25.00	0	1.19	61.13	25.00
120	6	0	0.66	30.42	62.67	4	12.72	3014.61	62.67
120	8	0	1.07	83.87	76.67	9	53.40	3600.00	76.67
120	10	0	5.56	465.39	120.67	9	271.11	3600.00	124.67
120	12	0	2.64	463.21	126.00	9	791.47	3600.00	129.78
120	14	0	4.82	1429.16	146.00	9	759.32	3600.00	167.11
120	16	2	4.88	1840.23	213.44	9	1994.69	3600.00	286.00
120	18	1	4.20	1947.06	221.11	9	2144.90	3600.00	328.00
160	2	0	0.13	0.13	16.33	0	0.10	0.22	16.33
160	4	0	0.58	8.86	25.00	0	2.33	110.77	25.00
160	6	0	1.08	103.41	62.67	9	23.64	3600.00	62.67
160	8	0	2.30	300.78	76.67	9	213.32	3600.00	76.67
160	10	1	5.17	1648.07	120.89	9	1001.60	3600.00	131.22
160	12	0	4.13	1595.81	126.00	9	1489.38	3600.00	165.00
160	14	5	15.68	2824.43	152.33	9	2050.57	3600.00	187.75
160	16	5	14.08	3332.66	225.56	9	2844.99	3600.00	247.33
160	18	7	11.54	3463.64	237.33	9	-	3600.00	-
200	2	0	0.23	0.23	16.33	0	0.17	0.35	16.33
200	4	0	1.02	13.08	25.00	0	3.76	398.17	25.00
200	6	0	2.25	323.10	62.67	9	45.90	3600.00	62.67
200	8	0	3.07	820.89	76.67	9	569.53	3600.00	76.67
200	10	3	11.50	2139.78	122.00	9	1467.65	3600.00	137.78
200	12	4	9.65	2740.72	133.11	9	1951.19	3600.00	166.25
200	14	7	48.99	3483.86	168.11	9	2963.01	3600.00	158.75
200	16	9	27.83	3600.00	253.00	9	-	3600.00	-
200	18	9	114.26	3600.00	285.56	9	-	3600.00	-

Source: from author (2015).

Table A.4: Dense Random Instances - High Demands

		Branch & Price				CPLEX			
$\lvert S\rvert$	$\lvert V\rvert$	#NF	time_int	time(s)	cost	#NF	time_int	time(s)	cost
20	2	0	0.00	0.00	19.67	0	0.01	0.02	19.67
20	4	0	0.01	0.02	59.00	0	0.09	0.28	59.00
20	6	0	0.02	0.23	184.11	0	0.13	0.99	184.11
20	8	0	0.01	0.38	183.67	0	0.25	2.34	183.67
20	10	0	0.03	2.38	252.00	0	1.11	6.26	252.00
20	12	0	0.09	12.53	348.14	0	2.03	10.32	348.14
20	14	1	17.49	506.43	581.00	0	13.07	40.35	573.00
20	16	1	147.92	827.49	719.40	0	20.92	35.50	719.40
20	18	0	-	0.10	-	0	-	0.79	-
40	2	0	0.01	0.01	19.67	0	0.01	0.04	19.67
40	4	0	0.01	0.11	56.33	0	0.21	0.61	56.33
40	6	0	0.02	7.15	135.44	0	0.57	13.48	135.44
40	8	0	0.03	1.59	159.67	1	1.21	508.28	159.67
40	10	0	0.03	8.47	191.00	3	2.79	2748.24	191.00
40	12	1	0.09	776.44	272.44	9	3.49	3600.00	287.00
40	14	3	0.13	1301.14	418.89	9	16.68	3600.00	435.22
40	16	6	0.26	2472.04	546.56	9	33.87	3600.00	603.78
40	18	6	0.25	2440.22	599.44	9	93.90	3600.00	667.44
80	2	0	0.01	0.01	19.67	0	0.03	0.07	19.67
80	4	0	0.05	0.42	56.33	0	0.42	3.79	56.33
80	6	0	0.09	8.74	130.67	0	3.01	193.01	130.67
80	8	0	0.12	7.50	159.67	6	9.88	2818.23	159.67
80	10	0	0.23	26.20	191.00	9	13.04	3600.00	191.00
80	12	1	0.25	649.88	269.56	9	78.74	3600.00	280.78
80	14	3	0.43	1516.65	387.00	9	752.22	3600.00	496.56
80	16	5	0.68	2298.35	517.22	9	950.62	3600.00	703.67
80	18	6	0.87	2458.66	523.33	9	2107.12	3600.00	837.38
120	2	0	0.03	0.03	19.67	0	0.06	0.13	19.67
120	4	0	0.12	1.97	56.33	0	1.08	13.03	56.33
120	6	0	0.20	94.95	130.67	0	7.97	1135.29	130.67
120	8	0	0.54	31.71	159.67	7	31.22	3351.69	159.67
120	10	0	0.74	86.37	191.00	9	87.64	3600.00	195.56
120	12	2	0.83	1221.87	269.56	9	930.39	3600.00	316.56
120	14	0	1.58	1190.82	377.00	9	1681.89	3600.00	547.56
120	16	3	1.34	2118.18	501.56	9	1731.38	3600.00	799.38
120	18	6	1.87	2781.59	532.22	9	987.24	3600.00	748
160	2	0	0.06	0.06	19.67	0	0.10	0.23	19.67
160	4	0	0.29	9.51	56.33	0	1.82	38.95	56.33
160	6	0	0.63	162.56	130.67	6	17.34	3294.34	130.67
160	8	0	0.92	118.15	159.67	9	70.71	3600.00	159.67
160	10	0	1.35	328.35	191.00	9	606.78	3600.00	198.22
160	12	2	1.92	1539.68	272.00	9	1373.49	3600.00	405.50
160	14	3	2.72	1949.36	388.78	9	1472.87	3600.00	551.00
160	16	6	3.50	2985.52	552.44	9	1796.00	3600.00	721.00
160	18	7	13.63	3128.94	554.44	9	3413.39	3600.00	479
200	2	0	0.10	0.10	20.57	0	0.17	0.37	20.57
200	4	0	0.52	19.08	56.88	0	3.74	84.64	56.88
200	6	0	1.00	452.59	138.88	8	27.60	3600.00	138.88
200	8	0	2.44	346.62	160.50	8	215.63	3600.00	160.50
200	10	0	2.38	736.39	187.38	8	1866.90	3600.00	207.12
200	12	3	3.98	2907.43	275.25	8	2714.27	3600.00	592.60
200	14	3	7.33	2398.76	432.62	8	2312.73	3600.00	709.25
200	16	9	15.42	3600.00	602.78	9	-	3600.00	-
200	18	9	9.73	3600.00	625.33	9	-	3600.00	-

Source: from author (2015).

Table A.5: Hierarchical Instances - Low Demands

		Branch & Price				CPLEX			
$\|S\|$	$\|V\|$	#NF	time_int	time(s)	cost	#NF	time_int	time(s)	cost
125	2	0	0.01	0.01	16.33	0	0.06	0.13	16.33
125	4	0	0.11	0.29	25.00	0	2.27	9.18	25.00
125	6	0	0.36	4.87	62.67	0	17.30	167.21	62.67
125	8	0	1.15	29.32	76.67	5	75.18	3056.65	78.22
125	10	4	13.63	1929.54	135.22	9	270.71	3600.00	151.78
125	12	3	65.85	1389.12	144.89	9	629.59	3600.00	169.33
125	14	9	182.92	3600.00	264.00	9	1538.64	3600.00	217.00
125	16	9	857.62	3600.00	488.50	9	3142.99	3600.00	406.33
125	18	9	916.42	3600.00	601.33	9	-	3600.00	-
150	2	0	0.01	0.01	16.33	0	0.06	0.15	16.33
150	4	0	0.03	0.28	25.00	0	2.06	10.77	25.00
150	6	0	0.35	6.15	62.67	0	13.70	260.67	62.67
150	8	0	0.64	33.88	76.67	8	68.72	3540.39	77.11
150	10	4	2.25	1888.93	131.89	9	218.09	3600.00	155.33
150	12	4	2.95	1998.31	135.44	9	922.26	3600.00	185.56
150	14	7	12.57	2961.95	215.89	9	951.29	3600.00	263.50
150	16	9	26.20	3600.00	410.33	9	855.97	3600.00	419.33
150	18	9	155.34	3600.00	448.11	9	-	3600.00	-
175	2	0	0.01	0.01	16.33	0	0.09	0.20	16.33
175	4	0	0.03	0.84	25.00	0	2.57	19.25	25.00
175	6	0	0.05	6.64	62.67	0	22.57	646.73	62.67
175	8	0	0.36	32.48	76.67	9	108.28	3600.00	77.67
175	10	4	2.23	1782.50	128.78	9	519.80	3600.00	166.00
175	12	4	5.99	1722.29	136.89	9	873.28	3600.00	233.33
175	14	7	13.50	2826.14	201.44	9	1429.26	3600.00	305.78
175	16	9	25.24	3600.00	335.00	8	2425.69	3600.00	514.00
175	18	9	19.76	3600.00	439.67	8	3201.44	3600.00	573.00
200	2	0	0.01	0.01	16.33	0	0.11	0.25	16.33
200	4	0	0.03	2.18	25.00	0	3.87	31.70	25.00
200	6	0	0.06	19.37	62.67	1	28.18	1236.67	62.67
200	8	0	0.25	83.65	76.67	9	161.43	3600.00	79.00
200	10	4	2.69	2088.51	128.44	9	1038.06	3600.00	166.22
200	12	2	3.44	1461.63	132.67	9	1293.03	3600.00	212.88
200	14	6	19.43	2565.80	199.89	7	1875.52	3600.00	283.33
200	16	9	26.12	3600.00	394.11	7	3016.97	3600.00	430.75
200	18	9	28.82	3600.00	595.67	8	-	3600.00	-
225	2	0	0.02	0.02	16.33	0	0.19	0.43	16.33
225	4	0	0.04	1.26	25.00	0	5.72	47.53	25.00
225	6	0	0.82	15.46	62.67	1	56.71	1890.65	62.67
225	8	0	0.47	71.10	76.67	9	499.16	3600.00	86.22
225	10	6	1.63	2466.65	136.00	9	1434.67	3600.00	180.00
225	12	5	9.57	2070.26	144.44	9	1578.52	3600.00	273.33
225	14	6	13.49	2598.64	228.11	9	2677.50	3600.00	291.29
225	16	9	33.93	3600.00	369.00	9	-	3600.00	-
225	18	9	53.11	3600.00	603.78	-	-	-	-
250	2	0	0.02	0.02	16.33	0	0.18	0.41	16.33
250	4	0	0.05	0.82	25.00	0	6.17	64.05	25.00
250	6	0	0.31	22.09	62.67	4	58.16	2697.23	62.67
250	8	0	1.18	143.34	76.67	9	586.85	3600.00	81.78
250	10	4	5.73	2781.41	130.44	9	1143.66	3600.00	183.22
250	12	3	3.96	1632.76	133.89	9	2035.18	3600.00	210.67
250	14	8	19.96	3228.64	196.89	9	3188.84	3600.00	295.75
250	16	9	16.32	3600.00	349.44	6	-	3600.00	-
250	18	9	6.83	3600.00	507.89	-	-	-	-

Source: from author (2015).

Table A.6: Hierarchical Instances - High Demands

		Branch & Price				CPLEX			
$\|S\|$	$\|V\|$	#NF	time_int	time(s)	cost	#NF	time_int	time(s)	cost
125	2	0	0.01	0.01	19.67	0	0.05	0.11	19.67
125	4	0	0.05	0.22	56.33	0	1.72	4.52	56.33
125	6	1	70.59	439.99	180.89	0	17.85	58.07	180.89
125	8	0	1.08	9.08	164.67	0	50.23	458.09	164.67
125	10	1	8.93	799.98	229.67	6	204.63	2701.86	237.33
125	12	9	57.62	3600.00	369.29	7	629.25	2889.93	389.00
125	14	4	350.70	1866.99	499.67	4	636.09	2227.06	445.83
125	16	7	1560.21	2827.93	672.33	6	1214.45	2692.10	726.00
125	18	7	126.055	2801.42	783	6	1549.83	2774.42	569.00
150	2	0	0.01	0.01	19.67	0	0.06	0.15	19.67
150	4	0	0.04	0.41	56.33	0	2.28	6.84	56.33
150	6	2	4.23	830.02	161.88	0	19.39	77.69	161.88
150	8	0	1.09	26.19	171.11	0	70.36	1479.70	171.11
150	10	1	2.58	1007.59	240.89	8	409.67	3261.10	266.33
150	12	9	76.04	3600.00	438.50	7	597.23	3072.05	468.75
150	14	7	197.97	3032.94	473.17	8	1262.33	3503.23	591.17
150	16	9	287.46	3600.00	660.67	9	1111.43	3600.00	792.00
150	18	7	129.69	2854.57	659.67	8	2747.84	3381.41	796
175	2	0	0.01	0.01	19.67	0	0.09	0.20	19.67
175	4	0	0.08	0.65	56.33	0	2.40	10.69	56.33
175	6	2	3.03	819.05	170.00	1	22.56	703.68	167.00
175	8	0	0.81	19.08	159.67	5	119.17	2675.24	161.22
175	10	1	2.42	868.06	202.78	9	509.45	3600.00	259.44
175	12	9	18.90	3600.00	425.00	9	1813.97	3600.00	493.17
175	14	7	57.18	3181.59	396.67	9	2819.76	3600.00	806.50
175	16	9	287.10	3600.00	986.75	9	-	3600.00	-
175	18	9	70.71	3600.00	954.75	9	-	3600.00	-
200	2	0	0.01	0.01	19.67	0	0.11	0.26	19.67
200	4	0	0.04	1.67	56.33	0	3.25	15.48	56.33
200	6	3	6.99	1205.31	149.38	1	17.34	659.21	148.25
200	8	0	3.33	38.29	162.56	6	197.57	2769.50	162.56
200	10	1	5.22	1554.55	204.33	9	821.48	3600.00	277.56
200	12	9	126.25	3600.00	360.62	9	2250.90	3600.00	554.86
200	14	8	103.62	3342.64	398.33	9	2324.11	3600.00	634
200	16	9	188.44	3600.00	578.50	9	-	3600.00	-
200	18	8	865.91	3218.07	1042.25	9	-	3600.00	-
225	2	0	0.01	0.01	19.67	0	0.15	0.34	19.67
225	4	0	0.15	1.40	56.33	0	3.52	23.51	56.33
225	6	3	3.18	1212.33	163.11	0	69.38	697.81	159.11
225	8	0	1.64	81.44	160.44	7	359.11	3273.55	162.56
225	10	4	2.54	2080.56	222.33	9	1689.94	3600.00	284.44
225	12	9	13.98	3600.00	439.78	9	2448.91	3600.00	951.75
225	14	9	79.15	3600.00	459.00	9	-	3600.00	-
225	16	9	333.86	3600.00	737.67	9	-	3600.00	-
225	18	9	39.84	3600.00	598.67	2	-	3600.00	-
250	2	0	0.01	0.01	19.67	0	0.19	0.42	19.67
250	4	0	0.09	1.58	56.33	0	5.34	33.30	56.33
250	6	3	1.80	1207.01	176.89	1	61.85	1209.78	169.89
250	8	0	0.77	42.30	160.11	7	611.69	3275.76	167.33
250	10	5	4.44	2128.35	205.11	9	1795.74	3600.00	295.71
250	12	9	96.17	3600.00	400.22	9	3442.45	3600.00	830.00
250	14	9	372.61	3600.00	775.25	9	3200.92	3600.00	866
250	16	9	514.31	3600.00	1275.00	9	-	3600.00	-
250	18	9	111.09	3600.00	940.80	-	-	-	-

Source: from author (2015).

Table A.7: Transit-Stub Instances - Low Demands

		Branch & Price				CPLEX			
$\|S\|$	$\|V\|$	#NF	time_int	time(s)	cost	#NF	time_int	time(s)	cost
90	2	0	0.01	0.01	16.33	0	0.04	0.08	16.33
90	4	0	0.02	0.10	25.00	0	1.43	3.92	25.00
90	6	0	0.15	1.10	62.67	0	6.16	80.43	62.67
90	8	0	0.71	11.62	76.67	5	14.09	3060.16	76.67
90	10	0	0.90	562.50	126.11	9	56.14	3600.00	135.44
90	12	4	0.90	1936.71	137.44	9	420.97	3600.00	158.44
90	14	6	4.25	2529.66	168.89	9	1295.66	3600.00	242.62
90	16	9	7.64	3600.00	261.00	9	1668.90	3600.00	347.17
90	18	8	54.70	3545.14	277.00	9	1630.65	3600.00	381.50
135	2	0	0.01	0.01	16.33	0	0.05	0.12	16.33
135	4	0	0.03	0.37	25.00	0	1.88	8.17	25.00
135	6	0	0.73	4.26	62.67	0	16.24	207.81	62.67
135	8	0	0.95	15.40	76.67	9	24.20	3600.00	78.11
135	10	1	4.86	1008.53	125.56	9	119.30	3600.00	145.11
135	12	4	11.38	1698.08	136.11	9	1072.44	3600.00	179.25
135	14	6	18.96	2527.99	171.33	9	2063.45	3600.00	259.67
135	16	8	28.05	3366.60	274.00	9	919.61	3600.00	405
135	18	7	106.01	3324.57	300.44	9	3232.04	3600.00	369
180	2	0	0.01	0.01	16.33	0	0.09	0.22	16.33
180	4	0	0.07	0.13	25.00	0	2.90	20.02	25.00
180	6	0	0.13	5.92	62.67	0	29.46	714.93	62.67
180	8	0	0.21	128.66	76.67	9	103.33	3600.00	80.11
180	10	1	3.48	489.77	121.56	9	798.14	3600.00	162.00
180	12	3	3.34	1558.32	131.67	9	1268.69	3600.00	184.56
180	14	6	19.92	2489.55	161.89	9	1271.54	3600.00	240.60
180	16	7	55.83	3160.88	257.44	9	2709.80	3600.00	420.50
180	18	7	33.59	2867.82	286.11	9	2696.72	3600.00	489.50
225	2	0	0.02	0.02	16.33	0	0.16	0.35	16.33
225	4	0	0.12	0.42	25.00	0	5.05	42.98	25.00
225	6	0	0.79	7.96	62.67	0	33.44	1389.65	62.67
225	8	0	1.37	18.82	76.67	9	421.83	3600.00	82.00
225	10	2	6.37	1248.10	121.78	9	1336.48	3600.00	172.22
225	12	3	15.60	1480.03	136.11	9	1700.60	3600.00	262.67
225	14	7	21.94	2835.99	172.33	9	2469.18	3600.00	273.00
225	16	8	272.49	3280.00	266.44	9	-	3600.00	-
225	18	7	84.19	2977.65	301.67	9	-	3600.00	-
270	2	0	0.02	0.02	16.33	0	0.23	0.52	16.33
270	4	0	0.04	0.47	25.00	0	7.51	76.94	25.00
270	6	0	0.08	10.08	62.67	1	107.51	2363.50	62.67
270	8	0	0.38	25.72	76.67	9	702.13	3600.00	94.89
270	10	3	3.13	1316.29	123.11	9	1916.61	3600.00	216.25
270	12	3	4.34	1410.66	130.44	9	2316.13	3600.00	209.80
270	14	7	21.83	2893.45	170.67	9	-	3600.00	-
270	16	9	70.29	3600.00	261.00	9	-	3600.00	-
270	18	7	63.97	3102.35	287.22	9	-	3600.00	-

Source: from author (2015).

Table A.8: Transit-Stub Instances - High Demands

		Branch & Price				CPLEX			
$\|S\|$	$\|V\|$	#NF	time_int	time(s)	cost	#NF	time_int	time(s)	cost
90	2	0	0.01	0.01	19.67	0	0.03	0.07	19.67
90	4	0	0.05	0.14	56.33	0	1.02	2.06	56.33
90	6	1	2.36	468.98	158.50	0	5.92	16.36	158.50
90	8	0	0.30	4.46	166.89	0	26.13	196.38	166.89
90	10	0	1.06	403.26	227.44	1	68.70	893.57	227.44
90	12	8	123.75	3417.56	394.22	4	241.60	2223.44	384.33
90	14	3	24.73	1714.26	384.33	1	350.68	953.60	455.00
90	16	5	98.56	2224.50	570.00	0	330.64	1714.93	618.00
90	18	7	16.41	2803.70	471.67	5	671.74	2163.21	547.00
135	2	0	0.01	0.01	19.67	0	0.05	0.12	19.67
135	4	0	0.12	0.40	56.33	0	2.17	5.80	56.33
135	6	0	2.00	402.40	179.56	0	14.80	135.11	179.56
135	8	0	0.90	4.15	165.22	0	62.51	658.16	165.22
135	10	0	2.38	336.54	216.33	6	263.55	3092.05	223.78
135	12	7	82.43	3291.35	334.78	7	505.15	3018.57	371.11
135	14	5	258.30	2385.65	458.29	5	800.86	2554.44	531.75
135	16	6	452.72	2684.76	582.80	6	1044.08	3216.73	713.75
135	18	8	522.89	3211.46	687.50	8	1655.28	3309.91	786.00
180	2	0	0.01	0.01	19.67	0	0.09	0.20	19.67
180	4	0	0.02	0.37	56.33	0	2.80	10.97	56.33
180	6	2	0.36	905.50	157.00	0	12.39	221.59	156.22
180	8	0	0.16	6.39	159.67	3	160.48	2381.46	159.67
180	10	0	1.28	556.60	200.67	9	712.76	3600.00	247.33
180	12	8	8.81	3393.55	335.44	9	1837.74	3600.00	400.29
180	14	6	192.28	2758.45	425.00	9	2549.79	3600.00	749.80
180	16	7	136.65	3064.30	629.80	9	3139.49	3600.00	1045.00
180	18	9	519.74	3600.00	559.00	9	-	3600.00	-
225	2	0	0.01	0.01	19.67	0	0.15	0.33	19.67
225	4	0	0.34	1.18	56.33	0	4.27	22.11	56.33
225	6	3	3.09	1203.99	158.11	0	86.50	481.99	157.44
225	8	0	1.72	19.86	160.11	6	231.23	2885.80	164.22
225	10	2	3.47	896.60	203.67	9	1331.91	3600.00	275.11
225	12	9	6.32	3600.00	337.00	9	2254.52	3600.00	495.00
225	14	6	170.09	2860.92	490.38	9	2298.47	3600.00	510.00
225	16	7	808.61	3092.19	612.50	9	3233.45	3600.00	827.50
225	18	8	767.36	3595.54	581.75	9	-	3600.00	-
270	2	0	0.02	0.02	19.67	0	0.22	0.48	19.67
270	4	0	0.16	0.65	56.33	0	6.24	41.78	56.33
270	6	3	2.97	1202.06	156.78	2	84.45	1373.08	159.11
270	8	0	3.33	14.23	159.67	7	1172.24	3350.94	167.56
270	10	1	8.81	1048.52	195.78	9	2632.52	3600.00	362.75
270	12	8	20.95	3345.70	335.00	9	2926.11	3600.00	322
270	14	7	433.33	2864.21	510.12	9	-	3600.00	-
270	16	7	1131.90	3292.89	770.00	9	-	3600.00	-
270	18	9	676.16	3600.00	909.17	9	-	3600.00	-

Source: from author (2015).