MIGUEL ANGELO TANCREDI MOLINA

**UTILIZAÇÃO DA ÁLGEBRA DE CAMINHOS PARA REALIZAR O
MAPEAMENTO DE REQUISIÇÕES VIRTUAIS SOBRE REDES DE
SUBSTRATO**

São Paulo

2012

MIGUEL ANGELO TANCREDI MOLINA

**UTILIZAÇÃO DA ÁLGEBRA DE CAMINHOS PARA REALIZAR O
MAPEAMENTO DE REQUISIÇÕES VIRTUAIS SOBRE REDES DE
SUBSTRATO**

**Tese apresentada à Escola Politécnica
da Universidade de São Paulo para
obtenção do título de Doutor em
Engenharia Elétrica.**

São Paulo

2012

MIGUEL ANGELO TANCREDI MOLINA

**UTILIZAÇÃO DA ÁLGEBRA DE CAMINHOS PARA REALIZAR O
MAPEAMENTO DE REQUISIÇÕES VIRTUAIS SOBRE REDES DE
SUBSTRATO**

**Tese apresentada à Escola Politécnica
da Universidade de São Paulo para
obtenção do título de Doutor em
Engenharia Elétrica.**

**Área de concentração:
Sistemas Eletrônicos**

**Orientador:
Prof. Dr. José Roberto de Almeida
Amazonas**

São Paulo

2012

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 10 de agosto de 2012.

Assinatura do autor _____

Assinatura do orientador _____

FICHA CATALOGRÁFICA

Molina, Miguel Ângelo Tancredi

Utilização da álgebra de caminhos para realizar o mapeamento de requisições virtuais sobre redes de substrato / M.A.T. Molina. -- ed.rev. -- São Paulo, 2012.

252 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Telecomunicações e Controle.

1. Algoritmos 2. Álgebra 3. Teoria dos grafos 4. Virtualização I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Telecomunicações e Controle II. t.

DEDICATÓRIA ESPECIAL

A minha querida esposa Jane Cássia e a minha filha Caroline Molina, fruto do nosso amor e razão do meu existir.

DEDICATÓRIA

A minha mãe Therezinha Bianca e ao meu pai Dr. Miguel Molina (*in memorium*) pela minha formação profissional, moral e ética.

A Jane Cássia e a futura engenheira Caroline.

Aos meus familiares e amigos, em especial a minha sogra e ao meu sogro por todas as suas orações, que torcem por minhas vitórias e se entristecem com minhas derrotas.

Ao nosso Senhor Jesus Cristo por toda a inspiração e sabedoria. E à Nossa Senhora mãe de Deus, minha protetora e guardiã dos necessitados.

AGRADECIMENTOS

Ao meu professor e grande mestre, Prof. Dr. José Roberto de Almeida Amazonas, que me orientou e incentivou durante a realização desta Tese.

Ao Prof. Dr. Cláudio Marte, pelas suas sugestões fornecidas no exame de qualificação e pela gentileza de novamente aceitar o convite de tomar parte da banca examinadora final deste trabalho.

Ao Prof. Dr. Denis Gabos, pelas suas sugestões fornecidas no exame de qualificação e pela gentileza de novamente aceitar o convite de tomar parte da banca examinadora final deste trabalho.

Ao Prof. Dr. Xavier Hesselbach, pela ajuda e cooperação nas diversas áudio-conferências realizadas no decorrer deste trabalho e pelo fornecimento da infraestrutura de laboratório utilizada em diversos testes e acessos remotos com a *Universitat Politècnica de Catalunya* (UPC), de Barcelona, Espanha.

Ao futuro doutor Juan Felipe Botero, pelo auxílio nos laboratórios e nas simulações.

Aos colegas e amigos do programa de Pós-Graduação em Engenharia Elétrica da Escola Politécnica da USP (PPGEE/EPUSP) em especial ao grupo de pesquisa coordenado pelo prof. Amazonas.

RESUMO

A tecnologia de virtualização de redes é um novo paradigma de redes que permite a múltiplas redes virtuais (VNs) compartilharem de uma forma eficiente e eficaz a mesma rede de infraestrutura denominada rede de substrato (SN). A implementação e o desenvolvimento de novos protocolos, testes de novas soluções e arquiteturas para a Internet atual e do futuro podem ser tratadas por meio da virtualização de redes. Com a virtualização de redes surge um desafio denominado problema VNE.

O problema de virtualização de redes embutidas (VNE) consiste em realizar o mapeamento dos nós virtuais e o mapeamento dos enlaces virtuais sobre uma rede de substrato (SN). O problema é conhecido como NP-Hard e a sua solução é realizada por meio de algoritmos heurísticos e aproximados que realizam o mapeamento de nós e enlaces virtuais em dois estágios de forma independente ou coordenada.

A presente tese tem o objetivo de resolver o mapeamento dos enlaces virtuais do problema VNE com a utilização da álgebra de caminhos. A solução apresentada fornece o melhor desempenho quando comparada com as demais soluções de virtualização de redes encontradas na literatura.

Os resultados obtidos nas simulações para o problema VNE foram avaliados e analisados com a utilização do algoritmo desenvolvido nesta tese denominado “*Path Algebra for Virtual Link Mapping*” (PAViLiM), que utiliza a álgebra de caminhos para realizar o mapeamento de enlaces virtuais para caminhos na rede de substrato.

A álgebra de caminhos é poderosa e flexível. Tal flexibilidade permite que ocorra uma exploração detalhada do espaço de soluções e a identificação do melhor critério e política que devem ser utilizados para a virtualização de redes.

Palavras-chave: Virtualização de redes, Álgebra de caminhos, Rede Virtual, Qualidade de Serviços, Complexidade Algorítmica.

ABSTRACT

The network virtualization technology is a new paradigm of network that allows multiple virtual networks (VNs) share in an efficient and effective way the same network infrastructure called substrate network (SN). The implementation and the development of new protocols, testing of new solutions and architectures for current and future Internet can be addressed through network virtualization. With the network virtualization arises a challenge called VNE problem.

The problem of virtual network embedded (VNE) is to conduct the mapping of the virtual nodes and mapping of the virtual links over a substrate network (SN). The problem is known as NP-Hard and its solution is accomplished by means of approximate and heuristic algorithms that perform the mapping of virtual nodes and links in two stages independently or coordinated.

This thesis aims to solve the mapping of virtual links for VNE problem using the paths algebra. The solution presented provides the best performance when compared with other networks virtualization solutions from the literature.

The results of simulation for the VNE problem were evaluated and analyzed using the algorithm developed in this thesis called "Path Algebra for Virtual Link Mapping" (PAViLiM), which uses the paths algebra to perform the mapping of virtual links to paths in substrate network.

The paths algebra is powerful and flexible. This flexibility allows the occurrence of a detailed exploration for identifying the best solutions and political criteria to be used for network virtualization.

Keywords: Network Virtualization, Path Algebra, Virtual Network, Quality of Service, Algorithmic Complexity.

LISTA DE FIGURAS

Figura 1 – Arquitetura com dois papéis de ISPs: Infraestrutura como Serviço.	2
Figura 2 – Arquitetura com quatro papéis de ISPs.	3
Figura 3 – Modelo de Virtualização.	4
Figura 4 – Redes Virtuais Embutidas (VNE) em uma Rede de Substrato (SN).	4
Figura 5 – VNE de duas requisições de rede virtual (VNRs).	14
Figura 6 – Representação de uma rede, seus vértices e aresta.	30
Figura 7 – Exemplo de uma rede, seus vértices e arestas.	42
Figura 8 – Rede do Exemplo Didático.	43
Figura 9 – (a) pesos associados com os nós e arestas. (b) pesos dos nós artificialmente inseridos na aresta. (c) métricas finais.	50
Figura 10 – SN e peso de nós: métrica CPU e peso de arestas: métrica bandwidth.	56
Figura 11 – SN e pesos de arestas: métrica delay.	56
Figura 12 – VNRs e peso de nós: métrica CPU e pesos de arestas: métrica bandwidth e delay.	56
Figura 13 – Atualização dos pesos na SN após o atendimento do $VL_{1,2}$ da VNR # 2.	61
Figura 14 – Atualização dos pesos na SN após o atendimento do $VL_{2,3}$ da VNR # 2.	61
Figura 15 – Atualização dos pesos na SN após o atendimento do $VL_{1,2}$ da VNR # 1.	62
Figura 16 – Atualização dos pesos na SN após o atendimento do $VL_{2,3}$ da VNR # 1.	63
Figura 17 – Profundidade mínima para conectividade completa, dígrafos de 20 nós – random flat.	78
Figura 18 – Profundidade mínima para conectividade completa, dígrafos de 50 nós - random flat.	79

Figura 19 – Profundidade mínima para conectividade completa, dígrafos de 100 nós - random flat.	79
Figura 20 – Profundidade mínima para conectividade completa, dígrafos de 20 nós – transit stub.	80
Figura 21 – Profundidade mínima para conectividade completa, dígrafos de 50 nós – transit stub.	80
Figura 22 – Profundidade mínima para conectividade completa, dígrafos de 100 nós – transit stub.	80
Figura 23 – Diagrama de Blocos do Algoritmo PAViLiM.	83
Figura 24 – Diagrama de blocos do programa de atendimento de VNRs do algoritmo PAViLiM.	88
Figura 25 – Layout do arquivo virtualStart.dat.	91
Figura 26 – Rede de Substrato.	97
Figura 27 – Requisições de Rede Virtual.	98
Figura 28 - Dígrafo com 10 nós utilizado nas simulações.	102
Figura 29 – Resultados da taxa média de receita mapeada – grupo 1.	117
Figura 30 – Resultados da taxa média de aceitação de VNRs – grupo 1.	118
Figura 31 – Resultados da relação custo/receita média – grupo 1.	118
Figura 32 – Resultados da taxa média de receita mapeada – grupo 2.	120
Figura 33 – Resultados da taxa média de aceitação de VNRs – grupo 2.	121
Figura 34 – Resultados da relação custo/receita média – grupo 2.	121
Figura 35 – Taxa média de receita mapeada – PAViLiM atendimento parcial.	131
Figura 36 – Taxa média de aceitação de VNRs – PAViLiM atendimento parcial.	131
Figura 37 – Resultados do atendimento multi-path.	136
Figura 38 – Dígrafo com 2 vértices.	156
Figura 39 – Caminhos com origem no vértice 1 em um dígrafo com 3 vértices.	156
Figura 40 – Caminhos com origem no vértice 4 em um dígrafo com 4 vértices.	157

Figura 41 – Caminhos com origem no vértice 5 em um dígrafo com 5 vértices.	159
Figura 42 – Quantidade total de caminhos para N vértices.	160
Figura 43 – Matriz de adjacência da rede virtual 1.	202
Figura 44 – Largura de Banda solicitada na VN1.	202
Figura 45 – Matriz de adjacência da rede virtual 2.	203
Figura 46 – Largura de Banda solicitada na VN2.	203
Figura 47 – Matriz de adjacência da rede virtual 3.	204
Figura 48 – Largura de Banda solicitada na VN3.	205
Figura 49 – Matriz de adjacência da rede virtual 4.	206
Figura 50 – Largura de Banda solicitada na VN4.	206
Figura 51 – Matriz de adjacência da rede virtual 5.	207
Figura 52 – Largura de Banda solicitada na VN5.	208
Figura 53 – Matriz de adjacência da rede virtual 6.	209
Figura 54 – Largura de Banda solicitada na VN6.	209
Figura 55 – Matriz de adjacência da rede virtual 7.	210
Figura 56 – Largura de Banda solicitada na VN7.	210
Figura 57 – Matriz de adjacência da rede virtual 8.	211
Figura 58 – Largura de Banda solicitada na VN8.	211
Figura 59 – Matriz de adjacência da rede virtual 9.	212
Figura 60 – Largura de Banda solicitada na VN9.	212
Figura 61 – Matriz de adjacência da rede virtual 10.	213
Figura 62 – Largura de Banda solicitada na VN10.	213
Figura 63 – Matriz de adjacência da rede virtual 11.	214
Figura 64 – Largura de Banda solicitada na VN11.	214
Figura 65 – Matriz de adjacência da rede virtual 12.	215

Figura 66 – Largura de Banda solicitada na VN12.	215
Figura 67 – Matriz de adjacência da rede virtual 13.	215
Figura 68 – Largura de Banda solicitada na VN13.	215
Figura 69 – Matriz de adjacência da rede virtual 14.	217
Figura 70 – Largura de Banda solicitada na VN14.	217
Figura 71 – Matriz de adjacência da rede virtual 15.	218
Figura 72 – Largura de Banda solicitada na VN15.	218
Figura 73 – Matriz de adjacência da rede virtual 16.	219
Figura 74 – Largura de Banda solicitada na VN16.	219
Figura 75 – Matriz de adjacência da rede virtual 17.	220
Figura 76 – Largura de Banda solicitada na VN17.	220
Figura 77 – Matriz de adjacência da rede virtual 18.	220
Figura 78 – Largura de Banda solicitada na VN18.	221
Figura 79 – Matriz de adjacência da rede virtual 19.	221
Figura 80 – Largura de Banda solicitada na VN19.	222
Figura 81 – Matriz de adjacência da rede virtual 20.	223
Figura 82 – Largura de Banda solicitada na VN20.	223

LISTA DE TABELAS

Tabela 1 - Problemas com a Virtualização de Redes.	6
Tabela 2 – Leis da Álgebra de Conjuntos.	28
Tabela 3 – Descrição das questões resolvidas pela Álgebra de Caminhos.	36
Tabela 4 – Métricas e funções de combinação de métricas.	44
Tabela 5 – Sínteses para os três caminhos.	45
Tabela 6 – Sínteses para os três caminhos com abreviações.	46
Tabela 7 – Mapeamento dos Nós Virtuais para Nós Reais.	57
Tabela 8 – Mapeamento dos Enlaces Virtuais para Caminhos na SN.	59
Tabela 9 – Valores das sínteses para cada caminho analisado na SN.	64
Tabela 10 – Programas do algoritmo ELND.	66
Tabela 11 – Nome de programas: ELND e PAViLiM.	66
Tabela 12 – Tempo de execução e caminhos descobertos - Algoritmo ELND.	68
Tabela 13 – Tempo de execução e caminhos descobertos com filtro de profundidade – Algoritmo PAViLiM.	69
Tabela 14 – Comparação do número médio de caminhos enumerados com e sem filtragem por profundidade - <i>random flat</i> .	71
Tabela 15 – Comparação do número médio de caminhos enumerados com e sem filtragem por profundidade – <i>transit stub</i> .	71
Tabela 16 – Comprimento Médio Mínimo (Profundidade Mínima) de caminhos para alcançar 100% de conectividade.	81
Tabela 17 – Sínteses e ordenação léxica multidimensional padrão.	87
Tabela 18 – Principais Programas do algoritmo PAViLiM.	90
Tabela 19 – Valores para Política de Atendimento.	92
Tabela 20 – Tabela Resumo de Arquivos da Integração.	96
Tabela 21 – Conteúdo dos arquivos de atendimento.	99

Tabela 22 – Resultados do algoritmo PAViLiM.	100
Tabela 23 – Atendimento das Requisições de Redes Virtuais (VN requests) utilizando políticas diferentes de atendimento.	104
Tabela 24 – Requisições de Redes Virtuais Atendidas x Políticas de Atendimento para Ordenação 1.	104
Tabela 25 – Requisições de Redes Virtuais Atendidas x Políticas de Atendimento para Ordenação 2.	104
Tabela 26 – Requisições de Redes Virtuais Atendidas x Políticas de Atendimento – Ordenação 1.	105
Tabela 27 – Requisições de Redes Virtuais Atendidas x Políticas de Atendimento – Ordenação 2.	105
Tabela 28 – Parâmetros escolhidos para os cenários de simulação.	113
Tabela 29 – Algoritmos utilizados nas simulações.	115
Tabela 30 – Resultados da taxa de receita mapeada – grupo 1.	116
Tabela 31 – Resultados da taxa de aceitação de VNRs – grupo 1.	116
Tabela 32 – Resultados da relação custo/receita – grupo 1.	116
Tabela 33 – Resultados da taxa de receita mapeada – grupo 2.	119
Tabela 34 – Resultados da taxa de aceitação de VNRs – grupo 2.	119
Tabela 35 – Resultados da relação custo/receita – grupo 2.	119
Tabela 36 – Comparação de dois mapeamentos possíveis.	124
Tabela 37 – Sumário dos melhores resultados obtidos para diferentes indicadores e políticas – grupo 1.	127
Tabela 38 – Sumário dos melhores resultados obtidos para diferentes indicadores e políticas – grupo 2.	127
Tabela 39 – Taxa de receita média mapeada – PAViLiM em modo de atendimento parcial.	130
Tabela 40 – Taxa de aceitação média de VNRs – PAViLiM em modo de atendimento parcial.	130

Tabela 41 – Taxa de aceitação de VNRs para atendimento multi-path.	135
Tabela 42 – Regra para mapeamento dos nós virtuais para nós reais (físicos) da rede de substrato.	201
Tabela 43 – CPU request para VN request 1.	203
Tabela 44 – CPU request para VN request 2.	204
Tabela 45 – CPU request para VN request 3.	205
Tabela 46 – CPU request para VN request 4.	207
Tabela 47 – CPU request para VN request 5.	208
Tabela 48 – CPU request para VN request 6.	209
Tabela 49 – CPU request para VN request 7.	210
Tabela 50 – CPU request para VN request 8.	211
Tabela 51 – CPU request para VN request 9.	213
Tabela 52 – CPU request para VN request 10.	213
Tabela 53 – CPU request para VN request 11.	215
Tabela 54 – CPU request para VN request 12.	216
Tabela 55 – CPU request para VN request 13.	216
Tabela 56 – CPU request para VN request 14.	217
Tabela 57 – CPU request para VN request 15.	218
Tabela 58 – CPU request para VN request 16.	219
Tabela 59 – CPU request para VN request 17.	220
Tabela 60 – CPU request para VN request 18.	221
Tabela 61 – CPU request para VN request 19.	222
Tabela 62 – CPU request para VN request 20.	223

LISTA DE ABREVIATURAS E SIGLAS

ALEVIN	Algorithms for Embedding of Virtual Networks
AS	Autonomous System
ASs	Autonomous Systems
AVAIL	Availability
BFS	Breadth-First Search
BW	Bandwidth
C/R	Cost/Revenue
CPU	Central Processing Unit
DFS	Depth-First Search
DVINE	Deterministic Rounding-based Virtual Network Embedding
DVINESP	Deterministic Rounding-based VNE Single-Path Algorithm
ELND	Eliminação de Loop pelo Nó de Destino
EPUSP	Escola Politécnica da Universidade de São Paulo
FIFO	First In First Out
GAR	Greeedy Available Resources
GARPS	Greeedy Available Resources Path-Splitting
GARSP	Greeedy Available Resources Single-Path
GT-ITM	Georgia Tech Internetwork Topology Models
HOP	Hand-off Point
IaaS	Infrastructure as a Service
ILP	Integer Linear Programming
InP	Infrastructure Provider
IP	Infrastructure Provider
IRTF	Internet Resource Task Force
ISP	Internet Service Provider
ISPs	Internet Service Providers
LCF	Least Consuming First
LCF-UP	Least Consuming First - USP Proposed

MATLAB	Matrix Laboratory
MCF	Most Consuming First
MCF-UP	Most Consuming First - USP Proposed
MFP	Multiple Flow Path
MIP	Mixed Integer Programming
NP	Non-Polynomial
NP Complete	Non-Polynomial Complete
NPC	Non-Polynomial Complete
NP-Hard	Non-Polynomial Hard
P	Polynomial
PA	Paths Algebra
PAViLiM	Paths Algebra for Virtual Link Mapping
PDT	Packet Delay Transfer
PDV	Packet Delay Variation
PIP	Physical Infrastructure Provider
PLR	Packet Loss Rate
POC	Proof of Concept
PPGEE	Programa de Pós-Graduação em Engenharia Elétrica
QoE	Quality of Experience
QoS	Quality of Service
RVINE	Randomized Rounding-based Virtual Network Embedding
SID	Subgraph Isomorphism Detection
SLA	Service Level Agreement
SN	Substrate Network
SP	Service Provider
SPF	Shortest Path First
SPR	Shortest Path Routing
THRU	Throughput
TS	Transit Stub
UFP	Unsplittable Flow Problem
UPC	Universitat Politècnica de Catalunya

USP	Universidade de São Paulo
VL	Virtual Link
VN	Virtual Network
VN request	Virtual Network Request
VNE	Virtual Network Embedded
VNO	Virtual Network Operator
VNP	Virtual Network Provider
VNR	Virtual Network Request
VNRG	Virtual Network Resource Group
VNRs	Virtual Network Requests
VNs	Virtual Networks

SUMÁRIO

1. Introdução	1
1.1 Introdução ao Trabalho de Pesquisa	1
1.2 Motivação.....	5
1.3 Problema.....	6
1.4 Objetivos	7
1.5 Metodologia.....	7
1.6 Organização do Trabalho.....	8
 2. Revisão da Virtualização de Redes	 9
2.1 Introdução à Virtualização de Redes	9
2.2 Conceitos e Terminologia.....	10
2.3 Elementos da Arquitetura de uma Rede Virtual (VN).....	11
2.4 Formulação do Problema VNE.....	12
2.5 Metodologias para a resolução do problema VNE	15
2.5.1 Utilização do <i>stress</i> do enlace e do nó na rede de substrato.....	16
2.5.2 Restrições de largura de banda em um cenário <i>offline</i>	17
2.5.3 Otimização da receita média.....	17
2.5.4 Mapeamento coordenado de nós e enlaces	19
2.5.5 Mapeamento de nós e enlaces em um único estágio	20
2.5.6 Introdução do conceito de nós escondidos (<i>hidden hops</i>)	21
2.5.7 Custo ótimo em um único estágio.....	22
2.5.8 Algoritmo inspirado no Google <i>PageRank</i>	22
2.5.9 Discussão do problema VNE em um ambiente <i>inter-domain</i>	23
2.5.10 Síntese das propostas para a solução do problema VNE.....	24
2.6 Considerações Finais do Capítulo	25

3. Álgebra de Caminhos	26
3.1 Conceitos da Álgebra Universal	26
3.1.1 Conjunto	26
3.1.2 Operações sobre Conjuntos	26
3.1.3 Álgebra de Conjuntos	27
3.1.4 Princípio da Dualidade.....	28
3.1.5 Teoria das Redes Complexas.....	29
3.1.6 Par Ordenado e Produto Cartesiano.....	30
3.1.7 Relação Binária.....	31
3.1.8 Propriedades das Relações Binárias	31
3.2 Estruturas de Ordem.....	32
3.2.1 Ordenação	32
3.2.2 Ordenação de Produto.....	33
3.2.3 Ordenação Lexicográfica ou Léxica.....	34
3.3 Álgebra de Caminhos.....	35
3.3.1 História da Álgebra de Caminhos	36
3.3.2 Conceitos da Álgebra de Caminhos.....	37
3.3.3 Novo <i>Framework</i> da Álgebra de Caminhos	38
3.3.3.1 Métricas	38
3.3.3.2 Função de Combinação de Métricas	39
3.3.3.3 Sínteses.....	39
3.3.3.4 Abreviação.....	41
3.3.3.5 Ordenação Léxica Multidimensional	41
3.3.4 Exemplo Didático da Álgebra de Caminhos.....	42
3.4 Álgebra de Caminhos e o Problema VNE	47
3.4.1 Métricas e Critérios de Otimização	48
3.4.2 Solução do Problema VNE	49
3.4.2.1 Transformação de métricas de nós em métricas de arestas	49
3.4.2.2 Enumeração de todos os caminhos na SN.....	51
3.4.2.3 Ordenação das requisições de rede virtual (VNRs).....	51

3.4.2.4	Atendimento de VNRs	53
3.4.2.5	Atribuição dos enlaces virtuais para caminhos na SN	53
3.5	Prova de Conceito: Álgebra de Caminhos e o Problema VNE	55
3.6	Considerações Finais do Capítulo	64
4.	Algoritmo para mapeamento de enlaces de rede virtual.....	65
4.1	Algoritmo ELND	65
4.2	Complexidade algorítmica.....	66
4.3	Estudo da filtragem por profundidade	70
4.3.1	Breve revisão da teoria dos grafos	72
4.3.1.1	Percurso	73
4.3.1.2	Caminho	74
4.3.2	Teorema das potências da matriz de adjacência.....	75
4.3.3	Fórmula da Filtragem por Profundidade	76
4.3.4	Programas para Filtragem por Profundidade	77
4.4	Algoritmo PAViLiM	82
4.5	Diagrama de blocos do algoritmo PAViLiM e solução do problema VNE	83
4.6	Principais programas do algoritmo PAViLiM	89
4.7	Integração entre PAViLiM e ALEVIN.....	90
4.7.1	Início da simulação	91
4.7.2	Rede de substrato (SN)	93
4.7.3	Requisições de rede virtual.....	93
4.7.4	Resultado da simulação.....	94
4.7.5	Término da simulação.....	95
4.7.6	Tabela resumo de arquivos da integração.....	95
4.7.7	Exemplo da Integração	96
4.8	Considerações Finais do Capítulo	100
5.	Simulação e Discussão de Resultados	101
5.1	Efeito da política e do critério de ordenação sobre o atendimento de VNRs	101
5.1.1	Exemplo dos Efeitos da Política.....	102

5.2	Metodologia de simulação.....	107
5.2.1	Topologia de criação de redes (SN e VNs).....	107
5.2.2	Definição de recursos e demandas.....	108
5.2.2.1	Demanda média sobre um nó virtual para uma dada carga (ρ).....	109
5.2.2.2	Demanda média sobre um enlace virtual para uma dada carga (ρ)	110
5.2.3	Condições para simulação.....	112
5.3	Simulação: PAViLiM versus programas do ALEVIN	113
5.3.1	Discussão dos Resultados.....	122
5.3.2	Total de Simulações	126
5.3.3	Análise Individual de Resultados	126
5.3.4	Considerações Finais das Simulações	128
5.4	Atendimento Parcial de VNRs.....	129
5.4.1	Considerações Finais do Atendimento Parcial	132
5.5	Atendimento <i>multi-path</i> de VNRs	133
5.5.1	Considerações Finais do Atendimento <i>multi-path</i>	137
5.6	Considerações Finais do Capítulo	138
6.	Conclusão e Trabalhos Futuros.....	140
6.1	Conclusão	140
6.2	Trabalhos Futuros	146
	Referências Bibliográficas	148
	Anexos	155
	Anexo A – Número Total de Caminhos em um Dígrafo	156
	Anexo B – Principais Programas do Algoritmo PAViLiM.....	166
	Anexo C – Matrizes de Adjacência e Mapeamentos de Nós	201
	Anexo D – Resultados Individuais para cada cenário	224

1. Introdução

Este capítulo faz uma introdução ao trabalho de pesquisa, apresenta a motivação pela escolha do tema, o problema e sua resposta, objetivos, justificativa, metodologia e organização do trabalho.

1.1 Introdução ao Trabalho de Pesquisa

A Internet tem se mostrado resistente a mudanças, impedindo os testes de novos protocolos, sua utilização e a implantação de novos serviços. Um dos motivos dessa resistência é a atual arquitetura dos provedores de serviços da Internet (*Internet Service Providers - ISPs*), que surgiram para fornecer ao usuário final tanto o acesso à rede física quanto um conjunto limitado de serviços. Agrupar esses dois papéis a uma única identidade, “provedor”, não corresponde à realidade e essa arquitetura está sendo questionada [1] [2] [4].

Os trabalhos [1] [3] [4] e [6] fornecem propostas de um novo paradigma para a Internet, o qual faz a separação dos provedores de serviço de Internet em múltiplos papéis e camadas de abstração. A proposta de Feamster [1] define uma arquitetura para os ISPs com dois papéis: provedores de infraestrutura e provedores de serviço, a Figura 1 exibe esses dois papéis. Essa arquitetura é denominada Infraestrutura como Serviço (*Infrastructure as a Service - IaaS*).

Figura 1 – Arquitetura com dois papéis de ISPs: Infraestrutura como Serviço (IaaS).

Segundo Schaffrath [3], a arquitetura da Internet ideal deveria ser baseada em provedores que possuam os seguintes papéis: provedor de infraestrutura física (*Physical Infrastructure Provider* - PIP), provedor de rede virtual (*Virtual Network Provider* - VNP), operador de rede virtual (*Virtual Network Operator* - VNO) e provedor de serviço (*Service Provider* - SP).

A Figura 2 traduz a arquitetura proposta por Schaffrath [3].

Figura 2 – Arquitetura com quatro papéis de ISPs.

Chowdhury e Rahman [4] recomendam para a arquitetura da Internet um modelo de virtualização de redes que é extremamente flexível, podendo atender e suportar a heterogeneidade de tecnologias e oferecer suporte a diferentes interesses comerciais dos diversos tipos de **stakeholders** que atuam na Internet. A Figura 3 apresenta o modelo de virtualização de redes para a Internet e mostra os seguintes papéis: *Infrastructure Provider* (provedor de infraestrutura), *Service Provider* (provedor de serviços), *Broker* (mediador) e *End User* (usuário final).

Figura 3 – Modelo de Virtualização de Redes [4].

No modelo de virtualização de redes, o provedor de infraestrutura gerencia os recursos da rede física no ambiente de virtualização de redes. O provedor de serviços pode contratar recursos de múltiplos provedores de infraestrutura para criar redes virtuais com serviços que proporcionem atendimento fim a fim, com qualidade e suporte a novos protocolos de comunicação. O mediador faz a interface entre solicitações do usuário final ou provedor de serviços e o provedor de infraestrutura.

Segundo Botero [6], a arquitetura ideal para a Internet é de uma rede **overlay**, com múltiplas camadas de abstração dando origem a múltiplas redes virtuais embutidas ou sobrepostas (VN *Embedded* ou VNE). A Figura 4 apresenta esse cenário.

Figura 4 – Redes Virtuais Embutidas (VNE) em uma Rede de Substrato (SN) [6].

Independente do número de camadas que será adotado na Internet, em qualquer das abordagens é utilizado o conceito de Virtualização de Redes e de Redes Virtuais Embutidas.

A **Virtualização de Redes** é um conceito que se traduz por transformar uma infraestrutura de rede física e real em uma infraestrutura lógica, virtual e que não é diretamente mapeada com todos os recursos físicos existentes; contudo, uma vez que cada nó virtual e enlace virtual forem logicamente definidos e as restrições de rede configuradas, o comportamento final para o provedor ou usuário é o de uma rede física exclusiva para o atendimento de suas requisições.

As **Redes Virtuais Embutidas** (VNEs ou VN *Embedded*) são redes virtuais administradas por um provedor que não possui acesso ou controle de uma infraestrutura física. A rede virtual embutida existe e obtém seus recursos de uma ou mais redes de substrato (SN – *Substrate Network*), que são contratados junto a provedores de infraestrutura [4].

1.2 Motivação

Com o advento da convergência das redes a alocação de recursos e o controle de congestionamento são questões complexas que têm sido assunto de várias áreas ativas de pesquisa [5].

Segundo Xu [7], as aplicações convergentes que integram vídeo, áudio e voz em uma única infraestrutura abrem as portas para novas maneiras de comunicação, aprendizado e trabalho.

Uma tendência recente na área de redes é o uso simultâneo de uma única rede física por múltiplas instâncias de rede. Essas redes, denominadas redes virtuais (*virtual networks*), fornecem isolamento entre as instâncias ou tipos de redes e suportam o uso compartilhado da mesma infraestrutura para propósitos diferentes [8].

Redes Virtuais podem melhorar a utilização da infraestrutura de rede física reutilizando os roteadores ou enlaces individuais para o atendimento de múltiplas requisições virtuais simultâneas ou permitem a agregação de múltiplos recursos físicos para obter um aumento dos recursos virtuais.

As redes virtuais auxiliarão na criação de um novo paradigma para a Internet. Nas propostas para a nova Internet, com a virtualização de redes, haverá múltiplas camadas de abstração e instâncias de rede, fornecendo vários tipos e níveis de serviços ao usuário final [3] [6].

1.3 Problema

Nesse trabalho de pesquisa foi tratado o problema da **Virtualização de Redes**.

Para o pleno funcionamento de um ambiente de **Virtualização de Redes** é necessário resolver os problemas listados na Tabela 1.

Tabela 1 - Problemas com a Virtualização de Redes.

Problemas	Perguntas
P1	Como realizar o mapeamento dos nós da rede virtual (VN) para os nós da rede de substrato (SN)?
P2	Como escolher os melhores caminhos na rede de substrato (SN) para atender as requisições virtuais (VNR – VN requests) baseado no tipo de atendimento (<i>online</i> ou <i>offline</i>), política de atendimento e critério de ordenação?
P3	Como executar o atendimento em tempo real (<i>online</i>) de requisições virtuais (VN request) com garantia de QoS?
P4	Como fazer o atendimento de requisições virtuais com garantia de QoS de forma programada ou agendada (<i>offline</i>)?
P5	Como realizar o atendimento de requisições virtuais sem garantia de QoS com uso mínimo de recursos?
P6	Como realizar o provisionamento offline da rede virtual para a rede de substrato com o objetivo de minimizar os recursos utilizados e maximizar as requisições virtuais atendidas?

As questões relacionadas a mapeamento e atendimento de requisições virtuais, com ou sem garantia de QoS, e a identificação dos melhores caminhos em uma rede de substrato com múltiplas restrições impostas pelas requisições de uma rede virtual são, segundo Fajjari [9], problemas computacionalmente difíceis de resolver e, dependendo do tamanho do problema, NP *hard*.

A análise desses problemas dentro da perspectiva de projeto de protocolos é também complexa e é preciso levar em consideração algoritmos ou heurísticas que garantam a convergência e eliminem laços de roteamento nessas redes, obedecendo a tipos diferentes de métricas de QoS. Um novo *framework* matemático chamado “**álgebra de caminhos**” pode ser utilizado para auxiliar essa análise e permitir a determinação de soluções e sua validação [10].

A álgebra de caminhos foi criada por Carré [10]. Estendida e modificada por Sobrinho [12], Gouda [11] e Herman [13,14] é indicada para a resolução de problemas que apresentam atendimento de requisições de rede mono e multirrestritivas.

Este trabalho de pesquisa resolve todos os problemas listados na Tabela 1, com exceção do problema P1, por meio do uso da álgebra de caminhos.

1.4 Objetivos

A tese tem como objetivos:

- ✓ Estudar as “redes virtuais embutidas” (VN *Embedded*);
- ✓ Resolver os problemas de virtualização de redes enumerados de P2 a P6, listados na Tabela 1, com o uso da álgebra de caminhos.

1.5 Metodologia

A técnica de pesquisa adotada foi exploratória, bibliográfica e baseada em estudo experimental (simulações). Segundo Giorgi [15], o planejamento de pesquisas exploratórias é bastante flexível, de modo que possibilita a consideração dos mais variados aspectos relativos ao fato estudado.

1.6 Organização do Trabalho

O presente trabalho está organizado da seguinte forma:

Capítulo 1 – **Introdução** - apresenta uma introdução ao trabalho de pesquisa, a motivação para a escolha do tema, o problema e sua resposta, objetivos, justificativa, metodologia e organização do trabalho.

Capítulo 2 - **Revisão da Virtualização de Redes** - faz uma revisão dos conceitos teóricos de virtualização de redes e dos principais algoritmos da literatura.

Capítulo 3 - **Álgebra de Caminhos** - apresenta a álgebra de caminhos e mostra como utilizá-la para a resolução de problemas de virtualização de redes.

Capítulo 4 - **Algoritmo para mapeamento de enlaces de rede virtual** - apresenta o algoritmo PAViLiM, desenvolvido para realizar o mapeamento de enlaces de rede virtual. É apresentado o ambiente de simulação ALEVIN e sua integração com os programas do algoritmo PAViLiM.

Capítulo 5 - **Simulação e Discussão de Resultados** - apresenta cenários de simulação para o atendimento de requisições de redes virtuais. É realizada a comparação de resultados de diferentes algoritmos com o algoritmo proposto.

Capítulo 6 - **Conclusão e Trabalhos Futuros** - apresenta a conclusão da tese de doutorado e fornece sugestões de trabalhos futuros para a continuidade desta pesquisa.

2. Revisão da Virtualização de Redes

Este capítulo faz uma revisão dos conceitos teóricos de virtualização de redes e dos principais algoritmos encontrados na literatura para a resolução do problema VNE.

2.1 Introdução à Virtualização de Redes

Virtualização de Redes fornece um mecanismo poderoso de utilização de múltiplas instâncias de rede virtual em uma única rede física que se torna compartilhada. A rede física é chamada de rede de substrato (*substrate network*). A implantação e o desenvolvimento de novos protocolos, testes de novas soluções e arquiteturas para a Internet atual e do futuro podem ser tratados por meio da utilização de virtualização de redes. Há muitos projetos em andamento na Internet seguindo esse novo paradigma. Como exemplo, os projetos CABO, GENI e VINI¹ [16] [17] [18].

Segundo o Grupo de Pesquisa de Redes Virtuais (VNGR) do IRTF (*Internet Resource Task Force*), uma rede virtual inclui todos os recursos existentes em uma rede física real, como roteadores, *hosts*, enlaces, serviços, comutadores e suas interconexões, porém de forma virtual. Há um mapeamento entre os recursos virtuais e os recursos físicos, mas esse mapeamento não é necessariamente de 1 para 1 (um recurso virtual mapeado para um recurso físico), o mapeamento pode ser também de 1 para N (um recurso virtual mapeado para N recursos físicos) ou de N para 1 (N recursos virtuais mapeados para um recurso físico – o que acaba sendo o mais comum) [8].

As instâncias de uma rede virtual (VN) são isoladas entre si e uma falha em uma instância de rede virtual não afeta, obrigatoriamente, outra instância de rede virtual. Contudo, a falha em um elemento físico pode afetar diversas redes virtuais que utilizam esse elemento [8].

¹ O projeto CABO tem como objetivo explorar múltiplas arquiteturas personalizadas e concorrentes sobre uma única infraestrutura física [16]. O projeto GENI, patrocinado pelo NSF, fomenta estudos das redes do futuro e uma de suas áreas de pesquisa trata da virtualização de redes [17]. O projeto VINI tem como objetivo fornecer suporte para a experimentação de novos protocolos sobre uma facilidade de rede compartilhada [18].

As redes virtuais (VNs) consistem de elementos ativos chamados nós virtuais e elementos passivos chamados enlaces virtuais, que são suportados diretamente por uma rede de substrato (SN) ou por outra rede virtual. Tanto a rede virtual quanto a rede de substrato podem ser representadas como um dígrafo (grafo direcionado) [19].

As “Redes Virtuais Embutidas” (VNE ou VN *Embedded*) são redes virtuais administradas por um provedor que não possui acesso ou controle de uma infraestrutura física. O problema encontrado nas redes virtuais embutidas (VNE) consiste no mapeamento de um conjunto de requisições virtuais (VN *requests* ou VNRs) sobre uma rede de substrato, de forma eficiente e que garanta a maior quantidade de requisições atendidas e/ou forneça a maior receita obtida e/ou a menor quantidade de recursos utilizada na rede de substrato [6].

2.2 Conceitos e Terminologia

O VNRG (*Virtual Network Research Group*) está padronizando os conceitos e terminologias encontrados nas redes virtuais (VNs) [8]. Os principais conceitos são: recurso virtual, virtualização e redes virtuais.

a) Recurso Virtual

Um recurso virtual ou lógico é criado ao selecionar um conjunto completo ou parcial de características de um recurso real. Esse subconjunto de características de um recurso real é oferecido ao usuário, que acredita ser proprietário (dono) exclusivo do recurso. Contudo, o acesso do usuário pode ser compartilhado com outros usuários.

Um recurso virtual é uma abstração de um recurso físico ou de outro recurso virtual, possui os mesmos mecanismos e ferramentas que um recurso físico e pode possuir várias interfaces para acesso e gerenciamento do recurso.

b) Virtualização e Estratificação (Layering)

O processo de criação de um recurso virtual - que pode ser um hardware, sistema operacional, dispositivo de armazenamento ou rede - é denominado virtualização. A meta da virtualização é centralizar tarefas administrativas, melhorar a escalabilidade e a utilização dos recursos físicos.

No contexto de rede, a pilha TCP/IP não é um exemplo de rede virtual ou virtualização de redes; é, na verdade, uma estratificação (*layering*) em camadas de protocolos de rede. As técnicas de virtualização de redes envolvem mecanismos dentro e entre os nós de uma rede.

c) Rede Virtual

É uma rede descrita por seus nós e enlaces virtuais e que apresenta as seguintes características:

- Isolamento: as redes virtuais (*virtual networks*) são isoladas entre si, o que fornece segurança aos dados que são transmitidos;
- Particionamento: ocorre o particionamento lógico de uma rede física ou de outra rede virtual para o fornecimento dos recursos virtuais;
- Abstração: é um modo de esconder a complexidade e os detalhes da implementação de uma rede virtual. Uma rede virtual é mapeada em um conjunto de nós e caminhos em uma rede de substrato, esse mapeamento não é visível para o usuário da VN.

2.3 Elementos da Arquitetura de uma Rede Virtual (VN)

- Elementos Tradicionais: host, roteador, comutador, enlace físico e nó físico.
- Elementos Virtuais: nó virtual (*virtual node*) e enlace virtual (*virtual link*).

Uma rede virtual deve possuir mecanismos de gerenciamento para a criação de partições lógicas sobre componentes lógicos ou físicos com o objetivo de permitir sua construção.

Uma requisição de rede virtual (VNR) se traduz em uma rede virtual com nós e enlaces virtuais com certas necessidades de rede, garantia de QoS (*Quality of Service*) e segurança. Os elementos virtuais (*virtual nodes e virtual links*) devem ser mapeados sobre os elementos tradicionais para obter uma solução ótima, que atenda a demanda da requisição e que consuma a menor quantidade de recursos físicos da rede física real, denominada rede de substrato (*substrate network - SN*).

O mapeamento dos nós virtuais para nós físicos e o mapeamento dos enlaces virtuais para caminhos na rede de substrato são definidos como um problema de virtualização de rede embutida, ou problema VNE. Segundo [1] e [2] o problema VNE é NP-Hard e a sua solução deve ser obtida por meio de heurísticas ou algoritmos aproximados. Utilizando técnicas de divisão e conquista (*divide and conquer*) é possível dividir o problema VNE em dois subproblemas: mapeamento de nós virtuais e mapeamento de enlaces virtuais, esses subproblemas são mais simples de resolver, mas cada um deles continua sendo um problema classificado como NP-Hard [1] [2].

2.4 Formulação do Problema VNE

Com a introdução na Internet do paradigma de Infraestrutura como Serviço (IaaS), conforme discutido no Capítulo 1, surgem novos papéis para o ISP tradicional, o provedor de infraestrutura (InP) que possui, implementa e mantém a rede de substrato (SN) e o provedor de serviço (SP) responsável pela implantação de protocolos e serviços fim a fim. Nesse modelo, serviços personalizados podem ser solicitados por grupos de usuários e atendidos por SPs que deverão realizar alocações ótimas sobre a rede de substrato que é mantida pelo InP.

Virtualização de redes é o mecanismo fundamental para fornecer um serviço fim a fim com garantia de QoS em uma arquitetura de redes baseada em IaaS. E o elemento básico para realizar a virtualização de redes é a rede virtual (VN).

Uma das tarefas do SP é a geração de requisições de rede virtual (VNRs) baseada na análise das demandas de serviço do usuário. Cada VNR contém um conjunto de demandas de rede e nenhum parâmetro, relacionado à roteamento, necessário para o fornecimento da qualidade de experiência (QoE) do usuário fim a fim.

Depois que a VNR é criada, um algoritmo é executado para escolher a alocação ótima, segundo um objetivo pré-definido, dessa requisição de rede virtual (demanda virtual) sobre os recursos da SN. O mapeamento que define o relacionamento dos elementos de rede virtual com os respectivos elementos da rede de substrato pode ser dividido em dois estágios:

- Mapeamento de nós virtuais;
- Mapeamento de enlaces virtuais.

O mapeamento de nós virtuais é aquele em que cada nó da VNR é mapeado para um nó na rede de substrato que tenha capacidade suficiente para atender a demanda de recursos do nó virtual. No mapeamento de enlaces virtuais, enlaces virtuais são mapeados para um conjunto de caminhos direcionados na rede de substrato com capacidade de recursos suficiente para atender a demanda do enlace virtual. A Figura 5 ilustra o mapeamento de nós e enlaces virtuais em uma rede de substrato. São apresentadas duas VNRs e é feito o seu respectivo mapeamento.

Figura 5 – VNE de duas requisições de rede virtual (VNRs).

O problema VNE pode ser descrito formalmente por meio da representação da SN e das VNRs como grafos direcionados (dígrafos).

Uma SN é representada por um grafo direcionado: $G=(V,A)$, em que V são os vértices que representam os nós da rede de substrato e A são os arcos do dígrafo que representam as arestas da rede de substrato. Sobre a rede de substrato são atendidas um conjunto de requisições de rede virtual, cada VNR pode ser representada pelo seu próprio dígrafo: $G^k=(V^k,A^k)$, onde V^k representa os vértices da rede virtual k e A^k representa as arestas dessa rede virtual. Cada VNR é embutida (*embedded*) sobre a SN por meio do mapeamento de nós virtuais para nós da rede de substrato e de enlaces virtuais sobre um ou mais caminhos na rede de substrato.

É possível definir duas funções que descrevem a operação de mapeamento de nós e enlaces realizada por algoritmos VNE.

Função de mapeamento de nós virtuais: $f: V^k \rightarrow V$

Função de mapeamento de enlaces virtuais: $g: A^k \rightarrow 2^P$

onde:

f atribui nós virtuais a nós na SN e g realiza o mapeamento de um ou mais caminhos direcionados na SN. Na definição da função g tem-se 2^P , que representa todos os conjuntos de *conjuntos de caminhos* direcionados possíveis na SN. Se a função g é capaz de atribuir para um enlace virtual um conjunto de mais de um elemento, ou seja, o problema VNE terá uma solução *multi-path* e irá permitir enlaces virtuais mapeados a múltiplos caminhos na SN. A solução do problema VNE sendo *multi-path* permitirá a utilização de roteamento também *multi-path*. Caso contrário, o resultado do mapeamento fornecerá uma solução *single-path* ao problema VNE, em que a função g mapeará um único caminho na SN para o enlace virtual.

Ambas as funções, ao realizarem o mapeamento, não devem exceder os recursos dos nós e enlaces na SN. Uma solução ótima ao problema VNE é aquela que realize o mapeamento de nós e enlaces e atenda a todas as restrições impostas pelas VNRs e, adicionalmente, alcance um dado objetivo de otimização: minimizar os custos sobre a rede de substrato, aumentar a receita no atendimento de VNRs, optar por soluções que sejam mais resilientes para a SN ou por soluções que utilizem recursos da SN em determinadas localidades.

A álgebra de caminhos, que será discutida no próximo capítulo, é um *framework* matemático que ajudará a resolver o problema VNE relacionado ao mapeamento de enlaces virtuais em que se possa utilizar um número flexível de restrições de rede, parâmetros de QoS, resiliência, consumo de energia, entre outros.

2.5 Metodologias para a resolução do problema VNE

Nessa seção serão discutidas as soluções propostas por [20] [21] [2] [4] [22] [23] [24] [25] e [26].

2.5.1 Utilização do *stress* do enlace e do nó na rede de substrato

No trabalho de Zhur [20] é feita uma proposta para a solução do problema VNE por meio da minimização e balanceamento do *stress* na rede de substrato. O *stress* de um elemento (nó ou enlace) é definido como o número de instâncias virtuais mapeadas sobre esse elemento. São resolvidos dois problemas:

- **VNA-I** – problema VNE, que realiza o atendimento de VNRs *online* e sob demanda e sem reconfiguração da rede de substrato;
- **VNA-II** – problema VNE, que realiza uma reconfiguração da SN por meio da reconfiguração das VNRs de maior prioridade que foram mapeadas *online*.

Para resolver esses problemas foram propostos dois algoritmos heurísticos:

- a) *Algoritmo de Atribuição de VNs Básico*: seu objetivo é minimizar o *stress* dos nós e enlaces na rede de substrato. É utilizada a seguinte equação, que deve ser minimizada em cada atendimento:

$$\alpha \cdot \max S_N + \beta \cdot \max S_L$$

onde

$$0 \leq \alpha, \beta \leq 1 \text{ e } \alpha + \beta = 1$$

e

$$S_N \text{ e } S_L$$

representam o *stress* do nó ou enlace respectivamente na rede de substrato.

- b) *Algoritmo de Subdivisão*: realiza a divisão de cada requisição de rede virtual (VNR) em um conjunto de sub-VNs conectadas em uma topologia estrela. Depois dessa divisão o mapeamento é realizado em cada sub-VN e novamente é atendido o objetivo de minimizar o *stress* dos nós e enlaces.

A limitação dessa abordagem é que em ambos os algoritmos só é levado em conta, como critério de otimização para que ocorram os mapeamentos, o *stress* dos nós e enlaces, não é considerada nenhuma restrição imposta pelos recursos existentes na rede de substrato e pode-se chegar a mapeamentos que não sejam válidos.

2.5.2 Restrições de largura de banda em um cenário *offline*

A proposta de Lu [21] considera para a solução do problema VNE apenas as restrições de largura de banda e um cenário com atendimento das requisições de rede virtual realizada de maneira *offline*. As topologias de VN solicitadas são limitadas, também, àquelas do tipo *backbone-star*. Essas topologias são compostas de dois tipos de nós: nós de acesso e nós de *backbone*. Os nós de *backbone* conectam os nós de acesso. O algoritmo que foi desenvolvido utiliza programação linear e tenta minimizar uma função objetivo quadrática. Em cada iteração do algoritmo ocorrem mudanças do mapeamento dos nós de *backbone* para realizar o atendimento de uma determinada VNR.

As limitações dessa abordagem são:

- Uma única restrição (largura de banda) é levada em conta para realizar o mapeamento de nós e enlaces;
- Cenário *offline* – não existe possibilidade de atendimento *online*;
- Topologias *backbone-star*, impossibilidade de trabalhar com qualquer topologia de VNs.

2.5.3 Otimização da receita média

A proposta de Yu [2] procura maximizar a receita média de longo prazo que é definida como a soma ponderada da largura de banda e de recursos de CPU das demandas de VNR que serão atendidas. Apenas as restrições de CPU e de largura de banda (BW) são levadas em consideração nos algoritmos propostos. Para cada requisição de rede virtual (VNR), os mapeamentos de nós e enlaces para a SN ocorrem

em dois estágios: mapeamento de nós virtuais para nós reais e, em seguida, mapeamento de enlaces virtuais para um ou múltiplos caminhos na SN. O mapeamento de nós utiliza um algoritmo “guloso” (*greedy algorithm*) que escolhe um conjunto elegível de nós SN para cada nó virtual e, então, faz o mapeamento desses nós baseado na quantidade de recursos disponíveis. O mapeamento escolhido é o primeiro mapeamento de nós virtuais para nós da rede de substrato que atende as restrições da VNR. O mapeamento de enlaces virtuais pode utilizar dois algoritmos: um algoritmo *single-path* e um algoritmo *multi-path*.

A solução *single-path* encontra um mapeamento de um enlace virtual para um único caminho na rede de substrato. O algoritmo utilizado é o ***k-shortest path***, que resolve o problema clássico de fluxo sem divisões (UFP) aumentando o valor de ***k*** até que se encontre um caminho na SN que possua largura de banda suficiente para mapear o respectivo enlace virtual. A complexidade do algoritmo para uma rede de substrato com ***N*** nós é $O(N \log N + kN)$, o que mostra a eficiência computacional do algoritmo para valores pequenos de ***k***.

A solução *multi-path* encontra para um enlace virtual um conjunto de caminhos na rede de substrato. O algoritmo utilizado realiza uma divisão flexível do tráfego em múltiplos caminhos sobre a rede de substrato, obedecendo a uma taxa máxima de divisão (*split ratio*). O algoritmo resolve o problema do fluxo de múltiplas divisões (MFP). A complexidade do algoritmo é polinomial.

As limitações dessa abordagem são:

- Otimização por receita, que utiliza um conceito subjetivo de receita;
- Conjunto limitado de restrições, em que se trabalha apenas com restrições de largura de banda e de recursos de CPU;
- O mapeamento de nós virtuais utiliza um algoritmo “guloso” que não garante que a solução encontrada fornecerá um mapeamento ótimo na segunda fase de mapeamento dos enlaces virtuais.

2.5.4 Mapeamento coordenado de nós e enlaces

A proposta de Chowdhury [4] é resolver as duas fases do problema VNE de forma coordenada e integrada. Cada fase dos mapeamentos não deve ser tratada de forma isolada. As metas dessa metodologia são minimizar os custos do VNE e maximizar a receita e o número de VNRs que são atendidos (taxa de aceitação). O custo do VNE é definido como a soma da largura de banda e CPU consumidos na SN ao realizar um atendimento de uma VNR. Informações da localização geográfica são restrições informadas nas VNRs e características dos recursos dos nós físicos na SN. Existe uma distância não negativa na VNR para cada localização de nó virtual. Essa distância deve ser entendida como a distância máxima a que um nó virtual pode estar de uma determinada localização física desejada.

O estágio do mapeamento de nós começa com a criação de um grafo SN aumentado usando um conjunto de meta-nós que representam cada um dos nós virtuais. Nesse grafo SN aumentado, o requisito de localização, presente em cada nó virtual, e as restrições de capacidade são atendidas em um *cluster* de nós da rede de substrato onde o nó virtual está conectado. O algoritmo desenvolvido utiliza a técnica de programação inteira mista (MIP) para resolver o problema VNE sobre o grafo SN aumentado e persegue o objetivo de minimização do custo de atendimento de uma VNR. A abordagem MIP proposta é NP-completo, mas, por meio de um mecanismo de relaxação, pode-se transformar a abordagem em um problema de programação linear inteira (ILP), que de forma aproximada faça com que o algoritmo de mapeamento de nós convirja em um tempo computacional eficiente. Existem duas formas de relaxação por meio de simplificação e arredondamentos do problema: a determinística e a aleatória, que geram dois algoritmos diferentes:

- DViNE – que utiliza arredondamento determinístico;
- RViNE – que utiliza arredondamento aleatório.

O mapeamento dos enlaces virtuais é realizado utilizando um dos algoritmos (*single-path* ou *multi-path*) propostos por Yu [2].

As limitações dessa abordagem são:

- Otimização por receita, que utiliza um conceito subjetivo de receita;
- Conjunto limitado de restrições, onde se trabalha apenas com restrições de largura de banda, CPU e localidade;
- Metas de otimização extremamente ambiciosas, que abrangem a minimização de custos, a maximização da receita e da taxa de aceitação, ao mesmo tempo.

2.5.5 Mapeamento de nós e enlaces em um único estágio

A proposta de Lischka [22] é resolver o problema VNE em um único estágio, ou seja, realizar o mapeamento de nós e enlaces virtuais em uma única fase. Por meio de uma simplificação, o problema VNE é transformado em um problema bem conhecido NP-completo de detecção de isomorfismo em subgrafos (SID).

Na teoria de grafos, um isomorfismo de grafos G e H ($G \simeq H$) é uma bijeção entre o conjunto de vértices de G e H e a função m , definida como:

$$m: V(G) \rightarrow V(H)$$

tal que quaisquer dois vértices i e j de G são adjacentes em G se e somente se $m(i)$ e $m(j)$ são adjacentes em H .

O problema SID tenta encontrar um subgrafo G_{sg} de G , onde $G_{sg} \subset G$, tal que $G_{sg} \simeq H$. O problema VNE é resolvido pela modificação de uma heurística SID, consistindo de encontrar-se um isomorfismo do subgrafo representando uma requisição de rede virtual (VNR) e, ao obedecer suas demandas, encontrar o mapeamento de nós e enlaces virtuais na rede de substrato.

As limitações dessa abordagem são:

- Necessidade de *backtrack*, que acontece toda vez que uma requisição de rede virtual não pode ser atendida. O *backtrack* desfaz a última VNR que foi mapeada na tentativa de mapear essa nova requisição que acabou de chegar;
- Redução do problema VNE por meio da suposição de *delay* igual a 1 (um) para todo enlace virtual e físico;
- Conjunto limitado de restrições, onde se trabalha apenas com restrições de largura de banda e CPU.

2.5.6 Introdução do conceito de nós escondidos (*hidden hops*)

No trabalho de Botero [23] é introduzido o conceito de nós escondidos (*hidden hops*), os quais devem ser considerados quando se realiza o mapeamento de enlaces virtuais para caminhos na rede de substrato. Um nó escondido é um nó físico que é intermediário em um caminho utilizado no mapeamento de enlaces virtuais para caminhos na SN. Esse nó intermediário consome recursos de processamento (CPU), uma vez que faz o encaminhamento de pacotes.

O trabalho apresenta um algoritmo de otimização que, em uma única fase, faz o mapeamento de nós e enlaces, implementando uma heurística de um algoritmo “guloso” modificado, que obedece ao consumo de recursos pelos nós escondidos. O problema VNE é modelado como um problema de fluxo sem divisão (UFP), no qual dois tipos de VNRs são suportados: sem restrições e com restrições de largura de banda e CPU. As VNRs com restrições são atendidas primeiro, mas o *backtrack* pode acontecer e VNRs que não possuem restrições de largura de banda ou CPU são desfeitas na tentativa de mapear uma nova requisição de rede virtual que acabou de chegar.

As limitações dessa abordagem são:

- Necessidade de *backtrack*, que acontece toda vez que uma requisição de rede virtual não pode ser atendida;
- Utilização de um algoritmo “guloso” para executar o mapeamento de nós e enlaces virtuais;
- Conjunto limitado de restrições, onde se trabalha apenas com restrições de largura de banda e CPU.

2.5.7 Custo ótimo em um único estágio

A proposta de Houidi [24] é realizar o mapeamento de nós e enlaces virtuais em um único estágio. A solução proposta utiliza uma heurística que se vale de uma modelagem do problema VNE por meio de programação inteira mista (MIP). A função objetivo deve ser otimizada para se obter o “custo ótimo”, que é, na verdade, o menor custo possível obtido após o mapeamento dos nós e enlaces virtuais sobre a SN.

Como os problemas MIP são NP-completos, a principal limitação da solução apresentada é a escalabilidade. O tempo de execução do algoritmo é proibitivo para redes grandes.

2.5.8 Algoritmo inspirado no Google *PageRank*

A proposta de Cheng [25] é inspirada no algoritmo que foi desenvolvido por Larry Page e Sergey Brin, usado no Google e chamado de ***PageRank***. A solução do problema VNE é obtida por meio da análise das topologias da SN e da VN, onde ocorre a classificação (*rank*) e ordenação dos nós e seu mapeamento. Os critérios utilizados para a classificação dos nós na SN e na VN são: capacidade de CPU disponível ou solicitada, quantidade e capacidade dos enlaces existentes em cada nó e relacionamento entre nós vizinhos. São propostas duas abordagens para a solução do problema VNE:

- Mapeamento de nós e enlaces em duas fases: por meio de um primeiro algoritmo que faz a classificação (*rank*) dos nós e o seu mapeamento, seguido de um algoritmo *single-path* ou *multi-path* para mapeamento dos enlaces;
- Mapeamento de nós e enlaces de forma coordenada: um único algoritmo que utiliza a classificação (*rank*) de nós e pesquisa por largura (*breadth-first search*) para fazer o mapeamento de nós e enlaces de forma integrada. Esse algoritmo pode realizar um procedimento de *backtrack* quando um novo mapeamento se mostra ineficaz ou impossível de ser atendido. O *backtrack* desfaz o último mapeamento realizado com sucesso na tentativa de atender uma nova solicitação que não pôde ser atendida anteriormente.

As limitações dessa abordagem são:

- Mapeamento de nós e enlaces em uma única fase, onde existe a necessidade de *backtrack* toda vez que uma requisição de rede virtual não pôde ser atendida. O *backtrack* tem o objetivo de aumentar a taxa de atendimento de requisições virtuais, mas não existe uma garantia de que isso realmente irá acontecer;
- Não existem simulações suficientes para se garantir que a técnica de mapeamento de nós por classificação (*rank*) é a mais eficaz e a que produz os melhores resultados quando comparada com as demais abordagens apresentadas;
- Conjunto limitado de restrições, no qual se trabalha apenas com restrições de largura de banda e de recursos de CPU.

2.5.9 Discussão do problema VNE em um ambiente *inter-domain*

Todas as propostas já vistas para a solução do problema VNE supõem um único provedor de infraestrutura de rede (InP). Esse cenário é conhecido como *intra-domain* VNE. Em um ambiente com múltiplos provedores de infraestrutura de rede ocorre o problema conhecido como **VNE *inter-domain***. O grande desafio desse problema é

realizar o mapeamento de VNs fim a fim utilizando InPs diferentes sem impor restrições que afetem a autonomia de cada provedor. Cada provedor deve ser capaz de mapear parte da VNR segundo suas políticas administrativas internas. Portanto, devem existir acordos de serviços entre os provedores que garantam a comunicação, troca de informação e utilização de recursos disponíveis na rede de infraestrutura de cada provedor. Duas propostas são citadas para resolver o problema VNE *inter-domain*: Chowdhury [26] e Houidi [24].

A primeira proposta, de Chowdhury [26], apresenta um *framework* chamado PolyViNE, baseado em políticas fim a fim que permitem o atendimento de VNs utilizando recursos de múltiplos provedores. PolyVine define um protocolo distribuído que irá coordenar o processo de mapeamento de VNs usando os recursos fornecidos por múltiplos InPs e com o objetivo de garantir preços competitivos para o SP.

A segunda proposta, de Houidi [24], utiliza algoritmos heurísticos e exatos para provisionar VNs em múltiplos provedores de infraestrutura. A proposta utiliza o modelo de virtualização de redes de [4], apresentado no capítulo 1 (seção 1.1) deste trabalho. Deve existir um provedor com papel de mediador (*broker*) na rede, o qual fará a negociação de recursos de rede virtual solicitados entre InPs. O problema VNE poderá ser resolvido por meio de técnicas de programação linear mista, fluxo máximo (*max-flow*) ou corte mínimo (*min-cut*). Para o mapeamento de enlaces virtuais os algoritmos utilizados poderão ser *single-path* ou *multi-path*.

2.5.10 Síntese das propostas para a solução do problema VNE

A maioria das propostas e metodologias discutidas neste capítulo para a solução do problema VNE, trabalham com restrições lineares (CPU nos nós e BW nos enlaces) nas VNRs. O mapeamento dos nós e enlaces pode ser realizado em uma ou duas fases. O mapeamento de enlaces pode ser feito por meio de algoritmos *single-path* ou *multi-path*.

Os algoritmos de mapeamento de enlaces *multi-path* utilizam programação linear exata. Os algoritmos de mapeamento de enlaces *single-path* buscam soluções por meio do roteamento que usa a métrica de caminhos mais curtos (*shortest path routing*).

Nenhum algoritmo que utilize a abordagem *multi-path* suporta restrições não lineares como disponibilidade ou taxa de perda de pacotes (*packet loss rate*).

Os algoritmos de roteamento em caminhos mais curtos são mono-restritivos; havendo suporte no algoritmo a mais de uma restrição, elas devem ser combinadas para se obter um único valor de custo, utilizado como métrica por esses algoritmos.

2.6 Considerações Finais do Capítulo

O capítulo apresentou o problema de virtualização de redes embutidas (VNE) que pode ser mapeado formalmente por grafos direcionados. A rede de substrato (SN) é representada por um grafo direcionado e os seus recursos são mapeados como métricas sobre esse grafo. As solicitações dos usuários são mapeadas como requisições de redes virtuais (VNRs) que devem ser atendidas de forma ótima para economizarem-se recursos na SN e obter-se maior receita no negócio “virtualização”.

Foram apresentadas diversas metodologias e algoritmos para resolução do problema VNE, todas as metodologias possuem limitações que se originam de simplificações do problema VNE e da falta de flexibilidade para poder se trabalhar com quaisquer tipos de métricas nas redes virtuais (VNs) e quaisquer funções de combinação dessas métricas, a álgebra de caminhos é um *framework* matemático que resolve essas limitações.

O próximo capítulo fará uma apresentação da álgebra de caminhos e mostrará como utilizá-la para resolução do problema de virtualização de redes embutidas (VNE).

3. Álgebra de Caminhos

Este capítulo faz uma apresentação da álgebra de caminhos e mostra como utilizá-la para a resolução de problemas de virtualização de redes embutidas (VNE). A habilidade da álgebra de caminhos para considerar restrições lineares e não lineares multirrestritivas fornece uma grande flexibilidade para o mapeamento de enlaces virtuais [27].

3.1 Conceitos da Álgebra Universal

A álgebra universal é o ramo da matemática que estuda as estruturas denominadas álgebras. Uma álgebra consiste em um conjunto associado a uma coletânea de operações [28].

As próximas seções apresentam algumas definições da álgebra universal que são utilizadas pela álgebra de caminhos.

3.1.1 Conjunto

O conceito de conjunto foi introduzido no século XIX por Georg Cantor [29] e desde então sua definição não foi alterada. Um conjunto consiste em uma coletânea de itens de qualquer natureza. Cada item do conjunto é denominado elemento ou membro. Como exemplos é possível citar [10]:

- ✓ o conjunto de todos os peixes do aquário de Mônaco;
- ✓ o conjunto de todos os símbolos fonéticos internacionais;
- ✓ o conjunto dos sete pecados capitais;
- ✓ o conjunto dos números inteiros;
- ✓ o conjunto de todos os caminhos entre um nó origem e um nó destino em uma rede.

3.1.2 Operações sobre Conjuntos

A álgebra define para cada conjunto uma coletânea de operações. As operações podem ser executadas sobre um ou mais elementos do conjunto e seus operadores são

classificados segundo sua “n-aridade” (*n-ary*). Uma operação que atua sobre um único elemento do conjunto utiliza um operador chamado *1-ary* ou unário. Uma operação que é executada entre dois elementos de um conjunto utiliza um operador *2-ary* ou binário. Quando a operação é entre três elementos do conjunto tem-se um operador *3-ary* ou ternário. Operações de grau superior ou com “n-aridade” não especificada são representadas por meio de símbolos de função com seus argumentos entre parênteses e separados por vírgula, por exemplo: $f(x_1, \dots, x_n)$.

3.1.3 Álgebra de Conjuntos

Os conjuntos satisfazem várias leis ou identidades, as quais são definidas a seguir [30]:

- *Lei da Idempotência:* diz que operações de união e intersecção, quando aplicadas a um único conjunto, fornecem como resultado o próprio conjunto;
- *Lei da Associatividade:* define que quando uma mesma operação é aplicada entre mais de dois conjuntos o resultado é o mesmo, independentemente da ordem em que as operações foram executadas;
- *Lei de Comutatividade:* diz que a ordem dos conjuntos, em uma operação de união ou intersecção, não altera o resultado;
- *Lei de Distributividade:* define a forma como a operação de união pode ser distribuída com relação à intersecção e vice-versa;
- *Leis de Identidade:* definem os casos específicos de união e intersecção de um conjunto qualquer com o conjunto vazio ($\emptyset$) ou com o conjunto universo (U);
- *Lei da Involução:* diz que o complementar do complementar de um conjunto é o próprio conjunto;
- *Leis Complementares:* definem os casos de operações aplicadas a conjuntos e a seus complementares;
- *Leis de De Morgan:* dizem que o complementar da união entre dois conjuntos é igual à intersecção dos complementares desses conjuntos e vice-versa, invertendo-se as operações de união e intersecção [31].

A Tabela 2 apresenta as leis da álgebra de conjuntos.

Tabela 2 – Leis da Álgebra de Conjuntos.

Álgebra de Conjuntos	
Lei da Idempotência	
1a) $A \cup A = A$	1b) $A \cap A = A$
Lei da Associatividade	
2a) $(A \cup B) \cup C = A \cup (B \cup C)$	2b) $(A \cap B) \cap C = A \cap (B \cap C)$
Lei da Comutatividade	
3a) $A \cup B = B \cup A$	3b) $A \cap B = B \cap A$
Lei da Distributividade	
4a) $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$	4b) $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
Leis de Identidade	
5a) $A \cup \emptyset = A$	5b) $A \cap U = A$
6a) $A \cup U = U$	6b) $A \cap \emptyset = \emptyset$
$\emptyset$ é o conjunto vazio e U é o conjunto universo	
Lei da Involução	
7) $(A')' = A$	
Leis Complementares	
8a) $A \cup A' = U$	8b) $A \cap A' = \emptyset$
9a) $U' = \emptyset$	9b) $\emptyset' = U$
Leis de De Morgan	
10a) $(A \cup B)' = A' \cap B'$	10b) $(A \cap B)' = A' \cup B'$

3.1.4 Princípio da Dualidade

Na matemática, uma dualidade traduz conceitos, teoremas ou estruturas matemáticas em outros conceitos, teoremas ou estruturas. A tradução é feita de uma forma um-para-um por meio de uma operação denominada involução. Por exemplo, se a equação dual de **A** é **B**, então a equação dual de **B** é **A** [32].

Cada linha da Tabela 2 possui duas colunas e dois pares de equações, identificadas por um número e um literal: **a)** ou **b)**. As equações da coluna **a)** e da coluna **b)** possuem um relacionamento de dualidade em cada linha. Com uma equação encontrada em uma coluna da Tabela 2, pode-se obter a equação dual da outra coluna através da substituição de cada ocorrência de $\cup$, $\cap$, U e $\emptyset$ por, respectivamente, $\cap$, $\cup$, $\emptyset$ e U [30].

3.1.5 Teoria das Redes Complexas

A teoria das redes complexas utiliza o formalismo matemático da teoria dos grafos sendo utilizada para se modelar e caracterizar uma estrutura de rede [33]. As redes complexas e os grafos são muitas vezes utilizados como sinônimos, mas existe uma diferença conceitual entre os dois termos. Os grafos são uma representação matemática das redes complexas, cujas propriedades de seus nós e de suas arestas são baseadas no sistema que se está estudando [33].

Uma rede é definida, do ponto de vista matemático, com base em quatro primitivas e dois axiomas [14]:

Primitiva 1 – **V** é um conjunto de elementos denominados vértices.

Primitiva 2 – **E** é um conjunto de elementos denominados arestas.

Primitiva 3 – **f** é uma função cujo domínio é **E** e cuja imagem está contida em **V**.

Primitiva 4 – **g** é uma função cujo domínio é **E** e cuja imagem está contida em **V**.

Axioma 1 – **V** é um conjunto finito e não vazio.

Axioma 2 – **E** é um conjunto finito.

A Figura 6 mostra a representação de uma rede com dois vértices e uma aresta. Na figura os conjuntos de vértices e arestas são: $V=\{(o),(d)\}$ e $E=\{(e)\}$.

Figura 6 – Representação de uma rede, seus vértices e aresta.

Uma aresta pode ser orientada ou não orientada. Na Figura 6, a aresta **e** é uma aresta orientada com o vértice **o** sendo um vértice de origem e o vértice **d** sendo o vértice de destino para a aresta. A função **f** pode ser definida como uma função que aplicada sobre uma aresta orientada fornece como resultado o vértice de origem **f(e): o = f(e)**. A função **g** pode ser definida como uma função que aplicada sobre uma aresta orientada fornece como resultado o vértice de destino **g(e): d=g(e)**.

3.1.6 Par Ordenado e Produto Cartesiano

Um par ordenado de um conjunto é formado por dois elementos em uma ordem fixa. Sejam x e y membros do conjunto **A**, existem dois pares ordenados possíveis: (x,y) e (y,x) . Estes pares ordenados são diferentes a menos que o elemento x seja igual ao elemento y [34].

Dados dois conjuntos **A** e **B**, não vazios, chama-se de produto cartesiano de **A** por **B** o conjunto indicado por $A \times B$, formado por todos os pares ordenados nos quais o primeiro elemento pertence ao conjunto **A** e o segundo elemento pertence ao conjunto **B**, conforme a Eq. (1) [35].

$$A \times B = \{(x,y) \mid x \in A \text{ e } y \in B\} \quad (1)$$

3.1.7 Relação Binária

Uma relação é uma associação ou comparação entre objetos. Dois objetos podem ou não estar relacionados por alguma regra ou por alguma forma de associação. Alguns exemplos de relações comuns na matemática são:

- Múltiplo de;
- Menor que;
- É paralelo a;
- É subconjunto de.

Uma relação é dita binária quando relaciona dois elementos de um conjunto ou dois elementos de dois conjuntos diferentes. Sejam x e y dois elementos dos conjuntos **A** e **B**, respectivamente. Existindo uma relação entre esses elementos utiliza-se a notação: $x \mathfrak{R} y$. Se o elemento y é único, essa relação é uma função representada por $y = f(x)$. [14].

Uma relação binária $\mathfrak{R}$ do conjunto **A** para o conjunto **B** pode ser representada como um par ordenado (x,y) de $A \times B$, ou seja, uma relação binária é um conjunto que está contido (subconjunto) ou é resultante do produto cartesiano entre dois conjuntos [34].

Uma aresta orientada **e** é uma relação binária $\mathfrak{R}$ de um par de vértices origem e destino sendo representada pelo par ordenado **(o,d)** ou por meio da notação: **o** $\mathfrak{R}$ **d**, onde **o** é o vértice de origem, ou vértice do qual a aresta orientada sai, e **d** é o vértice de destino, ou vértice no qual a aresta orientada entra [13].

3.1.8 Propriedades das Relações Binárias

Seja $\mathfrak{R}$ uma relação binária que associa elementos do conjunto **A** com elementos do mesmo conjunto **A**, diz-se que $\mathfrak{R}$ é uma relação definida em **A** ou que $\mathfrak{R}$ é uma relação sobre **A**.

Uma relação $\mathfrak{R}$ sobre A pode apresentar as seguintes propriedades:

- Reflexiva: quando $x \mathfrak{R} x$ para todo $x \in A$.
- Antirreflexiva: não existe nenhum $x \in A$ que satisfaça a relação $x \mathfrak{R} x$.
- Não-reflexiva: quando no conjunto A existe pelo menos um x que é reflexivo e pelo menos um y que é antirreflexivo.
- Simétrica: quando $x \mathfrak{R} y$ implica $y \mathfrak{R} x$ para todo $x, y \in A$.
- Antissimétrica: quando $x \mathfrak{R} y$ e $y \mathfrak{R} x$ apenas quando $x=y$ para todo $x, y \in A$.
- Não-simétrica: quando $x \mathfrak{R} y$ não implica $y \mathfrak{R} x$ para todo $x, y \in A$.
- Transitiva: quando $x \mathfrak{R} y$ e $y \mathfrak{R} z$ implica $x \mathfrak{R} z$ para x, y e $z \in A$.

3.2 Estruturas de Ordem

A teoria das estruturas de ordem é um ramo da matemática no qual as relações binárias que ocorrem nessas estruturas são relações de ordem [14] [36].

3.2.1 Ordenação

Uma relação sobre um conjunto A é chamada de relação de ordem ou ordenação de A quando essa relação é reflexiva, antissimétrica e transitiva.

Neste trabalho, uma ordenação será denotada pelos seguintes símbolos especiais: $\succsim, >, \preceq, <$. Esses símbolos têm o seguinte significado:

- Superior, mais otimizado ou equivalente: $\succsim$
- Estritamente superior ou estritamente mais otimizado: $>$
- Inferior, menos otimizado ou equivalente: $\preceq$
- Estritamente inferior ou estritamente menos otimizado: $<$

Exemplo da utilização dos símbolos de ordenação:

Sejam x e y dois elementos comparáveis pertencentes ao conjunto $\mathbf{A}$. Se $x \succcurlyeq y$, lê-se x é superior, mais otimizado ou equivalente a y . Quando $x \preccurlyeq y$, diz-se que x é inferior, menos otimizado ou equivalente a y .

Através do símbolo $\preccurlyeq$ é possível descrever as propriedades características de uma ordenação:

- Reflexiva: $x \preccurlyeq x$, para todo $x \in A$.
- Antissimétrica: se $x \preccurlyeq y$ e $y \preccurlyeq x$, então $x=y$.
- Transitiva: se $x \preccurlyeq y$ e $y \preccurlyeq z$, então $x \preccurlyeq z$.

Dois elementos x e y pertencentes a $\mathbf{A}$ são ditos comparáveis se $x \preccurlyeq y$ ou $y \preccurlyeq x$; caso contrário, esses elementos são incomparáveis. Se todos os elementos de $\mathbf{A}$ tomados dois a dois são comparáveis, a ordenação de $\mathbf{A}$ é total e o conjunto $\mathbf{A}$ é chamado de conjunto totalmente ordenado ou cadeia (*chain*) [10]. Caso o conjunto $\mathbf{A}$ possua apenas um subconjunto com elementos que são comparáveis, a ordenação é parcial e o conjunto $\mathbf{A}$ é chamado de conjunto parcialmente ordenado.

3.2.2 Ordenação de Produto

Uma ordenação de produto é um produto cartesiano ordenado, gerado a partir de dois conjuntos ordenados quaisquer, onde dois pares ordenados (a_1, b_1) e (a_2, b_2) pertencentes ao produto cartesiano de $A \times B$ obedecem a uma relação de ordem [14]:

$$(a_1, b_1) \succ (a_2, b_2) \text{ se e somente se } a_1 \succ a_2 \text{ e } b_1 \succ b_2$$

ou

$$(a_1, b_1) \prec (a_2, b_2) \text{ se e somente se } a_1 \prec a_2 \text{ e } b_1 \prec b_2$$

ou

$$(a_1, b_1) = (a_2, b_2) \text{ se e somente se } a_1 = a_2 \text{ e } b_1 = b_2$$

Uma ordenação de produto pode ser generalizada quando se aplica a produtos cartesianos generalizados sobre n conjuntos ordenados $A_1 \times A_2 \times A_3 \times \dots \times A_n$ onde dois elementos desse produto cartesiano:

$$(a_1, a_2, a_3, \dots, a_n) \text{ e } (b_1, b_2, b_3, \dots, b_n)$$

obedecem a seguinte relação de ordem:

$$(a_1, a_2, a_3, \dots, a_n) > (b_1, b_2, b_3, \dots, b_n)$$

$$\text{se e somente se } a_1 > b_1 \text{ e } a_2 > b_2 \text{ e } a_3 > b_3 \text{ e } \dots a_n > b_n$$

ou

$$(a_1, a_2, a_3, \dots, a_n) < (b_1, b_2, b_3, \dots, b_n)$$

$$\text{se e somente se } a_1 < b_1 \text{ e } a_2 < b_2 \text{ e } a_3 < b_3 \text{ e } \dots a_n < b_n$$

ou

$$(a_1, a_2, a_3, \dots, a_n) = (b_1, b_2, b_3, \dots, b_n)$$

$$\text{se e somente se } a_1 = b_1 \text{ e } a_2 = b_2 \text{ e } a_3 = b_3 \text{ e } \dots a_n = b_n$$

3.2.3 Ordenação Lexicográfica ou Léxica

Uma ordenação lexicográfica ou léxica é um produto cartesiano ordenado, gerado a partir de dois conjuntos ordenados quaisquer, onde dois pares ordenados (a_1, b_1) e (a_2, b_2) pertencentes ao produto cartesiano de $A \times B$ obedecem a uma relação de ordem [37]:

$$(a_1, b_1) > (a_2, b_2) \text{ se e somente se } a_1 > a_2 \text{ ou } a_1 = a_2 \text{ e } b_1 > b_2$$

ou

$$(a_1, b_1) < (a_2, b_2) \text{ se e somente se } a_1 < a_2 \text{ ou } a_1 = a_2 \text{ e } b_1 < b_2$$

ou

$$(a_1, b_1) = (a_2, b_2) \text{ se e somente se } a_1 = a_2 \text{ e } b_1 = b_2$$

A ordenação léxica também é chamada de ordenação de dicionário, pois um alfabeto pode ser interpretado como um conjunto ordenado de letras e as palavras como um subconjunto do produto cartesiano do alfabeto. Em um dicionário as palavras são ordenadas e seguem uma ordenação léxica. Dadas duas palavras nesse dicionário, formadas pela sequência de letras: " $a_1, a_2, a_3, \dots, a_n$ " e " $b_1, b_2, b_3, \dots, b_n$ " a palavra " $a_1, a_2, a_3, \dots, a_n$ " aparece antes no dicionário do que a palavra " $b_1, b_2, b_3, \dots, b_n$ " se e somente se:

$$a_1 < b_1, \text{ ou } a_1 = b_1 \text{ e } a_2 < b_2$$

ou

$$a_1 = b_1 \text{ e } a_2 = b_2 \text{ e } a_3 < b_3$$

ou $\dots$

$$a_1 = b_1 \text{ e } a_2 = b_2 \text{ e } a_3 = b_3 \text{ e } \dots \text{ e } a_{n-1} = b_{n-1} \text{ e } a_n < b_n$$

A ordenação léxica parte do princípio de que todas as palavras possuem o mesmo comprimento e, se isso não for verdade, basta acrescentar caracteres em “branco” em uma palavra que originalmente tem menos caracteres e considerar que o caracter “branco” é um elemento mínimo² para a ordenação no alfabeto.

3.3 Álgebra de Caminhos

A álgebra de caminhos foi proposta por Carré [10], modificada e estendida por Gouda [11], Sobrinho [12] e Herman [13,14]. Ela pode resolver diretamente as seguintes questões listadas na Tabela 3 [10].

² Quando o conjunto V é totalmente ordenado existe um elemento no conjunto, denominado elemento mínimo $\emptyset$, que satisfaz a relação de ordenação: $\emptyset < v$ para todo $v \in V$ [10].

Tabela 3 – Descrição das questões resolvidas pela Álgebra de Caminhos.

Questão	Descrição
Q1	Determinação de todos os caminhos acessíveis entre um nó origem e um nó destino.
Q2	Determinação de todos os caminhos mais curtos entre um nó origem e um nó destino.
Q3	Descoberta do caminho crítico (mais longo).
Q4	Identificação dos caminhos mais confiáveis.
Q5	Identificação dos caminhos de maior capacidade.
Q6	Enumeração de todos os caminhos existentes entre quaisquer pares de nós origem e destino – par ordenado (o,d).
Q7	Listagem de todos os caminhos simples ³ .
Q8	Listagem de todos os caminhos elementares ⁴ .

As próximas seções apresentam a história e os principais conceitos da álgebra de caminhos.

3.3.1 História da Álgebra de Caminhos

A álgebra de caminhos “nasce” em 1979 com Carré [10], que propôs uma nova álgebra para a validação e análise da convergência de problemas de roteamento monorrestritivos dentro de um único paradigma, independente da topologia de rede utilizada.

Sobrinho [12], em 2002, e Gouda [11], em 2003, modificaram a álgebra de caminhos para formalizar a utilização de múltiplas métricas que podem ser utilizadas em roteamento multirrestritivo em uma rede de telecomunicações.

Herman [13,14], em 2007, fez uma extensão aos trabalhos anteriores e criou um novo *framework* matemático que introduz um novo critério de desempate entre caminhos com sínteses iguais, o qual permite a distinção desses caminhos considerados anteriormente equivalentes. Esse *framework* utiliza o princípio da ordenação léxica multidimensional e pode ser utilizado sem modificações para

³ Um caminho é chamado simples se não passar duas vezes pela mesma aresta.

⁴ Um caminho é chamado elementar se não passar duas vezes pelo mesmo vértice.

diferentes aplicações que requerem um tratamento com múltiplas restrições e regras de negócio [13,14].

3.3.2 Conceitos da Álgebra de Caminhos

Segundo Carré [10], a álgebra de caminhos é definida por um conjunto **P** com duas operações binárias **V** e **•**. A operação **V** é chamada operação de junção ou **união** de **P**, e um elemento **x V y** é chamado de união de **x** e **y**. A operação **•** é chamada multiplicação ou **produto**, e um elemento **x • y** é descrito como o produto de **x** e **y**, nessa ordem. Por simplicidade, um produto de **x • y** pode ser escrito como **xy** [10].

A álgebra de caminhos possui as seguintes propriedades:

- (i) A operação **V** é idempotente, comutativa e associativa.

$x \vee x = x$	para todo $x \in P$,
$x \vee y = y \vee x$	para todo $x, y \in P$,
$(x \vee y) \vee z = x \vee (y \vee z)$	para todo $x, y, z \in P$.
- (ii) A operação **•** é associativa e distributiva sobre **V**.

$(x \bullet y) \bullet z = x \bullet (y \bullet z)$	para todo $x, y, z \in P$,
$x \bullet (y \vee z) = (x \bullet y) \vee (x \bullet z)$	para todo $x, y, z \in P$,
$(y \vee z) \bullet x = (y \bullet x) \vee (z \bullet x)$	para todo $x, y, z \in P$.
- (iii) O conjunto **P** contém um elemento $\emptyset$, denominado elemento zero, que é identidade ou neutro para **V** e nulo para a operação **•**.

$\emptyset \vee x = x$	para todo $x \in P$,
$\emptyset \bullet x = \emptyset = x \bullet \emptyset$	para todo $x \in P$.
- (iv) O conjunto **P** possui um elemento unitário ou unidade, denominado **e**, que é neutro ou identidade para o operador produto.

$e \bullet x = x = x \bullet e$	para todo $x \in P$.
---------------------------------	-----------------------

Como a operação **V** é idempotente, comutativa e associativa o conjunto **P** é um conjunto totalmente ordenado e quaisquer dois elementos desse conjunto são comparáveis e a regra da absorção de Herman é observada [14]:

$$x \leq y \Leftrightarrow x \vee y = y \quad \text{ou} \quad y \geq x \Leftrightarrow y \vee x = y$$

3.3.3 Novo *Framework* da Álgebra de Caminhos

A álgebra de caminhos de Herman [13,14] é definida pela estrutura matemática

$\langle \mathbf{M}, \mathbf{F}, \bar{\mathbf{S}}, \preceq_{ML} \rangle$, em que:

- $\mathbf{M}$ é um conjunto de métricas de roteamento;
- $\mathbf{F}$ é um conjunto de funções de combinação de métricas;
- $\bar{\mathbf{S}}$ é um conjunto de operações binárias denominadas sínteses;
- $\preceq_{ML}$ é uma relação de ordenação léxica multidimensional.

3.3.3.1 Métricas

Na álgebra de caminhos, $\mathbf{M}$ é um conjunto de m métricas de roteamento, ou seja:

$$\mathbf{M} = \{M_1, M_2, M_3, \dots, M_m\}$$

Uma métrica de roteamento é qualquer característica ou atributo que segundo uma estratégia de otimização permite que um roteador faça uma escolha e tome uma decisão do melhor caminho em uma rede. Uma métrica de roteamento $\mathbf{M}_j$ é definida como uma 2-upla (Q_j, w^{Q_j}) , em que:

- Q_j é o conjunto de valores possíveis que a métrica $\mathbf{M}_j$ pode assumir;
- w^{Q_j} são os valores ou pesos que a métrica $\mathbf{M}_j$ apresenta em cada arco⁵ de um dígrafo que representa uma dada rede.

⁵ Na maior parte da literatura de redes e neste trabalho os termos arcos, arestas e enlaces são utilizados como sinônimos, porém, segundo Chartrand [35], deve-se utilizar em um grafo o termo aresta e em um grafo orientado, ou dígrafo, o termo arco.

3.3.3.2 Função de Combinação de Métricas

A álgebra de caminhos permite que as m métricas atribuídas aos enlaces de uma rede sejam combinadas por meio de k funções de combinação de métricas, e o conjunto dessas funções é denominado $\mathbf{F}$, ou seja:

$$F = \{ f_1, f_2, \dots, f_k \}$$

Cada função de combinação de métricas retornará um único valor, que será considerado o novo valor ou peso mapeado para uma dada aresta e que será utilizado pelos algoritmos de roteamento para a escolha do melhor caminho em uma rede.

Os algoritmos de roteamento podem ser *monorrestritivos* ou *multirrestritivos*. Nos algoritmos de roteamento monorrestritivos uma única função de combinação de métricas é utilizada e os algoritmos trabalham apenas com um único valor de retorno dessa função, que é chamado de custo ou peso atribuído a cada enlace da rede. Nos algoritmos de roteamento multirrestritivos k funções de combinação de métricas são utilizadas e representam o número de restrições que serão impostas.

A aplicação das funções de combinação de métricas sobre os valores das métricas de uma aresta j definem um vetor de métricas combinadas $\bar{C}_j$, que será utilizado pelos algoritmos de roteamento multirrestritivos:

$$\bar{C}_j = \{ f_1(w_j^{Q_1}, w_j^{Q_2}, \dots, w_j^{Q_m}), f_2(w_j^{Q_1}, w_j^{Q_2}, \dots, w_j^{Q_m}), \dots, f_k(w_j^{Q_1}, w_j^{Q_2}, \dots, w_j^{Q_m}) \}$$

3.3.3.3 Sínteses

Na definição da estrutura matemática da álgebra de caminhos tem-se que $\bar{\mathbf{S}}$ é um conjunto de operações binárias, denominadas sínteses, que devem ser aplicadas sobre os valores resultantes das funções de combinação de métricas aplicadas aos enlaces ao longo de um caminho entre um nó origem e um nó destino.

Se o algoritmo de roteamento utilizado for monorrestritivo, a síntese para cada caminho entre um nó de origem e um de destino fornece um único valor, denominado palavra-peso. Quando o algoritmo de roteamento é multirrestritivo com k restrições impostas, onde: $k \in \mathbb{N} : k > 1$, obtém-se k valores resultantes de k sínteses aplicadas para um dado caminho. A palavra-peso resultante das k sínteses é denominada palavra-peso multidimensional.

A álgebra de caminhos permite r operações binárias diferentes de síntese, porém a proposta de Herman [13] utiliza um conjunto de quatro operações binárias de síntese, que são dependentes das métricas e das funções de combinação de métricas. O conjunto das sínteses denomina-se **S** e as operações binárias são:

$$S = \{ \text{minimizativa}, \text{maximizativa}, \text{aditiva}, \text{multiplicativa} \}$$

ou de forma abreviada:

$$S = \{ \min(), \max(), \text{add}(), \text{mul}() \}$$

As operações binárias, ou sínteses, determinam ou calculam o valor da palavra-peso da seguinte maneira:

Na síntese **minimizativa** a palavra-peso resultante para um caminho é determinada encontrando o *menor valor* ou peso atribuído a uma aresta por uma métrica ou função de combinação de métricas nesse caminho.

Na síntese **maximizativa** a palavra-peso resultante para um caminho é determinada encontrando o *maior valor* ou peso atribuído a uma aresta por uma métrica ou função de combinação de métricas nesse caminho.

Na síntese **aditiva** a palavra-peso resultante para um caminho é determinada por meio da *adição de todos* os valores ou pesos atribuídos às arestas por uma métrica ou função de combinação de métricas encontradas nesse caminho.

Na síntese ***multiplicativa*** a palavra-peso resultante para um caminho é determinada por meio da *multiplicação de todos* os valores ou pesos atribuídos às arestas por uma métrica ou função de combinação de métricas encontradas nesse caminho.

Todas as sínteses utilizadas são léxicas e um caminho pode fornecer vários valores de palavras-peso para a mesma restrição e síntese se for utilizado o conceito de abreviação.

3.3.3.4 Abreviação

Define-se abreviação de um caminho, chamada de $\bar{b}_j$, como sendo um conjunto de arestas do caminho que deverá ser utilizado pela síntese para o cálculo do valor da palavra-peso. Todo caminho possui um comprimento ou ordem e cada aresta desse caminho é chamada de letra. Quando se deseja utilizar um caminho com j letras de abreviação para cálculo da síntese, utiliza-se a nomenclatura: $\bar{b}_j(\bar{S}[\cdot])$ e quando não se quer utilizar nenhuma abreviação utiliza-se a nomenclatura: $\bar{b}_\infty(\bar{S}[\cdot])$.

3.3.3.5 Ordenação Léxica Multidimensional

A ordenação léxica multidimensional $\leq_{ML}$ tem o objetivo de ordenar todos os caminhos que possam existir entre um nó origem e um nó destino. Nessa ordenação é estabelecida uma priorização entre as k restrições impostas na escolha do melhor caminho.

Os caminhos são ordenados iniciando com a síntese de maior prioridade. Quando os caminhos são considerados equivalentes, pode-se utilizar como critério de desempate uma abreviação desse caminho com essa mesma síntese, ou realizar a ordenação dos caminhos com a próxima síntese menos prioritária.

A próxima seção apresenta um exemplo didático dessa álgebra.

3.3.4 Exemplo Didático da Álgebra de Caminhos

Uma rede é representada por um grafo orientado ou dígrafo $D=(V,A)$, onde V é um conjunto de vértices ou nós e A é um conjunto de arestas.

Um nó de origem e um nó de destino são descritos como um par ordenado (o,d) , e um caminho entre esses nós pode ser representado como uma sucessão de vértices ou como uma sucessão de arestas. Na Figura 7, o caminho entre os nós $(1,5)$ pode ser representado como uma sucessão de vértices $p_{1,5}$ ou como uma sucessão de arestas $p_{a,d}$.

Figura 7 – Exemplo de uma rede, seus vértices e arestas.

A Figura 8 mostra uma rede com os vértices $V = \{ 1,2,3,4,5,6,7,8 \}$ e as arestas $A = \{ a, b, c, d, e, f, g, h, i \}$. Nessa rede estão sendo utilizadas três métricas: m_1 , m_2 e m_3 , atribuídas a cada enlace, e três funções de combinação de métricas: f_1 , f_2 e f_3 , que seguem as seguintes regras:

$$M = \{ m_1, m_2, m_3 \}$$

$$f_1 = w^{Q_1}$$

$$f_2 = w^{Q_2}$$

$$f_3 = w^{Q_1} + w^{Q_2} + w^{Q_3}$$

$$F = \{ f_1, f_2, f_3 \} \text{ - funções de combinação de métricas}$$

Ou seja, as funções de combinação de métricas f_1 e f_2 são funções identidade que fornecem como valor de retorno da função o próprio peso ou valor da respectiva métrica

aplicada à aresta. A função de combinação de métricas f_3 realiza a soma das métricas aplicadas em cada aresta.

Figura 8 – Rede do Exemplo Didático.

Existem três caminhos entre o nó de origem 1 e o nó de destino 5:

- Caminho α : {1,2,3,4,5} ou {a,b,c,d}
- Caminho β : {1,6,7,5} ou {e,f,g}
- Caminho λ : {1,8,5} ou {h,i}

Os valores das métricas m_1 , m_2 e m_3 , atribuídos para cada enlace da rede e as funções de combinação de métricas f_1 , f_2 e f_3 , calculadas a partir dos valores dessas métricas, podem ser visualizados na Tabela 4.

Tabela 4 – Métricas e funções de combinação de métricas.

Arestas	m_1	m_2	m_3	f_1	f_2	f_3
a	5	10	5	5	10	20
b	5	5	10	5	5	20
c	5	20	15	5	20	40
d	5	5	20	5	5	30
e	5	10	15	5	10	30
f	5	10	10	5	10	25
g	5	5	5	5	5	15
h	10	15	5	10	15	30
i	10	10	10	10	10	30

A rede da Figura 8 apresenta três sínteses: $\bar{S}_1$, $\bar{S}_2$ e $\bar{S}_3$. Nas sínteses léxicas não serão utilizadas abreviações nos caminhos, ou seja, será utilizada apenas a palavra correspondente ao caminho completo e designada por: $\bar{b}_\infty(\bar{S}[\cdot])$.

Para cada uma das sínteses foi definida uma operação binária dada por:

$$\bar{S} = \{\bar{S}_1, \bar{S}_2, \bar{S}_3\}$$

$$\bar{S} = \{add(), \min(), \max()\}$$

A síntese $\bar{S}_1$ realizará uma operação binária aditiva, a síntese $\bar{S}_2$ realizará uma operação binária minimizativa e a síntese $\bar{S}_3$ uma operação binária maximizativa.

A Tabela 5 apresenta as sínteses obtidas para os três caminhos.

Tabela 5 – Sínteses para os três caminhos.

Caminhos	Síntese $\bar{S}_1$	Síntese $\bar{S}_2$	Síntese $\bar{S}_3$
$\alpha \{a,b,c,d\}$	20	5	40
$\beta \{e,f,g\}$	15	5	30
$\lambda \{h,i\}$	20	10	30

A ordenação léxica multidimensional $\leq_{ML}$ define como um caminho será pior ou menos otimizado do que outro caminho. Neste exemplo didático $\leq_{ML} = \{ \geq, \geq, \leq \}$, ou seja, um caminho $\mathbf{x}$ é pior do que um caminho $\mathbf{y}$ quando:

$$\bar{S}(x) \leq_{ML} \bar{S}(y)$$

$$\bar{S}_1(x) \leq_L \bar{S}_1(y) \Rightarrow \bar{S}_1(x) \geq \bar{S}_1(y)$$

ou

$$\bar{S}_1(x) = \bar{S}_1(y) \text{ e } \bar{S}_2(x) \leq_L \bar{S}_2(y) \Rightarrow \bar{S}_2(x) \geq \bar{S}_2(y)$$

ou

$$\bar{S}_1(x) = \bar{S}_1(y) \text{ e } \bar{S}_2(x) = \bar{S}_2(y) \text{ e } \bar{S}_3(x) \leq_L \bar{S}_3(y) \Rightarrow \bar{S}_3(x) \leq \bar{S}_3(y)$$

Ao analisar as sínteses da Tabela 5 percebe-se que o melhor caminho entre o nó de origem **1** e o nó de destino **5** é o caminho β porque:

$$\bar{S}_1(\beta) < \bar{S}_1(\alpha) \text{ e } \bar{S}_1(\beta) < \bar{S}_1(\lambda)$$

Os caminhos α e λ são equivalentes para a primeira síntese. Como houve um empate nos valores da primeira síntese, deve ser avaliada a segunda síntese, onde se percebe que o caminho α é melhor ou mais otimizado que o caminho λ pois:

$$\bar{S}_2(\alpha) < \bar{S}_2(\lambda)$$

A ordenação final dos caminhos é dada por: $\beta > \alpha > \lambda$, ou seja, o melhor caminho é o β , seguido pelo α e o pior caminho é o λ .

Se as sínteses ou a ordenação léxica multidimensional forem alteradas poderá ocorrer uma alteração dos resultados de qual será o melhor ou pior caminho.

Por exemplo, se a ordenação léxica multidimensional for alterada para $\leq_{ML} = \{ \leq, \geq, \leq \}$, a ordenação final dos caminhos ficará $\alpha > \lambda > \beta$, ou seja, o melhor caminho determinado pela segunda síntese é o α , seguido pelo λ e o pior caminho, já determinado pela primeira síntese, é o caminho β .

O exemplo didático não trabalhou com abreviações nas sínteses. A Tabela 6 ilustra os valores das sínteses $\bar{S}_1$, $\bar{S}_2$ e $\bar{S}_3$ para o caminho α quando se utilizam abreviações de três, duas e uma letra.

Tabela 6 – Sínteses para os três caminhos com abreviações.

Abreviação	Síntese $\bar{S}_1$	Síntese $\bar{S}_2$	Síntese $\bar{S}_3$	Explicação
$\bar{b}_\infty(\bar{S}[\alpha])$	20	5	40	sem abreviação – todas as letras do caminho são utilizadas.
$\bar{b}_3(\bar{S}[\alpha])$	15	5	40	três letras correspondentes às arestas {a,b,c}.
$\bar{b}_2(\bar{S}[\alpha])$	10	5	20	duas letras correspondentes às arestas: {a,b}
$\bar{b}_1(\bar{S}[\alpha])$	5	10	20	uma letra correspondente à aresta: {a}

A ordenação léxica multidimensional pode propor como critério de ordenação a utilização de abreviações, ao invés de uma síntese com o caminho completo, ou sugerir como critério de desempate o uso dessas abreviações.

A próxima seção apresenta o estágio de mapeamento dos enlaces virtuais do problema VNE modelado e resolvido por meio da utilização da álgebra de caminhos.

3.4 Álgebra de Caminhos e o Problema VNE

A álgebra de caminhos pode ser utilizada no estágio de mapeamento de enlaces virtuais para caminhos na rede de substrato do problema VNE, pois esse problema é em essência multirrestritivo (*multi-constraint*). Após a escolha de um conjunto de métricas e funções de combinação dessas métricas, mecanismos de priorização e ordenação das funções de métricas são utilizados segundo as necessidades do negócio “Virtualização de Redes” [27].

As métricas empregadas pela álgebra de caminhos são métricas lineares, como largura de banda (*bandwidth*), número de saltos (*hops*) e atraso (*delay*), ou métricas não-lineares, como disponibilidade (*availability*) e taxa de perda de pacotes (*packet loss rate*). Novas métricas podem ser criadas por meio da utilização de diversas funções de combinação de métricas, como, por exemplo, a função chamada qualidade de serviços ou **QoS** (*quality of service*), utilizada para quantificar a percepção subjetiva de um usuário em relação à qualidade do serviço oferecido em função de parâmetros físicos indicadores do desempenho da rede [55].

$$QoS = f(THRU, PDT, PDV, PLR)$$

onde:

- THRU – representa a vazão ou “*throughput*” de uma rede;
- PDT – atraso na transferência de pacotes (*packet delay transfer*);
- PDV – variação do atraso (*packet delay variation*);
- PLR – taxa de perda de pacotes (*packet loss rate*).

Como a virtualização de redes é uma estratégia de negócios, métricas importantes, como **custo** e **receita**, também devem ser consideradas e utilizadas [19].

Um problema importante nas redes virtuais é a determinação de caminhos alternativos na rede de substrato, os quais podem ser utilizados no evento de uma falha do caminho escolhido e mapeado. A álgebra de caminhos resolve facilmente esse problema, pois ela consegue encontrar todos os caminhos possíveis entre quaisquer pares de nós de origem e de destino, tornando a recuperação de uma VN rápida e de fácil implementação [27].

3.4.1 Métricas e Critérios de Otimização

A álgebra de caminhos fornece uma grande flexibilidade por permitir qualquer tipo de métrica e combinação de métricas. Múltiplos critérios de otimização podem ser simultaneamente empregados, com a qualidade de cada uma dessas soluções sendo avaliada.

O mapeamento dos enlaces virtuais para caminhos na rede de substrato pode utilizar as seguintes métricas:

- *Capacidade de CPU disponível*: é a capacidade de CPU disponível em um nó da rede de substrato depois do mapeamento do enlace virtual;
- *Capacidade de Memória disponível*: é a capacidade de memória disponível em um nó da rede de substrato depois do mapeamento do enlace virtual;
- *Capacidade de Largura de Banda (BW) disponível*: é a capacidade de largura de banda disponível em um enlace da SN depois do mapeamento do enlace virtual;
- *Atraso (Delay)*: o atraso individual de um enlace pertencente à rede de substrato;
- *Variação do Atraso (PDV ou Jitter)*: a variação do atraso de um enlace da rede de substrato utilizado para mapeamento de um enlace virtual;
- *Perdas*: é a porcentagem de perda de pacotes em um enlace pertencente à rede de substrato;

- *Confiabilidade*: é a probabilidade do mapeamento de um enlace virtual para um caminho na SN manter seu funcionamento em circunstâncias de rotina, bem como em situações hostis e inesperadas;
- *Energia*: a energia consumida por cada nó da SN para processar os pacotes;
- *Número de Saltos (hops)*: é a quantidade de nós que uma mensagem deve atravessar ao longo de um caminho na SN.
- *Custo*: é o custo da utilização de nós e caminhos na SN para cada VN atendida;
- *Receita*: é a receita solicitada de recursos por nós e enlaces virtuais presentes em uma VN.

Na álgebra de caminhos todas as métricas são associadas como pesos nas arestas da rede de substrato, até mesmo quando a métrica representa uma característica do nó, como capacidade de CPU ou memória disponível. Essa transformação de métricas de nós para arestas será discutida na Seção 3.4.2.1 deste capítulo.

3.4.2 Solução do Problema VNE

A metodologia⁶ proposta para resolver o problema VNE referente ao mapeamento de enlaces virtuais para caminhos na rede de substrato compreende quatro estágios:

- Transformação de métricas de nós em métricas de arestas;
- Enumeração de todos os caminhos na SN;
- Ordenação das requisições de rede virtual (VNRs);
- Atribuição dos enlaces virtuais a caminhos na SN.

3.4.2.1 Transformação de métricas de nós em métricas de arestas

A álgebra de caminhos foi desenvolvida para associar pesos às arestas de um dígrafo que representa uma rede. Para que seja possível a utilização de métricas atribuídas a características de nós, como CPU, memória ou consumo de energia, entre outras, deve-se fazer a transformação exemplificada a seguir.

⁶ Segundo o dicionário Houaiss, metodologia também é definida como corpo de regras e diligências estabelecidas para realizar uma pesquisa; método [57].

Considere os dois nós da Figura 9, que é uma parte de um dígrafo que possui métricas atribuídas a nós e arestas. No exemplo, deseja-se trabalhar com métricas de largura de banda disponível em uma aresta (BW) e capacidade disponível de CPU em um nó. Na Figura 9.a são mostrados os pesos associados à largura de banda entre os nós $BW(A-B)$ e aos recursos de CPU. Na Figura 9.b, os pesos dos nós são artificialmente colocados na aresta. A transformação acontece na Figura 9.c, onde, dependendo da métrica utilizada no nó, a álgebra de caminhos irá buscar uma solução com o objetivo de maximizar a capacidade disponível daquele recurso [19].

Figura 9 – (a) pesos associados com os nós e arestas. (b) pesos dos nós artificialmente inseridos na aresta. (c) métricas finais.

No exemplo, para maximizar os recursos de CPU disponíveis nos nós pode-se utilizar a condição mais restritiva, que é dada por $\min[CPU(A), CPU(B)]$, ou seja, é atribuído à aresta o menor valor entre os valores da CPU. Dessa maneira, ao comparar dois caminhos diferentes para o mesmo par origem/destino (o,d), o melhor caminho será aquele que resultar na maior soma dos valores de recursos de CPU disponíveis atribuídos às arestas de um caminho (síntese aditiva), ou aquele que resultar no maior valor entre os “menores valores” de recursos de CPU disponíveis obtidos em caminhos diferentes (síntese minimizativa) [19].

3.4.2.2 Enumeração de todos os caminhos na SN

O primeiro passo realizado pelo algoritmo que implementa a álgebra de caminhos é encontrar todos os caminhos que conectam todos os pares de nós em uma rede de substrato. É um procedimento explosivo em termos de complexidade computacional para redes de substrato completamente conectadas, mas para redes pequenas ou para grandes redes esparsas isso não é verdade [27].

Essa enumeração de todos os caminhos é realizada pelo algoritmo apenas uma vez e o seu tempo de execução pode ser negligenciado se o procedimento for executado “*offline*”, ou seja, antes do processo de atendimento e mapeamento de requisições de rede virtual (VNRs).

O processo de enumeração dos caminhos se “paga” pela qualidade da solução fornecida e seu tempo de execução pode ser reduzido por meio da imposição de um limite do número de caminhos que serão descobertos. Este assunto será abordado na Seção 4.3 (Estudo da filtragem por profundidade).

3.4.2.3 Ordenação das requisições de rede virtual (VNRs)

O mapeamento de VNs deve ser otimizado para realizar a maior quantidade de atendimentos de VNRs [2]. Cada requisição de rede virtual (VNR) é considerada como uma unidade, independentemente da quantidade de enlaces virtuais e de recursos solicitados.

Uma VNR deve ser sempre completamente atendida para que o problema de virtualização de redes seja considerado resolvido. Até o momento, não existe suporte para atendimento parcial de VNRs, ou seja, todas as requisições de recursos de nós e de enlaces virtuais devem ser atendidas e mapeadas para que a VNR seja considerada atendida.

Cada VNR atendida consome recursos físicos da rede de substrato, como, por exemplo, o poder de processamento (CPU) dos nós e a largura de banda (BW) dos

enlaces. Sendo assim, o atendimento de uma VNR nem sempre é possível, porque os recursos físicos restantes na SN podem não ser suficientes.

A maneira e a ordem como as VNRs são alocadas também afetam o número de VNRs atendidas. Existem diferentes maneiras de alocar requisições de rede virtual:

- Alocar VNRs na ordem em que elas chegam, como se fosse um mecanismo de enfileiramento FIFO (*First In First Out*);
- Alocar VNRs ordenadas, de forma crescente ou decrescente, pelo total de BW solicitada;
- Alocar VNRs ordenadas, de forma crescente ou decrescente, pelo total de CPU solicitada;
- Alocar VNRs ordenadas, de forma crescente ou decrescente, pela soma total de recursos de CPU e BW.

A política de ordenação afeta a quantidade de VNRs atendidas. Por exemplo, se existir uma rede de substrato com um total de BW de 20 e houver cinco VNRs para serem atendidas com solicitações de largura de banda iguais a $BW = \{2,4,6,8,10\}$, se for utilizada a política de ordenação e atendimento de VNRs em ordem crescente de solicitação de BW, as VNRs que solicitaram a banda de $\{2,4,6,8\}$ serão atendidas; caso a política de ordenação e atendimento de VNRs for em ordem decrescente de solicitação de BW, as VNRs que solicitaram a banda de $\{10,8,2\}$ serão atendidas.

Nesse exemplo, sendo o critério de otimização orientado pela quantidade de VNRs atendidas, a política de ordenação crescente mostrou-se melhor que a política de ordenação decrescente, uma vez que a primeira fez quatro atendimentos de VNRs e a segunda apenas três. Se o critério de otimização for a receita obtida com a virtualização e considerando a receita uma função da BW total atendida, percebe-se que os dois métodos de ordenação e atendimento são equivalentes.

O exemplo serviu para mostrar que a política de ordenação e atendimento afeta o número de VNRs atendidas e a receita obtida. Não é possível dizer qual é a melhor

ordenação para o atendimento de VNRs, uma vez que a melhor política depende do critério de otimização e de políticas comerciais utilizadas pelos provedores de VNs.

Dado que uma VNR deve ser atendida, os vários enlaces virtuais presentes nessa requisição de rede virtual podem ser ordenados e atendidos de diferentes maneiras, fazendo com que a qualidade dos resultados seja afetada.

As políticas de ordenação e atendimento dos enlaces virtuais presentes em uma VNR não são tratadas neste trabalho. Esse assunto é indicado para trabalhos futuros.

3.4.2.4 Atendimento de VNRs

Neste trabalho, as políticas de ordenação e atendimento de VNRs utilizadas são: FIFO para simular um atendimento *online* de chegadas de requisições e soma total de recursos de CPU e BW, em ordem crescente ou decrescente, para os casos de atendimento *offline* de requisições de rede virtual. Os enlaces virtuais de uma VNR são atendidos linha a linha considerando a matriz de transição ou adjacência representativa da VNR.

3.4.2.5 Atribuição dos enlaces virtuais para caminhos na SN

Um enlace virtual, ao ser mapeado para um caminho da rede de substrato, consumirá recursos da SN e, a cada atribuição realizada, os pesos das arestas no caminho utilizado devem ser atualizados em função dos recursos da SN que foram efetivamente utilizados no atendimento da VNR.

Os caminhos entre um nó origem e um nó destino na SN são ordenados segundo a álgebra de caminhos, utilizando um dado critério de ordenação. Os seguintes critérios multirrestritivos podem ser utilizados para a ordenação dos caminhos:

$$a) BW, CPU \Rightarrow \bar{S}_1 = \{min\} e \bar{S}_2 = \{min\} \Rightarrow \bar{S} = \{min, min\}$$

$$b) CPU, BW \Rightarrow \bar{S}_1 = \{min\} e \bar{S}_2 = \{min\} \Rightarrow \bar{S} = \{min, min\}$$

$$c) HOP, BW, CPU \Rightarrow \bar{S}_1 = \{add\} e \bar{S}_2 = \{min\} e \bar{S}_3 = \{min\} \Rightarrow \bar{S} = \{add, min, min\}$$

- d) *HOP, CPU, BW* $\Rightarrow \bar{S}_1 = \{add\} e \bar{S}_2 = \{min\} e \bar{S}_3 = \{min\} \Rightarrow \bar{S} = \{add, min, min\}$
e) *Delay, BW, CPU* $\Rightarrow \bar{S}_1 = \{add\} e \bar{S}_2 = \{min\} e \bar{S}_3 = \{min\} \Rightarrow \bar{S} = \{add, min, min\}$
f) *Delay, CPU, BW* $\Rightarrow \bar{S}_1 = \{add\} e \bar{S}_2 = \{min\} e \bar{S}_3 = \{min\} \Rightarrow \bar{S} = \{add, min, min\}$
g) *Avail, BW, CPU* $\Rightarrow \bar{S}_1 = \{mult\} e \bar{S}_2 = \{min\} e \bar{S}_3 = \{min\} \Rightarrow \bar{S} = \{mult, min, min\}$
h) *Avail, CPU, BW* $\Rightarrow \bar{S}_1 = \{mult\} e \bar{S}_2 = \{min\} e \bar{S}_3 = \{min\} \Rightarrow \bar{S} = \{mult, min, min\}$
i) *BW+CPU, HOP* $\Rightarrow \bar{S}_1 = \{min\} e \bar{S}_2 = \{add\} \Rightarrow \bar{S} = \{min, add\}$
j) *HOP, BW+CPU* $\Rightarrow \bar{S}_1 = \{add\} e \bar{S}_2 = \{min\} \Rightarrow \bar{S} = \{add, min\}$
k) *HOP, BW Total* $\Rightarrow \bar{S}_1 = \{add\} e \bar{S}_2 = \{min\} \Rightarrow \bar{S} = \{add, min\}$

onde:

BW refere-se a largura de banda disponível em um caminho;
BW Total é a soma da largura de banda disponível em um caminho;
CPU refere-se ao valor de CPU disponível em um caminho;
HOP é o número de saltos no caminho;
Delay é o atraso total em um caminho;
Avail ou **Availability** é a disponibilidade de um caminho.

A lista de critérios de ordenação multirrestritivos apresentados dos itens de a) até k) é um exemplo das possibilidades de ordenação de caminhos na SN por meio da álgebra de caminhos, sendo que outros critérios também podem ser criados e estudados para a resolução do problema VNE.

O critério de ordenação escolhido afeta a qualidade do atendimento da VN, o número de VNRs atendidas e os recursos na rede de substrato que são utilizados.

Um enlace virtual é mapeado para o melhor caminho na SN, dado um determinado critério de ordenação e síntese. Os recursos naquele caminho são atualizados pelos valores reais que foram consumidos pela VN. Esse procedimento é repetido para cada um dos enlaces virtuais encontrados na VNR.

A VNR é atendida com sucesso no caso de que todos os seus enlaces virtuais consigam ser mapeados para caminhos na SN. No caso de uma atribuição de enlace

virtual não conseguir obter recursos disponíveis suficientes na SN, a VNR não será atendida e todos os mapeamentos dos enlaces virtuais dessa mesma VNR que foram realizados serão desfeitos e os recursos solicitados devolvidos para a rede de substrato.

Quando um enlace virtual não consegue ser atendido, antes de se afirmar que a VNR não será atendida se pode fazer uma análise do último enlace virtual mapeado e tentar remapear esse último enlace virtual em outro caminho na SN, na tentativa de que recursos na rede de substrato sejam liberados para o próximo enlace virtual. Esse processo é chamado de *backtrack* e o mesmo pode desfazer o último enlace virtual mapeado, um conjunto de enlaces mapeados da VNR atual ou desfazer a última VNR atendida, com o objetivo de conseguir realizar o atendimento ao enlace virtual e à VNR a que o enlace virtual pertence. Essa análise e o processo de *backtrack* são indicados para trabalhos futuros.

Um último ponto relacionado à atribuição dos enlaces virtuais para caminhos na SN é que o modelo de negócios atual para virtualização de redes supõe que todos os enlaces virtuais encontrados em uma VNR devem ser atendidos para que a VNR seja atendida. É possível o atendimento parcial de VNRs, com uma quantidade menor de enlaces virtuais atendidos ou com todos os enlaces virtuais atendidos com uma quantidade de recursos alocada para cada um deles inferior ao que foi solicitado. A álgebra de caminhos pode auxiliar no atendimento de VNRs parciais desde que o modelo de negócios utilizado suporte esse novo tipo de serviços.

3.5 Prova de Conceito: Álgebra de Caminhos e o Problema VNE

Considere a rede de substrato representada pelos dígrafos nas Figuras 10 e 11, e, as requisições de rede virtual representadas na Figura 12.

Figura 10 – SN e peso de nós: métrica *CPU* e peso de arestas: métrica *bandwidth*.

Figura 11 – SN e pesos de arestas: métrica *delay*.

Figura 12 – VNRs e peso de nós: métrica *CPU* e pesos de arestas: métrica *bandwidth* e *delay*.

A SN possui peso nos nós que indica a capacidade de recursos de processamento disponível (CPU) naquele nó, e pesos nas arestas, que indicam a largura de banda disponível e o *delay* máximo que uma mensagem terá ao passar por aquela aresta.

Cada VNR indica nos nós virtuais e nos enlaces virtuais os recursos de processamento (CPU) e largura de banda (*bandwidth*) desejados. Existe também uma informação na VNR do *delay* máximo total tolerável em um enlace virtual mapeado para um caminho na rede de substrato.

A solução do problema VNE é iniciada com o mapeamento dos nós virtuais para nós na rede de substrato, que pode ser executado por algoritmos gulosos (*greedy*), conforme indicado em [2] e [6], ou por algoritmos de mapeamento coordenado, indicado em [4].

A análise, modelagem e o estudo da viabilidade técnica do mapeamento de nós virtuais para nós na rede de substrato utilizando a álgebra de caminhos são indicados para trabalhos futuros.

A Tabela 7 fornece o mapeamento de nós virtuais para nós reais na SN após a execução de algum dos algoritmos de mapeamento de nós descritos nas referências [2], [3], [4] ou [6]. Esse mapeamento será utilizado para a alocação dos enlaces virtuais para caminhos na rede de substrato por meio da álgebra de caminhos.

Tabela 7 – Mapeamento dos Nós Virtuais para Nós Reais.

Número da VNR	Nó Virtual	Nó Real
1	1	B
1	2	F
1	3	D
2	1	A
2	2	C
2	3	E

Para solucionar o problema VNE os seguintes objetivos devem ser cumpridos:

- Alocação do maior número de VNs à SN;
- Maximizar a quantidade de recursos disponíveis globais na SN após cada VN atendida, isto é, deve-se economizar nos recursos de largura de banda e na capacidade de CPU da SN. Este objetivo é atendido através do mapeamento dos enlaces virtuais para caminhos mais curtos disponíveis na SN e que atendam as restrições da VN;
- Minimizar o atraso (*delay*).

As métricas, funções de combinação de métricas, operações de síntese e relações de ordenação utilizadas para a solução deste problema são:

$M = \{hops, bw, cpu, delay\}$ - Conjunto de Métricas

$F = M$ - Funções de Combinação de Métricas

$\bar{S} = \{add, min, min, add\}$ - Operações de Síntese

$\preceq_{ML} = \{\geq, \leq, \leq, \geq\}$ - Relações de Ordenação

Nas sínteses léxicas não serão utilizadas abreviações nos caminhos, ou seja, será utilizada apenas a palavra correspondente ao caminho completo e designada por: $\bar{b}_{\infty}(\bar{S}[\cdot])$.

Uma vez modelado o problema pela álgebra de caminhos, deve-se seguir a metodologia de quatro estágios proposta na seção 3.4.2., que se inicia com a transformação das métricas dos nós para métricas de arestas e, em seguida, é feita a enumeração de todos os caminhos na SN. Esses dois estágios são independentes das VNRs que devem ser atendidas e podem, inclusive, ser feitos de forma *offline* para reduzir o tempo de atendimento de uma requisição de rede virtual.

O próximo estágio é a ordenação das VNRs necessária para identificar a ordem de atendimento. Nessa prova de conceito, as VNRs são ordenadas de forma decrescente pela soma total de recursos solicitados de CPU e BW. A fórmula utilizada para o cálculo da soma de recursos totais é dada por:

$$Total\ de\ Recursos\ da\ VN(i) = \alpha * (\sum BW)_{VN(i)} + \beta * (\sum CPU)_{VN(i)}$$

onde:

os valores de α e β obedecem a: $0 \leq \alpha \leq 1, 0 \leq \beta \leq 1$

Atribuindo às constantes α e β o valor igual a 1 (um), é possível calcular o total de recursos requisitado por VNR nessa prova de conceito, o que permite sua ordenação. As duas VNRs da Figura 12 são atendidas na seguinte ordem:

1º - VNR #2 - total de recursos solicitados = 200

2º - VNR #1 - total de recursos solicitados = 140

Os enlaces virtuais em cada VNR são atendidos utilizando a ordem dada pela matriz de transição ou adjacência da VNR lida linha a linha. Neste exemplo, primeiro será atendido o $VL_{1,2}$ e, depois, o $VL_{2,3}$. A Tabela 8 apresenta os enlaces virtuais de cada VNR e quais são os caminhos na rede de substrato que devem ser encontrados.

Tabela 8 – Mapeamento dos Enlaces Virtuais para Caminhos na SN.

Número da VNR	Enlace Virtual	Caminho na SN
1	$VL_{1,2}$	$P_{B,F}$
1	$VL_{2,3}$	$P_{F,D}$
2	$VL_{1,2}$	$P_{A,C}$
2	$VL_{2,3}$	$P_{C,E}$

A requisição de rede virtual número dois (VNR #2) será a primeira VNR a ser atendida. Todos os caminhos na rede de substrato serão obtidos para os pares de nós origem e destino: (A,C) e, depois, (C,E). Em seguida, uma operação de síntese e ordenação léxica ordenará esses caminhos. A fase de atribuição de enlaces virtuais para caminhos na SN é feita por meio da determinação do melhor caminho que possa atender as restrições do enlace virtual que está sendo analisado. Lembrar que as restrições no exemplo são: a largura de banda, o processamento (CPU) e o *delay total máximo* no caminho. O primeiro caminho ordenado que consiga satisfazer essas restrições será a solução do mapeamento da VN para um caminho na SN.

Quando ocorre o atendimento de uma VN, os pesos na rede de substrato devem ser atualizados da seguinte maneira: o peso da largura de banda encontrado em todas as arestas no caminho da SN deve ser subtraído da largura de banda requisitada no enlace virtual e o processamento no par de nós origem e destino da SN deve, também, ser subtraído pelo processamento requisitado pelos nós virtuais na VN. Notar que a subtração do peso dos nós virtuais acontece apenas uma vez, não importa quantos enlaces virtuais utilizam o mesmo nó virtual.

Após a atualização dos pesos na rede de substrato em decorrência do atendimento do primeiro enlace virtual mapeado para o caminho (A,C), ocorre uma nova operação de síntese e ordenação léxica com a finalidade de ordenar novamente todos os caminhos para que o próximo enlace virtual mapeado para o caminho (C,E) na rede de substrato possa ser atendido.

Da Figura 10 obtém-se todos os caminhos para (A,C):

(A,C): (A,B,E,C) , (A,B,D,F,E,C)

Utilizando as operações de síntese e ordenações léxicas determina-se que o melhor caminho é o caminho (A,B,E,C) pela síntese $\bar{S}_1$ aditiva que utiliza métrica *hops* e cuja ordenação léxica diz que o menor valor da síntese *hops* é melhor ou o mais otimizado.

Os pesos na SN são atualizados após esse atendimento, como pode ser visto na Figura 13.

Figura 13 – Atualização dos pesos na SN após o atendimento do $VL_{1,2}$ da VNR # 2.

O mapeamento do segundo enlace virtual da VNR #2 deve agora ocorrer. O caminho (C,E) deve ser obtido na rede de substrato e escolhido por meio da álgebra de caminhos.

Da Figura 10 obtêm-se todos os caminhos para (C,E):

(C,E): (C,A,B,E), (C,A,B,D,F,E)

Utilizando as operações de síntese e ordenações léxicas determina-se que o melhor caminho é o caminho (C,A,B,E) pela síntese $\bar{S}_1$ aditiva que utiliza métrica *hops* e cuja ordenação léxica diz que o menor valor da síntese *hops* é melhor ou o mais otimizado. Notar que o caminho (C,A,B,D,F,E) não pode ser utilizado em nenhuma situação, pois a síntese $\bar{S}_4$ aditiva que utiliza métrica *delay* apresenta um valor de 2400, que é maior do que a restrição do enlace virtual de 2000 para o *delay total máximo* no caminho. A Figura 14 apresenta os pesos na SN após serem atualizados por esse atendimento.

Figura 14 – Atualização dos pesos na SN após o atendimento do $VL_{2,3}$ da VNR # 2.

Como todos os enlaces virtuais da VNR #2 conseguiram ser mapeados para caminhos na rede de substrato, a requisição de rede virtual é considerada atendida e as atualizações dos valores nos pesos das arestas da SN não necessitam ser desfeitas e são definitivas. Cabe agora o atendimento da próxima requisição de rede virtual, a VNR #1.

A requisição de rede virtual número um (VNR #1) possui dois enlaces virtuais que são mapeados para os caminhos (B,F) e (F,D) na SN. Todos os caminhos na rede de substrato são obtidos para esses pares de nós origem e destino.

Da Figura 10 obtêm-se todos os caminhos para (B,F):

(B,F): (B,D,F), (B-E-F)

Utilizando as operações de síntese e ordenações léxicas determina-se que o melhor caminho é o caminho (B,D,F), pois ele apresenta o maior valor para síntese $\bar{S}_2$ minimizativa que utiliza a métrica largura de banda disponível na aresta e cuja ordenação léxica diz que o maior valor dessa síntese é melhor ou o mais otimizado. Notar que a segunda síntese foi utilizada, pois houve uma igualdade de valores para a síntese $\bar{S}_1$ *aditiva* para os dois caminhos. A Figura 15 apresenta os pesos na SN após serem atualizados por esse atendimento.

Figura 15 – Atualização dos pesos na SN após o atendimento do $VL_{1,2}$ da VNR # 1.

Após a atualização dos pesos na rede de substrato devido ao mapeamento do primeiro enlace virtual para o caminho (B,F) na SN, o mapeamento do segundo enlace virtual da VNR #1 deve ocorrer, e o caminho (F,D) deve ser obtido na rede de substrato e escolhido por meio da álgebra de caminhos.

Da Figura 10 obtêm-se todos os caminhos para (F,D):

(F,D): (F,E,B,D), (F,E,C,A,B,D)

Utilizando as operações de síntese e ordenações léxicas determina-se que o melhor caminho é o caminho (F,E,B,D) definido pela síntese $\bar{S}_1$ aditiva que utiliza métrica *hops* e cuja ordenação léxica diz que o menor valor da síntese *hops* é melhor ou o mais otimizado. Notar que o caminho (F,E,C,A,B,D) não pode ser utilizado em nenhuma situação, pois a síntese $\bar{S}_4$ aditiva que utiliza métrica *delay* apresenta um valor de 2400, que é maior do que a restrição do enlace virtual de 2000 para o *delay total máximo* no caminho. A Figura 16 apresenta os pesos finais na SN após serem atualizados por esse atendimento.

Figura 16 – Atualização dos pesos na SN após o atendimento do $VL_{2,3}$ da VNR # 1.

Como todos os enlaces virtuais da VNR #1 conseguiram ser mapeados para caminhos na rede de substrato, a requisição de rede virtual é considerada atendida e as atualizações dos valores nos pesos das arestas da SN não necessitam ser desfeitas e são definitivas.

As duas VNRs foram atendidas e o problema VNE foi resolvido por meio da utilização da álgebra de caminhos.

A Tabela 9 apresenta os valores de todas as sínteses que foram utilizadas na prova de conceito.

Tabela 9 – Valores das sínteses para cada caminho analisado na SN.

VNR #	Enlace Virtual	Caminho na SN	Caminho Completo	$\bar{S}_1$	$\bar{S}_2$	$\bar{S}_3$	$\bar{S}_4$
2	(1,2)	(A,C)	A-B-E-C	3	100	80	1700
2	(1,2)	(A,C)	A-B-D-F-E-C	5	70	50	2500
2	(2,3)	(C,E)	C-A-B-E	3	50	50	1600
2	(2,3)	(C,E)	C-A-B-D-F-E	5	70	50	2400
1	(1,2)	(B,F)	B-D-F	2	70	50	900
1	(1,2)	(B,F)	B-E-F	2	10	60	1200
1	(2,3)	(F,D)	F-E-B-D	3	65	50	1700
1	(2,3)	(F,D)	F-E-C-A-B-D	5	60	50	2400

3.6 Considerações Finais do Capítulo

O capítulo apresentou a formulação do problema VNE utilizando a álgebra de caminhos, sua modelagem e uma metodologia de quatro estágios para solução do problema.

No próximo capítulo, será apresentado o algoritmo proposto para atendimento de requisições de rede virtual utilizando a álgebra de caminhos.

4. Algoritmo para mapeamento de enlaces de rede virtual

Este capítulo apresenta o algoritmo **PAViLiM** (*Paths Algebra for Virtual Link Mapping*) proposto para realizar o mapeamento de enlaces virtuais para caminhos na rede de substrato. Inicialmente é apresentado o diagrama de blocos e o nome dos principais programas do algoritmo de roteamento ELND utilizado como referência para a criação do algoritmo proposto. Em seguida, discute-se a complexidade desse algoritmo e a solução adotada para resolver o problema de virtualização de redes.

4.1 Algoritmo ELND

O algoritmo de roteamento ELND é um algoritmo de roteamento multirrestritivo “*hop-by-hop*” que utiliza a álgebra de caminhos e possui uma estratégia de otimização definida pelo usuário ou pelo provedor, adotada pelo nó de origem, sendo imposta, quando necessário, a propriedade da coerência [14].

O algoritmo ELND pode ser dividido em quatro grandes blocos [14]:

- ✓ bloco 1: realiza a enumeração de todos os caminhos simples e elementares⁷ entre todos os pares de nós origem-destino da rede;
- ✓ bloco 2: cálculo das palavras-peso multidimensionais correspondentes a todos os caminhos que foram encontrados;
- ✓ bloco 3: ordenação dos caminhos entre cada par de nós origem-destino;
- ✓ bloco 4: busca, identificação e remoção de *loops*.

⁷ Caminhos simples e elementares são aqueles que atravessam cada nó e aresta do grafo uma única vez.

A Tabela 10 apresenta para cada bloco do algoritmo os principais programas.

Tabela 10 – Programas do algoritmo ELND.

Número do Bloco	Programas	Descrição
1	buscaPath separaPath	Enumeração dos caminhos.
2	calculaPath	Cálculo das palavras pesos.
3	ordenaPaths	Ordenação dos caminhos.
4	buscaLoop	Remoção dos loops.

Os programas e procedimentos do algoritmo ELND foram criados utilizando uma convenção de nomes em português. O algoritmo PAViLiM utiliza alguns desses programas com uma nova convenção de nomes em Inglês. A Tabela 11 apresenta o mapeamento dos nomes de programas entre ELND e PAViLiM.

Tabela 11 – Nome de programas: ELND e PAViLiM.

Nomes de Programas no ELND	Nomes de Programas no PAViLiM
buscaPath	searchPath
separaPath	sortPath
calculaPath	evaluatePath
ordenaPaths	orderPaths
buscaLoop	searchLoop

4.2 Complexidade algorítmica

A fase de enumeração de todos os caminhos existentes entre quaisquer pares de nós origem-destino, realizada pelos algoritmos de roteamento multirrestritivos baseados na álgebra de caminhos em um dígrafo completamente conectado é *NP-Hard*⁸ e, supondo um dígrafo com n nós e $n \times (n - 1)$ arestas, pode-se determinar a sua complexidade algorítmica como sendo [27]:

⁸ *NP-Hard*: são problemas mais difíceis que os problemas mais difíceis em *NP*, alguns pesquisadores afirmam que problemas *NP-Hard* englobam os problemas *NP-Completo* (NPC) [57].
NP-Completo (NPC): são problemas classificados como intratáveis cuja solução exata é desconhecida [38].

$$f(n) = O(n^n)$$

onde:

$f(n)$ indica o tempo de execução da fase de enumeração dos algoritmos da álgebra de caminhos que são limitados de forma assintótica pela função n^n . A notação **O** (*big-O*) impõe um limite superior para a função $f(n)$ e representa a complexidade algorítmica. O valor de **n** corresponde ao “tamanho do problema” e descreve a quantidade de nós existentes no dígrafo representativo da rede de substrato estudada.

Na análise da complexidade algorítmica é importante concentrar-se na taxa de crescimento do tempo de execução do algoritmo como uma função do tamanho do problema, obtendo-se um quadro geral do comportamento do algoritmo com três critérios de análise: pior caso, melhor caso e caso médio [39]. Nem sempre um algoritmo pode ser analisado empregando os três critérios. Por exemplo, nos programas de enumeração de caminhos em um dígrafo completamente conectado os três critérios de análise são iguais e fornecem o mesmo resultado de complexidade.

O tempo real de execução de um algoritmo é obtido por meio da multiplicação de um fator constante (f_c), que depende do software e do hardware em que os programas serão executados, pela complexidade algorítmica [40], ou seja, o tempo real para a enumeração de caminhos realizada pelos programas da álgebra de caminhos em um dígrafo completamente conectado é calculado pela expressão [40]:

$$(Fator\ Constante) * (Complexidade\ Algorítmica)$$

$$f_c * n^n$$

O valor exponencial **nⁿ** da complexidade algorítmica indica que os algoritmos da álgebra de caminhos em redes completamente conectadas possuem tempo de execução explosivo quando se aumenta o valor de **n**.

A complexidade algorítmica explosiva foi percebida em simulações realizadas com o algoritmo ELND em dígrafos completamente conectados. A Tabela 12 apresenta essas simulações em dígrafos com um tamanho de 2 a 10 nós.

Nota-se que quando o tamanho do dígrafo aumenta a quantidade de caminhos enumerados e o tempo de execução aumentam em uma taxa não-polinomial (NP) quando relacionada ao aumento da quantidade de nós [13,14].

Tabela 12 – Tempo de execução e caminhos descobertos⁹ - Algoritmo ELND.

Tamanho do Dígrafo	Quantidade de Caminhos	Tempo Buscapath (segundos)	Tempo Separapath (segundos)	Tempo Total (segundos)	Tempo Total (minutos)	Tempo Total (horas)	Tempo Total (dias e horas)
2	2	0,0058	0,02	0,03	-	-	-
3	12	0,0028	0,07	0,07	-	-	-
4	60	0,0121	0,39	0,41	-	-	-
5	320	0,0669	2,66	2,73	-	-	-
6	1950	0,3744	21,51	21,89	-	-	-
7	13692	3,4949	181,85	185,34	3,09	-	-
8	109592	26,2766	1701,44	1727,72	28,80	-	-
9	986400	326,2589	18749,15	19075,41	317,92	5,30	-
10	9864090	11266,3800	161076,17	172342,55	2872,38	47,87	1 dia e 23 horas

Nas redes reais, representadas por dígrafos com pequena quantidade de nós ou dígrafos com matrizes de adjacência esparsa, os programas de enumeração dos caminhos da álgebra de caminhos podem ser utilizados, pois apresentam tempo de execução baixo e finito devido à baixa conectividade [27].

Entretanto, não se pode garantir que a enumeração dos caminhos simples e elementares terá sempre um tempo finito de execução para qualquer rede analisada. Surge, assim, a necessidade de reduzir a quantidade de operações e caminhos enumerados pelos programas da álgebra de caminhos.

⁹ Os dados da Tabela 12 referentes aos dígrafos de tamanho 2 a 9 foram obtidos em uma estação de trabalho na Escola Politécnica da Universidade de São Paulo, Brasil. Os dados relativos ao dígrafo 10 foram obtidos por meio da execução do algoritmo ELND em um servidor remoto na Universitat Politècnica de Catalunya (UPC), em Barcelona, Espanha.

Um mecanismo para reduzir a complexidade do algoritmo e diminuir a quantidade de operações e caminhos enumerados é impor um limite de comprimento máximo que um caminho poderá ter ao ser descoberto. O mecanismo é chamado de filtragem por profundidade e foi implementado no algoritmo PAViLiM.

Novas simulações foram realizadas com o algoritmo PAViLiM e os tempos com diversos valores de filtragem por profundidade foram medidos e são mostrados na Tabela 13 para um dígrafo de 10 nós.

Tabela 13 – Tempo de execução e caminhos descobertos com filtro de profundidade – Algoritmo PAViLiM.

Tamanho do Dígrafo	Filtro por Profundidade	Quantidade de Caminhos	Tempo SearchPath (segundos)	Tempo SortPath (segundos)	Tempo Total (segundos)	Tempo Total (minutos)	Tempo Total (horas)
10	1	90	0,001	0,807	0,81	-	-
10	2	810	0,005	8,9	8,91	-	-
10	3	5850	0,28	91,97	92,25	1,54	-
10	4	36090	2,1	743,84	745,94	12,43	-
10	5	187290	16,13	4608,11	4624,24	77,07	1,28
10	6	792090	133,38	21833,72	21967,1	366,12	6,10
10	7	2606490	1345,21	74037,55	75382,76	1256,38	20,94

Na Tabela 13 percebe-se a redução da quantidade de caminhos enumerados e do tempo total de execução do algoritmo. A consequência imediata da redução do tempo total de execução do algoritmo PAViLiM é a possibilidade de tratar problemas maiores com dígrafos com centenas de nós.

O número total de caminhos enumerados, obtidos com algoritmos sem e com filtragem, em um dígrafo completamente conectado pode ser obtido analiticamente por meio do uso das equações (2) e (3) desenvolvidas neste trabalho e cujas demonstrações são apresentadas no Anexo A.

Algoritmo ELND – Total de caminhos sem filtragem – Eq. (2)

Total de Caminhos (sem filtragem): $2 \times n! + n!/2! + n!/3! + \dots + n!/(n-2)!$

ou

Total de Caminhos (sem filtragem): $\sum_{i=0}^{n-2} C_{n,i} \times (n-i)!, \quad \text{para } n > 3$

Algoritmo PAViLiM – Total de caminhos com filtragem – Eq. (3)

Total de Caminhos (com filtragem): $n!/(n-2)! + n!/(n-3)! + n!/(n-4)! + \dots$

Profundidade Desejada = k termos da expressão Para $n > 3$ e $k < (n-1)$

4.3 Estudo da filtragem por profundidade

Foi feito um estudo da filtragem por profundidade em redes sintéticas obtidas por dois geradores de topologias de rede que criam redes complexas. Os geradores utilizados foram o Waxman [41] e o GT-ITM [42].

O gerador Waxman fornece topologias de rede chamadas **random flat**, que possuem nós distribuídos de forma aleatória (*random*) em um plano, com as arestas ligando esses nós obedecendo a uma distribuição de probabilidade que depende da distância Euclidiana entre os nós [41].

O gerador GT-ITM fornece topologias de rede chamadas **transit stub**, que simulam uma rede hierárquica com múltiplos domínios de trânsito e domínios raiz interligados [42].

As Tabelas 14 e 15 mostram o número médio de caminhos enumerados sem e com filtragem com valor k de profundidade igual a seis ($k = 6$). Foram analisadas 100 topologias de redes *random flat* e *transit stub* com 20, 50 e 100 nós.

Tabela 14 – Comparação do número médio de caminhos enumerados com e sem filtragem por profundidade - *random flat*.

Topologias <i>Random Flat</i>		
Número de Nós	Média do Número Máximo de Caminhos Possíveis de ser Enumerados (sem filtragem)	Número Médio de Caminhos Descobertos com comprimento máximo igual a seis (com filtragem)
20	3.162	692
50	117.489	6.607
100	1.995.262	41.687

Tabela 15 – Comparação do número médio de caminhos enumerados com e sem filtragem por profundidade – *transit stub*.

Topologias <i>Transit Stub</i>		
Número de Nós	Média do Número Máximo de Caminhos Possíveis de ser Enumerados (sem filtragem)	Número Médio de Caminhos Descobertos com comprimento máximo igual a seis (com filtragem)
20	1.738	447
50	16.982	2.042
100	66.069	3.467

As Tabelas 14 e 15 comprovam que a filtragem por profundidade reduz a quantidade de operações, os caminhos enumerados e o tempo de execução dos programas da álgebra de caminhos em uma rede complexa. Será discutido qual é o valor da profundidade mínima que deve ser utilizada em um dado dígrafo para não comprometer a qualidade dos resultados obtidos.

Um dígrafo é dito conectado quando há pelo menos um caminho entre quaisquer pares de nós existentes. Se para esse dígrafo for aplicada a filtragem por profundidade e os caminhos descobertos puderem atingir todos os pares de nós existentes, o dígrafo continua conectado e a qualidade dos resultados pós-filtragem pode não ser severamente comprometida.

Um grafo ou dígrafo (grafo orientado) é completamente determinado por um conjunto de vértices ou nós e pelo conhecimento de quais pares de vértices são adjacentes ou interligados entre si por arestas. Essa informação pode ser obtida por meio de um gráfico ou de uma matriz denominada matriz de adjacência. A teoria dos grafos pode facilmente ser investigada como um tópico da teoria das matrizes [35].

4.3.1 Breve revisão da teoria dos grafos

Seja $G = (V, E)$ um grafo formado pelo conjunto V de n vértices e E de m arestas dado por:

$$V = \{v_1, v_2, v_3, \dots, v_n\}$$

$$E = \{e_1, e_2, e_3, \dots, e_m\}$$

Quando dois vértices estão ligados por uma aresta são chamados de vértices adjacentes e esta aresta pode ser interpretada como uma relação binária denominada adjacência. Quando o grafo é orientado é chamado de dígrafo e representado por $D = (V, E)$.

O número de vértices em um dígrafo é denominado ordem do dígrafo e a relação entre o número de arestas do dígrafo e o maior número de arestas que podem ser criadas se chama densidade do dígrafo [14].

Um dígrafo pode ser representado por uma matriz, denominada matriz de adjacência, em que as linhas dessa matriz representam os vértices “saintes”, ou de origem, e as colunas são os vértices de destino, entrantes ou de chegada. Cada elemento da matriz de adjacência representa uma aresta do dígrafo [43].

A matriz de adjacência $\mathbf{A}$ do dígrafo $\mathbf{D}$ é uma matriz quadrada de dimensão $n \times n$ onde o valor de cada elemento a_{ij} da matriz é:

$$a_{ij} = \begin{cases} 0, & \text{os vértices } v_i \text{ e } v_j \text{ não são adjacentes ou } i = j \text{ (} a_{ii} \text{)} \\ 1, & \text{os vértices } v_i \text{ e } v_j \text{ são adjacentes e possuem uma aresta de conexão.} \end{cases}$$

A matriz de adjacência $\mathbf{A}$ apresenta a diagonal principal com valores iguais a zero e em um dígrafo essa matriz é não-simétrica¹⁰. Ou seja, a_{ij} pode ser diferente de a_{ji} , mas não necessariamente.

4.3.1.1 Percurso

Um percurso é uma sequência qualquer de arestas adjacentes que ligam dois vértices [44]. Essa definição aplica-se a grafos e a dígrafos, ignorando a orientação das arestas.

Um percurso é chamado de elementar quando não passa duas vezes pelo mesmo vértice; é denominado simples se não passa duas vezes pela mesma aresta [44].

¹⁰ Em um grafo essa afirmação não é verdadeira. A matriz de adjacência para um grafo é simétrica.

4.3.1.2 Caminho

Um caminho é um percurso em que todas as arestas possuem a mesma orientação, ou seja, um caminho é um conjunto sucessivo de vértices adjacentes [14]:

$$v_1, v_2, v_3, \dots, v_n$$

onde:

o primeiro vértice (v_1) é o vértice de origem do caminho e o último vértice (v_n) é o vértice de destino, ou término do caminho.

Um caminho do vértice origem (v_1) para o vértice destino (v_n) pode ser representado por meio da notação $p(v_1, v_n)$ e descrito pela sequência de vértices ou arestas:

$$p(v_1, v_n) = v_1, v_2, v_3, \dots, v_n$$

$$p(v_1, v_n) = e_1, e_2, e_3, \dots, e_{n-1}$$

onde

$$e_1 = (v_1, v_2), e_2 = (v_2, v_3), e_3 = (v_3, v_4), \dots, e_{n-1} = (v_{n-1}, v_n)$$

O caminho $p(v_1, v_n)$ tem um comprimento ou distância

$$d(v_1, v_n) = n - 1$$

que indica o número de arestas e a ordem nesse caminho.

Em um dígrafo pode haver diferentes caminhos, com diferentes distâncias ou comprimentos entre um vértice de origem e um vértice de destino. A filtragem por profundidade irá impor o comprimento máximo **k** que um caminho poderá ter.

4.3.2 Teorema das potências da matriz de adjacência

O teorema das potências da matriz de adjacência permite entender a equação para a obtenção do valor de k para a filtragem por profundidade.

Teorema das Potências [35]

Seja A uma matriz de adjacência de um dígrafo G com p vértices, ou seja:

$$V = \{v_1, v_2, v_3, \dots, v_p\} \text{ e } A = A(G).$$

O valor de um elemento (i, j) da matriz potência A^n com n pertencente ao conjunto dos números naturais e $n \geq 1$, é o número de diferentes caminhos com comprimento n entre os vértices v_i e v_j .

Demonstração

A prova é feita por indução.

Para $n = 1$ o teorema é verdadeiro, pois A^1 é a matriz de adjacência e, por definição, os elementos (i, j) da matriz A que possuem o valor de (i, j) igual a 1 (um) são vértices adjacentes e o caminho entre eles é de comprimento 1 (um).

Supor, por hipótese de indução, que para $n = (w - 1)$, com $w \geq 2$, a matriz potência A^{w-1} apresenta elementos (i, j) com valores diferentes de zero que representam o número de caminhos diferentes de comprimento igual a $(w - 1)$ que podem ser encontrados entre os vértices v_i e v_j .

Deve-se provar que a matriz potência A^w possui elementos (i, j) diferentes de zero que fornecem o número de caminhos de comprimento w entre os vértices v_i e v_j .

A matriz potência A^w é obtida pela seguinte multiplicação:

$$A^w = A^{w-1} \cdot A$$

e, pela definição de multiplicação matricial, sabe-se que um elemento (i, j) da matriz é obtido por:

$$a_{ij}^{(w)} = \sum_{k=1}^p a_{ik}^{w-1} \cdot a_{kj}$$

Na equação, o elemento a_{ik}^{w-1} indica o número de caminhos de comprimento $(w - 1)$ entre os vértices v_i e v_k por hipótese de indução. Caso os vértices v_k e v_j sejam adjacentes, o elemento $a_{ij}^{(w)}$ indicará o número de caminhos de comprimento w entre os vértices v_i e v_j no dígrafo G .

Está provado, assim, o Teorema das Potências.

4.3.3 Fórmula da Filtragem por Profundidade

O comprimento de um caminho é o número de arestas que devem ser atravessadas por um nó origem para atingir um nó destino. O menor valor desse comprimento, denominado k , que garanta a conectividade de um dado dígrafo é o valor que será utilizado pelo filtro de profundidade no algoritmo PAViLiM. O valor de k é obtido pela seguinte fórmula:

$$A^0 + A^1 + A^2 + A^3 \dots A^k = X$$

ou

$$\sum_{i=0}^k A^i = X$$

onde:

A é a matriz de adjacência de um dígrafo $n \times n$

A^0 é a matriz identidade I de um dígrafo $n \times n$

A^w é a matriz potência de ordem w de um dígrafo $n \times n$

X é uma matriz que apresenta todos os seus elementos $x_{ij} \neq 0$

k é o menor expoente que produz uma matriz X com elementos diferentes de zero.

Como a matriz X apresenta todos os seus elementos com valores diferentes de zero conclui-se, pelo Teorema das Potências da Matriz de Adjacência, que existem caminhos de comprimento máximo k entre todos os vértices do dígrafo representados pela matriz de adjacência A .

Se uma rede de substrato é representada por sua matriz de adjacência A , ao impor caminhos de comprimento máximo k , obtido pela fórmula da filtragem por profundidade, haverá conectividade entre quaisquer nós na rede e a qualidade dos resultados fornecidos pelos algoritmos da álgebra de caminhos para a enumeração dos caminhos pós-filtragem não será comprometida, como será comprovado pelos resultados experimentais.

4.3.4 Programas para Filtragem por Profundidade

No algoritmo PAViLiM a filtragem por profundidade é realizada por dois programas:

- ✓ **connectivity** – programa que descobre o valor de k para fornecer conectividade completa em uma dada rede de substrato.
- ✓ **searchpath_filter** – programa *searchPath* modificado para suportar enumeração de caminhos com comprimento máximo especificado pelo valor de k .

Com a utilização do programa *connectivity* é possível determinar qual é o comprimento ou profundidade mínima de caminhos que garante a conectividade

completa em uma rede, ou seja, que forneça pelo menos um caminho entre quaisquer pares de nós origem e destino nessa rede.

Por meio dos geradores de topologia de redes Waxman e GT-ITM¹¹, foram criados cenários para a simulação de redes complexas com dois objetivos:

- Determinar a profundidade média para diferentes ordens de redes sinteticamente geradas.
- Comprovar, por meio de um grande número de simulações, que os resultados obtidos são representativos para classes de redes com características semelhantes a um dos dois geradores de topologia utilizados.

As Figuras 17, 18 e 19 apresentam a profundidade mínima necessária para permitir a conectividade de 100 dígrafos de topologias *random flat* com 20, 50 e 100 nós, obtidos com o gerador de topologias de rede *Waxman*. Foi utilizado o programa *connectivity* do algoritmo PAViLiM para obter esses resultados.

Figura 17 – Profundidade mínima para conectividade completa, dígrafos de 20 nós - *random flat*.

¹¹ Os geradores de topologia Waxman e GT-ITM foram criados e são utilizados para simular a topologia de rede da Internet [45] [46] [47][48]. Segundo pesquisas de [49] [50] a topologia de rede da Internet é melhor representada por meio de redes *transit stub* que apresentam o comportamento de interconexão de múltiplas redes e de vários sistemas autônomos (ASs). As redes *random flat* podem ser utilizadas para modelar e representar as topologias de redes interiores a um sistema autônomo (AS).

Figura 18 – Profundidade mínima para conectividade completa, dígrafos de 50 nós - *random flat*.

Figura 19 – Profundidade mínima para conectividade completa, dígrafos de 100 nós - *random flat*.

As Figuras 20, 21 e 22 apresentam a profundidade mínima necessária para permitir a conectividade de 100 dígrafos de topologias *transit stub* com 20, 50 e 100 nós, obtidos com o gerador de topologias de rede *GT-ITM*. Foi utilizado o programa *connectivity* do algoritmo PAViLiM para obter esses resultados.

Figura 20 – Profundidade mínima para conectividade completa, dígrafos de 20 nós – *transit stub*.

Figura 21 – Profundidade mínima para conectividade completa, dígrafos de 50 nós – *transit stub*.

Figura 22 – Profundidade mínima para conectividade completa, dígrafos de 100 nós – *transit stub*.

Nas topologias de rede com n nós os caminhos apresentam um comprimento máximo dado por $(n - 1)$. Observando as figuras de 17 a 22 percebe-se que a cobertura completa das redes é conseguida com valores de profundidade k muito abaixo do comprimento máximo possível. A Tabela 16 apresenta os valores médios da profundidade encontrada nessas simulações.

Tabela 16 – Comprimento Médio Mínimo (Profundidade Mínima) de caminhos para alcançar 100% de conectividade.

Número de Nós	Transit Stub	Random Flat	Comprimento máximo $(n - 1)$
20	8	7	19
50	9	5	49
100	11	4	99

Um dado interessante visto na Tabela 16 é referente à conectividade de uma rede: ora ela diminui, ora ela aumenta com o aumento dos nós da rede, dependendo da topologia de rede analisada. A topologia *random flat* representa o comportamento de múltiplas redes conectadas em um mesmo sistema autônomo (AS). Quanto maior a rede, maior o grau de conexão dos elementos nessa rede e a conectividade entre todos os elementos é possível com caminhos que são cada vez mais curtos [51]. Nas simulações percebe-se que o valor de k , que indica o comprimento mínimo dos caminhos para haver conectividade, nas redes com topologia *random flat* diminui quando se aumenta o tamanho da rede. A topologia *transit stub* representa múltiplas redes conectadas entre si, em sistemas autônomos que também são conectados entre si por elementos que simulam os roteadores de borda em um AS. Nessas topologias, quando se aumenta o número de nós aumenta-se o número de ASs e sistemas autônomos de trânsito adicionais podem surgir entre um nó origem e um nó destino, o que aumenta o comprimento mínimo dos caminhos para que haja a conectividade. Nas simulações percebe-se que o valor de k , que indica o comprimento mínimo dos

caminhos para haver conectividade, nas redes com topologia *transit stub*, aumenta quando se aumenta o tamanho da rede.

4.4 Algoritmo PAViLiM

O algoritmo PAViLiM foi criado para resolver o problema VNE de mapeamento de enlaces virtuais (*Virtual Links*) para caminhos físicos na rede de substrato (SN). Ele utilizou como referência o algoritmo ELND.

O algoritmo PAViLiM foi implementado no ambiente de programação e desenvolvimento de algoritmos, análise de dados, visualização e computação numérica MATLAB¹² [52]. A decisão pela utilização do MATLAB está relacionada com a rapidez no desenvolvimento de aplicações e resolução de problemas quando comparado a linguagens de programação tradicionais, como JAVA, C, C++ e Fortran [53].

PAViLiM resolve por meio da álgebra de caminhos a fase 2 (dois) do problema VNE, isto é, o mapeamento dos enlaces virtuais. A fase 1 (um), mapeamento dos nós virtuais, é resolvida por meio de algoritmos consagrados para a solução do problema VNE disponibilizados pelo ambiente ALEVIN [6] [54].

ALEVIN (*Algorithms for Embedding of Virtual Networks*) é um *framework* disponível como software livre utilizado para desenvolver, comparar e analisar algoritmos VNE. Um conjunto popular de algoritmos foram implementados no ALEVIN e, devido à sua natureza, novos softwares podem ser adicionados para permitir aos pesquisadores executar simulações, investigar e realizar modificações nos algoritmos com novos conjuntos de entradas e critérios de otimização [56].

¹² MATLAB – *Matrix Laboratory* - é um software interativo de alto desempenho voltado para o cálculo numérico. O MATLAB integra análise numérica, cálculo com matrizes, processamento de sinais e construção de gráficos em ambiente fácil de usar, onde problemas e soluções são expressos somente como eles são escritos, matematicamente ao contrário da programação tradicional [53].

4.5 Diagrama de blocos do algoritmo PAViLiM e solução do problema VNE

A solução do problema VNE ocorre por meio da integração do algoritmo PAViLiM com algoritmos existentes e implementados no ambiente ALEVIN.

A Figura 23 apresenta o diagrama de blocos do algoritmo PAViLiM.

Figura 23 – Diagrama de Blocos do Algoritmo PAViLiM.

De acordo com a Figura 23, o **bloco 1** inicia o algoritmo PAViLiM. Os **blocos 2 e 3** fazem a leitura da matriz de adjacência e das matrizes de métricas da rede de substrato fornecidas por algoritmos do ambiente ALEVIN. O **bloco 4** realiza a enumeração e pré-ordenação dos caminhos com filtragem por profundidade. A pré-ordenação utiliza como critério de otimização o número de saltos (*hop count*).

Os **blocos 5 e 6** fazem a leitura das políticas de atendimento das requisições virtuais e dos critérios de ordenação fornecidos por programas do ambiente ALEVIN. O **bloco 7** chama um subprograma que irá realizar o atendimento das requisições virtuais.

O atendimento de requisições de rede virtual (VNRs) pode ser feito de duas formas: *online* ou *offline*. Quando o atendimento é feito de forma *online*, as requisições de rede virtual que chegam para ser atendidas são inseridas em uma fila de atendimento e utiliza-se a política FIFO (*First-In First-Out*) para o atendimento da fila. Quando a forma de atendimento é *offline*, todas as requisições de rede virtual que devem ser atendidas são lidas e, por meio de uma política de atendimento, determina-se qual será a ordem em que cada VNR será atendida. São seis políticas de atendimento possíveis de solicitar:

- 1) MCF – *Most Consuming First*
- 2) LCF – *Least Consuming First*
- 3) FIFO – *First-In First-Out*
- 4) MCF-UP (*USP Proposed*)
- 5) LCF-UP (*USP Proposed*)
- 6) *Faça a sua escolha (Do your choice)*

Política **1) MCF** analisa e ordena todas as requisições em ordem decrescente de quantidade de recursos solicitados.

Política **2) LCF** analisa e ordena todas as requisições em ordem crescente de quantidade de recursos solicitados. Para calcular a quantidade de recursos solicitados, em ambas as propostas, deve-se realizar uma adição simples da largura de banda (BW) e de recursos de CPU solicitados para cada VNR.

Política **3) FIFO** faz o atendimento *online* ou simula esse atendimento. As requisições virtuais vão sendo atendidas, quando possível, na ordem em que chegam.

As políticas **4) MCF-UP e 5) LCF-UP** são duas políticas novas baseadas no MCF e LCF originais, que agora utilizam uma ponderação das métricas de largura de banda e CPU pelos recursos totais disponíveis na rede de substrato. Nessas novas políticas

ainda se somam os valores das métricas de largura de banda e CPU, porém, agora ponderados por dois fatores multiplicadores (α e β) que indicam, na política, a influência desejada pela métrica BW e pela métrica CPU. A expressão utilizada para o cálculo da quantidade de recursos solicitados por cada requisição de rede virtual (VNR ou VN request) é dada por:

$$\text{Recursos Solicitados} = \frac{\alpha * (\sum BW)_{VN(i)}}{(\sum BW)_{SUB}} + \frac{\beta * (\sum CPU)_{VN(i)}}{(\sum CPU)_{SUB}}$$

α e β obedecem a:

$$0 \leq \alpha \leq 1, 0 \leq \beta \leq 1 \text{ e} \\ \alpha + \beta = 1$$

onde:

$(\sum BW)_{VN(i)}$ - representa a somatória dos recursos solicitados de largura de banda para uma VNR.

$(\sum BW)_{SUB}$ - representa a somatória dos recursos de largura de banda existentes na SN.

$(\sum CPU)_{VN(i)}$ - representa a somatória dos recursos solicitados de CPU para uma VNR.

$(\sum CPU)_{SUB}$ - representa a somatória dos recursos de CPU existentes na rede de substrato (SN).

Política **6) “Faça a sua escolha”** (*Do your choice*) permite que os programas do ALEVIN indiquem qual será a ordem de atendimento das VNRs que deve ser utilizada.

Quando uma VNR deve ser atendida pelo programa de atendimento de requisições de rede virtual todos os seus enlaces virtuais devem ser mapeados para caminhos na rede de substrato. Os caminhos que são oferecidos na tentativa de realizar o mapeamento do enlace virtual são ordenados por meio da álgebra de caminhos com critérios de ordenação solicitados por programas do ALEVIN e identificados no **bloco 6**

do algoritmo PAViLiM. Os critérios de ordenação dos caminhos suportados pelo algoritmo PAViLiM são:

- 1) BW,CPU
- 2) CPU,BW
- 3) HOP,BW,CPU
- 4) HOP,CPU,BW
- 5) DELAY,BW,CPU
- 6) DELAY,CPU,BW
- 7) AVAILABILITY,BW,CPU
- 8) AVAILABILITY,CPU,BW
- 9) disponível ou faça a sua escolha (*do your choice*)
- 10) BW+CPU,HOP
- 11) HOP,BW+CPU
- 12) HOP, BW Total

Por meio da álgebra de caminhos é realizada a ordenação dos caminhos da rede de substrato que serão oferecidos para o atendimento de um enlace virtual segundo um dos critérios de ordenação selecionados.

Por exemplo, o critério de ordenação **3) HOP, BW, CPU** indica que devem ser consideradas três métricas e três funções de combinação de métricas para número de saltos (HOP), largura de banda disponível (BW) e recursos de CPU disponível na rede de substrato. As sínteses padrão (*default*) que devem ser consideradas para cada uma dessas métricas são:

$$M = \{M_1, M_2, M_3\}$$

$$M = \{HOP, BW, CPU\}$$

$$\bar{S} = \{\bar{S}_1, \bar{S}_2, \bar{S}_3\}$$

$$\bar{S} = \{add, min, min\}$$

Para o critério de ordenação selecionado no exemplo, a ordenação léxica multidimensional $\preceq_{ML}$ padrão (*default*) é dada por:

$$\preceq_{ML} = \{\geq, \leq, \leq\}$$

No algoritmo PAViLiM cada critério de ordenação possui valores padrão (*default*) para a síntese e ordenação léxica multidimensional, mas esses valores podem ser alterados, se desejado, quando da execução do algoritmo.

A Tabela 17 apresenta os valores padrão (*default*) utilizados pelos critérios de ordenação para as sínteses e para a ordenação léxica multidimensional $\preceq_{ML}$ que são suportados pelo algoritmo PAViLiM.

Tabela 17 – Sínteses e ordenação léxica multidimensional padrão.

Número do Critério	Métricas	Sínteses	Ordenação Léxica Multidimensional
1	BW,CPU	{min,min}	{ $\leq, \leq$ }
2	CPU,BW	{min,min}	{ $\leq, \leq$ }
3	HOP,BW,CPU	{add,min,min}	{ $\geq, \leq, \leq$ }
4	HOP,CPU,BW	{add,min,min}	{ $\geq, \leq, \leq$ }
5	DELAY,BW,CPU	{add,min,min}	{ $\geq, \leq, \leq$ }
6	DELAY,CPU,BW	{add,min,min}	{ $\geq, \leq, \leq$ }
7	AVAIL,BW,CPU	{mult,min,min}	{ $\leq, \leq, \leq$ }
8	AVAIL,CPU,BW	{mult,min,min}	{ $\leq, \leq, \leq$ }
9	-	-	-
10	BW+CPU,HOP	{min,add}	{ $\leq, \geq$ }
11	HOP,BW+CPU	{add,min}	{ $\geq, \leq$ }
12	HOP,BW Total	{add,add}	{ $\leq, \leq$ }

A Figura 24 apresenta o módulo de atendimento de requisições virtuais do algoritmo PAViLiM. A numeração dos blocos é sequencial àquela apresentada na Figura 23.

Figura 24 – Diagrama de blocos do programa de atendimento de VNRs do algoritmo PAViLiM.

De acordo com a Figura 24, o **bloco 7** inicia o programa de atendimento de requisições de rede virtual do algoritmo PAViLiM. O **bloco 8** faz a preparação (inicialização) para o atendimento de requisições virtuais, que consiste na validação do critério de ordenação dos caminhos solicitados e na atualização do vetor peso (w) dos enlaces dos caminhos obtidos. O **bloco 9** faz a leitura de uma VNR por vez, se o atendimento for *online*, ou de todas as VNRs, se o atendimento for *offline*.

No **bloco 10** ocorre o mapeamento dos nós virtuais para nós físicos na rede de substrato. Esse bloco é executado pelo programa ALEVIN. O **bloco 11** realiza o mapeamento dos enlaces virtuais para caminhos na rede de substrato. O **bloco 12** executa a atualização de métricas e reordenação dos caminhos. Se houver mais atendimentos por realizar e o atendimento que está sendo executado for *online*, tem-se que do bloco 12 deve-se retornar ao bloco 9; se o atendimento que está sendo executado for *offline*, tem-se que do bloco 12 deve-se retornar ao bloco 10, pois todas as VNRs já foram lidas.

4.6 Principais programas do algoritmo PAViLiM

Cada bloco das Figuras 23 e 24 é implementado por um ou mais programas desenvolvidos ou modificados para resolver o problema da virtualização de redes. A Tabela 18 apresenta os principais programas e procedimentos. O Anexo B fornece todos os programas mostrados nessa tabela.

Tabela 18 – Principais Programas do algoritmo PAViLiM.

Número do Bloco	Programas	Descrição
1	main_PavilimA	Programa Principal.
2	main_PavilimA	Leitura da Matriz de Adjacência da Rede de Substrato.
3	main_PavilimA	Leitura das Métricas da Matriz de Substrato.
4	connectivity searchPath_filter sortPath	Enumeração dos caminhos com filtragem de profundidade.
5	main_PavilimA	Leitura da Política de Atendimento das Requisições Virtuais.
6	main_PavilimA	Leitura do critério de ordenação dos caminhos.
7	main_PavilimB	Programa de Atendimento de Requisições Virtuais.
8	updateNewMetrics orderNewPaths (initialization)	Validação do critério de ordenação dos caminhos. Atualização do vetor peso (w) dos enlaces dos caminhos obtidos. Preparação (Inicialização) para atendimento de Requisições Virtuais.
9	main_PavilimB	Leitura das Requisições Virtuais (VN requests).
10	ALEVIN	Mapeamento de Nós Virtuais para a Rede de Substrato.
11	attendVNsHiddenX	Atendimento de uma requisição virtual.
12	updateNewMetrics orderNewPaths	Atualização de Métricas e Reordenação dos Caminhos.

4.7 Integração entre PAViLiM e ALEVIN

A integração entre o algoritmo PAViLiM e o ambiente ALEVIN é feita por meio da troca de arquivos em um diretório (pasta) compartilhado no servidor de simulação. O algoritmo PAViLiM foi desenvolvido para a plataforma MATLAB e os programas do ambiente ALEVIN foram escritos na linguagem de programação JAVA. A integração ideal, recomendada para trabalhos futuros, será a conversão do algoritmo PAViLiM para JAVA e a inserção desse algoritmo no ambiente ALEVIN. Como o presente trabalho é uma prova de conceito (POC) para demonstrar o grande potencial da álgebra

de caminhos para a resolução do problema VNE, a conversão do algoritmo PAViLiM para JAVA não foi priorizada nem executada.

Os programas do algoritmo PAViLiM comunicam-se com os programas do ambiente ALEVIN por meio da troca de arquivos sem formatação ou caracteres especiais. Arquivos são fornecidos por ambas as partes seguindo uma convenção de nomes que permitem um intercâmbio de informações e a interação entre os algoritmos.

A seguinte informação deve ser trocada entre os dois ambientes:

- Início da Simulação.
- Rede de substrato (SN).
- Requisições de rede virtual.
- Resultados da Simulação.
- Término da Simulação.

4.7.1 Início da simulação

A simulação inicia com a criação do arquivo ***virtualStart.dat*** feita pelo ALEVIN. A presença do arquivo, além de indicar o início da simulação, fornece a política de atendimento das requisições virtuais, o critério de ordenação dos caminhos e o número de requisições de rede virtual que serão fornecidas para o caso de atendimento *offline*. Quando o atendimento é *online*, a quantidade de requisições é ignorada e o término da simulação é controlado por meio do arquivo de término ***virtualFinish.dat***.

A Figura 25 mostra o formato (*layout*) do arquivo ***virtualStart.dat***.

Figura 25 – *Layout* do arquivo *virtualStart.dat*.

Os valores utilizados para a política de atendimento e o critério de ordenação estão, respectivamente, na Tabela 19 e na Tabela 17.

Tabela 19 – Valores para Política de Atendimento.

Política de Atendimento	Valor
MCF	1
LCF	2
FIFO	3
MCF-UP	4
LCF-UP	5
Faça a sua escolha	6

Exemplo de utilização do arquivo virtualStart.dat.

Suponha que se deseja simular o atendimento *offline* de 20 (vinte) requisições de rede virtual, segundo política de atendimento MCF e critério de ordenação de caminhos: saltos (HOP), largura de banda (BW) disponível e capacidade de processamento (CPU) disponível. O arquivo *virtualStart.dat*, para representar essa solicitação será dado por:

virtualStart.dat

1 3 20

O valor “1” no arquivo representa a política de atendimento MCF. O valor “3” indica o critério de ordenação dos caminhos: HOP, BW, CPU, e o valor “20” representa a quantidade de requisições de rede virtual que serão fornecidas.

4.7.2 Rede de substrato (SN)

A informação sobre a topologia da rede de substrato (SN) e sobre os pesos referentes a cada métrica utilizada é fornecida pelo ALEVIN por meio de três arquivos:

- `substrateNetwork.dat`
- `substrateMetric1.dat`
- `substrateMetric2.dat`

O arquivo ***substrateNetwork.dat*** fornece a topologia da rede de substrato representada por meio de uma matriz de adjacência.

Os arquivos ***substrateMetric1.dat*** e ***substrateMetric2.dat*** são os pesos aplicados às arestas e aos vértices, relativos, respectivamente, à largura de banda (BW) disponível na aresta e à capacidade de processamento (CPU) disponível no vértice. O arquivo *substrateMetric1.dat* é uma matriz de pesos indicativa do valor da largura de banda disponível em cada aresta. O arquivo *substrateMetric2.dat* é um vetor de pesos indicativo do valor da capacidade de processamento (CPU) disponível em cada vértice.

As simulações do capítulo 5 trabalham apenas com as métricas de largura de banda (BW) e de recursos de CPU. Porém, se houver necessidade de simulações com outras métricas é possível criar arquivos *substrateMetricX.dat* que representarão um vetor ou uma matriz de pesos da respectiva métrica. O algoritmo PAViLiM, por utilizar a álgebra de caminhos, é extremamente flexível para adição, remoção e modificação de métricas, e palavras-peso em uma rede física ou virtual.

4.7.3 Requisições de rede virtual

Os programas do ambiente ALEVIN devem fornecer, para cada requisição de rede virtual, cinco arquivos:

- Arquivo da requisição de rede virtual (VN request).
- Arquivo de solicitação de recursos da métrica 1 (BW).
- Arquivo de solicitação de recursos da métrica 2 (CPU).
- Arquivo de consumo de CPU em nós escondidos (*hidden nodes*).
- Mapeamento de nós virtuais para nós da rede de substrato.

A convenção de nomes dos arquivos adotada é:

- *virtualRequest_#* → requisição de rede virtual #.
- *virtualMetric1_#* → métrica 1 para a VNR #.
- *virtualMetric2_#* → métrica 2 para a VNR #.
- *virtualHidden_#* → consumo de CPU nos nós escondidos para a VNR #.
- *virtual_to_Real_#* → mapeamento de nós virtuais para nós físicos da VNR #.

O símbolo “#” deve ser substituído por um número sequencial que indica o número da requisição de rede virtual que será fornecida pelo ALEVIN e lida pelos programas do PAViLiM.

Os quatro primeiros arquivos - requisição de rede virtual, métricas 1 e 2, e *hidden nodes* - são fornecidos pelo ALEVIN e lidos pelo PAViLiM e representam o **bloco 9** “Leitura de Requisições Virtuais (VN *requests*)” do algoritmo PAViLiM apresentado na Figura 24.

O arquivo de mapeamento dos nós virtuais para nós físicos é fornecido pelo ALEVIN após cada atendimento de uma VNR. Essa tarefa é descrita no **bloco 10** da Figura 24.

4.7.4 Resultado da simulação

O algoritmo PAViLiM, no **bloco 11** da Figura 24, deve realizar o atendimento de uma requisição de rede virtual. Ele informa se o atendimento da requisição de rede virtual

número “#” foi atendida ou não por meio do arquivo ***virtualSuccess_#***. Caso a requisição tenha sido atendida, será gerado um conjunto de arquivos com o mapeamento dos enlaces virtuais para caminhos físicos que foram escolhidos na rede de substrato.

Os seguintes arquivos são criados:

- *virtualSuccess_#* → apresenta valor 0 para fracasso ou valor 1 para sucesso do atendimento da VNR #.
- *virtualAttend_#_X_Y* → fornece o caminho na SN para o enlace virtual entre os nós virtuais X e Y da VNR #.

4.7.5 Término da simulação

O algoritmo PAViLiM testa a existência de um arquivo de término da simulação, o qual será criado pelo ALEVIN quando este desejar finalizar a mesma.

O arquivo deve ter o formato ***virtualFinish.dat*** e basta a existência do arquivo com qualquer conteúdo. Os programas do algoritmo PAViLiM, ao perceberem a presença desse arquivo, irão parar sua execução e gerar uma atualização dos recursos disponíveis na rede de substrato para cada métrica que foi utilizada.

4.7.6 Tabela resumo de arquivos da integração

A integração entre os programas do algoritmo PAViLiM e os programas do ambiente ALEVIN acontece por meio dos arquivos apresentados na Tabela 20.

Tabela 20 – Tabela Resumo de Arquivos da Integração.

Nome do Arquivo	Quem deve fornecer	Descrição
virtualStart.dat	ALEVIN	indica início da simulação; fornece política de atendimento e critério de ordenação dos caminhos.
substrateNetwork.dat substrateMetric1.dat substrateMetric2.dat	ALEVIN	matriz de adjacência da rede de substrato e peso das métricas de largura de banda e CPU.
virtualRequest_# virtualMetric1_# virtualMetric2_# virtualHidden_# virtual_to_Real_#	ALEVIN	arquivos que representam a requisição de rede virtual número “#”. mapeamento de nós virtuais para nós físicos.
virtualSuccess_# virtualAttend_#_X_Y	PAViLiM	mapeamento de enlaces virtuais para caminhos físicos na rede de substrato.
virtualFinish.dat	ALEVIN	término da simulação.

4.7.7 Exemplo da Integração

Nesta seção é fornecido um exemplo dos arquivos que devem ser gerados no processo de integração entre o algoritmo PAViLiM e o ambiente ALEVIN.

A Figura 26 apresenta uma rede de substrato e suas respectivas métricas de largura de banda nos enlaces e de recursos de CPU entre parênteses para cada nó.

Figura 26 – Rede de Substrato.

Os seguintes arquivos devem ser gerados pelo ALEVIN:

- *substrateNetwork.dat*
- *substrateMetric1.dat*
- *substrateMetric2.dat*

E o conteúdo dos arquivos é apresentado:

substrateNetwork.dat

```
0 1 0
0 0 1
1 1 0
```

substrateMetric1.dat

```
0 10 0
0 0 5
30 20 0
```

substrateMetric2.dat

```
20 50 70
```

O algoritmo PAViLiM irá ler esses arquivos e realizar a enumeração e pré-ordenação dos caminhos com filtragem por profundidade. A pré-ordenação utiliza como critério de otimização o número de saltos (*hop count*).

A Figura 27 apresenta duas requisições de rede virtual que devem ser atendidas.

Figura 27 – Requisições de Rede Virtual.

Os pesos entre parênteses nos nós representam a solicitação de capacidade de processamento (CPU) para cada nó virtual. Os pesos entre parênteses nos enlaces representam a capacidade de processamento consumido nos nós escondidos (*hidden nodes*), que são nós intermediários em um caminho físico mapeado por um enlace virtual. Os pesos nos enlaces identificam a largura de banda solicitada pelo enlace virtual (*virtual link*) na VNR.

Os arquivos que serão gerados pelo ALEVIN para atendimento das requisições virtuais da Figura 27 são divididos em dois conjuntos de arquivos:

Arquivos para atendimento da requisição de rede virtual 1 (VNR1):

- virtualRequest_1
- virtualMetric1_1
- virtualMetric2_1
- virtualHidden_1
- virtual_to_Real_1

Arquivos para atendimento da requisição de rede virtual 2 (VNR2):

- virtualRequest_2
- virtualMetric1_2
- virtualMetric2_2
- virtualHidden_2
- virtual_to_Real_2

O conteúdo dos arquivos de atendimento das VNRs está na Tabela 21.

Tabela 21 – Conteúdo dos arquivos de atendimento.

Arquivos VNR 1	Arquivos VNR2
<i>virtualRequest_1</i> 0 1 1 0	<i>virtualRequest_2</i> 0 1 0 0
<i>virtualMetric1_1</i> 0 5 5 0	<i>virtualMetric1_2</i> 0 10 0 0
<i>virtualMetric2_1</i> 5 10	<i>virtualMetric2_2</i> 10 20
<i>virtualHidden_1</i> 0 1 1 0	<i>virtualHidden_2</i> 0 2 0 0
<i>virtual_to_Real_1</i> 1 3 2 1	<i>virtual_to_Real_2</i> 1 3 2 2

Para iniciar a simulação, o algoritmo PAViLiM precisa saber qual é a política de atendimento, o critério de ordenação e a quantidade de requisições de rede virtual que devem ser atendidas. Essa informação é fornecida pelo ALEVIN no arquivo *virtualStart.dat*, que, no exemplo, possui o seguinte conteúdo:

virtualStart.dat
3 3 2

O arquivo de início da simulação indica que a política de atendimento solicitada é a FIFO, com critério de ordenação de caminhos: HOP, BW, CPU e que duas VNRs devem ser atendidas. O algoritmo PAViLiM processa as requisições na ordem em que elas aparecem e produz os resultados mostrados na Tabela 22.

Tabela 22 – Resultados do algoritmo PAViLiM.

Requisição de Rede Virtual 1 (VNR 1)	Requisição de Rede Virtual 2 (VNR2)
virtualSuccess_1 1	virtualSuccess_2 1
virtualAttend_1_12 3 1	virtualAttend_2_12 3 2
virtualAttend_1_21 1 2 3	

No fim do atendimento das requisições virtuais o ALEVIN gera um arquivo com o nome *virtualFinish.dat* e a simulação termina. O algoritmo PAViLiM gera, no fim da simulação, novas matrizes de métricas da rede de substrato mostradas a seguir:

substrateMetric1.dat

0 5 0
0 0 0
25 20 0

substrateMetric2.dat

10 49 65

4.8 Considerações Finais do Capítulo

O capítulo apresentou o algoritmo PAViLiM que foi criado para resolver o problema VNE (fase 2) de mapeamento de enlaces virtuais (*Virtual Links*) para caminhos físicos na rede de substrato (SN). Foi discutido no capítulo a integração do PAViLiM com algoritmos do ambiente ALEVIN que resolvem o problema VNE (fase 1).

5. Simulação e Discussão de Resultados

Este capítulo apresenta resultados de diversas simulações para o atendimento de requisições de redes virtuais com a utilização do algoritmo PAViLiM e de programas disponíveis no ambiente ALEVIN. É demonstrado o efeito da política de atendimento e o critério de ordenação no conjunto de resultados. A metodologia e as condições das simulações são explicadas e os resultados das simulações mostrados e comentados. São apresentadas, nas últimas seções do capítulo, novas possibilidades de atendimentos de VNRs e mapeamento de enlaces virtuais.

5.1 Efeito da política e do critério de ordenação sobre o atendimento de VNRs

A política de atendimento e o critério de ordenação¹³ escolhidos afetam os resultados obtidos pelos algoritmos de virtualização de redes. Não é o escopo deste trabalho determinar qual é a melhor política ou indicar regras de otimização, o objetivo desta seção é apresentar um exemplo dessa influência.

A melhor política de atendimento depende do critério de otimização e de políticas comerciais utilizadas pelos provedores de rede virtual [4] [6] [59]. A álgebra de caminhos mostra-se como uma solução extremamente flexível para:

- Na avaliação de um atendimento, utilizar várias políticas de atendimento simultâneas e escolher a melhor, aquela que consegue atender a maior quantidade de requisições ou fornecer maior receita.
- Atender os requisitos e necessidades contratuais (SLA) do cliente.
- Mudar a política de atendimento para priorizar o atendimento de uma VNR.

A influência da política no atendimento de VNRs foi constatada por meio de um exemplo. Foram realizadas simulações utilizando uma rede de substrato com 10 nós, representada pelo dígrafo da Figura 28.

¹³ Nesta seção poderão ser utilizados os termos **política** ou **política de atendimento** com o significado de “*política de atendimento e critério de ordenação*”.

Figura 28 - Dígrafo com 10 nós utilizado nas simulações.

5.1.1 Exemplo dos Efeitos da Política

O dígrafo da Figura 28 foi obtido pelo gerador de topologias de rede Waxman, que utiliza o modelo de redes *random flat*. O gerador Waxman criou vinte requisições de rede virtual para serem atendidas. O Anexo C fornece as matrizes de adjacência das vinte requisições de rede virtual e os mapeamentos de nós virtuais utilizados para cada VNR.

O algoritmo PAViLiM foi utilizado nas simulações para mapeamento dos enlaces virtuais para caminhos na rede de substrato. Cinco políticas de atendimento e dois critérios de ordenação dos caminhos foram utilizados:

Políticas de Atendimento

- MCF
- LCF
- FIFO
- MCF-UP
- LCF-UP

Critério de Ordenação dos Caminhos

- Ordenação 1: BW e CPU
- Ordenação 2: CPU e BW

Pela álgebra de caminhos, as ordenações 1 e 2 utilizam as seguintes métricas:

Conjunto de Métricas

$$M_1 = \{BW, CPU\} \quad - \text{ordenação 1}$$

$$M_2 = \{CPU, BW\} \quad - \text{ordenação 2}$$

As funções de combinação de métricas, sínteses e ordenação léxica multidimensional são iguais para as duas ordenações:

$$F = M \quad - \text{Funções de Combinação de Métricas}$$

$$\bar{S} = \{min, min\} \quad - \text{Operações de Síntese}$$

$$\preceq_{ML} = \{\leq, \leq\} \quad - \text{Relações de Ordenação}$$

As Tabelas 23, 24, 25, 26 e 27 apresentam os resultados do atendimento das vinte requisições de rede virtual. As cinco políticas de atendimento são avaliadas para cada um dos dois critérios de ordenação dos caminhos: *ordenação 1* e *ordenação 2*.

Tabela 23 – Atendimento das Requisições de Redes Virtuais (*VN requests*) utilizando políticas diferentes de atendimento.

Tipos de Atendimento	Requisições Ordenadas	Requisições Atendidas Ordenação 1 BW e CPU	Requisições Atendidas Ordenação 2 CPU e BW
MCF	19;3;4;5;9;2;14;1;11;18; 16;15;20;12;6;13;7;8;17;10	19;3;4;5;11;8	19;3;4;5;2;6
LCF	10;17;8;7;13;6;12;20;15;16; 18;11;1;14;2;9;5;4;3;19	10;17;8;7;13;6;12;20;15;16;18	10;17;8;7;13;6;12;20;15;16;18
FIFO	1;2;3;4;5;6;7;8;9;10;11;12; 13;14;15;16;17;18;19;20	1;2;3;4;5;6;10	1;2;3;4;5;6;7
MCF-UP	19;3;4;9;5;2;14;11;1;18;12; 15;20;16;6;13;7;8;17;10	19;3;4;9;18	19;3;4;9;2
LCF-UP	10;17;8;7;13;6;16;20;15;12; 18;1;11;14;2;5;9;4;3;19	10;17;8;7;13;6;16;20;15;12;18	10;17;8;7;13;6;16;20;15;12;18

A Tabela 23 mostra as VNRs ordenadas para cada política de atendimento e exibe quais requisições de rede virtual foram atendidas para cada critério de ordenação.

As Tabelas 24 e 25 mostram, para as cinco políticas de atendimento, o efeito da política sobre o atendimento de VNRs. Nas tabelas o caractere 'X' representa uma VNR atendida e o caractere '–' uma VNR não-atendida.

Tabela 24 – Requisições de Redes Virtuais Atendidas x Políticas de Atendimento para Ordenação 1.

Tipo de Atendimento	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
MCF	-	-	X	X	X	-	-	X	-	-	X	-	-	-	-	-	-	-	X	-
LCF	-	-	-	-	-	-	X	X	-	X	-	X	X	-	X	X	X	X	-	X
FIFO	X	X	X	X	X	X	-	-	-	X	-	-	-	-	-	-	-	-	-	-
MCF-UP	-	-	X	X	-	-	-	-	X	-	-	-	-	-	-	-	-	X	X	-
LCF-UP	-	-	-	-	-	X	X	X	-	X	-	X	X	-	X	X	X	X	-	X

Tabela 25 – Requisições de Redes Virtuais Atendidas x Políticas de Atendimento para Ordenação 2.

Tipo de Atendimento	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
MCF	-	X	X	X	X	X	-	-	-	-	-	-	-	-	-	-	-	-	X	-
LCF	-	-	-	-	-	X	X	X	-	X	-	X	X	-	X	X	X	X	-	X
FIFO	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-	-	-	-	-	-
MCF-UP	-	X	X	X	-	-	-	-	X	-	-	-	-	-	-	-	-	-	X	-
LCF-UP	-	-	-	-	-	X	X	X	-	X	-	X	X	-	X	X	X	X	-	X

Na Tabela 24 observe que a requisição 14 não pôde ser atendida por nenhuma política de atendimento, enquanto na Tabela 25 foram duas as requisições não atendidas: 11 e 14.

As Tabelas 26 e 27 apresentam a relação C/R (custo/receita), o total de recursos solicitados (receita)¹⁴, os recursos realmente utilizados (custo)¹⁵ e os recursos finais disponíveis na rede de substrato para as cinco políticas e critérios de ordenação 1 e 2.

Tabela 26 – Requisições de Redes Virtuais Atendidas x Políticas de Atendimento – Ordenação 1.

Tipo de Atendimento	Total de Requisições Atendidas	C/R (custo / receita)	Total de Recursos Solicitados (Receita)	Total de Recursos Utilizados (Custo)	Recursos Disponíveis na SN
MCF	6	2,13	808	1724	1306
LCF	11	2,27	663	1477	1553
FIFO	7	2,42	740	1797	1233
MCF-UP	5	2,08	757	1579	1451
LCF-UP	11	2,22	663	1475	1555

Tabela 27 – Requisições de Redes Virtuais Atendidas x Políticas de Atendimento – Ordenação 2.

Tipo de Atendimento	Total de Requisições Atendidas	C/R (custo / receita)	Total de Recursos Solicitados (Receita)	Total de Recursos Utilizados (Custo)	Recursos Disponíveis na SN
MCF	6	2,10	851	1789	1241
LCF	11	2,20	663	1462	1568
FIFO	7	2,31	800	1849	1181
MCF-UP	5	2,07	786	1629	1401
LCF-UP	11	2,16	663	1433	1597

A diferença entre o total de recursos solicitados (receita) e o total de recursos utilizados (custo) nas Tabelas 26 e 27 deve-se ao processo de mapeamento dos

¹⁴ recursos solicitados: cada requisição de rede virtual atendida possui uma quantidade de recursos solicitados que é a soma da largura de banda e CPU requisitada para cada enlace e nó virtual. A quantidade de recursos solicitados por todas as requisições virtuais atendidas denomina-se *receita*.

¹⁵ recursos utilizados: o mapeamento de um enlace virtual para um caminho na SN consome recursos adicionais de largura de banda e CPU em nós intermediários (*hidden nodes*), a menos que o enlace virtual seja mapeado diretamente para uma aresta na SN. Por esse motivo, o atendimento de uma requisição de rede virtual sempre consumirá mais recursos do que aqueles que foram solicitados. A quantidade de recursos utilizados na rede de substrato por todas as VNRs atendidas é chamada de *custo*.

enlaces virtuais para caminhos na SN. Quando um caminho possui mais de uma aresta ligando o nó de origem ao nó de destino, existe um consumo de largura de banda (BW) extra para cada aresta utilizada pelo caminho e há um consumo de processamento (CPU) adicional nos nós intermediários, chamados de nós escondidos (*hidden nodes*), presentes no caminho.

Pelas Tabelas 26 e 27 percebe-se que a quantidade de VNRs atendidas e a qualidade dos mapeamentos realizados no que se refere à relação custo e receita (C/R) são dependentes da combinação da política de atendimento e do critério de ordenação dos caminhos escolhidos. A política de atendimento MCF mostra que, ao ser utilizada, proporciona a maior receita possível entre as demais políticas, independentemente do critério de ordenação. A política de atendimento LCF tem como vantagem o atendimento da maior quantidade de requisições de rede virtual (VNRs). As políticas MCF-UP e LCF-UP apresentam relação C/R menor do que as políticas das quais se originaram. A política de atendimento FIFO deve ser utilizada quando o atendimento é *online*. Com essa política, o *backtrack* e o remapeamento periódico precisam ser realizados para que ocorra uma melhor utilização dos recursos da rede de substrato.

Como já mencionado, a escolha da melhor combinação de política e critério de ordenação é função da estratégia de negócios e dos objetivos comerciais que devem ser atingidos.

Segundo Botero [23] e Papadimitriou [60], quando a política comercial for privilegiar o atendimento aos clientes mais rentáveis e houver uma busca pela redução de custos e maximização de lucros nesse atendimento, a política de atendimento utilizada é a MCF com o critério de ordenação que utilize o número de saltos (*HOPs*) e síntese minimizativa.

5.2 Metodologia de simulação

Virtualização de Redes é um assunto vasto e extremamente complexo [1] [2]. O algoritmo PAViLiM resolve a fase de mapeamento de enlaces virtuais do problema VNE com o uso da álgebra de caminhos. A fase de mapeamento de nós virtuais é feita por meio de algoritmos “gulosos” (*greedy algorithms*) ou coordenados (*coordinated node mapping*) executados no ambiente ALEVIN [6] [54].

Atualmente há uma cooperação científica internacional para projetos, estudos e pesquisas entre a Escola Politécnica da Universidade de São Paulo e a Universitat Politècnica de Catalunya (UPC), de Barcelona, Espanha. O estudo do problema VNE é o primeiro objeto dessa cooperação.

A definição da metodologia de simulação e a integração do algoritmo PAViLiM com os programas do ALEVIN foram feitas em conjunto com os pesquisadores da UPC.

A metodologia de simulação criada para comparar algoritmos VNE existentes com o algoritmo PAViLiM define cenários que são parametrizados por:

- topologia de criação de redes (SN e VNs);
- definição de recursos e demandas;
- condições para a simulação.

5.2.1 Topologia de criação de redes (SN e VNs)

Para a criação das topologias de rede de substrato e rede virtual foi utilizado o algoritmo Waxman, que é um gerador de topologias aleatórias denominadas topologias “*random flat*”. Com o Waxman os nós são uniformemente distribuídos em uma área [41]. O gerador Waxman possui dois parâmetros, α e β , que determinam a probabilidade de existência de uma aresta entre dois nós por meio da Eq. (4):

$$P(u, v) = \alpha \cdot e^{-\frac{d(u,v)}{\beta \cdot L}} \quad (4)$$

onde

$$0 < \alpha, \beta \leq 1$$

$d(u, v)$: é a distância Euclidiana entre os nós u e v .

L : é a distância Euclidiana máxima entre quaisquer dois nós.

Um aumento no parâmetro α aumentará a probabilidade de existência de uma aresta entre quaisquer dois nós no dígrafo. Um aumento no parâmetro β aumentará a taxa de arestas mais distantes em relação às arestas mais próximas. Tanto a rede de substrato quanto a rede virtual são geradas com os mesmos parâmetros do gerador Waxman.

5.2.2 Definição de recursos e demandas

Uma vez criadas as topologias das redes física e virtual, é necessário gerar, para essas redes, os recursos e as demandas de banda passante (BW) dos enlaces e de capacidade de processamento (CPU) dos nós. A obtenção desses recursos e demandas é feita em duas fases:

Na primeira fase, a rede de substrato obtém os recursos de uma forma uniformemente distribuída. Os valores atribuídos às métricas de nós e arestas estão compreendidos em dois intervalos:

$(0, NR_X^{max})$ - intervalo para valores da métrica X atribuído a cada nó da SN¹⁶.

$(0, LR_Y^{max})$ - intervalo para valores da métrica Y atribuído a cada aresta da SN¹⁷.

Nas simulações, o valor para a métrica de nó, capacidade de processamento (CPU) existente, está no intervalo de (0,100). O valor para a métrica de aresta, largura de banda (BW), está no intervalo de (0,100).

¹⁶ NR_X^{max} é o valor de recurso máximo relacionado à métrica X de nó (NR – node resource).

¹⁷ LR_Y^{max} é o valor de recurso máximo relacionado à métrica Y de aresta (LR – link resource).

Na segunda fase são geradas as demandas para todas as redes virtuais. A intenção é encontrar uma carga média que possa ser distribuída uniformemente nessas redes virtuais e que atendam as restrições de recursos máximos que podem ser fornecidos.

A criação das demandas de nós e de enlaces é feita de forma diferente, pois para determinar a carga média de um recurso, para que o recurso possa ser distribuído uniformemente na VN, é necessário saber o número de elementos existentes para aquele recurso. No caso dos nós virtuais, seu número é fixo no experimento. Contudo, nos enlaces virtuais o número é probabilístico, visto que a criação de enlaces virtuais segue a Eq. (4) do gerador de Waxman.

5.2.2.1 Demanda média sobre um nó virtual para uma dada carga (ρ)

Como a SN possui um número de nós fixo, o recurso médio sobre um nó da rede de substrato é:

$$E[NR_x] = \frac{0 + NR_x^{max}}{2} = \frac{NR_x^{max}}{2} \quad (5)$$

onde

$E[NR_x]$ – recurso médio sobre um nó da SN.

NR_x^{max} – recurso máximo que pode ser atribuído a um nó da SN.

A demanda média sobre um nó virtual para uma dada carga ρ é:

$$E[ND_x] = \rho \cdot E[NR_x] \cdot \frac{|V|}{|V^k| \cdot k^{max}} \quad (6)$$

onde

$E[ND_x]$ – demanda média sobre um nó virtual.

$E[NR_x]$ – recurso médio sobre um nó da SN.

ρ – fator de carga, com valores entre: $0 \leq \rho \leq 1$.

$|V|$ – número de nós da SN.

$|V^k|$ – número de nós virtuais por rede virtual.

k^{max} – número de VNs na simulação.

Como a distribuição de demandas é uniforme, com valores no intervalo de $(0, ND_x^{max})$, obtém-se a demanda máxima de um recurso por meio da equação:

$$ND_x^{max} = 2 \cdot E[ND_x] \quad (7)$$

5.2.2.2 Demanda média sobre um enlace virtual para uma dada carga (ρ)

Na geração da topologia Waxman a criação de enlaces é probabilística: cada enlace virtual possui uma probabilidade para ser criado. O número médio de enlaces $E[|A|]$ é obtido por meio da probabilidade média $E[p]$ de criação de um enlace entre quaisquer dois nós. O número médio de enlaces $E[|A|]$ pode ser calculado pela equação:

$$E[|A|] = E[p] \cdot |V| \cdot (|V| - 1) \quad (8)$$

onde

$E[|A|]$ – número médio de enlaces.

$E[p]$ – probabilidade média de criação de um enlace.

$|V|$ – número de nós da SN ou da VN.

Na rede de substrato o recurso médio sobre um enlace é dado por:

$$E[LR_y] = \frac{0 + LR_y^{max}}{2} = \frac{LR_y^{max}}{2} \quad (9)$$

onde

$E[LR_y]$ – recurso médio sobre um enlace da SN.

LR_y^{max} – recurso máximo que pode ser atribuído a um enlace da SN.

A demanda média sobre um enlace virtual para uma dada carga ρ é:

$$E[LD_y] = \rho \cdot E[LR_y] \cdot \frac{E[|A|]}{E[|A^k|] \cdot k^{max}} \quad (10)$$

onde

$E[LD_y]$ – demanda média sobre um enlace virtual.

$E[LR_y]$ – recurso médio sobre um enlace da SN.

ρ – fator de carga, com valores entre: $0 \leq \rho \leq 1$.

$E[|A|]$ – número médio de enlaces da SN.

$E[|A^k|]$ – número médio de enlaces virtuais por rede virtual.

Como a distribuição de demandas é uniforme, com valores no intervalo de $(0, LD_y^{max})$, obtém-se a demanda máxima de um recurso por meio da equação:

$$LD_y^{max} = 2 \cdot E[LD_y] \quad (11)$$

Existem algumas restrições que devem ser obedecidas para garantir que as demandas geradas no processo possam ser atendidas:

$$LD_y^{max} \leq LR_y^{max}$$

$$E[|A|] < E[|A^k|] \cdot k^{max} \quad (12)$$

$$|V| < |V^k| \cdot k^{max}$$

Nota-se, assim, que não é qualquer combinação de quantidades de nós na SN e na VN, número de VNs e valores máximos atribuídos para recursos que podem ser utilizados nas simulações.

5.2.3 Condições para simulação

Para que os resultados das simulações realizadas pudessem ser comparados com outros resultados da literatura [20] [21] [56] foram utilizados, para a Eq. (4), os seguintes parâmetros de simulação:

$$\alpha = 0,5$$

$$\beta = 0,5$$

Área quadrada 1×1 para a distribuição uniforme dos nós.

$$L = \sqrt{2} \quad - \text{distância máxima entre nós.}$$

Estudos empíricos para esses parâmetros indicam que a distância média entre quaisquer dois nós possui o valor de [27] [41]:

$$E[d] \cong 0,5 \quad - \text{distância média entre nós.}$$

Utilizando os parâmetros de simulação obtém-se, pela Eq. (4), a probabilidade média de criação de um enlace entre quaisquer dois nós: $E[p] \cong 1/4$.

Os ciclos de processamento (CPU) são considerados recursos de nós e indicados por NR_{CPU} . A largura de banda (BW) é considerada um recurso de enlace e indicada por LR_{BW} . Para a distribuição uniforme desses valores foram escolhidos:

$$NR_{cpu}^{max} = 100 \quad e \quad LR_{bw}^{max} = 100$$

A Tabela 28 mostra os parâmetros utilizados nas simulações. Os valores para o número de nós na rede de substrato (SN), o número de nós virtuais por rede virtual (VN) e o número de requisições de rede virtual (VNR) foram escolhidos para permitir a comparação dos resultados com outras simulações já realizadas na literatura [27] [56]. Os valores das cargas de 0,2 a 0,7 cobrem uma série de situações de baixa a alta utilização de uma rede.

Tabela 28 – Parâmetros escolhidos para os cenários de simulação.

Parâmetros	Valores Escolhidos
Número de nós na SN (V)	20
Número de nós virtuais por VN (V^k)	10
Número de VNRs (k^{max})	10
Número de Cenários¹⁸ (N)	5
Cargas utilizadas (ρ)	{0,2; 0,3; 0,4; 0,5; 0,6; 0,7}

Como o gerador de topologias Waxman é probabilístico, para cada valor de carga são criados 5 (cinco) cenários diferentes para simulação.

5.3 Simulação: PAViLiM versus programas do ALEVIN

As simulações ocorreram segundo as condições estabelecidas neste capítulo.

O desempenho do algoritmo PAViLiM foi comparado com algoritmos existentes no ambiente ALEVIN que resolvem por completo o problema VNE e realizam o mapeamento de nós e enlaces virtuais utilizando caminhos *single-path*. A análise de caminhos *multi-path* será discutida na Seção 5.5.

O algoritmo PAViLiM foi combinado com dois algoritmos de mapeamento de nós virtuais:

- Mapeamento de nós “guloso” (*greedy available resources* – GAR).
- Mapeamento de nós coordenado (*coordinated node*).

¹⁸ cenários de simulação: em cada cenário se alteram a topologia de rede da SN e as VNRs solicitadas.

A Tabela 29 resume as características dos algoritmos utilizados nas simulações. Todos os algoritmos, ao realizarem o mapeamento dos enlaces virtuais, produzem soluções *single-path*.

Notar que estão sendo empregadas duas políticas diferentes para a álgebra de caminhos. A diferença entre elas são o conjunto **M** de métricas, as sínteses e as relações de ordenação.

Nas simulações com o algoritmo PAViLiM para cada SN avaliada é determinado o comprimento máximo que um caminho poderá ter por meio dos programas de filtragem por profundidade.

Os seguintes indicadores foram utilizados para comparar os resultados obtidos pelos diferentes algoritmos:

- Taxa de receita mapeada (*Mapped revenue ratio*): é a porcentagem da receita mapeada pela receita total que poderia ter sido mapeada;
- Taxa de aceitação de VNRs (*VNR acceptance ratio*): é a porcentagem das requisições de rede virtual mapeadas perante o total de requisições solicitadas;
- Relação custo/receita (*cost/revenue – C/R*): quanto menor o valor, melhor o resultado do mapeamento e o atendimento das requisições de redes virtuais.

Os resultados das simulações foram separados em dois grupos de acordo com a estratégia de mapeamento de nós utilizada. O primeiro grupo compara os algoritmos da álgebra de caminhos PathsAlgebraAR políticas M1 e M2 com o algoritmo GARSP. O segundo grupo compara os algoritmos da álgebra de caminhos *PA Coordinated* políticas M1 e M2 com o algoritmo DViNESP.

Tabela 29 – Algoritmos utilizados nas simulações.

Nome do Algoritmo	Descrição
GARSP	Algoritmo guloso que busca recursos disponíveis em k caminhos mais curtos ¹⁹ .
DViNESP	Algoritmo que realiza o mapeamento de nós coordenado e o mapeamento de enlaces de forma determinística [49].
PathsAlgebraAR – política M1	Mapeamento de nós “guloso” e mapeamento de enlaces utilizando o algoritmo PAViLiM com política M1: M=(hops,bw,cpu), S=(add,min,min) e relação de ordem ($\geq, \leq, \leq$).
PathsAlgebraAR – política M2	Mapeamento de nós “guloso” e mapeamento de enlaces utilizando o algoritmo PAViLiM com política M2: M=(hops,total bw), S=(add,add) e relação de ordem ($\geq, \leq$).
PA Coordinated – política M1	Mapeamento de nós coordenado e mapeamento de enlaces virtuais utilizando o algoritmo PAViLiM com política M1: M=(hops,bw,cpu), S=(add,min,min) e relação de ordem ($\geq, \leq, \leq$).
PA Coordinated – política M2	Mapeamento de nós coordenado e mapeamento de enlaces virtuais utilizando o algoritmo PAViLiM com política M2: M=(hops,total bw), S=(add,add) e relação de ordem ($\geq, \leq$).

¹⁹ **k** caminhos mais curtos é chamado na literatura *k-shortest paths* [61], indica que o algoritmo procura por **k** caminhos mais curtos entre um nó origem e um nó destino e o valor de **k** é uma variável lida pelo algoritmo. Nos algoritmos *single-path* o valor de **k** é 1.

As Tabelas 30, 31 e 32 apresentam para o primeiro grupo de algoritmos os resultados dos indicadores analisados. Foram calculadas para cada carga as médias dos valores obtidos nos cinco cenários.

Tabela 30 – Resultados da taxa de receita mapeada – grupo 1.

<i>Mapped Revenue Ratio</i>			
Cargas	PathsAlgebraAR política M1	PathsAlgebraAR política M2	GARSP
0,20	100,00	100,00	94,96
0,30	80,64	84,28	75,37
0,40	59,11	58,06	53,58
0,50	47,43	44,27	41,16
0,60	37,25	41,16	34,54
0,70	30,02	30,95	26,61

Tabela 31 – Resultados da taxa de aceitação de VNRs – grupo 1.

<i>VNR Acceptance Ratio</i>			
Cargas	PathsAlgebraAR política M1	PathsAlgebraAR política M2	GARSP
0,20	100,00	100,00	94,00
0,30	78,00	82,00	72,00
0,40	56,00	54,00	50,00
0,50	44,00	40,00	38,00
0,60	34,00	38,00	32,00
0,70	28,00	28,00	24,00

Tabela 32 – Resultados da relação custo/receita – grupo 1.

<i>Cost/Revenue - C/R</i>			
Cargas	PathsAlgebraAR política M1	PathsAlgebraAR política M2	GARSP
0,20	1,46	1,48	1,51
0,30	1,54	1,56	1,59
0,40	1,52	1,56	1,57
0,50	1,63	1,60	1,66
0,60	1,61	1,60	1,64
0,70	1,63	1,66	1,65

As Tabelas 30 e 31 mostram que os algoritmos da álgebra de caminhos sempre apresentam os melhores resultados para todas as cargas avaliadas. As políticas M1 e M2 apresentam resultados e desempenhos diferentes para cada carga.

A Tabela 32 mostra que o algoritmo da álgebra de caminhos com política M1 sempre é melhor que o algoritmo GARSP. Já o algoritmo PathsAlgebraAR com política M2 é melhor que o GARSP para todas as cargas, com exceção da carga 0,7.

As Figuras 29, 30 e 31 apresentam os gráficos dos resultados para o primeiro grupo de algoritmos analisados. Foram utilizados os dados das Tabelas 30, 31 e 32.

Figura 29 – Resultados da taxa média de receita mapeada – grupo 1.

Figura 30 – Resultados da taxa média de aceitação de VNRs – grupo 1.

Figura 31 – Resultados da relação custo/receita média – grupo 1.

As Tabelas 33, 34 e 35 apresentam, para o segundo grupo de algoritmos, os resultados dos indicadores analisados. Foram calculadas, para cada carga, as médias dos valores obtidos nos cinco cenários.

Tabela 33 – Resultados da taxa de receita mapeada – grupo 2.

<i>Mapped Revenue Ratio</i>			
Cargas	PA Coordinated política M1	PA Coordinated política M2	DViNESP
0,20	77,12	65,91	69,63
0,30	70,67	66,61	62,31
0,40	51,29	49,95	49,90
0,50	44,50	44,44	40,96
0,60	41,49	42,28	37,92
0,70	33,67	30,45	32,23

Tabela 34 – Resultados da taxa de aceitação de VNRs – grupo 2.

<i>VNR Acceptance Ratio</i>			
Cargas	PA Coordinated política M1	PA Coordinated política M2	DViNESP
0,20	76,00	64,00	68,00
0,30	70,00	66,00	60,00
0,40	50,00	48,00	48,00
0,50	44,00	44,00	40,00
0,60	38,00	40,00	36,00
0,70	32,00	28,00	30,00

Tabela 35 – Resultados da relação custo/receita – grupo 2.

<i>Cost/Revenue - C/R</i>			
Cargas	PA Coordinated política M1	PA Coordinated política M2	DViNESP
0,20	1,42	1,47	1,45
0,30	1,46	1,47	1,51
0,40	1,48	1,50	1,53
0,50	1,50	1,48	1,54
0,60	1,55	1,53	1,57
0,70	1,56	1,65	1,62

As Tabelas 33, 34 e 35 mostram que sempre existe um algoritmo da álgebra de caminhos que apresenta o melhor resultado para todas as cargas avaliadas. As políticas M1 e M2 apresentam resultados e desempenhos diferentes para cada carga.

O algoritmo da álgebra de caminhos *PA Coordinated* com política M1 sempre é melhor que o algoritmo *DViNESP*. Já o algoritmo *PA Coordinated* com política M2 é melhor que o *DViNESP* para as cargas de 0,2 à 0,6.

As Figuras 32, 33 e 34 apresentam os gráficos dos resultados para o segundo grupo de algoritmos analisados. Foram utilizados os dados das Tabelas 33, 34 e 35.

Figura 32 – Resultados da taxa média de receita mapeada – grupo 2.

Figura 33 – Resultados da taxa média de aceitação de VNRs – grupo 2.

Figura 34 – Resultados da relação custo/receita média – grupo 2.

5.3.1 Discussão dos Resultados

As Figuras 29, 30 e 31 mostram os resultados para o grupo 1 de algoritmos. Os algoritmos da álgebra de caminhos, para o conjunto de testes realizados, sempre apresentaram os melhores resultados. Há casos em que a política M1 da álgebra de caminhos é aquela que produz os melhores resultados; em outros casos é a política M2.

Comportamento semelhante é percebido com o grupo 2 de algoritmos. As Figuras 32, 33 e 34 mostram os resultados. A álgebra de caminhos, para o conjunto de testes realizados, sempre apresentaram os melhores resultados. Novamente, ora a política M1 da álgebra de caminhos é a que apresenta os melhores resultados, ora a política M2.

A interpretação correta desses resultados é que a álgebra de caminhos permite a identificação da política adequada para otimizar o desempenho de um dado parâmetro. Além disso, não é necessário escolher a política de forma antecipada: várias políticas podem ser utilizadas simultaneamente sem nenhum aumento significativo na complexidade computacional e a política que apresentar o melhor desempenho será a escolhida para aquele atendimento. Lembrar que a enumeração de todos os caminhos elegíveis na SN é computacionalmente explosiva, porém, devido à imposição da filtragem por profundidade, o número de operações dos algoritmos é reduzido e a enumeração dos caminhos elegíveis poderá ser realizada inclusive de forma *offline*, pois é independente da estratégia de otimização empregada.

As Figuras 30 e 33 mostram os resultados da *taxa de aceitação de VNRs* para os algoritmos do grupo 1 e do grupo 2, respectivamente. Nas duas figuras é possível observar uma tendência de redução no número de VNRs aceitas quando a carga aumenta.

O comportamento observado pode ser entendido ao considerar os recursos disponíveis na SN para tratar do procedimento VNE. Os recursos totais que serão

utilizados no processo de mapeamento de nós e enlaces virtuais é uma fração dos recursos de largura de banda e processamento da SN.

Quando a carga ρ utilizada nas simulações é baixa, existem recursos suficientes na SN para absorver toda a demanda de VNRs que serão mapeadas com sucesso. Pode-se aumentar a carga sobre a SN até um valor de ρ_0 , que ainda assim irá permitir o mapeamento das VNRs, porém com a exaustão de todos os recursos existentes na rede de substrato. Acima de ρ_0 , a carga em excesso não poderá ser mapeada e VNRs não serão atendidas.

Para uma carga $\rho > \rho_0$ pode-se esperar que a taxa de aceitação de VNRs siga uma tendência: ρ_0/ρ .

Como a receita no processo de virtualização é proporcional ao número de requisições de máquinas virtuais atendidas, a *taxa de receita mapeada* terá o mesmo comportamento que a *taxa de aceitação de VNRs*. Os resultados da taxa de receita mapeada são mostrados nas Figuras 29 e 32.

Para entender os resultados da relação C/R é necessário considerar, por meio de um exemplo, como o custo e a receita são avaliados. Seja uma VNR representada por:

- Nós virtuais: A e B .
- Enlace virtual: $(A - B)$; A (nó origem), B (nó destino).
- Largura de banda solicitada: $BW(A - B) = 100$.
- Capacidade de processamento solicitado para cada nó virtual:
 $CPU(A) = 40$; $CPU(B) = 60$.

A receita associada a essa requisição de rede virtual é:

$$R = CPU(A) + CPU(B) + BW(A - B) = 40 + 60 + 100 = 200.$$

Supor que o algoritmo da álgebra de caminhos PAViLiM encontre dois caminhos para o mapeamento do enlace virtual apresentado na Tabela 36.

Tabela 36 – Comparação de dois mapeamentos possíveis.

Mapeamento	1	2
Mapeamento de Nós	$A \rightarrow 1; B \rightarrow 2$	$A \rightarrow 4; B \rightarrow 6$
Caminho	1; 2	4; 7; 6
Custo (C)	$CPU(1) + CPU(2) + BW(1 - 2)$ $= 40 + 60 + 100 = 200$	$CPU(4) + CPU(7) + CPU(6) +$ $+ BW(4 - 7) + BW(7 - 6)$ $= 40 + 0^* + 60 + 100 + 100$ $= 300$
Receita (R)	$CPU(A) + CPU(B) + BW(A - B)$ $= 40 + 60 + 100 = 200$	$CPU(A) + CPU(B) + BW(A - B)$ $= 40 + 60 + 100 = 200$
relação C/R	$200/200 = 1,0$	$300/200 = 1,5$
* Notar que o valor da CPU(7) é zero, pois o custo dos nós intermediários (<i>hidden nodes</i>) não estão sendo considerados.		

Pela Tabela 36 percebe-se que a melhor solução é encontrada quando um enlace virtual é mapeado diretamente sobre um enlace na rede de substrato. Neste caso, a relação custo/receita é 1,0 (um). Cada vez que um nó intermediário (*hidden node*) precisa ser utilizado em um caminho, o custo aumenta pela soma do custo de CPU mais o custo da largura de banda. Como nesse exemplo o custo de nós intermediários não está sendo considerado, o valor do custo $CPU(7)$ foi considerado igual a zero (0).

Se o mapeamento de um enlace virtual entre dois nós A e B utilizar k nós intermediários, o custo será dado por:

$$C = CPU(A) + CPU(B) + (1 + k) \times BW. \quad (13)$$

A receita para esse enlace virtual é um número fixo dado por:

$$R = CPU(A) + CPU(B) + BW. \quad (14)$$

A relação Custo/Receita (C/R) é obtida pelas equações (13) e (14):

$$C/R = 1 + k \times \frac{BW}{R}. \quad (15)$$

Pela equação (15), nota-se que a relação C/R aumenta linearmente quando o número de nós intermediários aumenta.

A minimização do custo é uma das prioridades utilizadas pelos algoritmos para a resolução do problema VNE. Isto significa que todas as estratégias de mapeamento de uma VNR utilizam o caminho mais curto disponível e, com isso, no processo de mapeamento de enlaces virtuais todos os caminhos mais curtos serão consumidos primeiro. Quando a carga aumenta, para realizar o atendimento de novos VNRs, caminhos mais longos serão utilizados e o custo do mapeamento aumentará. Após essas observações é esperado que a relação C/R apresente um aumento ou um comportamento constante conforme a carga aumenta.

As Figuras 31 e 34 mostram um aumento da relação C/R e, para algumas cargas, um comportamento constante quando a carga aumenta para os algoritmos analisados do grupo 1 e do grupo 2.

5.3.2 Total de Simulações

O número total de vezes que cada algoritmo foi executado é dado por:

$$\text{Total de Simulações: } N \times k^{max} \times |\rho| = 5 \times 10 \times 6 = 300$$

Como foram comparados os resultados de seis políticas diferentes, o número total de atendimentos de VNRs foi de 1800.

Ao utilizar a média dos resultados percebe-se que a álgebra de caminhos sempre fornece o melhor desempenho médio para todos os indicadores e cargas.

Como a quantidade de amostras é pequena, é apresentada na próxima Seção uma análise individual do desempenho da álgebra de caminhos no atendimento de cada cenário.

5.3.3 Análise Individual de Resultados

Cada algoritmo de atendimento de VNRs foi utilizado em trinta cenários²⁰. Os resultados de cada cenário foram separados em dois grupos, de acordo com a estratégia de mapeamento de nós utilizada. O Anexo D apresenta os resultados individuais para cada cenário.

A Tabela 37 resume para o grupo 1 os melhores resultados para diferentes indicadores e políticas. Para o indicador “taxa de aceitação de VNRs”, os algoritmos da álgebra de caminhos (política M1 ou M2) apresentam os melhores resultados em 29 dos 30 cenários analisados. Com o indicador “taxa de receita mapeada” os algoritmos da álgebra de caminhos apresentam 28 vezes os melhores resultados. Para o indicador “relação custo/receita” os algoritmos da álgebra de caminhos apresentam 27 vezes os melhores resultados.

²⁰ trinta cenários: como são cinco cenários por carga e seis cargas utilizadas obtém-se trinta cenários.

Tabela 37 – Sumário dos melhores resultados obtidos para diferentes indicadores e políticas – grupo 1.

Indicador	PathsAlgebraAR – política M1	GARSP
	PathsAlgebraAR – Política M2	
<i>VNR Acceptance Ratio</i>	29	1
<i>Mapped Revenue Ratio</i>	28	2
<i>Cost / Revenue – C/R</i>	27	3

A Tabela 38 resume para o grupo 2 os melhores resultados para diferentes indicadores e políticas. Para o indicador “taxa de aceitação de VNRs”, os algoritmos da álgebra de caminhos (política M1 ou M2) apresentam os melhores resultados em 28 dos 30 cenários analisados. Com o indicador “taxa de receita mapeada” os algoritmos da álgebra de caminhos apresentam 27 vezes os melhores resultados. Para o indicador “relação custo/receita” os algoritmos da álgebra de caminhos apresentam 27 vezes os melhores resultados.

Tabela 38 – Sumário dos melhores resultados obtidos para diferentes indicadores e políticas – grupo 2.

Indicador	PA Coordinated – Política M1	DVINESP
	PA Coordinated – Política M2	
<i>VNR Acceptance Ratio</i>	28	2
<i>Mapped Revenue Ratio</i>	27	3
<i>Cost / Revenue – C/R</i>	27	3

5.3.4 Considerações Finais das Simulações

A quantidade total de simulações realizadas não permite tirar conclusões quantitativas com significância estatística²¹, mas permite fornecer comparações qualitativas e obter um entendimento do comportamento observado.

O desempenho da álgebra de caminhos, para o conjunto de testes realizados, quase sempre excederam o desempenho observado por outros algoritmos de virtualização de redes.

Nas simulações realizadas não foi possível identificar a razão específica que faz com que as políticas da álgebra de caminhos não sejam sempre as melhores. Possíveis explicações para esse fato são:

- Políticas diferentes podem empregar diferentes mapeamentos de nós virtuais que afetam o mapeamento de enlaces virtuais. Se este for o caso, deve existir uma melhor coordenação entre mapeamento de nós e enlaces. Um mapeamento de nós virtuais que utiliza a álgebra de caminhos deve ser desenvolvido.
- Estratégias de mapeamento de enlaces que não utilizam a álgebra de caminhos podem encontrar melhores resultados utilizando caminhos mais longos. Se este for o caso, deve ser feita uma relaxação do limite imposto pelo filtro por profundidade e tentar realizar um mapeamento com caminhos mais longos quando uma VNR não puder ser atendida.
- A ordem em que as VNRs são processadas pode impactar no resultado final dos mapeamentos. Nesse caso, um procedimento de *backtrack* deve ser desenvolvido e utilizado.

²¹ Novas simulações estão sendo realizadas (abril 2012) com 20 cenários e 8 cargas, o que fornece um total de 1600 simulações. Como são seis algoritmos utilizados tem-se um total de 9600 VNRs que deverão ser atendidas e analisadas. Os resultados preliminares, com 20 cenários, indicam não haver diferença substancial dos resultados obtidos com 5 cenários. A álgebra de caminhos continua apresentando os melhores resultados para todos os indicadores utilizados.

A álgebra de caminhos é uma estratégia flexível e poderosa para a solução do problema VNE. Tal flexibilidade permite que múltiplas políticas de atendimento sejam simultaneamente utilizadas para escolher a melhor política dado um critério de desempenho.

5.4 Atendimento Parcial de VNRs

Atualmente, as políticas de atendimento de VNRs impõem que o atendimento seja completo [2] [62].

Não existe na literatura suporte para atendimentos de VNRs de forma parcial, com enlaces virtuais em uma VNR podendo ou não ser atendidos.

Um novo modelo ou estratégia de negócios pode ser criado para suportar atendimentos parciais. Atendimento parcial é uma proposta para um novo tipo de atendimento.

Com a política de atendimento parcial, clientes de virtualização poderão ter, em determinados horários ou períodos, um atendimento de VNRs completo e garantido. Em outros horários, o SLA com um provedor permitirá o atendimento de VNRs parciais.

O algoritmo da álgebra de caminhos PAViLiM suporta o atendimento de VNRs de forma completa e parcial.

O atendimento parcial resolve o problema P5 de virtualização de redes listado na Tabela 1 do primeiro Capítulo deste trabalho.

As Tabelas 39 e 40 apresentam os resultados do PAViLiM em modo de atendimento parcial, do GARSP e do DViNESP. Lembrando que estes dois últimos algoritmos não suportam atendimento parcial e tiveram seus resultados transportados das Tabelas 30, 31, 33 e 34.

Tabela 39 – Taxa de receita média mapeada – PAViLiM em modo de atendimento parcial.

<i>Mapped Revenue Ratio</i>				
Cargas	PathsAlgebraAR política M1	PathsAlgebraAR política M2	GARSP	DViNESP
0,2	100,00	100,00	94,96	69,63
0,3	97,45	97,17	75,37	62,31
0,4	89,13	87,96	53,58	49,90
0,5	75,57	74,91	41,16	40,96
0,6	63,82	63,25	34,54	37,92
0,7	54,35	53,53	26,61	32,23

Tabela 40 – Taxa de aceitação média de VNRs – PAViLiM em modo de atendimento parcial.

<i>VNR Acceptance Ratio</i>				
Cargas	PathsAlgebraAR política M1	PathsAlgebraAR política M2	GARSP	DViNESP
0,2	100,00	100,00	94,00	68,00
0,3	97,74	97,67	72,00	60,00
0,4	90,39	89,28	50,00	48,00
0,5	77,53	76,90	38,00	40,00
0,6	66,87	66,14	32,00	36,00
0,7	57,83	56,92	24,00	30,00

As Figuras 35 e 36 apresentam os gráficos dos resultados do algoritmo PAViLiM com atendimento parcial. Foram utilizados os dados das Tabelas 39 e 40.

Figura 35 – Taxa média de receita mapeada – PAViLiM atendimento parcial.

Figura 36 – Taxa média de aceitação de VNRs – PAViLiM atendimento parcial.

As Figuras 35 e 36 mostram que a diferença de resultados entre as políticas M1 e M2 da álgebra de caminhos é mínima quando o atendimento é parcial.

A diferença de resultados entre o PAViLiM e os outros algoritmos deve-se ao fato de que os algoritmos de virtualização tradicionais não suportam atendimento parcial e uma VNR é completamente atendida ou não atendida.

5.4.1 Considerações Finais do Atendimento Parcial

O atendimento parcial de VNRs é uma proposta.

Foram feitas simulações para demonstrar que esse atendimento é possível e já suportado no PAViLiM e com desempenho superior aos algoritmos tradicionais que realizam atendimento completo.

O atendimento parcial de VNRs abre uma série de novas possibilidades e novos problemas surgem, os quais devem ser estudados, discutidos e resolvidos. As seguintes questões devem ser discutidas para o suporte ao atendimento parcial:

- Cada VNR deve ter uma informação indicando se suporta ou não o atendimento parcial.
- O atendimento parcial pode ter que garantir uma porcentagem mínima de recursos totais solicitados na VNR.
- Quando uma VNR é atendida de forma parcial pode ocorrer periodicamente um procedimento de *backtrack* para tentar atender de forma completa essa VNR.
- Na falta de recurso na rede de substrato uma VNR que foi atendida de forma completa e que suporta atendimento parcial pode ser desfeita para ser atendida parcialmente e a utilização de recursos na SN pode ser reduzida para dar suporte ao atendimento de uma nova VNR.

5.5 Atendimento *multi-path* de VNRs

As simulações deste capítulo mostraram resultados referentes a três indicadores:

- Taxa de receita mapeada (*Mapped Revenue Ratio*);
- Taxa de aceitação de VNRs (*VNR Acceptance Ratio*);
- Relação Custo/Receita (*Cost/Revenue relationship*).

O algoritmo PAViLiM obtém um conjunto de resultados adicionais extremamente úteis e importantes, como será discutido. O resultado primário do algoritmo é o mapeamento de todos os enlaces virtuais de cada VNR para caminhos na rede de substrato. Os resultados secundários são a enumeração de todos os caminhos entre um nó origem e um nó destino na SN e a obtenção de todas as soluções possíveis para cada mapeamento de um enlace virtual. O melhor mapeamento de um enlace virtual é determinado segundo as políticas de atendimento; porém, podem existir outras soluções de mapeamento possíveis determinadas pela álgebra de caminhos.

A existência de múltiplas soluções já obtidas pela álgebra de caminhos para o problema de mapeamento de enlaces virtuais é importante e útil para implementar técnicas de rápida recuperação após uma falha. Além disso, se houver monitoramento de QoS e for percebida a degradação de desempenho em um caminho mapeado, o mesmo poderá ser desfeito e um caminho alternativo poderá ser utilizado.

A funcionalidade de ter determinado todas as soluções possíveis e caminhos elegíveis para cada mapeamento de um enlace virtual permite que se utilizem técnicas para implementar um novo tipo de atendimento chamado *multi-path*. O atendimento *multi-path* discutido na literatura [2] [4] possui três objetivos:

- Atender maior quantidade de VNRs que não conseguem ser atendidas com um atendimento *single-path*.
- Realizar o balanceamento de carga entre vários caminhos elegíveis alternativos.
- Fornecer tolerância a falhas para suportar a queda de um caminho até que outro caminho seja obtido e mapeado.

O atendimento *multi-path* pode utilizar diferentes estratégias [27]:

- Procurar sempre por uma solução de atendimento *multi-path* que faça um balanceamento de carga entre múltiplos caminhos da SN.
- Realizar o atendimento *single-path* para todas as VNRs, se uma solução *single-path* não puder ser encontrada para uma dada VNR, imediatamente procurar realizar um atendimento *multi-path* para aquela VNR.
- Realizar o atendimento *single-path* para todas as VNRs de uma lista de atendimentos. Ao final do processo, aquelas VNRs que não puderam ser atendidas, realizar uma nova tentativa de mapeamentos com atendimento *multi-path*.

A informação da estratégia que será utilizada para atendimento *multi-path* pode estar disponível em cada VNR que será atendida, ideia semelhante à proposta para atendimento parcial, ou a estratégia de atendimento pode ser pré-configurada no provedor de serviços de virtualização.

O algoritmo PAViLiM foi modificado para implementar, ao final do processo de atendimento *single-path*, um atendimento *multi-path* para aquelas VNRs que não puderam ser atendidas utilizando o atendimento *single-path*.

Nas simulações foram utilizados os algoritmos GARSP que realizam um atendimento *single-path* e GARPS com suporte a atendimento *multi-path* utilizando a mesma estratégia de atendimento *multi-path* do PAViLiM.

Os algoritmos utilizados possuem as seguintes características:

- PathsAlgebra-Multi: mapeamento de nós GAR e mapeamento de enlaces utilizando o algoritmo PAViLiM com atendimento *multi-path* e política M1.
- GARSP: algoritmo guloso que busca recursos disponíveis em *k caminhos mais curtos*.
- GARPS: utiliza a mesma tecnologia de mapeamento de nós do GARSP²² com suporte a atendimento *multi-path* para mapeamento de enlaces virtuais.

A Tabela 41 apresenta, para os algoritmos analisados, os resultados do indicador taxa de aceitação de VNRs (*VNR Acceptance Ratio*). Foram calculadas para cada carga as médias dos valores obtidos nos cinco cenários.

A Figura 37 apresenta o gráfico dos resultados do algoritmo PAViLiM com atendimento *multi-path*. A Tabela 41 foi utilizada como referência para a criação do gráfico.

Tabela 41 – Taxa de aceitação de VNRs para atendimento *multi-path*.

<i>VNR Acceptance Ratio</i>			
Cargas	PathsAlgebra-Multi	GARSP	GARPS
0,2	100	94	100
0,3	84	72	78
0,4	62	50	54
0,5	52	38	50
0,6	47	32	46
0,7	38	24	34

²² GARSP x GARPS – o SP na sigla do GARSP significa atendimento *single-path* e o PS na sigla do GARPS significa atendimento *path-split* ou *multi-path*.

Figura 37 – Resultados do atendimento *multi-path*.

A Tabela 41 mostra que o algoritmo PAViLiM em modo de atendimento *multi-path* é a melhor solução entre os algoritmos analisados.

Ao fazer uma análise dos resultados individuais obtidos em cada um dos trinta cenários para o indicador “taxa de aceitação de VNRs” percebe-se que o algoritmo da álgebra de caminhos em modo *multi-path* apresenta os melhores resultados em 28 dos 30 cenários analisados.

5.5.1 Considerações Finais do Atendimento *multi-path*

Novamente a álgebra de caminhos fornece os melhores resultados para mapeamento de enlaces virtuais utilizando agora uma estratégia de atendimento *multi-path*.

Nas simulações realizadas nem todas as estratégias de *multi-path* foram implementadas pelo PAViLiM nesse momento e um conjunto maior de resultados precisa ser obtido e analisado.

Um estudo completo das estratégias e do atendimento *multi-path* à luz da álgebra de caminhos está fora do escopo deste trabalho. Várias questões não foram respondidas e também são indicadas para trabalhos futuros:

- Identificação nas VNRs do suporte para atendimento *multi-path*. As VNRs podem indicar se aceitam ou proíbem o atendimento *multi-path*.
- Análise de desempenho dos algoritmos *multi-path*. O atendimento *multi-path* é diferente do atendimento *single-path*, outras métricas e indicadores devem ser estudados para medir e comparar algoritmos.
- A fase de mapeamento de nós virtuais deve levar em conta que o atendimento possível é *multi-path*. Muitos mapeamentos de nós poderão ser criados de forma coordenada para que os enlaces virtuais possam ser mapeados em múltiplos caminhos.

5.6 Considerações Finais do Capítulo

O algoritmo PAViLiM com o uso da álgebra de caminhos resolve os seguintes problemas de virtualização de redes: P2, P3, P4, P5 e P6, listados na Tabela 1 e descritos a seguir:

Problema P2

“Como escolher os melhores caminhos na rede de substrato (SN) para atender as requisições virtuais (VNR – VN requests) baseado no tipo de atendimento (*online* ou *offline*), política de atendimento e critério de ordenação?”

A álgebra de caminhos permite o mapeamento de enlaces virtuais para caminhos na rede de substrato com a escolha dos melhores caminhos baseado em políticas de atendimento e critérios de ordenação definidos pelo usuário.

Problema P3

“Como executar o atendimento em tempo real (*online*) de requisições virtuais (VN request) com garantia de QoS?”

A álgebra de caminhos ao utilizar a política de atendimento FIFO está realizando um atendimento em tempo real (*online*) com garantia de QoS.

Problema P4

“Como fazer o atendimento de requisições virtuais com garantia de QoS de forma programada ou agendada (*offline*)?”

A álgebra de caminhos ao utilizar uma política de atendimento diferente de FIFO (LCF, LCF-UP, MCF, MCF-UP) associada a métricas adequadas (perda de pacotes, confiabilidade, atraso fim a fim, variação do atraso) permite o atendimento de forma programada ou agendada (*offline*) com garantia de QoS.

Problema P5

“Como realizar o atendimento de requisições virtuais sem garantia de QoS com uso mínimo de recursos? “

O algoritmo PAViLiM com o uso da álgebra de caminhos, em modo de atendimento parcial de VNRs, resolve esse problema.

Problema P6

“Como realizar o provisionamento offline da rede virtual para a rede de substrato com o objetivo de minimizar os recursos utilizados e maximizar as requisições virtuais atendidas?”

A álgebra de caminhos através da política de atendimento, priorização de métricas e critério de ordenação adequados pode realizar um provisionamento *offline* da rede virtual com os objetivos apresentados pelo problema P6.

6. Conclusão e Trabalhos Futuros

Este capítulo apresenta a conclusão do presente trabalho e fornece sugestões de trabalhos futuros para dar continuidade a esta pesquisa.

6.1 Conclusão

A virtualização de redes fornece um mecanismo poderoso para executar múltiplas instâncias de rede virtual em uma única rede física que se torna compartilhada. A rede física é chamada de rede de substrato (*substrate network*). A implementação e o desenvolvimento de novos protocolos, testes de novas soluções e arquiteturas para a Internet atual e do futuro podem ser tratadas por meio de virtualização de redes.

Para a implementação de virtualização de redes surge um desafio denominado problema VNE. O problema de virtualização de redes embutidas (VNE) consiste em realizar o mapeamento dos nós virtuais e o mapeamento dos enlaces virtuais sobre uma rede de substrato (SN).

A rede de substrato (SN) é representada por um dígrafo (grafo direcionado) e seus recursos são mapeados como métricas sobre esse dígrafo. As solicitações dos usuários são mapeadas como requisições de redes virtuais (VNRs) que devem ser atendidas segundo critérios de otimização que permitam economizar recursos na SN e obter as maiores receitas no negócio “virtualização”.

Neste trabalho foi apresentada uma nova maneira de realizar o mapeamento de enlaces virtuais para caminhos na rede de substrato com a utilização da álgebra de caminhos.

A álgebra de caminhos foi originalmente desenvolvida para resolver o problema de roteamento multirrestritivo. É uma técnica poderosa que pode trabalhar simultaneamente com métricas lineares e não-lineares, permitindo critérios de otimização orientados a diversas necessidades técnicas ou comerciais. Ao ser utilizada na solução do problema de mapeamento de enlaces virtuais mostra-se extremamente eficiente e eficaz quando comparada a outras soluções.

Foram estudadas diversas metodologias e algoritmos da literatura para a resolução do problema VNE. Todas as metodologias apresentam limitações que se originam de simplificações do problema VNE²³. A falta de flexibilidade para poder se trabalhar com quaisquer tipos de métricas nas redes virtuais (VNs) e a dificuldade ou impossibilidade de lidar com diversas funções de combinação de métricas são outros problemas. A álgebra de caminhos é um *framework* matemático que resolve essas questões.

A solução do problema VNE por meio da álgebra de caminhos utiliza uma metodologia de quatro estágios:

- transformação de métricas de nós para métricas de arestas;
- enumeração de caminhos com filtro por profundidade;
- ordenação das requisições de rede virtual (VNRs);
- atribuição dos enlaces virtuais para caminhos na SN.

A transformação das métricas de nós para métricas de arestas é feita por meio de um artifício,²⁴ pois a álgebra de caminhos trabalha apenas com métricas de arestas.

Os programas de enumeração de caminhos utilizados pela álgebra de caminhos possuem uma complexidade algorítmica com tempo de execução explosivo limitado pela função n^n , onde n é a quantidade de vértices ou nós existentes no dígrafo. Fica claro neste trabalho a necessidade de diminuir a complexidade algorítmica para que os programas de enumeração de caminhos possam ser executados em grandes redes. Foi apresentada uma proposta de filtragem por profundidade que reduz a complexidade algorítmica e permite que redes com centenas de nós possam ser utilizadas.

Com a filtragem por profundidade empregada nos algoritmos da álgebra de caminhos, impõe-se um limite de comprimento máximo em qualquer caminho considerado. Esse procedimento fornece uma importante redução do número de caminhos que são descobertos e enumerados. Essa modificação foi introduzida no algoritmo PAViLiM.

²³ o problema VNE é *NP-Hard*.

²⁴ o artifício pode ser visto na Seção 3.4.2.

A ordenação das requisições de rede virtual e a atribuição dos enlaces virtuais para caminhos na SN dependem da política de atendimento e do critério de ordenação dos caminhos. Foi apresentado um conjunto de cinco políticas de atendimento, sendo que duas dessas políticas são propostas neste trabalho: MCF-UP e LCF-UP²⁵.

A política de atendimento e o critério de ordenação escolhido afetam os resultados obtidos pelos algoritmos de virtualização de redes. Não foi o escopo deste trabalho determinar qual é a melhor política ou indicar regras de otimização.

Foi desenvolvido neste trabalho o algoritmo PAViLiM, que simula a metodologia de quatro estágios da álgebra de caminhos para a solução do problema VNE.

Com o algoritmo PAViLiM foram resolvidos os problemas de P2 a P6 listados na Tabela 1, do Capítulo 1, e reproduzida neste capítulo para facilitar a leitura.

Tabela 1 (reproduzida do Capítulo 1) - Problemas com a Virtualização de Redes.

Problemas	Perguntas
P1	Como realizar o mapeamento dos nós da rede virtual (VN) para os nós da rede de substrato (SN)?
P2	Como escolher os melhores caminhos na rede de substrato (SN) para atender as requisições virtuais (VNR – VN requests) baseado no tipo de atendimento (<i>online</i> ou <i>offline</i>), política de atendimento e critério de ordenação?
P3	Como executar o atendimento em tempo real (<i>online</i>) de requisições virtuais (VN request) com garantia de QoS?
P4	Como fazer o atendimento de requisições virtuais com garantia de QoS de forma programada ou agendada (<i>offline</i>)?
P5	Como realizar o atendimento de requisições virtuais sem garantia de QoS com uso mínimo de recursos?
P6	Como realizar o provisionamento <i>offline</i> da rede virtual para a rede de substrato com o objetivo de minimizar os recursos utilizados e maximizar as requisições virtuais atendidas?

²⁵ MCF-UP – Most Consuming First – USP Proposed.
LCF-UP – Most Consuming First – USP Proposed.

A álgebra de caminhos permite o mapeamento de enlaces virtuais para caminhos na rede de substrato com a escolha dos melhores caminhos baseado em políticas de atendimento e critérios de ordenação definidos pelo usuário. O atendimento pode ser *online* ou *offline*, as requisições de redes virtuais (VNRs) atendidas de forma completa ou parcial e a qualidade de serviço (QoS), se desejado, satisfeita.

Os problemas resolvidos pelo PAViLiM e descritos na Tabela 1 podem ser sumarizados:

- Atendimento de requisições de rede virtual (VNRs) *online* ou *offline*. A álgebra de caminhos realiza o mapeamento de enlaces virtuais para caminhos na SN utilizando políticas de atendimento e critérios de ordenação definidos pelo usuário.
- Atendimento de requisições de rede virtual (VNRs) *online* ou *offline* com garantia de QoS. A álgebra de caminhos garante a QoS por meio do fornecimento de caminhos na SN que garantam o SLA do cliente.
- Atendimento parcial de VNRs sem qualquer garantia de QoS ao cliente no qual a requisição de rede virtual é atendida quando existem recursos na SN. Neste trabalho foi feita uma proposta de atendimento parcial de VNRs.
- Flexibilização de políticas. A álgebra de caminhos é extremamente flexível com relação às métricas, funções de combinação de métricas, sínteses e ordenação léxica que podem ser utilizadas.

No capítulo 5 são descritas inúmeras simulações que comparam o algoritmo PAViLiM da álgebra de caminhos com algoritmos consagrados disponíveis no ambiente ALEVIN.

Vários indicadores podem ser utilizados para realizar a comparação dos algoritmos VNE. Neste trabalho foram escolhidos três indicadores:

- Taxa de receita mapeada – que indica a porcentagem de recursos das VNRs que foram atendidas.
- Taxa de aceitação de VNRs – que indica a porcentagem de atendimentos de VNRs realizadas.
- Relação Custo / Receita – quanto menor o valor dessa relação melhor é o resultado do mapeamento que foi realizado.

Resultados das Simulações

As estratégias que utilizam a álgebra de caminhos, para o conjunto de testes realizados, sempre produzem os melhores resultados quando comparadas com outras estratégias que utilizam a mesma abordagem de mapeamento de nós virtuais.

Dependendo do critério de desempenho escolhido, a melhor estratégia pode não ser a mesma para todas as cargas verificadas. Contudo, a melhor estratégia sempre é uma das estratégias da álgebra de caminhos.

O resultado dessas afirmações pode ser resumido nos seguintes aspectos:

- Mesmo que os resultados numéricos não variem tanto, é importante perceber que a álgebra de caminhos fornece um limite corrente para qualquer medida utilizada para desempenho.
- Diferentes estratégias da álgebra de caminhos podem ser simultaneamente avaliadas para verificar qual é a melhor estratégia para um determinado atendimento.

A álgebra de caminhos é poderosa e flexível. Tal flexibilidade permite que ocorra uma exploração detalhada do espaço de soluções e a identificação do melhor critério e política que devem ser utilizados para um provedor de infraestrutura ou provedor de serviços. Os requisitos da escolha podem ser técnicos, como, por exemplo, a garantia de QoS a um determinado cliente. Ou podem ser comerciais, como o atendimento privilegiado de uma grande conta de um provedor.

A álgebra de caminhos não impõe qualquer restrição sobre o tipo e número de métricas que podem ser utilizadas. É possível utilizar métricas lineares e não-lineares juntas, assim como qualquer combinação de métricas pode ser empregada.

A álgebra de caminhos realiza a enumeração e ordenação de todos os caminhos elegíveis e, portanto, não somente a melhor solução está disponível, mas todas as soluções podem ser acessadas. Esse resultado é muito importante para implementar técnicas de recuperação no caso de uma falha na rede de substrato.

6.2 Trabalhos Futuros

No desenvolvimento deste trabalho, dentro do contexto da virtualização de redes e do problema VNE foram detectadas diversas necessidades de pesquisa e desenvolvimento, que ficam como sugestão para que seja dada continuidade em trabalhos futuros:

- a) Eliminação da integração entre os programas da álgebra de caminhos e o ambiente ALEVIN. O programa PAViLiM deve ser convertido para JAVA e seu download disponibilizado como software livre no repositório de código fonte *sourceforge* [63][64]. O algoritmo PAViLiM convertido deve ser integrado nativamente com o ambiente ALEVIN.
- b) Mapeamento de nós virtuais por meio da utilização da álgebra de caminhos. O mapeamento de enlaces virtuais para caminhos na rede de substrato é sempre mais eficiente e com melhor desempenho quando são utilizadas as estratégias da álgebra de caminhos. Uma proposta de trabalhos futuros é realizar o mapeamento de nós e de enlaces coordenado com a utilização da álgebra de caminhos e verificar o sucesso desta abordagem.
- c) Estudo do atendimento parcial de VNRs. Os algoritmos da literatura realizam apenas o atendimento completo de VNRs. Não existe suporte a atendimento parcial. Uma proposta de trabalhos futuros é estudar todas as possibilidades de atendimento parcial de VNRs com a utilização de políticas definidas pelo usuário e suporte a mapeamentos “elásticos”²⁶.

²⁶ Mapeamentos elásticos são mapeamentos variáveis com o tempo e dependem da taxa de utilização da SN. Quando a SN está com recursos disponíveis os mapeamentos de todos os enlaces virtuais serão completos, e quando a SN estiver com escassez de recursos, os mapeamentos de requisições poderão ser parciais, ou seja, os graus de atendimento serão diferentes e “elásticos”.

- d) Estudo do atendimento *multi-path* de VNRs. Existem várias estratégias para mapeamento de enlaces virtuais para caminhos na rede de substrato utilizando técnicas *multi-path*. O algoritmo PAViLiM foi modificado para permitir o atendimento *multi-path* pós-atendimento de todas as VNRs *single-path*. É possível implementar outras estratégias de *multi-path* para o algoritmo PAViLiM, por exemplo, o *multi-path* para o balanceamento de carga ou o *multi-path* concomitante com o mapeamento *single-path*, que pode identificar, inclusive, qual dos mapeamentos é o mais eficaz.
- e) Análise do efeito da ordenação dos enlaces virtuais quando do atendimento de uma VNR. Os enlaces virtuais de uma VNR são atendidos de forma sequencial seguindo a numeração dos nós virtuais. Uma proposta para trabalhos futuros é realizar um atendimento de enlaces virtuais utilizando as mesmas políticas já utilizadas para a ordenação dos atendimentos de requisições de redes virtuais.
- f) Estudo e implementação de técnicas de rápida recuperação (*fast recovery*²⁷) e rápido reroteamento (*fast reroute*²⁸) após a detecção de uma falha na rede de substrato ou após a detecção de um caminho na rede de substrato com alta ou baixa utilização. A álgebra de caminhos realiza a enumeração e ordenação de todos os caminhos elegíveis e, portanto, pode escolher rapidamente um caminho alternativo após a falha de um caminho ou após a detecção de uma utilização inadequada na infraestrutura de rede determinada pela engenharia de tráfego.

²⁷ rápida recuperação (*fast recovery*): são técnicas que permitem a determinação de um caminho alternativo de forma extremamente rápida (milissegundos) após o evento de uma falha do caminho principal [64].

²⁸ rápido reroteamento (*fast reroute*): são técnicas que permitem a mudança de um caminho na rede após uma solicitação externa vinda, por exemplo, da engenharia de tráfego [65].

Referências Bibliográficas

1. Feamster, N.; Gao, L. e Rexford, J. [ed.]. How to Lease the Internet in Your Spare Time. ACM. New York, NY, USA. In: **SIGCOMM Computer Communication Review**, janeiro de 2007.
2. Milan, Y. et al. [ed.] Rethinking virtual network embedding: substrate support for path splitting and migration. ACM. In: **SIGCOMM Computer Communication Review**, v. 38, n. 2, abril 2008.
3. Schaffrath, G. et al. Network virtualization architecture: proposal and initial prototype. New York, NY, USA: ACM, 2009. In: **VISA'09 - Proceedings** of the 1st ACM workshop on virtualized infrastructure systems and architectures.
4. Chowdhury, N. M. e Rahman, M. Virtual network embedding with coordinated node and link mapping. Rio de Janeiro, Brasil, 2009. In: **Proc. IEEE INFOCOM 2009**.
5. Peterson, L. e Davie, B. S. **Computer networks: a systems approach**. 4. ed. s.l.: Elsevier, 2007.
6. Botero, J. F. et al. ALEVIN - a *framework* to develop, compare and analyze virtual network embedding algorithms. **Electronic Communications of the EASST**, p. 12, 2011.
7. Xu, L. Efficient and scalable on-demand data streaming using UEP codes. s.l.: ACM. In: **Multimedia Conference**. p. 70-78, 2001.
8. **IRTF**. Virtual Networks Research Group (VNRG). Disponível em: <http://www.irtf.org/vnrg>. Acessado em: abril de 2011.]
9. Fajjari, I. et al. VNE-AC: Virtual Network Embedding Algorithm based on Ant Colony metaheuristic. Kyoto, Japan: s.n., 2011. In: **IEEE International Conference on Communications ICC2011**.
10. Carré, B. **Graphs and networks**. Oxford, UK: Oxford University Press, 1979.

11. Gouda, M. G. e Schneider, M. [ed.] Maximizable routing metrics. IEEE ACM. 2003, **Transactions Networking**, p. 663-675.
12. Sobrinho, J. L. [ed.] An algebraic theory of dynamic network routing. IEEE ACM. 2005, **Transaction Networking**, p. 1160-1173.
13. Herman, W. e Amazonas, J. R. Hop-by-Hop routing convergence analysis based on paths algebra. In: **Proceedings 2007 Eletronics, Robotics and Automative Mechanics Conference**, p. 9-14, 2007.
14. Herman, W. Tese (Doutorado) **Formulação algébrica para modelagem de algoritmos de roteamento multi-restritivos**. Universidade de São Paulo: EPUSP, 2009.
15. Giorgi, R. **Metodologia para avaliação de segurança de servidores baseada em ferramentas de ataque e detecção de vulnerabilidades**. São Paulo: IPT, 2004.
16. **Princeton. Network virtualization**. Disponível em:
<http://www.cs.princeton.edu/~jrex/virtual.html>. Acessado em: abril de 2011.
17. **GENI. Geni projects**. Disponível em: http://www.geni.net/?page_id=38. Acessado em: abril de 2011.
18. **VINI. A virtual network infrastructure**. Disponível em: <http://www.vini-veritas.net/>. Acessado em: abril de 2011.
19. Amazonas, J. R., Hesselbach, X. e Botero, J. F. [ed.]. A new strategy for virtual network mapping optimization based on paths algebra. In: IEEE. 2010.
20. ZHU, Y e Ammar, M. Algorithms for assigning substrate network resources to virtual network components. In: **Proc. IEEE INFOCOM 2006**, p. 2812-2823.

21. Lu, J e Turner, J. **Efficient mapping of virtual networks onto a shared substrate.** [<http://cse.seas.wustl.edu/Research/FileDownload.asp?503>] [prod.] Washington. Washington: s.n., 2006. Disponível em:
<http://cse.seas.wustl.edu/Research/FileDownload.asp?503>. Acessado em: dezembro 2011.
22. Lischka, J e Karl, H. A virtual network mapping algorithm based on subgraph isomorphism detection. In: **VISA '09: Proceedings** of the 1st ACM workshop on Virtualized infrastructure systems and architectures. p. 81-88.
23. Botero, J.F., Hesselbach, X. e Fischer, A. Optimal mapping of virtual networks with hidden hops. **Telecommunication Systems**, v. 52, n. 3, p. 1-10, 2013 (no prelo). original publication - www.springerlink.com (march 2011).
24. Houidi, I. et al. [ed.] Virtual network provisioning across multiple substrate networks. **Computer Networks**. Special Issue on Architectures and Protocols for the Future Internet. v. 55, n. 4, p. 1011-1023, 2011.
25. Cheng, X., et al. Virtual network embedding through topology-aware node ranking. In: **SIGCOMM Comput. Commun.**, v. 41, p. 38-47, 2011.
26. Chowdhury, M., Samuel, F. e Boutaba, R. [ed.] Policy-based virtual network embedding across multiple domains. In: **ACM SIGCOMM**. Workshop on Virtualized infrastructure systems and architectures - VISA '10. p. 49-56, 2010.
27. Botero, J. F. et al. **A novel paths algebra-based strategy to solve the VNE problem.** p. 32, 2012 (no prelo).
28. Whitehead, A. N. **A treatise on universal algebra:** with applications. s.l.: Cornell University Library Historical Mathematics Monographs collection, 1898.
29. Abstract Math. abstractmath.org. Disponível em:
<http://www.abstractmath.org/MM/MMSets.htm>. Acessado em: dezembro 2011.
30. Camargo, H. A. **Matemática discreta I. Notas de aula** - Matemática discreta para fundamentos da computação. São Carlos: UFSCAR, 2011.

31. De Morgan. **MTSU University**. Disponível em:
http://frank.mtsu.edu/~phys2020/Lectures/L19-L25/L3/DeMorgan/body_demorgan.html. Acessado em: 5 de janeiro de 2012.]
32. **Wikimedia Foundation**. *Duality*. Disponível em:
http://en.wikipedia.org/wiki/Duality_%28mathematics%29. Acessado em: janeiro de 2012.
33. **Wikimedia Foundation**. *Rede complexa*. Disponível em:
http://pt.wikipedia.org/wiki/Rede_complexa. Acessado em: janeiro de 2012.
34. Stoll, R.R. **Set Theory and Logic**. New York, EUA: Dover Books on Mathematics, 1979.
35. Chartrand, G. **Introductory graph theory**. Boston, EUA: Weber & Schmidt, 1977.
36. Abe, J. M. **Estudos Avançados - A noção de estrutura em matemática e física**. Scielo. [Online] Disponível em: http://www.scielo.br/scielo.php?pid=S0103-40141989000200007&script=sci_arttext. Acessado em: 7 de janeiro de 2012.
37. **Wikimedia Foundation**. *Ordem lexicográfica*. disponível em:
http://pt.wikipedia.org/wiki/Ordem_lexicogr%C3%A1fica. Acessado em: janeiro de 2012.
38. Cormen, T. H. et al. **Introduction to Algorithms**. Third Edition. Massachusetts, EUA: Massachusetts Institute of Technology Press, 2009.
39. Oliveira, A. C. M. **Estrutura de dados – Notas de Aula**. Disponível em:
http://www.deinf.ufma.br/~acmo/grad/ED_complexidade_2005.pdf . Acessado em março, 2012.
40. Aho, A. V.; Hopcroft, J. E. e Ulmann, J. D. **Data Structures and Algorithms**. USA: Addison-Wesley, 1987.
41. Waxman, B. M. Routing of multipoint connections. **IEEE Journal on Selected Areas in Communications**, v. 6, n. 9, december 1988.

42. Calvert, K. L.; Doar, M. B. e Zegura, E. W. Modeling internet topology. **IEEE Communications Magazine**, June 1997.
43. Nicoletti, M. C.; Hruschka Jr., E. R. **Fundamentos da teoria dos grafos para computação**. São Carlos: EDUFSCar, 2006.
44. Trudeau, R. J. **Introduction to graph theory**. 2nd. Ohio: Dover Books on Mathematics, 1993.
45. Quoitin, B. Topology generation based on network design heuristics. In: **Proceedings of the 2005 ACM Conference on Emerging Network Experiment and Technology**. Toulouse, France, CoNEXT'05, October 24-27, 2005
46. Tangmunarunkit, H. Network topology generators: degree-based vs. structural. In: **Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and protocols for Computers Communications**. SIGCOMM02, 2002.
47. Vukadinovic, D.; Huang, P. On the spectrum and structure of internet topology graphs. **Computer Science. Lecture Notes in Computer Science (LNCS)**, v. 2346, 2002.
48. Medina, A.; Matta, I. e Byers, J. On the origin of power laws in internet topologies. **SIGCOMM Computer Communication**, 2000.
49. Zegura, E. W.; Calvert, K. L. A quantitative comparison of graph-based models for internet topology. **IEEE/ACM Transactions on Networking**, v. 5, n. 6, December 1997.
50. Zegura, E. W. Disponível em:
<http://orfe.princeton.edu/courses/orf401/handouts/ModelingInternetTopologyZegura.pdf>. Acessado em: março 2012.
51. Leskovec, J. PhD thesis. **Dynamics of large networks / dynamismus magnorum retium**. USA: Carnegie Mellon University, 2008.

52. **MATLAB**. Disponível em: <http://www.mathworks.com/products/matlab/>. Acessado em: fevereiro 2012.
53. **Wikimedia Foundation**. *MATLAB*. Disponível em: <http://pt.wikipedia.org/wiki/MATLAB> . Acessado em: fevereiro 2012.
54. **ALEVIN** – Algorithms for Embedding Virtual Networks. Disponível em: <http://alevin.sf.net> . Acessado em: fevereiro 2012.
55. Barra, C. R. Tese (Doutorado) **Caracterização experimental e por simulação e modelagem da qualidade de serviço obtida na transmissão de áudio e vídeo em tempo real**. Universidade de São Paulo: EPUSP, 2005.
56. Duelli, M.; Botero, J. F.; Hesselbach, X. VNREAL: Virtual Network Resource Embedding Algorithms in the *Framework* ALEVIN. In: **Proc. of the 7th Euro-NF Conf. on Next Generation Internet (NGI 2011)**. p. 1-2, IEEE, 2011.
57. Houaiss, A.; Villar, M. S. **Houaiss da Língua Portuguesa**. Rio de Janeiro, Brasil: Editora Objetiva Ltda, 2007.
58. Knuth, D. E. **The art of Computer Programming – Fundamental Algorithms v.1**. Third Edition. Massachusetts, EUA: Addison Wesley Longman, 1997.
59. Kuipers, F., et al. **An overview of constraint-based path selection algorithms for QoS routing**. In: IEEE Communications Magazine 40, 50-55, IEEE, 2002.
60. Papadimitriou, P., et al. **Implementing network virtualization for a future internet – Concept and Performance Aspects**. In: 20th ITC Specialist Seminar on Network Virtualization, 2009.
61. Gallo, G., Pallottino, S. **Shortest-path algorithms**. In: Business and Economics, v.3, Holanda: Springer Netherlands, 1998.
62. Chowdhury, N. M. e Boutaba, R. **A survey of network virtualization**. In: Computer Networks, 862-876, 2010.
63. **Wikimedia Foundation**. *SourceForge*. Disponível em: <http://en.wikipedia.org/wiki/SourceForge> . Acessado em: março 2012.

64. **SourceForge** – *Repositório de código fonte baseado na Web*. Disponível em: <http://sourceforge.net>. Acessado em: março 2012.
65. Ma, M. **Current Research Progress of Optical Networks**. Singapura: Springer Science Business, 2009.
66. Minei, I., Lucek, J. **MPLS-enabled applications: Emerging developments and new technologies**. United Kingdom: John Wiley & Sons, 2011.

ANEXOS

O Anexo A apresenta uma prova por indução das equações 2 e 3 citadas na Seção 4.2 – “Complexidade Algorítmica” que calculam o número máximo de caminhos simples que podem ser enumerados sem filtragem e com filtragem em um dígrafo completamente conectado.

O Anexo B apresenta os principais programas do algoritmo PAViLiM.

O Anexo C fornece as matrizes de adjacência de vinte requisições de rede virtual e os mapeamentos de nós virtuais para nós físicos na SN utilizados para cada VNR.

O Anexo D apresenta os resultados individuais para cada cenário que foi utilizado nas simulações.

Anexo A – Número Total de Caminhos em um Dígrafo

O total de caminhos simples sem filtragem em um dígrafo completamente conectado é dado pela Equação 2.

Total de Caminhos (sem filtragem): $2 \cdot n! + n!/2! + n!/3! + \dots + n!/(n-2)!$, para $n > 3$

(2)

Demonstração por Indução

O dígrafo completamente conectado com 2 (dois) vértices possui 2 caminhos simples, como pode ser visto pela Figura 38.

Figura 38 – Dígrafo com 2 vértices.

Figura 39 – Caminhos com origem no vértice 1 em um dígrafo com 3 vértices.

O dígrafo completamente conectado com três vértices possui quatro caminhos simples iniciando em cada vértice como pode ser visto na Figura 39, e o número total de caminhos simples encontrados é dado por:

$$(\text{número de vértices}) * (\text{quantidade de caminhos por vértice}) = 3 * 4 = 12$$

O dígrafo completamente conectado com 4 (quatro) vértices possui 60 caminhos simples, a Figura 40 exibe todos os caminhos com origem no vértice 4.

Figura 40 – Caminhos com origem no vértice 4 em um dígrafo com 4 vértices.

O dígrafo completamente conectado com quatro vértices possui quinze caminhos simples iniciando em cada vértice como pode ser visto na Figura 40, e o número total de caminhos simples encontrados é dado por:

$$(\text{número total de vértices}) * [(\text{quantidade total de caminhos para 3 vértices}) + 3]$$

$$4 * [(4 * 3) + 3] = 2 * [(4 * 3 * 2) + 3 * 2] = 2 * (4 * 3 * 2) + 2 * (3 * 2) =$$

$$2 * (4 * 3 * 2) + 2 * \frac{(4 * 3 * 2)}{4} = 2 * (4 * 3 * 2) + 2 * \frac{(4 * 3 * 2)}{2 * 2} =$$

$$2 * (4 * 3 * 2 * 1) + \frac{(4 * 3 * 2 * 1)}{2 * 1} = 2 * 4! + \frac{4!}{2!} = 60 \text{ caminhos}$$

Analiticamente, pode-se utilizar a equação:

$$2n! + \frac{n!}{2!} = 2 * 4! + \frac{4!}{2!} = 48 + 12 = 60 \text{ caminhos}$$

Por hipótese de indução, supor que a Equação 2 é obedecida para um dígrafo de n vértices, com $n > 3$. A Figura 42 apresenta em uma “caixa” o número total de caminhos desse dígrafo.

Figura 42 – Quantidade total de caminhos para N vértices

Deve-se provar que a Equação 2 é válida para um dígrafo de $(n+1)$ vértices. O dígrafo completamente conectado com $(n+1)$ vértices possui um número total de caminhos simples encontrados dado por:

$$(\text{número total de vértices}) * [(\text{quantidade total de caminhos para } N \text{ vértices}) + N]$$

$$(n + 1) [2n! + n!/2! + n!/3! + \dots + n!/(n - 2)! + n] =$$

$$2(n + 1)! + (n + 1)!/2! + (n + 1)!/3! + \dots + (n + 1)!/[(n + 1) - 3]! + (n + 1) * n =$$

$$2(n + 1)! + \frac{(n + 1)!}{2!} + \frac{(n + 1)!}{3!} + \dots + \frac{(n + 1)!}{[(n + 1) - 3]!} + \frac{(n + 1)!}{[(n + 1) - 2]!}$$

A equação acima é a Equação 2 para um dígrafo de $(n+1)$ vértices.

Prova-se assim a Equação 2 por indução.

Total de Caminhos (sem filtragem): $2 \cdot n! + n!/2! + n!/3! + \dots + n!/(n-2)!$, para $n > 3$

(2)

O total de caminhos simples **com filtragem** em um dígrafo completamente conectado é dado pela Equação 3.

Total de Caminhos (com filtragem): $n!/(n-2)! + n!/(n-3)! + n!/(n-4)! + \dots$

Profundidade Desejada = k termos da expressão

Para $n > 3$ e $k < (n-1)$

(3)

Demonstração por Indução

O dígrafo completamente conectado com quatro vértices pode ser visto na Figura 40. O número total de caminhos simples com filtragem para $k=1$ e $k=2$ é dado por:

*(número total de vértices) * [(quantidade total de caminhos com filtragem)]*

para profundidade $k = 1 \rightarrow 4 * 3 = 12$

para profundidade $k = 2 \rightarrow 4 * 3 + 4 * 3 * 2 = 36$

Analiticamente, pode-se utilizar a equação:

$$k = 1 \rightarrow \frac{n!}{(n-2)!} = \frac{4!}{2!} = 12 \text{ caminhos}$$

$$k = 2 \rightarrow \frac{n!}{(n-2)!} + \frac{n!}{(n-3)!} = \frac{4!}{2!} + \frac{4!}{1!} = 36 \text{ caminhos}$$

O dígrafo completamente conectado com cinco vértices pode ser visto na Figura 41. O número total de caminhos simples com filtragem para $k=1$, $k=2$ e $k=3$ é dado por:

$$(\text{número total de vértices}) * [(\text{quantidade total de caminhos com filtragem})]$$

$$\text{para profundidade } k = 1 \rightarrow 5 * 4 = 20$$

$$\text{para profundidade } k = 2 \rightarrow 5 * 4 + 5 * 4 * 3 = 80$$

$$\text{para profundidade } k = 3 \rightarrow 5 * 4 + 5 * 4 * 3 + 5 * 4 * 3 * 2 = 200$$

Analiticamente, pode-se utilizar a equação:

$$k = 1 \rightarrow \frac{n!}{(n-2)!} = \frac{5!}{3!} = 5 * 4 = 20 \text{ caminhos}$$

$$k = 2 \rightarrow \frac{n!}{(n-2)!} + \frac{n!}{(n-3)!} = \frac{5!}{3!} + \frac{5!}{2!} = 80 \text{ caminhos}$$

$$k = 3 \rightarrow \frac{n!}{(n-2)!} + \frac{n!}{(n-3)!} + \frac{n!}{(n-4)!} = 200 \text{ caminhos}$$

Por hipótese de indução, supor que a Equação 3 é obedecida para um dígrafo de n vértices, com $n > 3$ e a profundidade $k < (n-1)$. Abaixo a Equação 3 que fornece o total de caminhos simples com profundidade de $k=1$, $k=2$ até $k=n-2$:

$$k = 1 \rightarrow \frac{n!}{(n-2)!} \text{ caminhos}$$

$$k = 2 \rightarrow \frac{n!}{(n-2)!} + \frac{n!}{(n-3)!} \text{ caminhos}$$

$$k = n-2 \rightarrow \frac{n!}{(n-2)!} + \frac{n!}{(n-3)!} + \dots + \frac{n!}{n - [(n-2) + 1]} \text{ caminhos}$$

$$k = n-2 \rightarrow \frac{n!}{(n-2)!} + \frac{n!}{(n-3)!} + \dots + \frac{n!}{1!} \text{ caminhos}$$

Deve-se provar que a Equação 3 é válida pra um dígrafo de $(n+1)$ vértices com filtro de profundidade: $k=1$, $k=2$, $k=3$ até $k=\lfloor(n+1)/2\rfloor$.

Sabe-se que a quantidade de caminhos com profundidade $k=1$ é dada por:

$$k = 1 \rightarrow (n + 1) * (n) = \frac{(n + 1)!}{[(n + 1) - 2]!}$$

A quantidade de caminhos com profundidade $k=2$ é dada por:

$$\begin{aligned} & (\text{profundidade para } k = 1 \text{ do dígrafo com } (n + 1) \text{ vértices}) + \\ & (n + 1) * (\text{profundidade para } k = 1 \text{ do dígrafo com } n \text{ vértices}) = \\ & \frac{(n + 1)!}{[(n + 1) - 2]!} + (n + 1) * \frac{n!}{(n - 2)!} = \\ & \frac{(n + 1)!}{[(n + 1) - 2]!} + \frac{(n + 1)!}{[(n + 1) - 3]!} \end{aligned}$$

A quantidade de caminhos com profundidade $k=3$ é dada por:

$$\begin{aligned} & (\text{profundidade para } k = 1 \text{ do dígrafo com } (n + 1) \text{ vértices}) + \\ & (n + 1) * (\text{profundidade para } k = 2 \text{ do dígrafo com } n \text{ vértices}) = \\ & \frac{(n + 1)!}{[(n + 1) - 2]!} + (n + 1) * \left[\frac{n!}{(n - 2)!} + \frac{n!}{(n - 3)!} \right] = \\ & \frac{(n + 1)!}{[(n + 1) - 2]!} + \frac{(n + 1)!}{[(n + 1) - 3]!} + \frac{(n + 1)!}{[(n + 1) - 4]!} \end{aligned}$$

A quantidade total de caminhos simples para o dígrafo completamente conectado com $(n+1)$ vértices e com profundidade $k=[(n+1) - 2]$ é dada por:

(quantidade de caminhos com profundidade $k = 1$ do dígrafo com $(n + 1)$ vértices) +

*$(n + 1) * (quantidade de caminhos com profundidade $\{[(n + 1) - 2] + 1\}$ do dígrafo com n vértices)$*

$$\frac{(n + 1)!}{[(n + 1) - 2]!} + (n + 1) * \left[\frac{n!}{(n - 2)!} + \frac{n!}{(n - 3)!} + \dots + \frac{n!}{1!} \right]$$

Que pode ser transformada na equação:

$$\frac{(n + 1)!}{[(n + 1) - 2]!} + \frac{(n + 1)!}{[(n + 1) - 3]!} + \frac{(n + 1)!}{[(n + 1) - 4]!} + \dots + \frac{(n + 1)!}{1!}$$

Se obteve a Equação 3 para um dígrafo de $(n+1)$ vértices com filtros de profundidade: $k=1$, $k=2$, $k=3$ e $k=[(n+1) -2]$.

Prova-se assim que a Equação 3 é válida para qualquer número de vértices n , com $n>3$, por indução.

Total de Caminhos (com filtragem): $n!/(n-2)! + n!/(n-3)! + n!/(n-4)! + \dots$

Profundidade Desejada = k termos da expressão

Para $n > 3$ e $k < (n-1)$

(3)

Anexo B – Principais Programas do Algoritmo PAViLiM

O código fonte dos seguintes programas será apresentado:

- *main_PavilimA*
- *main_PavilimB*
- *attendVNsHiddenSeptemberX*
- *connectivity*
- *searchPath_filter*
- *sortPath*
- *readAttendOrderX*
- *readMetricsOrderX*
- *main_attendVNs*
- *updateNewMetrics*
- *orderNewPaths*
- *VNrequestX*
- *attendVNsHiddenX*

B.1 – Programa *main_PavilimA*

```
% Mapeamento de Enlaces de Rede Virtual
% algoritmo main_PAViLiMA
% Programa Principal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Matriz de Substrato
% Matriz T
clear all
echo off
load 'caminho\substrateNetwork.dat'
T=substrateNetwork;
M{1}=T;
```

[illegible]

```

N{1}=T;
N{2}=substrateMetric1;
N{3}=cpu2matrix(T,Rcpu);
N{5}=N{2}+N{3};      % metrica da soma de banda e cpu
% N{1} - hop
% N{2} - bw
% N{3} - cpu
% N{4} - delay
% N{5} - bw+cpu
% N{6} - reliability
% N{7} - energy consupcion
% N{8} - PLR (se tiver)
% G{1} - síntese e ordenação de hop
% G{2} - síntese e ordenação de bw
% G{3} - síntese e ordenação de cpu
% G{4} - síntese e ordenação de delay
% G{5} - síntese e ordenação de bw+cpu
% G{6} - sintese e ordenacao de bw total
% wword{1} - word de hop
% wword{2} - word de bw
% ...
[G wword]=readSubstrateMetricsSeptember;
% readMetricsOrder
%
% 1- bw, cpu
% 2- cpu, bw
% 3- hop, bw, cpu
% 4- hop, cpu, bw
% 5- delay, bw, cpu
% 6- delay, cpu, bw

```

```

% 7- availability, bw, cpu
% 8- availability, cpu, bw
% 9- do your choice: availability, delay, jitter, hop, bw, cpu
% 10 - bw+cpu,hop
% 11 - hop,bw+cpu
% 12 - hop, bw Total
% no momento 1 a 4 e 10 a 12
MO= virtualStart(2);
if MO == 1
    [M F word iType]=metricsOrder1(N,G,wword);
elseif MO == 2
    [M F word iType]=metricsOrder2(N,G,wword);
elseif MO == 3
    [M F word iType]=metricsOrder3(N,G,wword);
elseif MO == 4
    [M F word iType]=metricsOrder4(N,G,wword);
elseif MO == 10
    [M F word iType]=metricsOrder10(N,G,wword);
elseif MO == 11
    [M F word iType]=metricsOrder11(N,G,wword);
elseif MO == 12
    [M F word iType]=metricsOrder12(N,G,wword);
else
    % Outros não disponível
    % [M F word iType]=metricsOrder5(N,G,wword);
    fprintf('\nOthers is not Available\n');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[w] = updateNewMetrics(P,M,Nn);
orderNewPaths1;

```



```

% Inicializacao de Variaveis

totalNos=0;

totalMetrica1=0;

totalMetrica2=0;

totalGeral=0;

% Ordenacao dos Caminhos

for i=1:virtualStart(3)

    % leitura das VNs

    arq1 = strcat('caminho\','virtualRequest_',int2str(i));

    VN{i}=load(arq1);

    % leitura da Metrica 1 - MBW

    arq2 = strcat('caminho\','virtualMetric1_',int2str(i));

    Mbw{i}=load(arq2);

    % leitura da Metrica 2 - Vcpu

    arq3 = strcat('caminho\','virtualMetric2_',int2str(i));

    Vcpu{i}=load(arq3);

    % leitura da Hidden Matrix

    arq4 = strcat('caminho\','virtualHidden_',int2str(i));

    HiddenMatrix{i}=load(arq4);

    % Soma da Capacidade de Nos e Recursos

    totalNos=totalNos+VN{i};

    totalMetrica1=totalMetrica1+addMatrix(Mbw{i});

    totalMetrica2=totalMetrica2+addVector(Vcpu{i});

end

% Total Geral da Capacidade dos Recursos

totalGeral=totalMetrica1+totalMetrica2;

% inicialização de variável

Attend=[];

RV={};

RO=zeros(1,size(VN,2)); % vetor de ordenação dos caminhos.

```

```

fid=fopen('C:\Users\User\Documents\MATLAB\requestOrder','wt+'); % arquivo ordenado.

% laço para atendimento das requisições virtuais
if strcmp(AttendOrder,'FIFO')
    for i=1:size(VN,2)
        RO(i)=i;
        fprintf(fid,'%d \n', RO(i));
    end
elseif strcmp(AttendOrder,'MCFUSP')
    request = computeTotalRequests2May(Mbw,Vcpu,N{2},Rcpu);
    request = bubblesort(request);
    k=1;
    for j=size(request,2):-1:1
        i=request{j}(2);
        RO(k)=i;
        fprintf(fid,'%d \n', RO(k));
        k=k+1;
    end
elseif strcmp (AttendOrder,'LCFUSP')
    request = computeTotalRequests2May(Mbw,Vcpu,N{2},Rcpu);
    request = bubblesort(request);
    for j=1:size(request,2)
        i=request{j}(2);
        RO(j)=i;
        fprintf(fid,'%d \n', RO(j));
    end
elseif strcmp(AttendOrder,'MCF')
    request = computeTotalRequests(Mbw,Vcpu);
    request = bubblesort(request);
    k=1;
    for j=size(request,2):-1:1

```

```

        i=request{j}(2);
        RO(k)=i;
        fprintf(fid,'%d \n', RO(k));
        k=k+1;
    end
else
    request = computeTotalRequests(Mbw,Vcpu);
    request = bubblesort(request);
    for j=1:size(request,2)
        i=request{j}(2);
        RO(j)=i;
        fprintf(fid,'%d \n', RO(j));
    end
end

end

fclose(fid);

% Salva as variáveis em questão
save 'caminho\searchPath.mat'

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Resultados:
% Po{m}{o}{d}{:} array de caminhos não ordenados
% So{m}{o}{d}{:} array das sínteses correspondentes do Po
% Pfo {1}{o}{d}{:} array de caminhos ordenados
% Sfo {m}{o}{d}{:} array das sínteses correspondentes do Pfo
% NH matriz de próximo nó

```

B.2 – Programa *main_PavlimB*

```
% main_PavlimB

% Leitura de Requisições Virtuais - VN requests

% plug-in ou arquivo que será carregado.

% opção no momento - arquivo carregado.

clear all

load 'caminho\searchPath.mat'

% Inicialização de Matrizes.

VN={};

Vcpu={};

Mbw={};

VN2T={};

% Inicialização de Variável.

i=1;

% Preparação para leitura da primeira requisição

while i <= virtualStart(3)

    % leitura do arquivo virtualnodemapping_x

    arq6 = strcat('caminho','virtualnodemapping_',int2str(RO(i)),'.dat');

    if ((exist(arq6) == 2))

        arq6 = strcat('caminho','virtualnodemapping_',int2str(RO(i)),'.dat');

        nodeattend=load(arq6);

        if (int2str(nodeattend) == '1')

            % leitura do Mapeamento de Nós Virtual para Real

            arq5 = strcat('caminho','virtual_to_Real_',int2str(RO(i)));

            if ((exist(arq5) == 2))

                % Leitura das VNs, Métricas, Hidden, Mapeamento

                % leitura das VNs

                arq1 = strcat('caminho','virtualRequest_',int2str(RO(i)));

                VN{RO(i)}=load(arq1);

                % leitura da Metrica 1 - MBW
```

```

    arq2 = strcat('caminho\','virtualMetric1_',int2str(RO(i)));
    Mbw{RO(i)}=load(arq2);
    % leitura da Métrica 2 - Vcpu
    arq3 = strcat('caminho\','virtualMetric2_',int2str(RO(i)));
    Vcpu{RO(i)}=load(arq3);
    % leitura da Hidden Matrix
    arq4 = strcat('caminho\','virtualHidden_',int2str(RO(i)));
    HiddenMatrix{RO(i)}=load(arq4);
    % virtual to Real - node mapping
    VN2T{RO(i)}=load(arq5);
    % inicialização de variável
    Attend=[];
    RV={};
    % Rotina de Atendimento de Requisições
    [M,Rcpu,Attend,RV] =
attendVNsHiddenSeptember2(Pfo,M,iType,Mbw{RO(i)},Rcpu,Vcpu{RO(i)},VN{RO(i)},VN2T{RO(i)},
HiddenMatrix{RO(i)},RO(i),Attend,T,RV);

    % Atualização de Índices de Métricas e Reordenação
    if i < virtualStart(3)
        [w] = updateNewMetrics(P,M,Nn);
        orderNewPaths1;
    end

    % Atualização do contador de requisições
    i=i+1;
end
else
    % Esperar pela próxima requisição
    % Atualização do contador de requisições
    i=i+1;
end
end

```

```

        end
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Resultados:
%   Po{m}{o}{d}{:}   array de caminhos não ordenados
%   So{m}{o}{d}{:}   array das sínteses correspondentes do Po
%   Pfo {1}{o}{d}{:} array de caminhos ordenados
%   Sfo {m}{o}{d}{:} array das sínteses correspondentes do Pfo
%   NH matriz de próximo nó

```

B.3 – Programa *attendVNsHiddenSeptemberX*

```

function [M,Rcpu,Attend,RV] =
attendVNsHiddenSeptemberX(Pfo,M,iType,Mbw,Rcpu,Vcpu,VN,VN2T,HiddenMatrix,rn,Attend,T,RV)

% Ordena caminhos iguais para uma mesma métrica, ou seja ordena entre métricas
% Pfo{o}{d} contém os caminhos entre 'o' e 'd' ordenados entre as métricas
% Atendimento Completo da requisicao virtual
% Convenção para as métricas e síntese:
% iType{1} - Hop
% iType{2} - BW
% iType{3} - CPU
% iType{4} - Delay
% iType{N} - a definir
%
% Pfo - caminhos ordenados entre origem O e destino D
%
% M - matriz de métricas
% Mbw - matriz de métricas de BW para a VN request
% Vcpu - vetor de métricas de CPU para a VN request
% Rcpu - vetor de métricas de CPU da matriz de substrato
%

```

```

% Mcpu - matriz de métricas de CPU convertida pelo programa cpu2matrix
%
% VN - matriz de requisição para a rede virtual
% VN2T - matriz size(VN,2)x 2 onde em cada linha tem-se nós virtual e real
% rn - número da requisição
% T - matriz de substrato
% Matrizes e Vetores para resolução do problema de Hidden Nodes.
% HiddenMatrix - Matriz com pesos aplicados em Hidden Nodes.
% HiddenUsed - Vetor que indica se um nó físico foi utilizado para cálculo
% como nó intermediário no Hidden Node.
% D{n} - cria vetor linha com o nó origem n e 1's que identificam em sua posição os nós destino.
% C{n} - vetor linha com o valor dos nós virtuais destinos atingidos pelo nó origem n.
% C{n}(i) - nós diretamente conectados ao no nó origem nó n destino cada nó virtual.
% Attend - vetor que contem 1 para sucesso e 0 para fracasso de atendimentos.
% RV - resposta virtual
% Conversão do Vetor CPU para Matriz de CPU
Mcpu=cpu2matrix(VN,Vcpu);
% Matrizes temporárias de trabalho
TempMcpu=Mcpu;
TempVcpu=Vcpu;
TempRcpu=Rcpu;
TempM=M;
% Solução do Problema de Hidden Nodes
% HiddenUsed = ones(1,size(Rcpu,2));
% HiddenUsed = updateHidden(VN2T,HiddenUsed);
% inicialização de variáveis
n=1;
flag1=1;
flag2=1;
C={};

```

```

D={};
sucesso=0;
while (n<= size(VN,2)) && (flag1 ~= 0)
% flag1 é o sucesso analisar os links virtuais.
D{n}=VN(n,:);
C{n}=find(D{n}==1);
i=1;
while (i<=size(C{n},2)) && (flag1 ~=0)
    % Destino n para C{n}(i)
    % VL{n}{C{n}(i)}
    % Pfo{1}{o}{d}
    sucesso=0;
    j=1;
    while (j<= size(Pfo{1}{VN2T(n,2)}{VN2T(C{n}(i),2)},2)) && (sucesso ~=1)
        if (Mbw(n,C{n}(i)) > minimumBW(Pfo{1}{VN2T(n,2)}{VN2T(C{n}(i),2)}{j},TempM{iType{2}})) |
TempVcpu(n) > TempRcpu(VN2T(n,2)) | TempVcpu(C{n}(i)) > TempRcpu(VN2T(C{n}(i),2))
            j=j+1;
        else
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            % Attend VNs Hidden
            TempRcpu2=TempRcpu;
            % atualização de matrizes temporárias CPU
            TempRcpu2(VN2T(n,2))=TempRcpu2(VN2T(n,2)) - TempVcpu(n);
            TempRcpu2(VN2T(C{n}(i),2))= TempRcpu2(VN2T(C{n}(i),2)) - TempVcpu(C{n}(i));
            % Inicialização de Variável
            k=2;
            while (k < size(Pfo{1}{VN2T(n,2)}{VN2T(C{n}(i),2)}{j},2)) && flag2~=0
                if HiddenMatrix(n,C{n}(i)) >
                    TempRcpu2(Pfo{1}{VN2T(n,2)}{VN2T(C{n}(i),2)}{j}(k))
                    flag2=0;

```



```

else
    TempRcpu2(Pfo{1}{VN2T(n,2)}{VN2T(C{n}(i),2)}{j})(k) =
TempRcpu2(Pfo{1}{VN2T(n,2)}{VN2T(C{n}(i),2)}{j})(k) - HiddenMatrix(n,C{n}(i));
    flag2=1;
end
k=k+1;
end
if flag2 == 1
    sucesso=1;
    % atualização das métricas de CPU da Hidden Nodes
    TempRcpu = TempRcpu2;
    % atualização de requisições virtuais CPU
    TempVcpu(n)=0;
    TempVcpu(C{n}(i))=0;
    % atualização da matriz temporária de Banda

TempM{iType{2}}=updateBW(TempM{iType{2}},Mbw(n,C{n}(i)),(Pfo{1}{VN2T(n,2)}{VN2T(C{n}(i),2)}{j}));

    % salvar caminho
    RV{rn}{n}{C{n}(i)}=Pfo{1}{VN2T(n,2)}{VN2T(C{n}(i),2)}{j};
    % Indicar caminhos escolhidos para cada
    % enlace virtual - virtual link - em uma
    % requisicao atendida com sucesso ou não
    arq6 =
    strcat('caminho\','virtualAttend_',int2str(rn),'_',int2str(n),'_',int2str(C{n}(i)));
    AuxRV = RV{rn}{n}{C{n}(i)};
    save (arq6,'AuxRV','-ASCII');
end
j=j+1;
end
end
if sucesso == 0

```

```

        flag1 = 0;
    end

    i=i+1;

end

n=n+1;

end

arq4 = strcat('caminho\','virtualSuccess_',int2str(rn));
arq8='caminho\requestFail';
arq9='caminho\requestSuccess';

if flag1 == 0

    Attend(1,rn)=0;

    fprintf('\nVN request %d, was not attended! FAIL! \n',rn);

    % Indicar o Fracasso em Arquivo virtualSucess_#

    success=0;

    save (arq4,'success','-ASCII');

    % request fail

    save (arq8,'rn','-ASCII','-APPEND');

else

    Rcpu=TempRcpu;

    M{iType{3}}=cpu2matrix(T,Rcpu);

    M{iType{2}}=TempM{iType{2}};

    Attend(1,rn)=1;

    fprintf('\nVN request %d, was attended! SUCCESS!\n',rn);

    % Indicar o Sucesso em Arquivo virtualSucess_#

    success=1;

    save (arq4,'success','-ASCII');

    % request success

    save(arq9,'rn','-ASCII','-APPEND');

    % Indicar as Métricas Resultantes após o atendimento da requisição de
    % rede virtual rn nos arquivos: substrate Metric1_#.dat – Largura de Banda

```

```

% substrateMetric2_#.dat - CPU

% Metrica Final da rede de substrato

    arq5 = strcat('caminho\','substrateMetric1_',int2str(rn));

    AuxBW = M{iType{2}};

    save (arq5,'AuxBW','-ASCII');

% Métrica Final de CPU da rede de substrato

    arq5 = strcat('caminho\','substrateMetric2_',int2str(rn));

    save (arq5,'Rcpu','-ASCII');

end

```

B.4 – Programa *connectivity*

```

function [depth]=connectivity(Nn,G)

% A função deve ser chamada:

% [depth]=connectivity(Nn,G)

A1=G;

A2=G;

B=ones(Nn);

C=A1;

depth=1;

N=Nn*Nn;

status = 1;

% Parada se a conectividade não for atingida com M

% Inicialização de M

M=valor; % valor utilizado da tese de doutorado de [51]

if M>Nn

    M=Nn;

end

while (status) && depth<M

    depth=depth + 1;

    A2=A2*A1;

```

```

C = C + A2;
for i=1:N
    if C(i) >= 1
        C(i) = 1;
    end
end
if C == B
    status = 0;
end
end

% se o filtro por profundidade for menor ou igual a 2
% fazer com que seja 3 para aumentar a quantidade de caminhos enumerados.
if depth <=2
    depth=3;
end

fprintf('Conectividade atingida com profundidade: %d \n', depth);
% Conectividade = I + A + A*A + A*A*A .....

```

B.5 – Programa *searchPath_filter*

```

function [Nn,l] = searchPath_filter(T,graph)
%
% [Nn,l] = searchPath_filter(T,graph)
% Uma topologia é modelada em uma Matriz T em que as:
% - linhas são a origem das conexões.
% - colunas são o destino das conexões.
% Os valores das métricas de cada enlace é modelado em uma matriz M em que o array.
% M{m} corresponde a matriz da métrica m onde m pode ser 1 a 10 por exemplo => 10 métricas.
% Nn=length(T); % número total de nós da rede.
% Caminhos obtidos entre "o" e "d":
% P{o}{d}={[o, ..., d],...,[o, ..., d]} em que:

```

```

% P{o}{d}={[]} para o=d
% Vetores dos pesos dos enlaces dos caminhos obtidos em P{o}{d}{:}:
% w{m}{o}{d}(:) para m=1,2,..ou., m onde m é a métrica escolhida.
% F: array que contém para cada metrica a sintese e relação de ordem a ser aplicada.
% F{m}={'sintese', 'relação de ordem'}
% Ex: F{1}={'MIN','>'};
%     F{2}={'MAX','<'};
%     F{3}={'MUL','>'};
% F{M}={'ADD','<'}; em que size(F,2)=size(M,2)= número de métricas usadas.
% A relação de ordem léxica multidimensional é aplicada na sequencia em que se define o
% array F, ou seja:F{1} tem prioridade sobre F{2}
%           F{2} tem prioridade sobre F{3}, etc...
% Levantamento de árvores por cada nó
Nn=size(T,1);      % número total de nós da rede
C={};
D={};
I={};
% cálculo da profundidade para obtenção da conectividade.
depth=connectivity(size(T,1),T);
% rotina que salva variavel em arquivo
% saveFile(variavel,nome_arquivo)
arq='depth_random.dat';
saveFile(depth,arq);
for n=1:Nn          % para cada nó existente como origem busca as árvores possíveis
                    % em que n é o nó de origem da árvore
    h=1;            %número de hops
    D{h}{n}= T(n,:); % conexões diretas do nó "n"
    C{h}{n}= find(D{h}{n}==1); % busca nós con conexão direta ao no "n"
    % o comprimento indica inicialmente o número de árvores com 1 hop a partir do nó n.
    if ~isempty(C{h}{n})      % verifica se o nó "n"tem conexão

```

```

for a=1:size(C{h}{n},2), % se tem conexão, para 1o hop
    l{n}{a}=[n,C{h}{n}(1,a)]; % índice de nós de cada árvore com origem no nó "n"
end

% para 2o hop busca nós com conexão direta ao no "n+1" para cada nó n+1 contido em C1
NI=cell(1,1);
cp=0;
while cp~=size(l{n},2) % Não funcionou && h <= 4
    h=h+1;
    cp=0;
    for i=1:size(l{n},2), % para cada nó n+1
        if size(l{n}{i},2)>=h && h<=depth % profundidade - caminhos com profundidade + 1
            D{h}{i}= T(l{n}{i}(h,:));
            C{h}{i}= find(D{h}{i}==1);
            if ~isempty(C{h}{i})
                % eliminar loop de volta!!!!
                for j=1:size(l{n}{i},2)
                    k=find(C{h}{i}==l{n}{i}(1,j));
                    if isempty(k)
                        elseif k==1
                            C{h}{i}=[C{h}{i}(1,k+1:size(C{h}{i},2))];
                        elseif k>1
                            C{h}{i}=[C{h}{i}(1,1:k-1) C{h}{i}(1,k+1:size(C{h}{i},2))];
                        end
                    end
                end
            end
            if isempty(NI{:})
                rasc=0;
            else
                rasc=size(NI{:},2);
            end
            if ~isempty(C{h}{i})

```

```

        for j=1:size(C{h}{i},2)
            NI{:}{rasc+j}=[I{n}{i} C{h}{i}(1,j)];
        end
    else
        NI{:}{rasc+1}=[I{n}{i}];
    end
else
    if isempty(NI{:})
        rasc=0;
    else
        rasc=size(NI{:},2);
    end
    NI{:}{rasc+1}=[I{n}{i}];
    cp=cp+1;
end
else
    if isempty(NI{:})
        rasc=0;
    else
        rasc=size(NI{:},2);
    end
    NI{:}{rasc+1}=[I{n}{i}];
    cp=cp+1;
end
end
I{n}=NI{:};
NI=cell(1,1);
end
else % se o nó "n" não tem conexão
    I{n}{1}=[n];

```

```
    end  
end  
fprintf('O número do grafo no searchPath_filter é: %d', graph);  
fprintf('\n \n')
```


B.6 – Programa *sortPath*

```

function [P,w,A] = sortPath(M,I,Nn,graph);

%
% [P,w,A] = sortPath(M,I,Nn,graph);
%
% Inicialização variaveis
A=eye(Nn); % Matriz de Contagem de Caminhos
B=eye(Nn); % Matriz de Conectividade
C=eye(Nn); % Matriz de Conectividade Simétrica
control=0;
control1=0;
tempo=0;
tempo1=0;
path=Nn; % Variável que conta os caminhos descobertos
elapsedTime=clock;
P={};
w={};
for o=1:Nn % varre todas as origens
    for d=1:Nn % varre todas os destinos
        if o==d
            P{o}{d}{1}=[]; % define caminho nulo quando origem = destino
            for m=1:size(M,2) % define peso nulo para o enlace do caminho quando origem = destino
                w{m}{o}{d}{1}=[];
            end
        else
            if size(I{o}{1},2)>=2 % verifica se há pelo menos 1 hop
                k=1;
                for i=1:size(I{o},2)% varre os caminhos encontrados para a origem o
                    if ~isempty(find(I{o}{i}(:)==d)) %verifica se o caminho tem o nó d de destino
                        P{o}{d}{k}=[];
                    end
                end
            end
        end
    end
end

```

```

f=0;
for j=1:size(l{o}{i},2)    % varre dentro do caminho buscando o destino d
    if f==0 % se ainda não encontrou o destino
        if l{o}{i}(1,j)~=d
            P{o}{d}{k}=[P{o}{d}{k} l{o}{i}(1,j)];
            switch j
                case 1    % pula primeiro nó pois não há peso para montar
                case 2    % verifica se é o segundo nó
                    for m=1:size(M,2)% monta o peso do enlace
                        w{m}{o}{d}{k}=[M{m}(P{o}{d}{k}(1,j-1),l{o}{i}(1,j))];
                    end
                otherwise
                    for m=1:size(M,2) % monta o peso do enlace
                        w{m}{o}{d}{k}=[w{m}{o}{d}{k} M{m}(P{o}{d}{k}(1,j-1),l{o}{i}(1,j))];
                    end
                end
            end
        else
            P{o}{d}{k}=[P{o}{d}{k} l{o}{i}(1,j)];
            O=o;
            D=d;
            A(O,D)= A(O,D) +1;    % caminho descoberto
            path=path+1;
            arq2=strcat('arquivo',int2str(graph));
            saveFile(path,arq2);
            [B]=addconnect(O,D,B);
            [C]=addsymmetric(O,D,C);
            addtime2(elapsedTime,graph);
            [control,tempo]=printconnect1(Nn,control,tempo,elapsedTime,B,graph);
            [control1,tempo1]=printsymmetric1(Nn,control1,tempo1,elapsedTime,C);

```

```

switch j
case 1      % nunca deve ocorrer
    'Erro'
case 2      % verifica se é o segundo nó
    for m=1:size(M,2) % monta o peso do enlace
         $w_{\{m\}\{o\}\{d\}\{k\}} = [M_{\{m\}}(P_{\{o\}\{d\}\{k\}}(1,j-1), l_{\{o\}\{i\}}(1,j))];$ 
    end
otherwise
    for m=1:size(M,2) % monta o peso do enlace
         $w_{\{m\}\{o\}\{d\}\{k\}} = [w_{\{m\}\{o\}\{d\}\{k\}} M_{\{m\}}(P_{\{o\}\{d\}\{k\}}(1,j-1), l_{\{o\}\{i\}}(1,j))];$ 
    end
end
if k~=1      %verifica se há caminhos duplicados
    if all(size(P_{o}{d}{k})==size(P_{o}{d}{k-1}))
        if all(P_{o}{d}{k}==P_{o}{d}{k-1}) % verificar o caso em que os caminhos são iguais
            P_{o}{d}=P_{o}{d}(1:k-1);      %exclui caminho duplicado
            A(o,d)=A(o,d) - 1;      %exclui caminho duplicado - modificação Miguel Molina
            path=path -1;
            for m=1:size(M,2) % define peso nulo para o enlace do caminho
                 $w_{\{m\}\{o\}\{d\}} = w_{\{m\}\{o\}\{d\}}(1:k-1);$  % monta o peso do enlace
            end
            k=k-1;
        end
    end
end
k=k+1;
f=1;
end
end
end

```

```

else % se o caminho não tem o nó d de destino
    if k==1
        P{o}{d}{k}=[]; % define caminho nulo
        for m=1:size(M,2) % define peso
            w{m}{o}{d}{k}=[];
        end
    end
end
end

else % se não há pelo menos 1 hop
    P{o}{d}{1}=[]; % define caminho nulo
    for m=1:size(M,2) % define peso nulo
        w{m}{o}{d}{1}=[];
    end
end
end
end

% Contagem dos Caminhos utilizando a Matriz de Contagem de Caminhos
printTotalPath(Nn,A)

% Exibição dos Tempos em Horas, Minutos e Segundos
fprintf('O tempo para conseguir-se a Conectividade Simétrica é: \n \n')
timeFormat(tempo1)
fprintf("\n \n")
fprintf('O tempo para conseguir-se a Conectividade Normal é: \n \n')
timeFormat(tempo)
fprintf("\n \n")
fprintf('O número do grafo no separapath é: %d', graph);
fprintf("\n \n")

```

B.7 – Programa *readAttendOrderX*

```
function AttendOrder=readAttendOrderX

% function AttendOrder=readAttendOrderX

% função que define a ordem de atendimento

% o valor default é FIFO

AttendOrder=menu('Select VN request serve order','1) Most Consuming First', '2) Least Consuming
First', '3) FIFO (default)', '4) Most Consuming First (USP Proposed)', '5) Least Consuming First (USP
Proposed)', '6) Supply your VN request order');
```

B.8 – Programa *readMetricsOrderX*

```
function MO=readMetricsOrderX

MO = menu('Select the Sorting Path criterion','1) BW (first) and CPU (second)', '2) CPU (first) and BW
(second)', '3) HOP (first), BW (second) and CPU (third)', '4) HOP (first), CPU (second) and BW
(third)', '5) DELAY (first), BW (second) and CPU (third)', '6) DELAY (first), CPU (second) and BW
(third)', '7) AVAILABILITY (first), BW (second) and CPU (third)', '8) AVAILABILITY (first), CPU (second)
and BW (third)', '9) Do your choice: Avalability, Delay, Jitter, Packet Loss, HOP, BW, CPU');

% 1- bw, cpu
% 2- cpu, bw
% 3- hop, bw, cpu
% 4- hop, cpu, bw
% 5- delay, bw, cpu
% 6- delay, cpu, bw
% 7- availability, bw, cpu
% 8- availability, cpu, bw
% 9- do your choice: avalability, delay, jitter, packet loss, hop, bw, cpu
% 10 – bw+cpu,hop
% 11 – hop,bw+cpu
% 12 – hop, bw total
```

B.9 – Programa *main_attendVNs*

```
% Programa main_attendVNs

% Leitura de Requisições Virtuais - VN requests
% plug-in ou arquivo que será carregado.
% Inicialização de Matrizes
VN={};
Vcpu={};
Mbw={};
VN2T={};

% atendimento offline - número de atendimento
for i=1:virtualStart(3)
    % leitura das VNs
    arq1 = strcat('caminho\','virtualRequest_',int2str(i));
    VN{i}=load(arq1);
    % leitura da Metrica 1 - MBW
    arq2 = strcat('caminho\','virtualMetric1_',int2str(i));
    Mbw{i}=load(arq2);
    % leitura da Metrica 2 - Vcpu
    arq3 = strcat('caminho\','virtualMetric2_',int2str(i));
    Vcpu{i}=load(arq3);
    % leitura da Hidden Matrix
    arq4 = strcat('caminho\','virtualHidden_',int2str(i));
    HiddenMatrix{i}=load(arq4);
    % leitura do Mapeamento de Nós Virtual para Real
    arq5 = strcat('caminho\','virtual_to_Real_',int2str(i));
    VN2T{i}=load(arq5);
end

% inicialização de variável
Attend=[];
RV={};
```

```

% laço para atendimento das requisições virtuais
if strcmp(AttendOrder,'FIFO')
    for i=1:size(VN,2)
        % Rotina de Atendimento de Requisições
        [M,Rcpu,Attend,RV] =
attendVNsHiddenX(Pfo,M,iType,Mbw{i},Rcpu,Vcpu{i},VN{i},VN2T{i},HiddenMatrix{i},i,Attend,T,RV)
        % Atualização de Índices de Métricas e Reordenação
        [w] = updateNewMetrics(P,M,Nn);
        orderNewPaths;
    end
elseif strcmp(AttendOrder,'MCFUSP')
    request = computeTotalRequests2May(Mbw,Vcpu,N{2},Rcpu);
    request = bubblesort(request);
    for j=size(request,2):-1:1
        i=request{j}(2);
        [M,Rcpu,Attend,RV] =
attendVNsHiddenX(Pfo,M,iType,Mbw{i},Rcpu,Vcpu{i},VN{i},VN2T{i},HiddenMatrix{i},i,Attend,T,RV)
        % Atualização de Índices de Métricas e Reordenação
        [w] = updateNewMetrics(P,M,Nn);
        orderNewPaths;
    end
elseif strcmp (AttendOrder,'LCFUSP')
    request = computeTotalRequests2May(Mbw,Vcpu,N{2},Rcpu);
    request = bubblesort(request);
    for j=1:size(request,2)
        i=request{j}(2);
        [M,Rcpu,Attend,RV] =
attendVNsHiddenX(Pfo,M,iType,Mbw{i},Rcpu,Vcpu{i},VN{i},VN2T{i},HiddenMatrix{i},i,Attend,T,RV)

```

```

        % Atualização de Índices de Métricas e Reordenação
        [w] = updateNewMetrics(P,M,Nn);
        orderNewPaths;
    end
elseif strcmp(AttendOrder,'MCF')
    request = computeTotalRequests(Mbw,Vcpu);
    request = bubblesort(request);
    for j=size(request,2):-1:1
        i=request{j}(2);
        [M,Rcpu,Attend,RV] =
attendVNsHiddenX(Pfo,M,iType,Mbw{i},Rcpu,Vcpu{i},VN{i},VN2T{i},HiddenMatrix{i},i,Attend,T,RV)
        % Atualização de Índices de Métricas e Reordenação
        [w] = updateNewMetrics(P,M,Nn);
        orderNewPaths;
    end
else
    request = computeTotalRequests(Mbw,Vcpu);
    request = bubblesort(request);
    for j=1:size(request,2)
        i=request{j}(2);
        [M,Rcpu,Attend,RV] =
attendVNsHiddenX(Pfo,M,iType,Mbw{i},Rcpu,Vcpu{i},VN{i},VN2T{i},HiddenMatrix{i},i,Attend,T,RV);
        % Atualização de Índices de Métricas e Reordenação
        [w] = updateNewMetrics(P,M,Nn);
        orderNewPaths;
    end
end
end

```


B.10 – Programa *updateNewMetrics*

```

function [w] = updateNewMetrics(P,M,Nn);

%
% [w] = updateNewMetrics(M,P);
% Inicialização variaveis
w={};
for m=1:size(M,2)
    for o=1:Nn
        for d=1:Nn
            if o==d
                w{m}{o}{d}{1} = [];
            else
                for c=1:size(P{o}{d},2)
                    for e=1:(size(P{o}{d}{c},2) - 1)
                        if e==1
                            w{m}{o}{d}{c}=[M{m}(P{o}{d}{c}(e),P{o}{d}{c}(e+1))];
                        else
                            w{m}{o}{d}{c}= cat(2,w{m}{o}{d}{c},M{m}(P{o}{d}{c}(e),P{o}{d}{c}(e+1)));
                        end
                    end
                end
            end
        end
    end
end
end
end

```

B.11 – Programa *orderNewPaths*

```
[Po,So] = orderNewPaths(F,P,w,Nn,word);
```

```
[Pfo,Sfo] = ordenapaths_miguel(F,Po,So,Nn);
```

```
[Nh,Nht] = geraAnexthop(Pfo,Nn); % gera array de next hops.
```

```
[NH] = geraMnexthop(Nh,Nn,1); % gera matriz de next hop (NH) para os melhores caminho no  
% estado inicial da rede.
```

```
NHo=NH; % salva estado inicial da rede.
```

```
[Cd] = calcdtree(NH,Nn); % calcula árvore de destino a partir de NH.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Resultados:
```

```
% Po{m}{o}{d}{:} array de caminhos não ordenados
```

```
% So{m}{o}{d}{:} array das sínteses correspondentes do Po
```

```
% Pfo {1}{o}{d}{:} array de caminhos ordenados
```

```
% Sfo {m}{o}{d}{:} array das sínteses correspondentes do Pfo
```

```
% NH matriz de próximo nó
```

B.12 – Programa *VNrequestX*

```
% VN requests
```

```
%
```

```
% Ajustes para execução no programa
```

```
VN={};
```

```
Vcpu={};
```

```
Mbw={};
```

```
VN2T={};
```

```
readVNs(VN,Vcpu,Mbw,VN2T);
```

B.13 – Programa *attendVNsHiddenX*

```
function [M,Rcpu,Attend,RV] =
attendVNsHiddenX(Pfo,M,iType,Mbw,Rcpu,Vcpu,VN,VN2T,HiddenMatrix,rn,Attend,T,RV)
%
% Atendimento Completo da requisicao virtual
% Convenção para as métricas e síntese:
% iType{1} - Hop
% iType{2} - BW
% iType{3} - CPU
% iType{4} - Delay
% iType{N} - a definir
% Pfo - caminhos ordenados entre origem O e destino D
% M - matriz de métricas
% Mbw - matriz de métricas de BW para a VN request
% Vcpu - vetor de métricas de CPU para a VN request
% Rcpu - vetor de métricas de CPU da matriz de substrato
% Mcpu - matriz de métricas de CPU convertida pelo programa cpu2matrix
% VN - matriz de requisição para a rede virtual
% VN2T - matriz size(VN,2)x 2 onde em cada linha tem-se nós virtual e real
% rn - número da requisição
% T - matriz de substrato
% Matrizes e Vetores para resolução do problema de Hidden Nodes
% HiddenMatrix - Matriz com pesos aplicados em Hidden Nodes
% HiddenUsed - Vetor que indica se um nó físico foi utilizado para cálculo
% como nó intermediário no Hidden Node.
% D{n} - cria vetor linha com o nó origem n e 1's que identificam em sua posição os nós destino.
% C{n} - vetor linha com o valor dos nós virtuais destinos atingidos pelo nó origem n.
% C{n}(i) - nós diretamente conectados ao no nó origem nó n destino cada nó virtual.
% Attend - vetor que contem 1 para sucesso e 0 para fracasso de atendimentos.
% RV - resposta virtual
```

```

% Conversão do Vetor CPU para Matriz de CPU
Mcpu=cpu2matrix(VN,Vcpu);

% Matrizes temporárias de trabalho
TempMcpu=Mcpu;
TempVcpu=Vcpu;
TempRcpu=Rcpu;
TempM=M;

% inicialização de variáveis
n=1;
flag1=1;
flag2=1;
C={};
D={};
sucesso=0;
while (n<= size(VN,2)) && (flag1 ~= 0)
% flag1 é o sucesso
    D{n}=VN(n,:);
    C{n}=find(D{n}==1);
    i=1;
    while (i<=size(C{n},2)) && (flag1 ~=0)
        % Destino n para C{n}(i)
        % VL{n}{C{n}(i)}
        % Pfo{1}{o}{d}
        sucesso=0;
        j=1;
        while (j<= size(Pfo{1}{VN2T(n,2)}{VN2T(C{n}(i),2)},2)) && (sucesso ~=1)
            if (Mbw(n,C{n}(i)) > minimumBW(Pfo{1}{VN2T(n,2)}{VN2T(C{n}(i),2)}{j},TempM{iType{2}})) |
                TempVcpu(n) > TempRcpu(VN2T(n,2)) | TempVcpu(C{n}(i)) > TempRcpu(VN2T(C{n}(i),2))
                j=j+1;

```

else

% Attend VNs Hidden

TempRcpu2=TempRcpu;

% atualização de matrizes temporárias CPU

TempRcpu2(VN2T(n,2))=TempRcpu2(VN2T(n,2)) - TempVcpu(n);

TempRcpu2(VN2T(C{n}(i),2))= TempRcpu2(VN2T(C{n}(i),2)) - TempVcpu(C{n}(i));

% Inicialização de Variável

k=2;

while (k < size(Pfo{1}{VN2T(n,2)}{VN2T(C{n}(i),2)}{j},2)) && flag2~=0

if HiddenMatrix(n,C{n}(i)) >

TempRcpu2(Pfo{1}{VN2T(n,2)}{VN2T(C{n}(i),2)}{j}(k))

flag2=0;

else

TempRcpu2(Pfo{1}{VN2T(n,2)}{VN2T(C{n}(i),2)}{j}(k)) =

TempRcpu2(Pfo{1}{VN2T(n,2)}{VN2T(C{n}(i),2)}{j}(k)) - HiddenMatrix(n,C{n}(i));

flag2=1;

end

k=k+1;

end

if flag2 == 1

sucesso=1;

TempRcpu = TempRcpu2;

% atualização de requisições virtuais CPU

TempVcpu(n)=0;

TempVcpu(C{n}(i))=0;

% atualização da matriz temporária de Banda

TempM{iType{2}}=

updateBW(TempM{iType{2}},Mbw(n,C{n}(i)),(Pfo{1}{VN2T(n,2)}{VN2T(C{n}(i),2)}{j}));

% salvar caminho

RV{rn}{n}{C{n}(i)}=Pfo{1}{VN2T(n,2)}{VN2T(C{n}(i),2)}{j};

```

        arq6 =
        strcat('caminho',virtualAttend_',int2str(rn),'_',int2str(n),'_',int2str(C{n}(i)));
        AuxRV = RV{rn}{n}{C{n}(i)};
        save (arq6,'AuxRV','-ASCII');
    end
    j=j+1;
end
end
if sucesso == 0
    flag1 = 0;
end
i=i+1;
end
n=n+1;
end
arq4 = strcat('caminho',virtualSuccess_',int2str(rn));
arq8='caminho\requestFail';
arq9='caminho\requestSuccess';
if flag1 == 0
    Attend(1,rn)=0;
    fprintf('\nVN request %d, was not attended! FAIL! \n',rn);
    % Indicar o Fracasso em Arquivo virtualSucess_#
    success=0;
    save (arq4,'success','-ASCII');
    % request fail
    save (arq8,'rn','-ASCII','-APPEND');
else
    Rcpu=TempRcpu;
    M{iType{3}}=cpu2matrix(T,Rcpu);
    M{iType{2}}=TempM{iType{2}};

```

```

Attend(1,rn)=1;
fprintf('\nVN request %d, was attended! SUCCESS!\n',rn);
success=1;
save (arq4,'success','-ASCII');
% request success
save(arq9,'rn','-ASCII','-APPEND');
arq5 = strcat('caminhol',substrateMetric1_',int2str(rn));
AuxBW = M{iType{2}};
save (arq5,'AuxBW','-ASCII');
% Metrica Final de CPU da rede de substrato
arq5 = strcat('caminhol',substrateMetric2_',int2str(rn));
save (arq5,'Rcpu','-ASCII');
end

```

Anexo C – Matrizes de Adjacência e Mapeamentos de Nós

Anexo C fornece as matrizes de adjacência de vinte requisições de rede virtual e os mapeamentos de nós virtuais para nós físicos na SN utilizados para cada VNR.

Os dígrafos das requisições de rede virtual (VNRs) são apresentados em matrizes de adjacência de rede virtual, matriz de requisição de largura de banda (BW), tabelas de mapeamento dos nós virtuais para nós físicos na rede de substrato e tabela da necessidade de processamento para cada nó virtual.

C.1 – Mapeamento de Nós para as VNRs do exemplo

As VNRs do exemplo utilizam uma única regra para mapeamento de nós virtuais para nós físicos na rede de substrato (SN). A Tabela 42 mostra a regra utilizada.

Tabela 42 – Regra para mapeamento dos nós virtuais para nós reais (físicos) da rede de substrato.

<i>Virtual Node</i>	<i>Real Node (Substrate Network)</i>
1	$(10-1)+1=10$
2	$(10-2)+1=9$
3	$(10-3)+1=8$
N	$(10-N)+1$

No exemplo, não foi utilizado nenhum algoritmo de mapeamento de nós virtuais para nós físicos na rede de substrato. A regra utilizada para mapeamento de nós não se mostrou a mais eficaz para todas as simulações.

O mapeamento de nós não aproveita bem os recursos da rede de substrato. Deve-se fazer uma investigação, recomendada para trabalhos futuros, de como realizar o mapeamento de nós virtuais com o uso da álgebra de caminhos.

C.2 – VN request 1

A Figura 43 apresenta a matriz de adjacência da rede virtual 1.

$$\text{Virtual Network 1} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Figura 43 – Matriz de adjacência da rede virtual 1

A largura de banda solicitada é exibida na Figura 44.

$$\text{BW1} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 5 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 3 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6 & 0 & 0 & 0 & 7 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 8 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 1 & 0 & 0 & 6 & 4 & 4 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figura 44 – Largura de Banda solicitada na VN1

A Tabela 43 apresenta a requisição de CPU para cada nó virtual.

Tabela 43 – CPU request para VN request 1

<i>Virtual Node</i>	<i>CPU request</i>
1	6
2	4
3	7
4	3
5	5
6	1
7	2
8	2
9	4

C.3 – VN request 2

A Figura 45 apresenta a matriz de adjacência da rede virtual 2.

$$\text{Virtual Network 2} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Figura 45 – Matriz de adjacência da rede virtual 2

A largura de banda solicitada é exibida na Figura 46.

$$\text{BW2} = \begin{bmatrix} 0 & 10 & 7 & 10 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 8 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 3 & 0 \\ 0 & 0 & 9 & 0 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 7 & 0 & 0 \end{bmatrix}$$

Figura 46 – Largura de Banda solicitada na VN2

A Tabela 44 apresenta a requisição de CPU para cada nó virtual.

Tabela 44 – CPU request para VN request 2

<i>Virtual Node</i>	<i>CPU request</i>
1	1
2	9
3	3
4	9
5	10
6	3
7	5

C.4 – VN request 3

A Figura 47 apresenta a matriz de adjacência da rede virtual 3.

$$\text{Virtual Network 3} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Figura 47 – Matriz de adjacência da rede virtual 3.

A largura de banda solicitada é exibida na Figura 48.

BW3 =
$$\begin{bmatrix} 002706446 \\ 000000500 \\ 700060000 \\ 900020000 \\ 007400000 \\ 200000000 \\ 840000009 \\ 1000000000 \\ 600000800 \end{bmatrix}$$

Figura 48 – Largura de Banda solicitada na VN3.

A Tabela 45 apresenta a requisição de CPU para cada nó virtual.

Tabela 45 – CPU request para VN request 3

<i>Virtual Node</i>	<i>CPU request</i>
1	4
2	10
3	4
4	8
5	10
6	8
7	1
8	3
9	8

C.5 – VN request 4

A Figura 49 apresenta a matriz de adjacência da rede virtual 4.

$$\text{Virtual Network 4} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Figura 49 – Matriz de adjacência da rede virtual 4.

A largura de banda solicitada é exibida na Figura 50.

$$\text{BW 4} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 8 & 5 & 6 & 0 \\ 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 & 0 \\ 0 & 9 & 0 & 9 & 0 & 0 & 7 & 0 & 0 \\ 6 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 6 & 0 & 0 & 9 & 0 \\ 7 & 0 & 0 & 0 & 0 & 0 & 8 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \end{bmatrix}$$

Figura 50 – Largura de Banda solicitada na VN4.

A Tabela 46 apresenta a requisição de CPU para cada nó virtual.

Tabela 46 – CPU request para VN request 4.

<i>Virtual Node</i>	<i>CPU request</i>
1	1
2	8
3	4
4	9
5	5
6	8
7	1
8	7
9	4

C.6 – VN request 5

A Figura 51 apresenta a matriz de adjacência da rede virtual 5.

$$\text{Virtual Network 5} = \begin{bmatrix} 0 & 0 & 4 & 5 & 0 & 1 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 7 & 0 & 0 & 0 & 0 & 8 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 8 & 0 & 0 & 0 \\ 0 & 0 & 7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figura 51 – Matriz de adjacência da rede virtual 5.

A largura de banda solicitada é exibida na Figura 52.

BW 5 =
$$\begin{bmatrix} 001101100 \\ 000000010 \\ 100000101 \\ 100000000 \\ 000001000 \\ 100010000 \\ 101000010 \\ 010000100 \\ 001000000 \end{bmatrix}$$

Figura 52 – Largura de Banda solicitada na VN5.

A Tabela 47 apresenta a requisição de CPU para cada nó virtual.

Tabela 47 – CPU request para VN request 5.

<i>Virtual Node</i>	<i>CPU request</i>
1	1
2	10
3	5
4	7
5	1
6	5
7	9
8	1
9	4

C.7 – VN request 6

A Figura 53 apresenta a matriz de adjacência da rede virtual 6.

$$\text{Virtual Network 6} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Figura 53 – Matriz de adjacência da rede virtual 6.

A largura de banda solicitada é exibida na Figura 54.

$$\text{BW6} = \begin{bmatrix} 0 & 0 & 0 & 6 & 0 \\ 0 & 0 & 7 & 4 & 10 \\ 0 & 5 & 0 & 0 & 0 \\ 2 & 4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Figura 54 – Largura de Banda solicitada na VN6.

A Tabela 48 apresenta a requisição de CPU para cada nó virtual.

Tabela 48 – CPU request para VN request 6.

Virtual Node	CPU request
1	2
2	5
3	10
4	5
5	3

C.8 – VN request 7

A Figura 55 apresenta a matriz de adjacência da rede virtual 7.

$$\text{Virtual Network 7} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

Figura 55 – Matriz de adjacência da rede virtual 7.

A largura de banda solicitada é exibida na Figura 56.

$$\text{BW7} = \begin{bmatrix} 0 & 0 & 0 & 2 \\ 0 & 0 & 2 & 5 \\ 0 & 1 & 0 & 0 \\ 2 & 2 & 0 & 0 \end{bmatrix}$$

Figura 56 – Largura de Banda solicitada na VN7.

A Tabela 49 apresenta a requisição de CPU para cada nó virtual.

Tabela 49 – CPU request para VN request 7.

<i>Virtual Node</i>	<i>CPU request</i>
1	9
2	4
3	2
4	7

C.9 – VN request 8

A Figura 57 apresenta a matriz de adjacência da rede virtual 8.

$$\text{Virtual Network 8} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Figura 57 – Matriz de adjacência da rede virtual 8.

A largura de banda solicitada é exibida na Figura 58.

$$\text{BW 8} = \begin{bmatrix} 0 & 7 & 0 \\ 4 & 0 & 2 \\ 0 & 1 & 0 \end{bmatrix}$$

Figura 58 – Largura de Banda solicitada na VN8.

A Tabela 50 apresenta a requisição de CPU para cada nó virtual.

Tabela 50 – CPU request para VN request 8.

<i>Virtual Node</i>	<i>CPU request</i>
1	1
2	10
3	7

C.10 – VN request 9

A Figura 59 apresenta a matriz de adjacência da rede virtual 9.

$$\text{Virtual Network 9} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Figura 59 – Matriz de adjacência da rede virtual 9.

A largura de banda solicitada é exibida na Figura 60.

$$\text{BW 9} = \begin{bmatrix} 0 & 6 & 0 & 0 & 3 & 0 & 0 & 0 & 3 \\ 6 & 0 & 0 & 0 & 0 & 0 & 9 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 6 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 9 \\ 0 & 4 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 9 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \end{bmatrix}$$

Figura 60 – Largura de Banda solicitada na VN9

A Tabela 51 apresenta a requisição de CPU para cada nó virtual.

Tabela 51 – CPU request para VN request 9.

<i>Virtual Node</i>	<i>CPU request</i>
1	10
2	7
3	10
4	10
5	4
6	6
7	1
8	7
9	2

C.11 – VN request 10

A Figura 61 apresenta a matriz de adjacência da rede virtual 10.

$$\text{Virtual Network 10} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Figura 61 – Matriz de adjacência da rede virtual 10.

A largura de banda solicitada é exibida na Figura 62.

$$\text{BW 10} = \begin{bmatrix} 0 & 9 \\ 3 & 0 \end{bmatrix}$$

Figura 62 – Largura de Banda solicitada na VN10.

A Tabela 52 apresenta a requisição de CPU para cada nó virtual.

Tabela 52 – CPU request para VN request 10.

<i>Virtual Node</i>	<i>CPU request</i>
1	3
2	4

C.12 – VN request 11

A Figura 63 apresenta a matriz de adjacência da rede virtual 11.

$$\text{Virtual Network 11} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Figura 63 – Matriz de adjacência da rede virtual 11.

A largura de banda solicitada é exibida na Figura 64.

$$\text{BW 11} = \begin{bmatrix} 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 9 \\ 5 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 4 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 7 & 5 & 0 & 9 & 0 & 1 & 0 \end{bmatrix}$$

Figura 64 – Largura de Banda solicitada na VN11.

A Tabela 53 apresenta a requisição de CPU para cada nó virtual.

Tabela 53 – CPU request para VN request 11.

<i>Virtual Node</i>	<i>CPU request</i>
1	3
2	8
3	2
4	3
5	4
6	10
7	5
8	1

C.13 – VN request 12

A Figura 65 apresenta a matriz de adjacência da rede virtual 12.

$$\text{Virtual Network 12} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Figura 65 – Matriz de adjacência da rede virtual 12.

A largura de banda solicitada é exibida na Figura 66.

$$\text{BW 12} = \begin{bmatrix} 0 & 3 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 6 \\ 0 & 9 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Figura 66 – Largura de Banda solicitada na VN12.

A Tabela 54 apresenta a requisição de CPU para cada nó virtual.

Tabela 54 – CPU request para VN request 12.

<i>Virtual Node</i>	<i>CPU request</i>
1	10
2	9
3	4
4	7
5	2
6	1

C.14 – VN request 13

A Figura 67 apresenta a matriz de adjacência da rede virtual 13.

$$\text{Virtual Network 13} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Figura 67 – Matriz de adjacência da rede virtual 13.

A largura de banda solicitada é exibida na Figura 68.

$$\text{BW 13} = \begin{bmatrix} 0 & 0 & 3 & 0 \\ 0 & 0 & 10 & 0 \\ 4 & 9 & 0 & 1 \\ 0 & 0 & 10 & 0 \end{bmatrix}$$

Figura 68 – Largura de Banda solicitada na VN13

A Tabela 55 apresenta a requisição de CPU para cada nó virtual.

Tabela 55 – CPU request para VN request 13

<i>Virtual Node</i>	<i>CPU request</i>
1	1
2	5
3	5
4	7

C.15 – VN request 14

A Figura 69 apresenta a matriz de adjacência da rede virtual 14.

$$\text{Virtual Network 14} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Figura 69 – Matriz de adjacência da rede virtual 14.

A largura de banda solicitada é exibida na Figura 70.

$$\text{BW 14} = \begin{bmatrix} 0 & 0 & 10 & 0 & 0 & 5 \\ 0 & 0 & 10 & 0 & 1 & 0 \\ 6 & 9 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 8 & 9 \\ 0 & 5 & 0 & 4 & 0 & 0 \\ 10 & 0 & 0 & 6 & 0 & 0 \end{bmatrix}$$

Figura 70 – Largura de Banda solicitada na VN14

A Tabela 56 apresenta a requisição de CPU para cada nó virtual.

Tabela 56 – CPU request para VN request 14.

<i>Virtual Node</i>	<i>CPU request</i>
1	8
2	8
3	7
4	4
5	10
6	5

C.16 – VN request 15

A Figura 71 apresenta a matriz de adjacência da rede virtual 15.

$$\text{Virtual Network 15} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Figura 71 – Matriz de adjacência da rede virtual 15.

A largura de banda solicitada é exibida na Figura 72.

$$\text{BW 15} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 9 & 0 & 4 & 0 & 0 \\ 0 & 9 & 0 & 8 & 7 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Figura 72 – Largura de Banda solicitada na VN15.

A Tabela 57 apresenta a requisição de CPU para cada nó virtual.

Tabela 57 – CPU request para VN request 15.

<i>Virtual Node</i>	<i>CPU request</i>
1	7
2	9
3	3
4	3
5	6

C.17 – VN request 16

A Figura 73 apresenta a matriz de adjacência da rede virtual 16.

$$\text{Virtual Network 16} = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Figura 73 – Matriz de adjacência da rede virtual 16.

A largura de banda solicitada é exibida na Figura 74.

$$\text{BW 16} = \begin{bmatrix} 0 & 10 & 30 & 2 \\ 30 & 0 & 0 & 0 & 0 \\ 100 & 0 & 90 & 0 \\ 00 & 100 & 9 & 0 \\ 800 & 90 & 0 & 0 \end{bmatrix}$$

Figura 74 – Largura de Banda solicitada na VN16.

A Tabela 58 apresenta a requisição de CPU para cada nó virtual.

Tabela 58 – CPU request para VN request 16.

<i>Virtual Node</i>	<i>CPU request</i>
1	5
2	2
3	2
4	3
5	1

C.18 – VN request 17

A Figura 75 apresenta a matriz de adjacência da rede virtual 17.

$$\text{Virtual Network 17} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Figura 75 – Matriz de adjacência da rede virtual 17.

A largura de banda solicitada é exibida na Figura 76.

$$\text{BW 17} = \begin{bmatrix} 0 & 6 \\ 9 & 0 \end{bmatrix}$$

Figura 76 – Largura de Banda solicitada na VN17.

A Tabela 59 apresenta a requisição de CPU para cada nó virtual.

Tabela 59 – CPU request para VN request 17.

Virtual Node	CPU request
1	6
2	5

C.19 – VN request 18

A Figura 77 apresenta a matriz de adjacência da rede virtual 18.

$$\text{Virtual Network 18} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Figura 77 – Matriz de adjacência da rede virtual 18.

A largura de banda solicitada é exibida na Figura 78.

$$\text{BW 18} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 7 & 4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 8 \\ 0 & 2 & 0 & 0 & 5 & 0 & 0 \\ 7 & 0 & 0 & 8 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figura 78 – Largura de Banda solicitada na VN18.

A Tabela 60 apresenta a requisição de CPU para cada nó virtual.

Tabela 60 – CPU request para VN request 18.

<i>Virtual Node</i>	<i>CPU request</i>
1	5
2	1
3	10
4	10
5	4
6	8

C.20 – VN request 19

A Figura 79 apresenta a matriz de adjacência da rede virtual 19.

$$\text{Virtual Network 19} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figura 79 – Matriz de adjacência da rede virtual 19.

A largura de banda solicitada é exibida na Figura 80.

BW 19 =
$$\begin{bmatrix} 010000800 \\ 908000000 \\ 0801050000 \\ 006000100 \\ 003000409 \\ 000000600 \\ 900882050 \\ 000000700 \\ 000010000 \end{bmatrix}$$

Figura 80 – Largura de Banda solicitada na VN19.

A Tabela 61 apresenta a requisição de CPU para cada nó virtual.

Tabela 61 – CPU request para VN request 19.

Virtual Node	CPU request
1	10
2	7
3	9
4	7
5	4
6	4
7	5
8	10
9	1

C.21 – VN request 20

A Figura 81 apresenta a matriz de adjacência da rede virtual 20.

$$\text{Virtual Network 20} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Figura 81 – Matriz de adjacência da rede virtual 20.

A largura de banda solicitada é exibida na Figura 82.

$$\text{BW 20} = \begin{bmatrix} 0 & 7 & 0 & 0 \\ 7 & 0 & 9 & 0 \\ 0 & 9 & 0 & 7 \\ 0 & 0 & 10 & 0 \end{bmatrix}$$

Figura 82 – Largura de Banda solicitada na VN20.

A Tabela 62 apresenta a requisição de CPU para cada nó virtual.

Tabela 62– CPU request para VN request 20.

<i>Virtual Node</i>	<i>CPU request</i>
1	7
2	4
3	8
4	9

Anexo D – Resultados Individuais para cada cenário

O Anexo D apresenta os resultados individuais para cada cenário que foi utilizado nas simulações.

D.1 – Cenários com Carga (ρ) 0,2

Substrate Network: 20 nodes

Virtual Requests - virtual networks : 10 nodes

Quantidade de Virtual Requests: 10 requisições

Cenários: 5

load: 0.2

Cenários: 0 a 4

VNR Acceptance Ratio

Cenário	PathsAlgebraAR policy M1	PathsAlgebraAR policy M2	PA Coordinated policy M1	PA Coordinated policy M2	GARSP	DViNESP
0	100,00	100,00	70,00	80,00	100,00	80,00
1	100,00	100,00	70,00	70,00	100,00	60,00
2	100,00	100,00	80,00	60,00	100,00	60,00
3	100,00	100,00	80,00	30,00	80,00	70,00
4	100,00	100,00	80,00	80,00	90,00	70,00
Média	100,00	100,00	76,00	64,00	94,00	68,00

Mapped Revenue Ratio

Cenário	PathsAlgebraAR policy M1	PathsAlgebraAR policy M2	PA Coordinated policy M1	PA Coordinated policy M2	GARSP	DViNESP
0	100,00	100,00	74,87	84,72	100,00	85,72
1	100,00	100,00	70,69	70,69	100,00	61,39
2	100,00	100,00	80,33	61,99	100,00	59,95
3	100,00	100,00	80,55	32,98	83,72	70,86
4	100,00	100,00	79,16	79,15	91,07	70,22
Média	100,00	100,00	77,12	65,91	94,96	69,63

D.2 – Cenários com Carga (ρ) 0,3

Substrate Network: 20 nodes

Virtual Requests - virtual networks : 10 nodes

Quantidade de Virtual Requests: 10 requisições

Cenários: 5

load: 0.3

Cenários: 0 a 4

VNR Acceptance Ratio

Cenário	PathsAlgebraAR policy M1	PathsAlgebraAR policy M2	PA Coordinated policy M1	PA Coordinated policy M2	GARSP	DViNESP
0	90,00	90,00	70,00	70,00	70,00	70,00
1	70,00	70,00	60,00	50,00	70,00	60,00
2	70,00	90,00	70,00	70,00	60,00	50,00
3	60,00	70,00	70,00	60,00	80,00	50,00
4	100,00	90,00	80,00	80,00	80,00	70,00
Média	78,00	82,00	70,00	66,00	72,00	60,00

Mapped Revenue Ratio

Cenário	PathsAlgebraAR policy M1	PathsAlgebraAR policy M2	PA Coordinated policy M1	PA Coordinated policy M2	GARSP	DViNESP
0	91,30	91,00	73,52	73,52	73,73	73,22
1	73,22	71,75	58,77	49,13	73,22	61,69
2	74,44	92,89	70,06	70,06	64,93	53,53
3	64,23	73,37	70,53	59,89	81,26	51,34
4	100,00	92,38	80,45	80,45	83,69	71,76
Média	80,64	84,28	70,67	66,61	75,37	62,31

D.3 – Cenários com Carga (ρ) 0,4

Substrate Network: 20 nodes

Virtual Requests - virtual networks : 10 nodes

Quantidade de Virtual Requests: 10 requisições

Cenários: 5

load: 0.4

Cenários: 0 a 4

VNR Acceptance Ratio

Cenário	PathsAlgebraAR policy M1	PathsAlgebraAR policy M2	PA Coordinated policy M1	PA Coordinated policy M2	GARSP	DViNESP
0	60,00	50,00	50,00	50,00	50,00	40,00
1	50,00	60,00	40,00	40,00	50,00	40,00
2	50,00	50,00	50,00	50,00	40,00	50,00
3	40,00	50,00	50,00	50,00	40,00	50,00
4	80,00	60,00	60,00	50,00	70,00	60,00
Média	56,00	54,00	50,00	48,00	50,00	48,00

Mapped Revenue Ratio

Cenário	PathsAlgebraAR policy M1	PathsAlgebraAR policy M2	PA Coordinated policy M1	PA Coordinated policy M2	GARSP	DViNESP
0	62,35	54,39	52,55	51,20	54,02	42,87
1	55,35	64,32	42,84	42,84	56,11	42,84
2	51,95	54,15	49,53	49,53	41,85	51,21
3	44,85	54,80	53,15	53,78	44,85	53,15
4	81,06	62,62	58,36	49,32	71,05	59,43
Média	59,11	58,06	51,29	49,33	53,58	49,90

D.4 – Cenários com Carga (ρ) 0,5

Substrate Network: 20 nodes

Virtual Requests - virtual networks : 10 nodes

Quantidade de Virtual Requests: 10 requisições

Cenários: 5

load: 0.5

Cenários: 0 a 4

VNR Acceptance Ratio

Cenário	PathsAlgebraAR policy M1	PathsAlgebraAR policy M2	PA Coordinated policy M1	PA Coordinated policy M2	GARSP	DViNESP
0	50,00	40,00	40,00	30,00	40,00	40,00
1	60,00	40,00	50,00	60,00	50,00	50,00
2	40,00	50,00	40,00	40,00	30,00	30,00
3	50,00	50,00	70,00	60,00	50,00	60,00
4	20,00	20,00	20,00	30,00	20,00	20,00
Média	44,00	40,00	44,00	44,00	38,00	40,00

Mapped Revenue Ratio

Cenário	PathsAlgebraAR policy M1	PathsAlgebraAR policy M2	PA Coordinated policy M1	PA Coordinated policy M2	GARSP	DViNESP
0	52,79	44,74	36,01	27,77	42,66	36,01
1	63,76	45,57	55,38	64,37	54,82	55,38
2	45,36	55,57	36,99	36,99	34,15	29,04
3	53,16	53,84	71,01	61,29	53,00	61,29
4	22,10	21,61	23,10	31,79	21,19	23,10
Média	47,43	44,27	44,50	44,44	41,16	40,96

D.5 – Cenários com Carga (ρ) 0,6

Substrate Network: 20 nodes

Virtual Requests - virtual networks : 10 nodes

Quantidade de Virtual Requests: 10 requisições

Cenários: 5

load: 0.6

Cenários: 0 a 4

VNR Acceptance Ratio

Cenário	PathsAlgebraAR policy M1	PathsAlgebraAR policy M2	PA Coordinated policy M1	PA Coordinated policy M2	GARSP	DViNESP
0	40,00	40,00	40,00	30,00	30,00	30,00
1	30,00	30,00	20,00	20,00	30,00	30,00
2	30,00	30,00	30,00	40,00	30,00	30,00
3	20,00	30,00	40,00	50,00	20,00	40,00
4	50,00	60,00	60,00	60,00	50,00	50,00
Média	34,00	38,00	38,00	40,00	32,00	36,00

Mapped Revenue Ratio

Cenário	PathsAlgebraAR policy M1	PathsAlgebraAR policy M2	PA Coordinated policy M1	PA Coordinated policy M2	GARSP	DViNESP
0	43,36	45,25	42,97	33,76	33,06	33,76
1	33,34	33,53	23,16	23,16	32,98	33,22
2	33,26	33,33	33,28	38,14	30,60	30,57
3	23,11	30,48	45,96	53,14	22,55	38,86
4	53,19	63,22	62,09	63,22	53,51	53,19
Média	37,25	41,16	41,49	42,28	34,54	37,92

D.6 – Cenários com Carga (ρ) 0,7

Substrate Network: 20 nodes

Virtual Requests - virtual networks : 10 nodes

Quantidade de Virtual Requests: 10 requisições

Cenários: 5

load: 0.7

Cenários: 0 a 4

VNR Acceptance Ratio

Cenário	PathsAlgebraAR policy M1	PathsAlgebraAR policy M2	PA Coordinated policy M1	PA Coordinated policy M2	GARSP	DViNESP
0	40,00	40,00	40,00	40,00	30,00	50,00
1	20,00	30,00	30,00	20,00	30,00	30,00
2	20,00	20,00	30,00	30,00	20,00	30,00
3	40,00	30,00	30,00	30,00	20,00	20,00
4	20,00	20,00	30,00	20,00	20,00	20,00
Média	28,00	28,00	32,00	28,00	24,00	30,00

Mapped Revenue Ratio

Cenário	PathsAlgebraAR policy M1	PathsAlgebraAR policy M2	PA Coordinated policy M1	PA Coordinated policy M2	GARSP	DViNESP
0	42,60	42,27	41,44	43,28	31,55	52,11
1	22,87	30,90	32,17	21,09	33,06	33,47
2	23,12	23,89	27,75	28,70	23,89	26,59
3	40,11	35,78	36,12	36,12	23,43	25,08
4	21,38	21,89	30,87	23,05	21,11	23,89
Média	30,02	30,95	33,67	30,45	26,61	32,23