UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Samuel Moreira Abreu Araujo

Algoritmos para o Problema de Posicionamento e Encadeamento de Funções Virtuais de Rede

Belo Horizonte
2023

Samuel Moreira Abreu Araujo

Algoritmos para o Problema de Posicionamento e Encadeamento de Funções Virtuais de Rede

Versão Final

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais, como requisito parcial à obtenção do título de Doutor em Ciência da Computação.

Orientador: Geraldo Robson Mateus
Coorientadora: Fernanda S. Hojo de Souza

Belo Horizonte
2023

© 2023, Samuel Moreira Abreu Araújo.
Todos os direitos reservados

Araújo, Samuel Moreira Abreu.

A663a Algoritmos para o problema de posicionamento e encadeamento de funções virtuais de rede [recurso eletrônico] / Samuel Moreira Abreu Araújo –2023.
1 recurso online (208 f. il, color.) : pdf.

Orientador: Geraldo Robson Mateus.

Tese (Doutorado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciências da Computação.
Referências: f.146-160

1. Computação – Teses. 2. Virtualização das funções de rede – Teses. 3. Otimização matemática -Teses. 4. Aprendizado do computador – Teses. I. Mateus, Geraldo Robson. II. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Computação. III.Título.

CDU 519.6*22(043)

Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa
CRB 6/1510 Universidade Federal de Minas Gerais - ICEx

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Algoritmos para o Problema de posicionamento e encadeamento de funções virtuais de rede

SAMUEL MOREIRA ABREU ARAÚJO

Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. GERALDO ROBSON MATEUS - Orientador
Departamento de Ciência da Computação - UFMG

PROFA. FERNANDA SUMIKA HOJO DE SOUZA - Coorientadora
Departamento de Computação - UFOP

PROF. LUIZ FERNANDO BITTENCOURT
Departamento de Sistemas de Computação - Unicamp

PROF. RAFAEL AUGUSTO DE MELO
Departamento de Ciência da Computação - UFBA

PROFA. MICHELE NOGUEIRA LIMA
Departamento de Ciência da Computação - UFMG

PROF. THIAGO FERREIRA DE NORONHA
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 9 de fevereiro de 2023.

Resumo

Impulsionada pelo crescimento da internet, é gerada uma nova onda de aplicações e dados que podem ser acessados de qualquer lugar. Nesse meio, as redes virtuais emergem como tecnologias inovadoras, propiciando a implementação de novas funcionalidades de rede, com um baixo custo, e facilitando o gerenciamento de recursos. O problema abordado nesta tese é relativo ao posicionamento e encadeamento de funções virtuais de rede em um ambiente *online* em relação à chegada de requisições. Por se tratar de um problema combinatório NP-difícil, enquanto o número de componentes da rede cresce linearmente, o processamento computacional e o tempo de execução aumentam exponencialmente. Em um ambiente *online*, o provedor precisa lidar de maneira rápida com as requisições à medida que elas chegam, em vez de atender a um conjunto de requisições conhecidas de uma só vez (*offline*). Tais características aumentam a dificuldade de sua resolução em virtude do grande número de componentes processados. O objetivo deste trabalho é definir modelos, apresentá-los, bem como discutir e resolver o problema de posicionamento e encadeamento de funções virtuais de rede com diferentes algoritmos. Em uma linha clássica de otimização, são propostos um algoritmo exato, baseado em Programação Linear Inteira; e outro heurístico. São propostos mecanismos de hibridização entre técnicas de Aprendizado de Máquina e algoritmos clássicos de otimização. São aplicadas técnicas de clusterização para se reduzir o espaço de soluções e, consequentemente, o tempo de resolução do problema. Complementarmente aos algoritmos propostos, alguns trabalhos da literatura são pesquisados para identificar serviços de rede comumente usados e utilizados nas experimentações. Aplicando um algoritmo exato, percebe-se que reotimizar todo o modelo é custoso computacionalmente e inviável. Experimentos com a heurística geraram resultados promissores, como altos lucros e taxa de aceitação, com um baixo tempo de execução. Experimentos com as técnicas de clusterização confirmam a hipótese de que um espaço de soluções menor diminui consideravelmente o tempo de convergência do algoritmo exato proposto.

Palavras-chave: Virtualização das Funções da Rede. Otimização. Posicionamento. Encadeamento. Aprendizado de Máquina. Classes de Serviço.

Abstract

The expansion of the internet has resulted in the creation of a new wave of applications and data that can be accessed from anywhere. In this scenario, virtual networks are emerging as a disruptive technology, enabling the implementation of new network functions at low cost and facilitating the management of resources. The research problem approached in this thesis is concerned with Virtual Networks Functions Placement and Chaining in an online environment in relation to the arrival of the requests. Because it is a NP-hard problem, while the quantity of network components grows linearly, the computer processing and execution time for computing tasks have an exponential growth. In an online environment, the provider needs to quickly handle requests as they come in, rather than serving a set of known requests all at once (offline). These features increase its difficulty in solving issues because of the large amount of components being processed. This research is aimed at defining models, presenting them, as well as to analyze and solve the problem referred to virtual network functions placement and chaining through different types of algorithms. In a traditional optimization line, an algorithm based on Integer Linear Programming and another heuristic are proposed. Hybridization between Machine Learning techniques and classical optimization algorithms are proposed. It has been carried out clustering methods to decrease the solution space and, therefore, the time complexity. In terms of complementing the proposed algorithms, some papers from the literature have been reviewed to identify network services that are commonly used and considered in experiments. Regarding the optimal treatment, it has been observed that reoptimizing the entire model is computationally expensive and unfeasible. Experiments using the heuristic showed promising results, with high profits and acceptance rates despite a higher time complexity and running time. Experiments using cluster analysis confirmed the hypothesis that a smaller solution space can significantly reduce the runtime of the exact algorithm proposed.

Keywords: Network Functions Virtualization. Optimization. Placement. Chaining. Machine Learning. Service Classes.

Lista de Ilustrações

1.1 Exemplo de um fluxo *online* de SFCs 28

3.1 Posicionamento de duas instâncias de VNFs 51
3.2 Compartilhamento de VNFs 51
3.3 Fluxo de dados de uma SFC 52
3.4 Exemplo do compartilhamento de recursos entre SFCs 53
3.5 Mapeamento de uma SFC 53

4.1 Distribuição topológica dos nós e arcos físicos da rede *CompuServe* 63
4.2 Lucro e receita final (algoritmo exato) 70
4.3 Tempo médio de processamento por SFC (algoritmo exato) 71
4.4 Taxa de aceitação (algoritmo exato) 72
4.5 Espalhamento dos nós virtuais (algoritmo exato) 73
4.6 Espalhamento dos arcos virtuais (algoritmo exato) 74
4.7 Atraso fim a fim (algoritmo exato) 75

5.1 Cálculo da Função $h(x)$ para diferentes valores de α 83
5.2 Movimento de vizinhança N_1 84
5.3 Movimento de vizinhança N_2^{α} 85
5.4 Distribuição topológica dos nós e arcos físicos da rede *Cogent* 86
5.5 Lucro final no cenário denso (heurística) 89
5.6 Tempo de execução (heurística) 90
5.7 Taxa de aceitação no cenário denso (heurística) 92
5.8 Espalhamento dos nós virtuais (heurística) 93
5.9 Espalhamento dos arcos virtuais (heurística) 94
5.10 Atraso fim a fim (heurística) 95

6.1 Modelo operacional da Abordagem de Clusterização por Localização Espacial . 99
6.2 Visualização da clusterização com o algoritmo K-*Means* 102
6.3 Variação no número de nós dos *clusters* com o algoritmo K-*Means* 103
6.4 Visualização da clusterização com o algoritmo *Hierarchical Clustering* 104
6.5 Variação no número de nós dos *clusters* com o algoritmo *Hierarchical Clustering* 104
6.6 Lucro final (algoritmo de Clusterização Espacial) 106
6.7 Tempo de execução (algoritmo de Clusterização Espacial) 107
6.8 Taxa de aceitação (algoritmo de Clusterização Espacial) 108

6.9 Espalhamento dos nós virtuais (algoritmo de Clusterização Espacial) 108
6.10 Espalhamento dos arcos virtuais (algoritmo de Clusterização Espacial) 109
6.11 Atraso fim a fim (algoritmo de Clusterização Espacial) 110

7.1 Modelo operacional da Abordagem de Clusterização por Consumo de Recursos 113
7.3 Mapa de calor da correlação entre as características analisadas 121
7.4 Variação no número de nós dos *clusters* com o algoritmo K-*Means* 123
7.5 Lucro final (algoritmo de Clusterização por Consumo de Recursos) 125
7.6 Tempo de execução (algoritmo de Clusterização por Consumo de Recursos) . . 126
7.7 Taxa de aceitação (algoritmo de Clusterização por Consumo de Recursos) . . . 127
7.8 Espalhamento dos nós virtuais (algoritmo de Clusterização por Consumo de Recursos) 127
7.9 Espalhamento dos arcos virtuais (algoritmo de Clusterização por Consumo de Recursos) 128
7.10 Atraso fim a fim (algoritmo de Clusterização por Consumo de Recursos) . . . 129
7.11 Monitoramento da variação dos nós de origem e destino 132
7.12 Monitoramento da variação do atraso fim a fim 132
7.13 Monitoramento da variação da banda residual do SN 133
7.14 Impacto do retreinamento na taxa de aceitação 133

A.1 Transformações do VNF-PC para os 21 problemas NP-completos de Karp (1972) 162

B.1 Modelo de Negócio das redes NVF 164

D.1 Exemplos de ótimo local e global 169

E.1 Lucro final no cenário esparso (heurística) 174
E.2 Tempo de execução no cenário esparso (heurística) 175
E.3 Taxa de aceitação no cenário esparso (heurística) 176

F.2 *Elbow method* gerado pela clusterização dos nós da Figura F.1. 182
F.3 *Silhouette Coefficient* gerado pela clusterização dos nós da Figura F.1 183
F.4 Dendrograma gerado pela clusterização dos nós da Figura F.1 184

G.1 WCSS (*Elbow Method*) da clusterização com o algoritmo *K-Means* 189
G.2 *Silhouette Coefficient* da clusterização com o algoritmo *K-Means* 190
G.3 Dendrograma da clusterização com o algoritmo *Hierarchical Clustering* 190
G.4 Visualização da clusterização realizadas com o algoritmo DBSCAN 192
G.5 Variação no número de nós dos *clusters*, gerado com o DBSCAN 192

I.1 Variação das caraterísticas do *dataset* (parte I) 198
I.2 Variação das caraterísticas do *dataset* (parte II) 199

I.3 WCSS (*Elbow Method*) da clusterização com o algoritmo *K-Means* 200
I.4 *Silhouette Coefficient* da clusterização com o algoritmo *K-Means* 200
I.5 Impacto do retreinamento na taxa de aceitação, computado até $50000t$ 205
I.6 Impacto do retreinamento no lucro, computado até $50000t$ 205
I.7 Impacto do retreinamento na taxa de aceitação, computado até $80000t$ 206
I.8 Impacto do retreinamento no lucro, computado até $80000t$ 207
I.9 Impacto do retreinamento na taxa de aceitação, computado até $110000t$ 208
I.10 Impacto do retreinamento no lucro, computado até $110000t$ 208

Lista de Tabelas

1.1 Diferenças entre redes legadas e baseadas em NFV . . . 22
1.2 Demandas de QoS de diferentes aplicações . . . 25

2.1 Classificação dos artigos *surveys* . . . 38
2.2 Trabalhos abordados com otimização exata e heurística . . . 44
2.3 Comparação desta tese com outros trabalhos da literatura . . . 45
2.4 Trabalhos abordados com Aprendizado de Máquina . . . 48

3.1 Glossário de Conjuntos, Funções, Parâmetros e Variáveis . . . 55

4.1 Configurações dos servidores físicos adotados . . . 64
4.2 Funções de rede utilizadas . . . 65
4.3 Configurações de VMs de uso geral da Amazon EC2 . . . 65
4.4 NSs que são muito sensíveis à métricas de QoS . . . 66
4.5 NSs que são sensíveis à métricas de QoS . . . 66
4.6 NSs que são menos sensíveis à métricas de QoS . . . 67
4.7 Características das classes de serviços . . . 67

7.1 Descrição das características analisadas . . . 119
7.2 Impacto da variação do parâmetro Υ (resumo) . . . 124

8.1 Atraso fim a fim com origem em Belo Horizonte para diversos destinos . . . 143

C.1 Resumo do Gráfico 4.5a . . . 167
C.2 Resumo do Gráfico 4.5b . . . 167
C.3 Resumo do Gráfico 4.5c . . . 167
C.4 Resumo do Gráfico 4.5d . . . 167
C.5 Resumo do Gráfico 4.6a . . . 167
C.6 Resumo do Gráfico 4.6b . . . 167
C.7 Resumo do Gráfico 4.6c . . . 167
C.8 Resumo do Gráfico 4.6d . . . 167
C.9 Resumo do Gráfico 4.7a . . . 168
C.10 Resumo do Gráfico 4.7b . . . 168
C.11 Resumo do Gráfico 4.7c . . . 168
C.12 Resumo do Gráfico 4.7d . . . 168

E.1 Relação percentual de SFCs mapeadas . . . 177

E.2 Resumo do Gráfico 5.8a . . . 178
E.3 Resumo do Gráfico 5.8b . . . 178
E.4 Resumo do Gráfico 5.9a . . . 178
E.5 Resumo do Gráfico 5.9b . . . 178
E.6 Resumo do Gráfico 5.10a . . . 179
E.7 Resumo do Gráfico 5.10b . . . 179

F.1 Comparação entre os diferentes Algoritmos de clusterização . . . 188

H.1 Figura 6.8a estratificada em classes de serviços . . . 195
H.2 Figura 6.8b estratificada em classes de serviços . . . 195
H.3 Resumo do Gráfico 6.9 . . . 196
H.4 Resumo do Gráfico 6.10 . . . 196
H.5 Resumo do Gráfico 6.11 . . . 197

I.1 Impacto da variação do parâmetro Υ (completo) . . . 201
I.2 Taxa de aceitação estratificada em classes de serviços . . . 202
I.3 Resumo do Gráfico 7.8 . . . 202
I.4 Resumo do Gráfico 7.9 . . . 203
I.5 Resumo do Gráfico 7.10 . . . 203
I.6 Dados dos mapeamentos das SFCs de $0t$ a $50000t$ . . . 205
I.7 Dados dos mapeamentos das SFCs de $0t$ a $80000t$ . . . 206
I.8 Dados dos mapeamentos das SFCs de $0t$ a $110000t$ . . . 208

Lista de Algoritmos

1 SGBL 79
2 Construtivo 80
3 RVNS 86
4 VNF-PC por Localização Espacial 100
5 Mapeamento por Localização Espacial 100
6 VNF-PC por Consumo de Recursos 115
7 Mapeamento por Consumo de Recursos 116
8 Guloso 170
9 Construtivo Semiguloso 170
10 Multipartida 171
11 VNS 172
12 K-*Means* 181
13 *Hierarchical Clustering* 184
14 DBSCAN 187

Lista de Abreviaturas e Siglas

ADA, *Ada-boost* ou *Adaptive Boosting*
AM, Aprendizado de Máquina
B, *Byte*
BFS, *Breadth-First Search*
BL, *Busca Local*
CAPEX, *CAPital EXpediture*
CC, *Continuous Computing*
CCR, Clusterização por Consumo de Recursos
CDNs, *Content Delivery Networks*
CLE, Clusterização por Localização Espacial
CPP, *Controller Placement Problem*
DBSCAN, *Density-Based Spatial Clustering of Applications with Noise*
DPI, *Deep Packet Inspection*
DT, *Decision Trees*
EC, *Edge Computing*
FI, *Future Internet*
FLP, *Facility Location Problem*
FM, *Flow Monitor*
FW, *Firewall*
GB, *Gigabyte*
GBT, *Gradient Boosted Trees*
GRASP, *Greedy Randomized Adaptive Search Procedure*
IA, Inteligência Artificial
IaaS, *Infrastructure as a Service*
IC, Intervalo de Confiança
IDS, *Intrusion Detection System*
ILP, *Integer Linear Programming*
ILPrk , Algoritmo de Clusterização por Consumo de Recursos
ILPsk , Algoritmo de Clusterização por Localização Espacial
ILS, *Iterated Local Search*
ILP^{i}, ILP com redimensionamento das instâncias de VNFs
ILP^{ic}, ILP^{i} com reroteamento dos enlaces
ILP^{icp}, ILP^{ic} com realocação das instâncias de VNFs

InPs, *Infrastructure Providers*
IoT, *Internet of Things*
IPS, *Intrusion Prevention System*
ISGs, *Industry Specification Groups*
KL, *Kullback-leibler*
KNN, *K-Nearest Neighbours*
LC, Lista de Candidatos
LRC, Lista Restrita de Candidatos
LRP, *Location-Routing Problems*
MB, *Megabytes*
MEC, *Mobile Edge Computing*
MIP, *Mixed-Integer Program*
MIQCP, *Mixed Integer Quadratically Constrained Programs*
MLP, *Multilayer Perceptron*
N-PoPs, *Points of Presence*
NAT, *Network Address Translation*
NB, *Naive Bayes*
NF, *Network Function*
NFV, *Network Function Virtualization*
NFV-RA, *NFV Resource Allocation*
NP, *Nondeterministic Polynomial time*
NS, *Network Service*
OPEX, *OPerational EXpediture*
QoS, *Quality of Service*
RA, Realidade Aumentada
RF, *Random Forest*
RVNS, *Reduced VNS*
SDN, *Software Defined Network*
SFC, *Service Function Chain*
SG, Algoritmo heurístico semiguloso
SGBL, Algoritmo heurístico semiguloso com busca local
SGBLILP, Algoritmo heurístico semigulosa com a busca local e Algoritmo exato
SN, *Substrate Network*
SOM, *Self-Organizing Map*
SPs, *Service Providers*
SVM, *Support Vector Machine*
VND, *Variable Neighborhood Descent*
VNE, *Virtual Network Embedding*
VNF, *Virtual Network Function*

VNF-CC, *Chain Composition*
VNF-FG , *VNF Forwarding Graph*
VNF-OP, *VNF Orchestration Problem*
VNF-P, *VNF Placement*
VNF-PC, *VNF Placement and Chaining*
VNS, *Variable Neighborhood Search*
VoIP, *Voice over Internet Protocol*
VPN, *Virtual Private Networks*
VRP, *Vehicle Routing Problem*
WAN, *Wide Area Network*
WCSS, *Within-Cluster-Sum-of-Squares*
XGBoost, *eXtreme Gradient Boosting*
5G, Rede de Quinta Geração
6G, Rede de Sexta Geração

Sumário

1 Introdução **19**

1.1 Contextualização . . . 19

1.1.1 Arquitetura Legada e NFV . . . 21

1.1.2 Aplicações . . . 22

1.1.3 Taxonomia . . . 23

1.1.4 Problemas Relacionados às Tecnologias NFV . . . 25

1.2 Definição do Problema de Posicionamento e Encadeamento de VNFs *online* 27

1.3 Motivação . . . 30

1.4 Soluções Propostas: Algoritmos Exatos, Heurísticos e Hibridizações com AM 31

1.5 Objetivos . . . 33

1.5.1 Geral . . . 33

1.5.2 Específicos . . . 33

1.6 Contribuições . . . 34

1.7 Organização . . . 35

2 Trabalhos Relacionados **36**

2.1 Artigos *Surveys* . . . 36

2.2 Otimização Exata e Heurísticas . . . 38

2.2.1 Abordagens *Offline* e *Online* . . . 38

2.2.2 Abordagens Estáticas e Dinâmicas . . . 42

2.3 Aprendizado de Máquina . . . 45

3 Modelos real e computacional para o problema VNF-PC **49**

3.1 Definição Formal do Problema . . . 49

3.1.1 SN . . . 49

3.1.2 Funções Virtuais de Rede . . . 50

3.1.3 SFCs . . . 52

3.1.4 Mapeamento de uma SFC . . . 54

3.2 Formulação Proposta . . . 54

3.2.1 Restrições do Modelo *Offline* . . . 56

3.2.2 Restrições do Modelo *Online* . . . 57

3.2.3 Função Objetivo . . . 59

3.3 Considerações Finais . . . 60

4 Ambiente de simulação e experimentos de reotimização **61**
4.1 Configuração do Ambiente de Simulação . . . 61
4.1.1 Métricas . . . 61
4.1.2 Substrato de Rede . . . 63
4.1.3 Funções de Redes . . . 64
4.1.4 Classes de SFCs . . . 66
4.2 Experimentos Computacionais . . . 69
4.2.1 Algoritmos Avaliados . . . 69
4.2.2 Comparação entre as Estratégias de Reotimização . . . 70
4.3 Considerações Finais . . . 76

5 Heurística Semigulosa com Busca Local **78**
5.1 Heurística Semigulosa com Busca Local para o VNF-PC . . . 78
5.1.1 Heurística Construtiva Semigulosa . . . 80
5.1.2 Vizinhanças . . . 84
5.1.3 Busca Local . . . 85
5.2 Experimentos Computacionais . . . 86
5.2.1 Algoritmos Avaliados . . . 87
5.2.2 Comparação entre a Heurística e o Algoritmo Exato . . . 88
5.3 Considerações Finais . . . 95

6 Redução do espaço de solução: Abordagem de Clusterização por Localização Espacial **97**
6.1 Clusterização por Localização Espacial . . . 98
6.2 Experimentos Computacionais . . . 101
6.2.1 Interpretação dos Modelos de Clusterização . . . 101
6.2.2 Algoritmos Avaliados . . . 105
6.2.3 Análise das Estratégias de Clusterização . . . 105
6.3 Considerações Finais . . . 110

7 Redução do espaço de solução: Abordagem de Clusterização por Consumo de Recursos com Retreinamento **112**
7.1 Clusterização por Consumo de Recursos . . . 112
7.2 Experimentos Computacionais . . . 117
7.2.1 Algoritmos Avaliados . . . 118
7.2.2 *Dataset* . . . 118
7.2.3 Pré-processamento . . . 119
7.2.4 Seleção de características . . . 120
7.2.5 Interpretação dos Modelos de Clusterização . . . 122
7.2.6 Análise da Estratégia de Clusterização . . . 124

7.3 Alterações do *Dataset* . . . 129
7.3.1 Divergência de KL . . . 130
7.4 Experimentos Computacionais com demandas de SFCs desconhecidas . . . 130
7.4.1 Cenário de Experimentos e Tomada de Decisão . . . 130
7.4.2 Análise dos Momentos de Retreinamento . . . 132
7.5 Considerações Finais . . . 136

8 Conclusões e Trabalhos Futuros 138
8.1 Observações Finais . . . 138
8.2 Pesquisas Futuras . . . 142
8.2.1 Contextualização . . . 142
8.2.2 *Background* . . . 144
8.2.3 Problema . . . 145

Referências 160

Apêndice A Reduções Polinomiais: VNF-PC 161

Apêndice B Conceitos adicionais do VNF-PC 163
B.1 Entidades Envolvidas no Ambiente do VNF-PC . . . 163
B.2 Fórmula de Haversine . . . 164

Apêndice C Análise Complementar dos Experimentos do Capítulo 4 165
C.1 Relação Lucro e Receita . . . 165
C.2 Tempo de Execução . . . 166
C.3 Espalhamento dos Nós Virtuais . . . 166
C.4 Espalhamento dos Arcos Virtuais . . . 167
C.5 Atraso Médio Fim a Fim . . . 167

Apêndice D Introdução à Heurística Gulosa, Semigulosa e Metaheurística 169

Apêndice E Análise Complementar dos Experimentos do Capítulo 5 173
E.1 Lucro Total . . . 173
E.2 Tempo de Execução . . . 174
E.3 Taxa de Aceitação . . . 175
E.4 Espalhamento dos Nós Virtuais . . . 177
E.5 Espalhamento dos Arcos Virtuais . . . 178
E.6 Atraso Médio Fim a Fim . . . 178

Apêndice F Clusterização: Introdução a Algoritmos e Métricas 180
F.1 K-*Means*: Introdução . . . 180
F.2 Algoritmo *Hierarchical Clustering*: Introdução . . . 184

F.3 Algoritmo DBSCAN: Introdução 186

Apêndice G Clusterização: Análise Complementar do Capítulo 6 189
G.1 Algoritmo K-*Means* 189
G.2 Algoritmo *Hierarchical Clustering* 190
G.3 Algoritmo DBSCAN 191

Apêndice H Análise Complementar dos Experimentos do Capítulo 6 194
H.1 Lucro Total 194
H.2 Taxa de Aceitação 195
H.3 Espalhamento dos Nós Virtuais 195
H.4 Espalhamento dos Arcos Virtuais 196
H.5 Atraso Médio Fim a Fim 196

Apêndice I Análise Complementar dos Experimentos do Capítulo 7 198
I.1 *Dataset*: Análise de Variância 198
I.2 Clusterização: Análise Complementar 199
I.3 Ponderação de Valores: Análise Complementar 200
I.4 Taxa de Aceitação 202
I.5 Espalhamento dos Nós Virtuais 202
I.6 Espalhamento dos Arcos Virtuais 203
I.7 Atraso Médio Fim a Fim 203
I.8 Divergência de KL: Introdução 203
I.9 Análise dos Momentos de Retreinamento 204
I.9.1 Momento 1 204
I.9.2 Momento 2 206
I.9.3 Momento 3 207

Capítulo 1

Introdução

Neste Capítulo é realizada uma introdução à aplicação de virtualização em redes de computadores. Uma contextualização abordando conceitos, taxonomia e problemas relacionados é realizada na Seção 1.1. O problema tratado nesta tese é apresentado na Seção 1.2. A Seção 1.3 é dedicada à motivação deste trabalho. Os objetivos e as soluções propostas são apresentados respectivamente nas Seções 1.4 e 1.5. Na Seção 1.6 são especificadas as contribuições, e, por fim, na Seção 1.7 a estrutura desta tese é descrita.

1.1 Contextualização

A automação e mobilidade presentes no dia a dia das pessoas implica, o surgimento de novas demandas *web*, requerendo que o ser humano esteja sempre conectado à internet (Nokia, 2016). Essas demandas podem ser voltadas às mais diferentes áreas, *e.g.*, saúde, cidades inteligentes, indústrias de manufatura, logística, mídia e automotiva. Essas demandas impulsionam uma nova onda de aplicações e dados portáveis que podem ser acessados de qualquer lugar, gerando grandes desafios associados à demanda por *links* com estabilidade, disponibilidade e confiabilidade (He et al., 2022; Kokkinos, 2022). De modo a oferecer suporte a essas demandas, as tecnologias de virtualização de redes têm sido alvo de diferentes pesquisas nos últimos anos, impulsionando o desenvolvimento da computação em nuvem e das redes de comunicação de quinta e sexta geração (5G e 6G) (Fischer et al., 2013; Babbar et al., 2022; He et al., 2022; Tärneberg et al., 2022).

Segundo Sun et al. (2022), as redes de comunicação ocupam um espaço de destaque na literatura e no mercado pela sua rápida evolução, abrangência e penetração. Torna-se evidente um amplo conjunto de novos problemas, como: ubiquidade, mobilidade, localização de facilidades e multi-domínios. Um dos primeiros problemas relacionados à virtualização em redes de computadores é definido como *Virtual Network Embbeding* (VNE) (Chowdhury et al., 2009). A definição do VNE, apesar de ser inovadora à época, derivava de dois problemas que o antecederam: o *Virtual Private Networks* e o *Network Testbed*

Mapping Problem (Chowdhury et al., 2009; Gupta et al., 2001; Ricci et al., 2003).

Para Fischer et al. (2013), na definição clássica do VNE, os Provedores de Serviço (*Service Providers*, SPs) devem alocar eficientemente os recursos residuais do substrato físico de rede (*Substrate Network*, SN), para atender às requisições virtuais demandadas pelos clientes. Em sua definição, o VNE é empregado para mitigar problemas relacionados à ossificação da internet, provendo uma camada de virtualização adaptável às mudanças arquitetônicas demandadas; e permitindo que os SPs gerenciem as redes de forma flexível.

Posteriormente ao surgimento do VNE, um novo modelo de arquitetura de redes começou a ganhar espaço: as Redes Definidas por *Software* (*Software Defined Network*, SDN). A Open Networking Foundation (2019) é um consórcio dedicado ao desenvolvimento, padronização e comercialização da tecnologia SDN. Dentre as características de tal tecnologia estão: desacoplar o plano de controle do plano de dados da rede, e empregar um controlador logicamente centralizado com uma visão global do SN. Desse modo, o plano de controle consegue prover uma gerência facilitada e estratégica, gerando benefícios inerentes à programabilidade, flexibilidade, etc (Mijumbi et al., 2016). Nesse modelo de negócios, o plano de dados é responsável por encaminhar o tráfego da rede através de componentes de *hardware*, como *switches* e roteadores; e o plano de controle encarrega-se de executar as Funções de Rede (*Network Functions*, NFs) via *software*.

As redes SDNs e o VNE emergem contiguamente como tecnologias inovadoras, ajudando a trazer flexibilidade na implantação de novas funcionalidades de rede e facilitando o gerenciamento de recursos. Neste contexto, uma nova tecnologia emerge: a Virtualização de Funções de Rede (*Network Function Virtualization*, NFV). A adoção/transição das redes legadas para as redes baseadas em arquiteturas NFV começou em 2012, e desde então tem sido foco de pesquisas em todo o mundo (Bhamare et al., 2016; He et al., 2022).

As redes baseadas em tecnologias NFV são aplicáveis a diferentes processamentos de controle de dados e de rede, seja em uma infraestrutura fixa, ou móvel (White Paper NFV, 2012). De acordo com He et al. (2022), as redes baseadas em virtualização de funções desvinculam as NFs dos dispositivos de *hardware* legados (*middleboxes*), e as transfere para servidores virtualizados, que as instanciam por emulação sobre servidores físicos comerciais genéricos, os chamados *Commercial-Off-The-Shelf hardware* (COTS). Explorando o conceito de NFV, as NFs podem ser emuladas (virtualizadas) em quaisquer roteadores/servidores COTS (nomeados *Points of Presence*, N-PoPs) distribuídos pelo SN. Cada NF virtualizada (*Virtual Network Function*, VNF) é responsável por um tratamento específico a determinados fluxos de dados. Como exemplo de NFs que podem ser virtualizadas, tem-se: *Wide Area Network* (WAN), *Network Address Translation* (NAT), *Flow Monitor* (FM), *Intrusion Detection System* (IDS), *Intrusion Prevention System* (IPS), *Deep Packet Inspection* (DPI), *WAN Optimizer*, *Traffic Manager* (TM), *Video Optimization Controller* (VOC), etc (Laghrissi and Taleb, 2018; Zoure et al., 2022).

O uso de VNFs pode gerar benefícios como o desacoplamento entre *software* e *hard-*

ware, permitindo que o *software* evolua independentemente do *hardware* (e vice-versa). Outro ponto é a implantação flexível das NFs, de modo que um conjunto de dispositivos físicos possa executar diferentes NFs simultaneamente. Existe ainda o provisionamento dinâmico de recursos, no qual a rede pode ser dimensionada conforme as demandas das requisições e as condições residuais dos recursos físicos (Laghrissi and Taleb, 2018).

Complementarmente, a SDN fornece um ambiente propício à gerência de regras de roteamento entre VNFs; além de gerar flexibilidade de instanciação de novas NFs (Laghrissi and Taleb, 2018). Outro elemento pertencente ao plano de controle é o controlador de rede, que otimiza o gerenciamento de VNFs com flexibilidade. Apesar dos benefícios gerados pela integração entre SDN e NFV, as NFs podem ser virtualizadas sem a necessidade da SDN (White Paper NFV, 2012).

Um conjunto de VNFs sequenciadas (encadeadas) com o propósito de atender um determinado serviço de rede (*Network Service*, NS), é chamado Cadeia de Funções de Serviços (*Service Function Chain*, SFC) (Laghrissi and Taleb, 2018). Uma SFC que provê entrega de conteúdo de vídeo para os usuários, por exemplo, pode requerer codificadores e decodificadores instanciados em servidores estrategicamente posicionados no SN; ou ainda para fornecer suporte a dispositivos de Realidade Aumentada (RA) em redes 5G deve-se instanciar sequencialmente as VNFs NAT, *Firewall*, TM, VOC e IDS (Askari et al., 2019).

1.1.1 Arquitetura Legada e NFV

Buscando reduzir os custos de operação e implementação (*OPerational EXpediture*, OPEX e *CAPital EXpediture*, CAPEX)[1] envolvidos na gerência de dispositivos das redes de computadores, algumas empresas de telecomunicações do mundo uniram-se e formaram os *Industry Specification Groups* (ISGs)[2]. Os ISGs buscam construir conceitos e padrões para substituir os dispositivos de *hardware* dedicados, por equipamentos que possam ser virtualizados, dando origem ao conceito atual de NFV (White Paper NFV, 2012; He et al., 2022). Algumas diferenças entre os modelos de redes legados e a infraestrutura NFV são mostrados na Tabela 1.1.

[1] Capital gasto na operação e implantação de determinado sistema (Laghrissi and Taleb, 2018).
[2] https://www.etsi.org/newsroom/news/644-2013-01-isg-nfv-created

Tabela 1.1: Diferenças entre redes legadas e baseadas em NFV

Redes legadas	**NFV**
Middleboxes proprietários	Equipamentos genéricos (COTS)
Alto acoplamento *hardware/software*	Baixo acoplamento *hardware/software*
Baixo compartilhamento de *hardware*	Alto compartilhamento de *hardware*
Infraestrutura engessada e de baixa escalabilidade	Flexibilidade no gerenciamento
Expansão e manutenção difíceis, desperdício de recursos	Fácil manutenção e melhor utilização de recursos

Fonte: Elaborado pelo autor.

1.1.2 Aplicações

A evolução das redes de comunicação acontece em uma frente que dissocia o *hardware* do *software*. Por um lado, existe um foco na evolução de *hardware*, visando melhorias quanto à eficiência das redes, *e.g,* volume de tráfego e dispositivos, e taxa de transferência. Por outro lado, o foco é dado nos serviços prestados, em que sistemas devem conseguir atender a uma ampla e variada gama de serviços *online* (Askari et al., 2019; Tariq et al., 2022). Nesse ponto as tecnologias SDN e NFV são empregadas, usadas como facilitadoras no provimento de vários e novos serviços demandados (He et al., 2022; Tariq et al., 2022).

Quando são tratados aspectos inerentes ao suporte às novas demandas *web*, as principais preocupações são a flexibilidade e a capacidade de possíveis alterações na infraestrutura física. Neste sentido, a implementação de recursos virtuais em nuvens computacionais está se tornando crucial para a implantação bem-sucedida de um NS em tempo real (Gupta et al., 2017). As redes baseadas em virtualização de funções podem ser empregadas em diferentes ambientes, *e.g.*, redes de SPs, redes de infraestrutura híbrida, redes ópticas, nuvens de dados, redes locais e redes inter e intra *Data Centers* (Herrera and Botero, 2016). Nesta tese, similarmente a Breitgand et al. (2021); Gupta et al. (2017), é abordado um cenário de computação em nuvem. Um dos pontos destacados é nas aplicações fora de *datacenters*, em que a importância das restrições de fluxo de dados é maior, pois as necessidades de novos roteamentos e alterações de fluxo são maiores.

A virtualização de NFs oferecida pelas redes baseadas em virtualização de funções, também beneficia os setores de redes móveis, como as aplicações que usam as redes 5G (Babbar et al., 2022). Em redes 5G, *slicing* é um termo usado para referenciar o particionamento flexível e organização dos recursos computacionais de um SN. De maneira simplificada, uma *slice* possui um alto grau de flexibilidade para uma escolha sob demanda do conjunto de VNFs a serem usadas, e tem o objetivo de otimizar o uso do SN. Uma *slice* de rede permite criar várias redes virtuais e instanciar VNFs usando uma infraestrutura de rede compartilhada. Nesse modelo, cada *slice* pode coexistir lateralmente com outro *slice*, compartilhando recursos computacionais e reduzindo custos, *e.g.,* pode-se prover

um *slice* dedicado à realização de *streaming* de vídeo, e outro a serviços de mídia social.

Em um ambiente de rede legado, os SPs possuem um alto gasto com o CAPEX e o OPEX envolvidos na implantação/atualização dos recursos físicos do SN (Herrera and Botero, 2016; Tariq et al., 2022). Esse custo é devido à necessidade de compra e manutenção de *hardware* especializados para o atendimento das novas demandas. Um *hardware* dedicado requer mão de obra técnica especializada, tem alto consumo de energia, e não permite a adição de novas funcionalidades. Em contrapartida, as aplicações das tecnologias NFV mitigam grande parte desse problema, evitando uma proliferação de dispositivos ossificados, e trazendo maior flexibilidade ao projeto e gerenciamento de redes.

Em redes baseadas em NFV, a tarefa de sequenciar e instanciar as NFs em uma estrutura física é referenciada como mapeamento de funções de rede (He et al., 2022; Yi et al., 2018). Antes das tecnologias SDN/NFV, o mapeamento era executado manualmente, criando-se entradas de roteamentos em uma tabela estática e propensa a erros (Bari et al., 2015; Swami et al., 2019). Neste caso, qualquer alteração das NFs poderia acarretar uma reprogramação manual de todas as *middleboxes*, e em altos custos (Foukas et al., 2017; Vassilaras et al., 2017). Com a tecnologia NFV, os SPs não precisam mais adquirir/manipular *middleboxes* para atender os NS. Outra facilidade é que as VNFs podem ser mapeadas de forma independente e remota, o que potencialmente reduz os custos (Bari et al., 2015; Tariq et al., 2022). Notadamente, o modelo legado torna-se ineficiente perto da diversidade de NFs demandadas pelas aplicações que surgem a cada dia.

1.1.3 Taxonomia

Com o intuito de formalizar e unificar as terminologias existentes relativas ao tratamento do VNF-PC, a taxonomia adotada nesta tese é definida como segue:

- Mapeamento: a virtualização de redes permite que várias requisições de redes virtuais coexistam em um SN com compartilhamento de recursos. A alocação de recursos do SN para atender às demandas das requisições de rede em questão, respeitado as restrições do problema a um nível de nós e links físicos, é chamada de mapeamento (Chowdhury et al., 2012);

- Estático ou dinâmico: diz respeito ao mapeamento da requisição ser fixo, ou poder ser alterado conforme as necessidades do provedor (Laghrissi and Taleb, 2018). Tal aspecto é chamado nesta tese de reotimização. No mapeamento estático, cada requisição é mapeada sobre o SN de maneira fixa, neste caso, o mapeamento é mantido usando os mesmos nós e enlaces físicos até o término do seu tempo de vida.

Em mapeamentos dinâmicos, mesmo que uma requisição já esteja mapeada, o SN pode sofrer reotimizações e realocar recursos em prol de algum benefício (Fischer et al., 2013). Ao se aplicar reotimizações em ambientes reais de internet, necessita-se interromper os mapeamentos e fluxos de dados, remover as conexões ativas e reconfigurá-las, o que pode gerar um gasto de tempo, ser custoso computacionalmente, e causar problemas na qualidade do serviço prestado. Apesar de, introdutoriamente, nesta tese serem feitos alguns experimentos com um tratamento de reotimização a nível de posicionamento e encadeamento de VNFs, é dado foco no tratamento estático;

- *Online* ou *offline*: propriedade que se refere à forma de chegada e processamento das SFCs. Cenários *online* são caracterizados pela chegada das SFCs de maneira aleatória e desconhecida, na qual, *a priori*, não se conhece nenhum aspecto sobre a topologia, tempo de vida e requisitos demandados pelas SFCs. Neste caso, cada SFC é processada unitariamente, utilizando os recursos residuais do SN, de acordo com o seu tempo de chegada. Conceitualmente o modelo *offline* não possui princípios de tempo de entrada, saída ou duração. Em cenários *offline* se aplica o mapeamento em todo o conjunto de SFCs existentes de uma vez, neste caso o algoritmo beneficia-se do fato de conhecer integralmente as topologias, tempos de vida e requisitos (Fischer et al., 2013). Assumindo que um cliente pode a qualquer momento solicitar um serviço de rede, nesta tese é dado foco em uma chegada *online* de SFCs, contudo, a formulação em ILP proposta também pode ser aplicada em um ambiente *offline*;

- Computação em nuvem e contexto de aplicação: a computação em nuvem é um paradigma computacional em que um conjunto de VNFs são dinamicamente alocados e entregues sob demanda para os clientes (Gupta et al., 2017). Nesta situação, os provedores envolvidos podem alugar recursos de computação em nuvem sob um acordo com pagamento sob demanda (*serverless*). Tal ação permite que os provedores tenham um uso mais racional de recursos, propiciando reduções nos custos envolvidos. Neste contexto, as demandas de serviço podem se distinguir como:

  I Conteúdo da aplicação: *streaming* de vídeo, IA, *smart cities*, etc;

  II Contexto da aplicação: diferentes SFCs podem requerer diferentes tempos de processamentos de recursos, com envios para servidores e clientes diferentes;

  III Contexto do usuário: o acesso ao serviço pode ser executado de diferentes dispositivos de usuário como *smartphones*, *laptops*, IoT *devices*, e *desktops*.

- Qualidade de Serviço (*Quality of Service*, QoS): propriedade que se refere a alguns parâmetros necessários para uma determinada aplicação funcionar, são eles: largura de banda, atraso, variação do atraso, e perda (Tanenbaum and Wetherall, 2011).

Desta maneira, para a aplicação ser viável, os provedores precisam atender as demandas das requisições, respeitando os limites de parâmetros de QoS impostos. O objetivo destes parâmetros é oferecer um serviço de rede mais estável, com largura de banda dedicada, e com a variação do atraso controlada, garantindo o correto funcionamento do serviço *web*. Segundo Tanenbaum and Wetherall (2011), as aplicações de transferência de arquivos não são sensíveis ao atraso, neste caso, não há problema no atraso de alguns segundos para se iniciar o serviço. Por outro lado, as aplicações de tempo real, como VoIP e videoconferências possuem a necessidade de um baixo atraso. Como exemplos, algumas aplicações são mostradas abaixo:

Tabela 1.2: Demandas de QoS de diferentes aplicações

	Parâmetro de QoS			
	largura de banda	atraso	variação do atraso	perda
Compartilhamento de arquivo	média	baixa	baixa	baixa
Audio on demand	média	baixa	alta	baixa
Video on demand	alta	baixa	alta	baixa
Videoconferência	alta	alta	alta	baixa
Telefonia VoIP	baixa	alta	alta	baixa

Fonte: Elaborado pelo autor.

1.1.4 Problemas Relacionados às Tecnologias NFV

Nesta tese, os problemas relacionados às tecnologias NFV são tratados em duas linhas de pesquisa: *i*) através de otimização exata e heurística, onde os algoritmos buscam resolver os problemas de modo exato ou aproximado; e *ii*) por abordagens baseadas em Inteligência Artificial (IA), que buscam tirar proveito de conhecimentos aprendidos em ações anteriores, para uma tomada de decisão mais assertiva em ações futuras.

Problemas tratados através de otimização exata e heurísticas: para prover uma alocação de recursos ofertados pelos Provedores de Infraestrutura (*Infrastructure Providers*, InPs) e atender aos requisitos demandados, os InPs e SPs oferecem camadas de virtualização como um serviço (*Infrastructure as a Service*, IaaS). Na IaaS, os recursos demandados podem ser virtualizados em *Virtual Machines* (VMs) por meio das nuvens. Como exemplo, a Amazon EC2 fornece IaaS com base em variadas VMs com armazenamento e processamento para variados serviços[3] (Elfatih et al., 2022). A alocação dos recursos demandados pelos usuários sobre os ofertados pelos InPs, que, em geral, são onerosos e limitados, caracterizam o problema chamado NFV *Resource Allocation* (NFV-RA) (Laghrissi and Taleb, 2018; Yi et al., 2018; Herrera and Botero, 2016; Zoure et al., 2022).

[3]A Amazon EC2 oferece uma ampla gama de instâncias de servidores otimizados para atender aos diferentes requisitos demandados, disponível em https://aws.amazon.com/

O problema NFV-RA é segmentado em diferentes abordagens, nas quais podem ser consideradas várias formulações, dependentes do SN utilizado, das VNFs requisitadas, das SFCs demandadas, e das restrições (Yi et al., 2018). Outro ponto é que, em ambientes reais de rede, todos esses aspectos devem ser providos com uma alta taxa de dados transmitidos, confiabilidade, baixo atraso fim a fim, escalabilidade, e um consumo inteligente de recursos (Nguyen et al., 2017).

Um dos problemas combinatórios integrante do NFV-RA, e normalmente tratado com otimização, é a geração da composição da cadeia de VNFs (VNF *Chain Composition*, VNF-CC). O problema VNF-CC visa definir a melhor composição e encadeamento virtual das VNFs que constituem a sequência ideal para requisição. Neste caso, a ordem de encadeamento das VNFs pode implicar uma demanda menor ou maior de recursos físicos. Como saída, essa etapa gera um grafo de encadeamento chamado VNF *Forwarding Graph* (VNF-FG ou SFC), processado em outro estágio do NFV-RA (Laghrissi and Taleb, 2018).

A alocação de VNFs sobre o SN (*VNF Placement*, VNF-P) possui como objetivo posicionar as VNFs sobre os servidores existentes na rede, de maneira a atender a uma lista fixa de clientes previamente posicionados. O VNF-P é um problema pertencente à classe NP-difícil, e deriva de dois problemas já conhecidos como sendo NP-difíceis: o Problema de Localização de Facilidades, e o Problema de Atribuição Generalizada (Cohen et al., 2015).

Para manter as restrições originais do VNF-P e agregar novos desafios como posicionar os pontos de origem e destino (*endpoints*), posicionar as VNFs demandadas, e gerar um encadeamento ordenado (roteamento), é formulado o problema chamado Posicionamento e Encadeamento de VNFs (VNF *Placement and Chaining*, VNF-PC, ou VNF-FGE, VNF *Forwarding Graph Embedding*) (Herrera and Botero, 2016; Luizelli et al., 2015; Sharma et al., 2020). Essa variante do problema recebe como entrada o grafo VNF-FG (SFC) gerado pela resolução do problema VNF-CC. O VNF-PC, com uma chegada *online* de SFCs, e com um mapeamento estático, é o problema central tratado nesta tese, sendo apresentado na Seção 1.2.

Problemas tratados através de IA: a IA pode potencialmente reduzir o esforço computacional relacionado ao processamento das SFCs e gerar uma tomada de decisão mais assertiva em relação aos objetivos (Gebremariam et al., 2019; Xie et al., 2019). Mais estritamente, o Aprendizado de Máquina (AM, *Machine Learning*, ML) é a subárea da IA que provê aos algoritmos uma capacidade de aprender e tomar decisões sem serem explicitamente programados (Das and Nene, 2017).

Devido à complexidade de alguns problemas associados às redes baseadas em tecnologias NFV, e ao grande crescimento da dimensionalidade de dados a serem processados, uma alternativa é a aplicação de conceitos de AM. O raciocínio central dos algoritmos de AM baseia-se na indução e síntese de padrões. Nesse sentido, algumas das técnicas de AM podem ser usadas para coletar e analisar os dados de ações passadas da rede, e aprender

modelos de comportamentos (ações) para utilizar em momentos futuros. Neste tratamento, as tecnologias de controle da rede auxiliam na aplicação desses comportamentos aprendidos, transformando-os em ações algorítmicas e configurações da rede.

Em uma visão geral, os modelos de AM transitam em esquemas de agrupamento, regressão, classificação, e aprendizado estruturado (Zhao et al., 2019). Os algoritmos de aprendizado supervisionado realizam inferências com base no ajuste de parâmetros do modelo, utilizando dados já conhecidos. Dado um conjunto de dados para o treinamento, o objetivo é prever a classe de uma ou mais variáveis de saída.

Algumas aplicações de aprendizado supervisionado são encontradas no controle de admissão das redes. Esse problema tem o objetivo de otimizar a admissão de requisições com base em histórico de mapeamentos. O controle de admissão pode ter como objetivo tanto a tomada de decisão quanto atender ou não uma requisição dependendo do benefício gerado. Para tal, podem ser usadas diferentes métricas, *e.g.,* taxa de aceitação e lucro. O aprendizado supervisionado também pode ser aplicado para prever aspectos relativos à classificação de fluxos de dados. Um algoritmo proposto por Comaneci and Dobre (2018) recorre a um modelo de regressão para identificar os fluxos e diferenciar quais acontecem no mesmo período e quais não, de modo a otimizar o roteamento. Outras aplicações de aprendizado supervisionado são encontradas na detecção de Ataques Distribuídos de Negação de Serviço (DDoS); classificação de tráfegos, fluxos e pacotes de dados; otimização de rotas, previsão de falhas, detecção de anomalias de funcionamento, predição de QoS, segurança, controle de congestionamento etc (Gebremariam et al., 2019; Xie et al., 2019).

Os algoritmos de AM não supervisionado, não fazem inferências ajustando os parâmetros do modelo usado de acordo com rótulos, pois os rótulos são desconhecidos. Nesse caso, o objetivo é revelar informações intrínsecas aos agrupamentos formados (*clusters*). Existem diversas categorias de algoritmos de aprendizado não supervisionado, *e.g.,* sequenciais, hierárquicos, baseados na otimização de funções de custo e *Self-Organizing Map* (SOM) (Xie et al., 2019). Um algoritmo não supervisionado bastante usado em redes é o *K-means*, e suas aplicações podem ser encontradas na detecção de anomalias de rede.

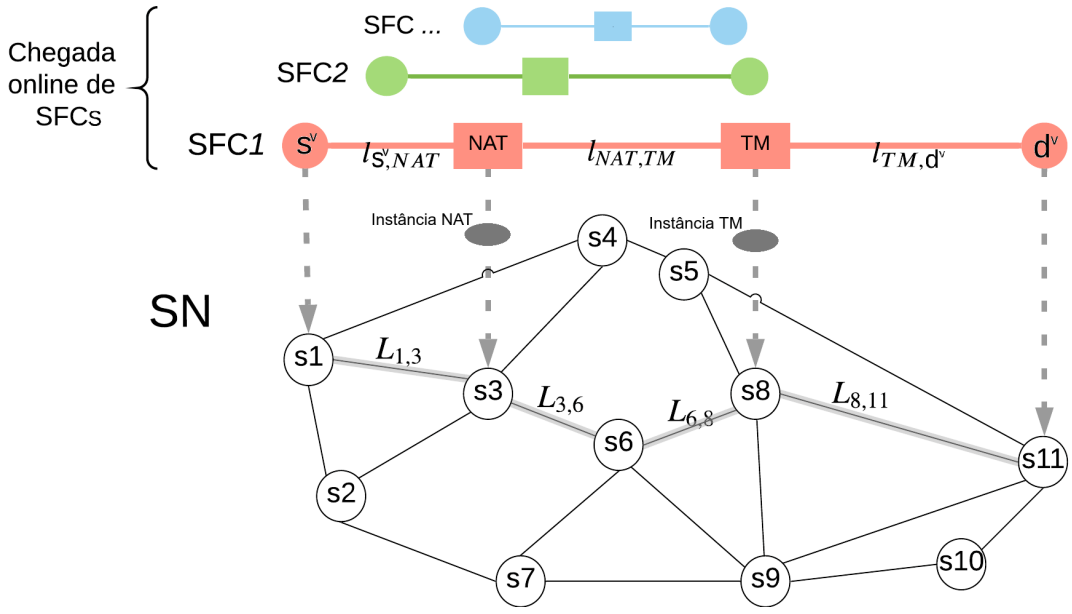
1.2 Definição do Problema de Posicionamento e Encadeamento de VNFs *online*

Dentre os problemas explicados na Subseção 1.1.4, o VNF-PC é a variante escolhida para ser tratada nesta tese. No modelo real do problema, os SPs devem mapear sobre o SN, as SFCs requisitadas pelos clientes de forma eficiente, e respeitando capacidades

limitadas de recursos disponíveis (Luizelli et al., 2017; Laghrissi and Taleb, 2018). As dificuldades de resolução do VNF-PC estão ligadas a sua natureza combinatória. Dada a complexidade de sua resolução, e sua função intrínseca no desenvolvimento de novas tecnologias NFV, ele torna-se ideal para ser estudado e tratado em um ambiente de hibridização entre AM, heurísticas e otimização exata.

A Figura 1.1 exemplifica uma chegada *online* de SFCs. De maneira simplificada, para cada SFC a ser mapeada, ela deve ser processada no SN no momento de sua chegada (definição de *online* apresentada na Subseção 1.1.3). Para esta ação, cada NF demandada deve ser disposta em uma instância de VNF emulada em um servidor físico, e cada arco virtual roteado em um ou mais arcos físicos do SN. No exemplo da Figura 1.1, o ponto de origem s^v da SFC_1 é mapeado no servidor $s1$, o ponto de destino d^v é mapeado no servidor $s11$, a VNF NAT em uma instância posicionada no servidor $s3$, a VNF TM em uma instância posicionada no servidor $s8$, o arco virtual $l_{s^v,NAT}$ no arco físico $L_{1,3}$, o arco virtual $l_{NAT,TM}$ nos arcos físicos $L_{3,6}$ e $L_{6,8}$, e o arco virtual l_{TM,d^v} no arco físico $L_{8,11}$. No exemplo, o servidor físico $s6$ é usado como nó ponte no roteamento dos arcos, e só repassa o tráfego de dados para o próximo nó da rede. Um explicação detalhada sobre o processo de mapeamento é apresentado na Seção 3.1. Vale ressaltar que o SN possui uma variação de recursos residuais disponíveis, pois cada SFC possui um tempo de vida, e quando uma SFC é encerrada, deve-se reintegrar ao SN residual os recursos antes alocados para se tornarem disponíveis para outras SFCs utilizarem. Neste caso, um mapeamento que antes era bom ou ótimo, pode ser potencialmente rearranjado e melhorado (reotimizado).

Figura 1.1: Exemplo de um fluxo *online* de SFCs



Fonte: Elaborado pelo autor.

Complementarmente, o problema VNF-PC é explicado de forma detalhada, a nível conceitual, na Seção 3.1, e seu respectivo modelo computacional é definido matematica-

mente com variáveis, parâmetros e restrições, sendo mostrado na Seção 3.2.

Complexidade: formalmente, um problema x é pertencente à classe NP-difícil se todos os problemas da classe NP são polinomialmente redutíveis a x, *i.e.*, se y $\propto$ x $\forall$y $\in$ NP (Cormen et al., 2001). Nessa definição, os problemas relacionados ao NFV-RA apresentados na Subseção 1.1.4 são em sua maioria problemas de otimização sendo improvável obter uma solução ótima em um tempo polinomial. No Apêndice A é possível observar as clássicas reduções dos 21 problemas NP-completos de Karp (1972), e suas reduções polinomiais que mostram que o problema VNF-PC é NP-difícil.

Em ambientes reais, as redes baseadas em virtualização normalmente possuem um conjunto numeroso de dispositivos a serem gerenciados, deixando as instâncias com uma alta cardinalidade de componentes a serem administrados, e dificultando uma alocação ótima de recursos. Nesta tese é tratado o problema VNF-PC de maneira similar a Luizelli et al. (2017); Fischer et al. (2019). Tal variante é um problema integrado aos problemas NP-difíceis: *Location Routing Problem* (LRP) e *Virtual Network Embbeding* (VNE) (Mehraghdam et al., 2014; Cohen et al., 2015).

O problema LRP considera um conjunto de instalações e um conjunto de clientes com demandas conhecidas. Neste caso, as decisões a serem tomadas são: o número e a localização das instalações a serem estabelecidas para atender os clientes; quais clientes são atendidos por quais instalações; e o roteamento das instalações para atender os clientes utilizando uma frota de veículos. Em uma abstração do LRP junto ao VNF-PC, cada instalação é modelada como uma VNF ou cliente final, e a roteirização transformada no roteamento encadeado entre as VNFs e os clientes. Nesse caso, o problema de roteamento transforma-se em um problema de fluxo com dependências entre mercadorias (Mehraghdam et al., 2014).

No VNE, o SN é composto por nós e enlaces, e são consideradas as capacidades de processamento e de largura de banda. De forma similar, as requisições de redes virtuais possuem demandas de processamento associadas a cada nó, e de largura de banda a cada enlace. Cada nó virtual de uma mesma requisição deve ser mapeado em um nó diferente do SN, e deve ser posicionado dentro de um raio de mapeamento. Cada enlace virtual de uma requisição pode ser mapeado em mais de um enlace físico e formar um caminho. Para se efetivar um mapeamento é necessário que todos os componentes alocados no SN possuam recursos suficientes para servir às demandas das requisições (Chowdhury et al., 2009). O VNF-PC também é um problema integrado ao VNE (Beck and Botero, 2015; He et al., 2022). No VNF-PC, as SFCs posicionadas e encadeadas são modeladas como requisições a serem mapeadas sobre um SN genérico no VNE. Mas, diferentemente do VNE, no VNF-PC não existe um raio de localização para alocar os nós virtuais, o que aumenta o número de variáveis a serem consideradas para o posicionamento das VNFs. Outras adições de restrições do VNF-PC em relação ao VNE, são referentes a latência de encaminhamento, atraso fim a fim, e alterações no fluxo de entrada e saída a cada

VNF encadeada (Gao et al., 2018). Khebbache et al. (2017) também provam que o VNF-PC pertence à classe NP-difícil mostrando que uma instância do VNE pode ser obtida relaxando as restrições de sequenciamento de VNFs, e com isso, deduz-se que o VNF-PC é NP-difícil, visto que o VNE também o é.

1.3 Motivação

Progressos na área de virtualização em redes, além de trazer benefícios para as áreas acadêmica e científica, promovem um amparo tecnológico ao desenvolvimento e ao aprimoramento de diversas tecnologias relacionadas à área (Boutaba et al., 2018; Tärneberg et al., 2022). As redes baseadas em NFV possuem um enorme potencial em reduzir os custos, e aumentar os lucros dos provedores de serviços e infraestrutura. Pelo lado dos usuários, as redes baseadas em tecnologias VNF podem gerar inúmeros benefícios como uma conexão melhor planejada, uma maior qualidade de serviço, e um custo menor (Laghrissi and Taleb, 2018). Dados todos esses benefícios, há uma real tendência à adoção das redes baseadas em NFV na atualidade (Li et al., 2020; Zoure et al., 2022).

A Research And Markets (2020) espera que o lucro do mercado global gerado pelo provisionamento de serviços das redes baseadas em NFV aumente $34,9\%$ a cada ano, atingindo 122 bilhões de dólares em 2027. Além dos benefícios financeiros, existem outros pontos a serem considerados. Suponha que uma empresa precise de uma NF específica em um local remoto. Se for feita a implantação de um *hardware* físico dedicado, esta ação pode ter um alto custo, ser demorada e de difícil gerenciamento. Por outro lado, se for implantada uma VNF, o provisionamento da função requerida é quase instantâneo, pois é automatizado. Fato que se torna mais interessante ainda se o provisionamento das VNFs ocorrer em vários locais diferentes (Bhamare et al., 2016; Laghrissi and Taleb, 2018).

As redes baseadas em NFV habilitam o desenvolvimento de uma nova gama de serviços, como as redes 5G e 6G (Kokkinos, 2022; Tärneberg et al., 2022). As três principais características que as redes 5G devem possuir são: *i*) uma alta capacidade de banda larga móvel disponível, *ii*) suporte a uma comunicação massiva entre usuários e dispositivos; e *iii*) o fornecimento de serviços de comunicação com baixa latência (Yousaf et al., 2017). Nesse âmbito, as redes 5G devem ser projetadas para fornecer um serviço flexível, escalável e facilmente programável, sendo que esses serviços possuem rigorosos requisitos de QoS e devem suportar uma grande gama de dispositivos conectados. Por introduzir uma grande mudança na maneira como os serviços de rede são implantados e operados, as redes baseadas em tecnologias NFV e SDN são facilitadoras centrais no desenvolvimento tecnológico das redes 5G e 6G (Gebremariam et al., 2019; Tärneberg et al., 2022).

Abordando aspectos orientados a serviços, a tecnologia NFV torna-se central em aplicações *web* como dispositivos de internet das coisas (*Internet of Things*, IoT) (Askari et al., 2019; Nguyen et al., 2020). Nessa linha, existem muitos desafios a serem tratados, como o problema VNF-PC. O VNF-PC é um problema com um alto tempo computacional para instâncias médias e grandes. Mas, os usuários demandam soluções rápidas, sendo que essas soluções devem ser geradas em redes cada vez maiores. Nesse sentido, um motivante desafio é o de atender a esses clientes com boas conexões, gerar baixos custos para os provedores, e realizar esse processamento em um baixo tempo computacional (Wang et al., 2019; He et al., 2022). Outro ponto que deve ser considerado é que essas ações devem ser benéficas tanto para os clientes, que contratam os serviços; quanto para as empresas provedoras, que oferecem os serviços e a infraestrutura.

Em uma prévia revisão de literatura, percebe-se que muitos trabalhos abordam o VNF-PC exclusivamente de maneira exata ou heurística (Capítulo 2). Em outros trabalhos percebe-se uma tendência pela aplicação de técnicas de AM. Mas, existe uma lacuna de trabalhos relativa à integração das técnicas de AM e otimização. As principais motivações desta tese são contribuir na consolidação de um material de referência para outras pesquisas, e na potencial aplicação de novas abordagens no ambiente de telecomunicações.

1.4 Soluções Propostas: Algoritmos Exatos, Heurísticos e Hibridizações com AM

Devido à quantidade e diversidade de SFCs demandadas, às estruturas de redes físicas cada vez maiores, e à necessidade de bons roteamentos com instanciações de VNFs bem planejadas, torna-se cada vez mais custosa a resolução do VNF-PC em um tempo viável. Na resolução do VNF-PC de modo exato, o número de variáveis e restrições a serem consideradas está diretamente ligado à quantidade de dispositivos presentes no SN e nas SFCs.

Para reduzir o tempo de resolução do VNF-PC, mas mantendo as características originais das restrições, ou reduz-se a dimensionalidade da SFC ou do SN. Como pressuposto, é adotado que reduzir a dimensionalidade de uma SFC não é uma tarefa trivial. Na prática, o número de componentes de uma SFC está ligado diretamente aos recursos demandados pela aplicação em questão. E mesmo que o problema VNF-CC seja resolvido na otimalidade, não implicaria uma redução significativa do tempo computacional gasto para se mapear uma SFC. Por outro lado, o SN muitas vezes é subutilizado nos mapeamentos de SFCs, *i.e.,* um SN com um número grande de roteadores e arcos físicos

é usado no processamento de uma SFC, mas, na prática, somente um subconjunto desses dispositivos são de fato bons candidatos para atender às demandas em questão.

Um dos princípios empregados nesta tese é que SFCs com características próximas tendam a utilizar os mesmos recursos. Assim, com base em um histórico de mapeamentos efetuados com sucesso no passado, propõe-se criar uma estratégia de clusterização que determine bons subconjuntos de dispositivos do SN a serem usados no processo de mapeamento de uma nova SFC. Como cada componente do SN é representado por um conjunto de variáveis em um modelo matemático, essa ação potencialmente reduziria o número de variáveis a serem examinadas para entrar na base da solução do problema, e, consequentemente, diminuir-se-ia o tempo de processamento para encontrar uma solução para o problema de otimização.

Com base nas afirmações anteriores, começam a se formar algumas questões de pesquisa: *i*) SFCs com características parecidas tendem a utilizar recursos similares? Com base em um histórico clusterizado de mapeamentos, pode-se aprender comportamentos, e induzir diferentes conjuntos de dispositivos do SN aptos para atender a uma determinada SFC. Tal ação reduziria o número de variáveis e restrições a serem processadas, auxiliando o algoritmo, seja exato ou heurístico, a investigar mais assertivamente o espaço de soluções; *ii*) aplicar um algoritmo exato em um espaço de soluções diminuído reduziria o tempo de tomada de decisão sobre o mapeamento de uma SFC no VNF-PC? Devido à característica NP-difícil do problema, ao se processar uma fração do espaço de soluções, o tempo de processamento será reduzido significativamente, gerando ganhos, como economia de processamento, e menos espera para o cliente final.

Em suma, os algoritmos de mapeamento propostos buscam resolver o problema VNF-PC, e atuam integrando AM, algoritmos exatos e heurísticos. Para tal, pretende-se resolver o VNF-PC de maneira *online*, com restrições atuais de mercado, em um tempo computacionalmente aceitável, e com um resultado próximo ao ótimo. Dentre as contribuições, são explorados:

I Modelo baseado em Programação Linear Inteira (PLI, *Integer Linear Programming*, ILP), capaz de resolver o VNF-PC de modo *online* (Capítulo 3);

II Ambiente de simulação baseado em demandas reais de rede, e experimentações com um algoritmo baseado em ILP e diferentes estratégias de reotimização (Capítulo 4);

III Algoritmo heurístico Semiguloso com busca local, baseado na heurística de Hart and Shogan (1987) e na metaheurística *Variable Neighborhood Search* (VNS) de Mladenović and Hansen (1997) (Capítulo 5);

IV Algoritmos de agrupamento baseado em AM. Neste caso, com base nas características geográficas do SN, e em mapeamentos anteriores, pretende-se prever quais regiões do

espaço de solução são promissoras para serem investigadas para mapear as novas SFCs (Capítulos 6 e 7);

1.5 Objetivos

1.5.1 Geral

Desenvolver diferentes algoritmos para solucionar o VNF-PC, com hibridizações entre otimização exata, heurísticas e AM, atuando em um ambiente *online*.

1.5.2 Específicos

I Desenvolver uma formulação, baseada em ILP, capaz de modelar computacionalmente o problema VNF-PC em um contexto *online* e *offline*;

II Desenvolver um algoritmo exato, baseado em um modelo matemático, para resolução do VNF-PC em um cenário *online*;

III Desenvolver estratégias de reotimização para resolução exata do VNF-PC em um cenário *online*;

IV Desenvolver um algoritmo eficiente, baseado em adaptações de heurísticas, para resolução do VNF-PC em um cenário *online*;

V Desenvolver um algoritmo baseado em AM não supervisionado, que gere diferentes clusters de servidores para serem usados nos mapeamentos de novas requisições;

VI Desenvolver um algoritmo baseado em AM, que tire proveito de informações obtidas da topologia de rede e das requisições mapeadas no passado, de modo a reduzir a dimensionalidade do espaço de solução, e consequentemente, reduzir o número de variáveis, restrições e o tempo de processamento.

1.6 Contribuições

Este trabalho gerou até o momento os seguintes trabalhos científicos:

I *Composition Selection Mechanism for Chaining and Placement of Virtual Network Functions*, no *International Conference on Network and Service Management* (Araújo et al., 2019a). Tal trabalho explora a propriedade que um SP pode potencialmente ter de diferentes opções de composições de SFCs para atender a um determinado NS;

II *Flexible Compositions for the Virtual Network Function Chain Placement in Online Environments*, no *International Teletraffic Congress* (Araújo et al., 2019b). Este trabalho é uma extensão do trabalho Araújo et al. (2019a), e apresenta um modelo completo para a resolução do VNF-PC. O modelo usado em tal artigo foi aprimorado até o apresentado nesta tese. As pesquisas com composições flexíveis foram colocadas em segundo plano, e optou-se por trabalhar estritamente com o VNF-PC;

III Alocação de Recursos para Redes Virtuais com Seleção de Método de Resolução via Aprendizado de Máquina, no Simpósio Brasileiro de Redes de Computadores (Araújo et al., 2020). Neste artigo é proposta uma heurística que tem como objetivo prever em quais situações se pode preterir um algoritmo heurístico em prol de um exato. Trata-se de um algoritmo para o VNE, e que em trabalhos futuros será adaptado para o VNF-PC;

IV *On the Analysis of Online and Periodic Virtual Network Embedding in Multi-Domain Environments*, no *International Journal of Networking and Virtual Organisations* (Araújo et al., 2021c). Trata-se de um estudo aplicado ao VNE, mas que fundamentou decisões inerentes ao modelo em ILP, conteúdo presente no Capítulo 3;

V *A Hybrid Optimization-Machine Learning Approach for the VNF Placement and Chaining Problem*, na revista *Computer Networks* (Araújo et al., 2021b). Artigo referente às abordagens de redução de dimensionalidade, apresentada nos Capítulos 6 e 7;

VI Uma Abordagem Heurística para o Posicionamento e Encadeamento de Funções Virtuais de Rede em Ambientes *Online*, no Simpósio Brasileiro de Redes de Computadores (Araújo et al., 2021a). O algoritmo proposto neste artigo é baseado nas metaheurísticas *Greedy Randomized Adaptive Search Procedure* e *Variable Neighborhood Search*, referente ao conteúdo apresentado no Capítulo 5.

VII *A demand aware strategy for a machine learning approach to VNF-PC problem*, no *International Conference on Cloud Networking* (Araujo et al., 2022). Neste artigo é

apresentada uma heurística que consiste na combinação de um algoritmo do artigo (Araújo et al., 2021b), com a aplicação de conceitos de re-treinamento. Conteúdo apresentado no Capítulo 7;

VIII *Modeling and Analysis of Different Reconfiguration Strategies for Virtual Network Function Placement and Chaining with Service Classes Identification* na revista *IEEE Latin America Transactions* (Araujo et al., 2023). Neste trabalho é apresentado um modelo matemático para resolver o VNF-PC com diferentes estratégias de reconfiguração. Conteúdo apresentado no Capítulo 4.

1.7 Organização

O restante desse texto está organizado como segue: é apresentado no Capítulo 2 um levantamento dos trabalhos relacionados. No Capítulo 3 é apresentada uma definição do problema VNF-PC e seu modelo computacional formulado com ILP. No Capítulo 4 são definidas as configurações do ambiente de experimentação, e realizados alguns experimentos com um algoritmo baseado no modelo matemático e diferentes estratégias de reotimização. O Capítulo 5 apresenta um algoritmo heurístico chamado SGBL. Nos Capítulos 6 e 7 são descritas duas abordagens de redução do espaço de solução do VNF-PC através de técnicas de AM. O Capítulo 8 é destinado a conclusões e propostas de trabalhos futuros.

Capítulo 2

Trabalhos Relacionados

Neste Capítulo são apresentados alguns trabalhos relacionados à pesquisa desenvolvida. Na Seção 2.1 investigam-se artigos que analisam extensamente problemas no contexto da virtualização em redes desde sua origem, os chamados artigos *Surveys*. Na Seção 2.2, são enfatizados alguns trabalhos que tratam a virtualização em redes através de algoritmos de otimização exata e heurística. Na Seção 2.3 examina-se trabalhos que aplicam algoritmos de AM na resolução de problemas relativos às redes baseadas em NFV e SDN.

2.1 Artigos *Surveys*

Os artigos chamados *surveys* cobrem a evolução dos pesquisadores sobre a tecnologia NFV desde seu início até o atual estado da arte. O artigo *survey* de Fischer et al. (2013) apresenta os papéis dos diferentes provedores envolvidos no paradigma da virtualização de redes, métricas de otimização, e uma taxonomia aplicada à área de virtualização. Seguindo a taxonomia citada, esta tese realiza o posicionamento e encadeamento de VNFs de forma heurística e exata, *online* e centralizada. Fischer et al. definem a chamada fragmentação de recursos. Um SN encontra-se fragmentado quando possui um conjunto de servidores e arcos com recursos disponíveis para utilização, mas que não podem ser utilizados, pois, os componentes adjacentes à região não possuem recursos disponíveis para se estabelecer um posicionamento de uma VNF ou encadeamento de uma SFC. O problema da fragmentação ocorre devido ao problema ser *online*, neste caso, quando o tempo de vida de uma SFC se encerra, os recursos antes utilizados são reintegrados ao SN, e ficam disponíveis para outra SFC utilizar. Quando várias SFCs são mapeadas e terminadas, o SN tende a fragmentar-se. Tal problema pode ser mitigado por algoritmos que realizam a reotimização das SFCs sobre o SN.

Desde o início dos anos 2000 as tecnologias de virtualização de redes têm sido foco de interesses acadêmicos e industriais. Atualmente, devido à sua complexidade computacional, o foco principal de trabalhos na comunidade de pesquisa são algoritmos heurísticos,

metaheurísticas ou baseados em AM (Fischer et al., 2013; Laghrissi and Taleb, 2018; Yi et al., 2018; Xie et al., 2019; Gebremariam et al., 2019). Muitos pesquisadores têm usado informações explícitas das SFCs e do SN para melhorar o desempenho dos algoritmos (Fischer et al., 2013). Alguns dos algoritmos propostos nesta tese diferem-se de heurísticas clássicas da literatura por empregar técnicas de AM e extrair informações implícitas às instâncias do problema. Após, conforme o conhecimento aprendido, são utilizadas essas informações para guiar o processo de busca do algoritmo no espaço de soluções.

Yi et al. (2018) fundamentam alguns conceitos elementares nessa área, apresentam terminologias, e definem casos de uso; além de explicar como as redes baseadas em NFV diferem-se do modelo de rede legado. Conforme Yi et al., apesar de diferentes artigos considerarem objetivos distintos, as restrições do VNF-PC são geralmente as mesmas, e incluem a alocação de recursos, mapeamento de VNFs e encadeamento. Também são abordadas todas essas restrições nesta tese. Para Yi et al., devido ao tempo de execução de uma heurística ser inferior ao da solução ótima, cada vez mais heurísticas têm sido propostas para resolver o VNF-PC. Este trabalho segue tal linha de desenvolvimento heurístico, mas em vez de propor uma heurística tradicional, como as apresentadas por Fischer et al. (2013); Laghrissi and Taleb (2018), opta-se por também empregar conceitos de AM para gerar algoritmos híbridos, buscando integrar as vantagens da otimização clássica e heurística aos conceitos de AM.

Xie et al. (2019) apresentam um artigo *survey* sobre algoritmos de AM empregados em redes. Xie et al. estudam como o AM é usado em aplicações envolvendo as redes SDN e NFV, e descrevem uma perspectiva geral de diferentes técnicas de AM aplicadas para a otimização de rotas, gerenciamento de recursos, segurança, e classificação de tráfego. Nesta tese, assim como em outras áreas, considera-se que os algoritmos devam promover características de autoadaptabilidade no mapeamento de uma SFC. Assume-se que, com o uso de técnicas de AM, a ação de mapear uma SFC pode se tornar mais automatizada e autoadaptável em relação às heurísticas tradicionais.

Zoure et al. (2022) colocam que o conceito de NFV surgiu como uma arquitetura de rede disruptiva, cuja a evolução está levando as empresas a terceirizar funções de rede para a computação em nuvem. Contudo, neste ambiente de migração, muitas empresas se mostram relutantes à adoção das redes NFV devido à preocupações com segurança e confiabilidade nos serviços prestados. Zoure et al. apresentam uma lista da taxonomia adotada em relação a referidos aspectos. Alguns pontos tratados por Zoure et al., e que estão presentes nesta tese, são a preocupação com o atendimento dos requisitos de QoS, mapeamento correto das instâncias de NFs, encadeamento das VNFs, localização geográfica e regras de afinidade/anti-afinidade. Como exemplo, a regra de anti-afinidade define que, por motivos de segurança, algumas VNFs devem ser instanciadas em diferentes servidores do SN. Outro ponto tratado diz respeito às violações em nível de mapeamento. Neste caso o algoritmo deve garantir a qualidade do mapeamento, garantindo aspectos

como um atraso máximo fim a fim a ser respeitado, e a confiabilidade do provisionamento de recursos demandados nas instâncias virtuais. Os artigos *surveys* utilizados como referência nesta tese são sumarizados na Tabela 2.1.

Tabela 2.1: Classificação dos artigos *surveys*

	Contribuição	**Referência**
VNE	- Definições clássicas de virtualização em redes e formulações; - Estratégias de otimização exata, heurística e metaheurística; - Conceitos: *online*, periódica e *offline*; concisa ou redundante; estática ou dinâmica; centralizada ou distribuída; - Definições de fragmentação e decomposição do VNE; - Métricas e *Green Networking*; - 125 trabalhos analisados.	(Fischer et al., 2013)
NFV/ SDN	- Definição clássica de VNF-P, VNF-PC, mapeamento *online* e *offline*; - Definições de redes móveis e sua interação com NFV e SDN; - Conceitos de ciclo de vida e migração de VMs; - Conceitos de economia de energia, custo e lucro; - Estratégia de otimização exata, heurística e metaheurística; - Diferentes tipos de restrições, objetivos e métricas; - 176 trabalhos analisados.	(Laghrissi and Taleb, 2018)
NFV	- Definição clássica de VNF-RA, VNF-CC, VNF-FGE e VNF-SCH; - Definições de casos de uso, arquitetura e relações entre NFV e SDN; - Estratégias de otimização exata, heurística e metaheurística; - Objetivos e métricas; - Definição de custo de energia, confiabilidade e segurança; - 446 trabalhos analisados.	(Yi et al., 2018)
SDN	- Relação entre AM e redes SDN; - Modelos e exemplos de AM supervisionado, não supervisionado, semi-supervisionado e por reforço aplicados em redes de SDN; - Vantagens e desvantagens dos algoritmos de AM; - Definição de problemas em redes SDN abordados com AM; - 278 trabalhos analisados.	(Xie et al., 2019)
NFV	- Definição sobre a tecnologia NFV; - Anomalias de serviço de rede; - Violações de requisitos de QoS; - 121 trabalhos analisados.	(Zoure et al., 2022)

Fonte: Elaborado pelo autor.

2.2 Otimização Exata e Heurísticas

2.2.1 Abordagens *Offline* e *Online*

Algoritmos que atuam em um contexto *offline* devem ser considerados, pois, geram uma análise de fatores que podem ser despercebidos em ambientes *online*. Em ambientes *offline*, mede-se a sensibilidade do algoritmo de mapeamento conforme são variados alguns

fatores, como a quantidade de VNFs em uma SFC, ou de SFCs processadas simultaneamente.

Mehraghdam et al. (2014) propõem uma heurística baseada em *Mixed Integer Quadratically Constrained Programs* para resolver o VNF-PC como uma extensão do VNE. No trabalho proposto, três objetivos são otimizados: maximizar a banda residual; minimizar o número de servidores usados; e minimizar o atraso dos encadeamentos. No final, um estudo dos três objetivos é efetuado. A abordagem proposta foca em uma análise de Pareto para mostrar os *trade-offs* presentes entre as métricas aferidas. Referido trabalho considera que os requisitos demandados pelas SFCs são conhecidos à *priori* e omitem os aspectos *online* das redes baseadas NFV, como atraso fim a fim, além de considerar que uma SFC pode ter vários *endpoints*. Baumgartner et al. (2015) também utilizam um algoritmo originalmente criado para solucionar VNE no VNF-PC. O objetivo do trabalho é minimizar os custos de mapeamento. Similarmente a Baumgartner et al., nesta tese são aplicadas restrições de largura de banda, processamento e armazenamento. No entanto, Baumgartner et al. não consideram as restrições de atraso fim a fim dos roteamentos realizados. E por ser uma variante do VNE, o algoritmo apresentado por Baumgartner et al. é voltado para o mapeamento das requisições do VNE, não tratando questões específicas ao compartilhamento de instâncias de VNFs.

Feng et al. (2017) apresentam um algoritmo, baseado no problema *multi-commodity chain*, para posicionar as VNFs em redes distribuídas nas nuvens e rotear os fluxos de serviço entre as VNFs. Visto a complexidade do problema, o tratamento aplicado é baseado em algoritmos de aproximação para gerar um tempo de execução polinomial. Feng et al. adotam um modelo de mapeamento simples, em que o compartilhamento de VNF entre diferentes SFCs e o atraso fim a fim não são considerados. Diferentemente do trabalho de Feng et al., e da mesma forma que Luizelli et al. (2015) e Gao et al. (2018), nesta tese aplicam-se as restrições de atraso e o compartilhamento de VNFs é estimulado para mitigar os custos financeiros de se manter um servidor do SN ativo.

Gao et al. (2018) tratam o VNF-PC com dois algoritmos distintos: um exato, baseado em Programação Linear Inteira Mista; e uma heurística matemática. Como objetivo, são minimizados os recursos alocados, mantendo o uso dos arcos em um nível satisfatório. Ambos os algoritmos, além de possuir restrições clássicas do VNP-PC *e.g.*, capacidades suportadas por servidor, atraso e roteamento orientado; possuem restrições exclusivas de compressão e descompressão do fluxo de uma SFC. De modo similar a Gao et al., referidas restrições são tratadas nesta tese, sendo considerado que o fluxo passante em um arco virtual é alterado pela VNF a que está associado. O trabalho de Gao et al. tem um tempo de execução da ordem de segundos, o que pode ser apontado como alto, considerando SFCs que são sensíveis ao atraso fim a fim, como aplicações de RA. Diferentemente, esta tese foca em algoritmos que gerem um tempo de execução da ordem de milissegundos.

O mapeamento de SFCs de modo estático e *offline*, considerando questões como alta confiabilidade e baixa latência é realizado em Wei et al. (2022b). Com o foco em um rápido processamento para atendimento à classes de serviços sensíveis ao atraso, o problema é dividido em duas partes: *i*) as VNFs são mapeadas através de um algoritmo baseado em *simulated annealing*; e *ii*) é feito o roteamento entre as VNFs. No algoritmo proposto são realizadas transmissões paralelas de dados entre os roteamentos, dividindo o fluxo entre as VNFs em múltiplos sub-fluxos. A ideia de sub-fluxos trafegando em paralelo e consolidando-se em um *endpoint* é colocada para reduzir o volume de tráfego passante em um só caminho, mitigando problemas de fragmentação do SN. Outra justificativa para os sub-fluxos, apresentada pelos autores, é a melhora da velocidade e da garantia de transmissão. Porém, em cenários reais, tal algoritmo tem uma aplicabilidade limitada, na qual poucos NS permitem que os fluxos de conexões sejam divididos. Nesta tese, de modo semelhante a Bari et al. (2019), é adotado que o fluxo de uma SFCs não pode ser dividido.

De modo a caracterizar cenários reais, algoritmos de alocação de recursos devem lidar com as SFCs conforme elas são requisitadas. Neste ambiente, um tratamento *online* se faz necessário, e gera uma análise focada em fatores inerentes ao processamento sequencial das SFCs. Tratamento adotado em Bari et al. (2015); Sun et al. (2016); Khebbache et al. (2017); Kuo et al. (2016); Bari et al. (2019); Li et al. (2020); Wang et al. (2021a).

Bari et al. (2015) propõem um algoritmo exato para resolver o problema VNF-PC, em um ambiente de integração entre ISPs e redes corporativas. Os autores também apresentam uma heurística baseada no algoritmo de Viterbi para resolver o problema em um baixo tempo computacional. Ambos os algoritmos minimizam o OPEX envolvido no mapeamento das SFCs, sendo que a função objetivo minimiza na mesma métrica os custos de instanciação de VNFs, energia usada, e encaminhamento de dados. Na função objetivo aplicada, tais custos são ponderados por parâmetros calibrados manualmente para priorizar determinados valores em detrimento a outros. Similarmente, nesta tese, a função objetivo minimiza o OPEX gerado, mas com o objetivo de maximizar o lucro dos provedores. Porém, ao invés de utilizar parâmetros de ponderação, define-se o lucro dos provedores como a receita gerada pelos mapeamentos das SFCs, decrementado pelas despesas de se prover tais serviços. Diferentemente de Bari et al., também são considerados nesta tese o atraso de propagação gerado nos arcos e de processamento nas VNFs.

Quando não se considera a distância entre o posicionamento das VNFs na cadeia de serviço e os *endpoints* no mapeamento de uma SFC, o atraso fim a fim pode aumentar e gerar uma degradação no serviço prestado. Para minimizar o atraso fim a fim, Kuo et al. (2016) apresentam uma heurística baseada em programação dinâmica para o problema *VNF Placement and Path Selection* (variante do VNF-PC), com um tratamento estático, e que possui como objetivo balancear a utilização de arcos e servidores do SN. Similarmente a Kuo et al., esta tese trata de aspectos relativos ao atraso fim a fim, mas opta-se

por ponderar essa métrica de maneira que seja viável para o usuário final, respeitando os limites de QoS, mas que também incentive o compartilhamento de recursos do SN, maximizando o lucro gerado.

Khebbache et al. (2017) apresentam duas heurísticas, e um algoritmo exato para solucionar o VNF-PC. A primeira heurística, chamada *Matrix-Based optimization*, é baseada em matrizes de instanciação e ordem de VNFs, em que o objetivo é descobrir pontos mais relevantes do SN para efetuar-se o mapeamento de uma SFC. A ideia da segunda heurística, chamada *Multi-Stage*, é construir um grafo estendido com foco no número de servidores disponíveis e nas VNFs existentes. Por ser um problema de minimização, um algoritmo exato é usado para gerar um limite inferior, criando um limite para fins de comparação. Ambas as heurísticas propostas priorizam pela eficiência do modelo, atribuindo uma quantidade mínima de recursos do SN para atender às demandas. Do mesmo modo que Khebbache et al., esta tese apresenta uma heurística e utiliza o solver CPLEX para resolver o modelo em ILP, e gerar um limite de otimização usado como *baseline*. Entretanto, diferentemente desta tese, as SFCs usadas na avaliação do algoritmo proposto por Khebbache et al. são aleatórias, e os algoritmos apresentados, caso necessário, não podem ser aplicados em um ambiente *offline*.

Bari et al. (2019) desenvolveram um orquestrador de SFCs que efetua o posicionamento, encadeamento, e migração de VNFs. O algoritmo proposto consiste em uma heurística baseada em Busca Tabu. Referido trabalho atua com uma chegada de SFCs *online*, onde cada SFC possui um tempo de vida, e as VNFs demandam recursos de CPU, restrições também adotadas nesta tese. O objetivo do algoritmo é reduzir a emissão de carbono recorrendo aos componentes do SN que usam energias limpas. Nesse aspecto, o algoritmo efetua migrações de SFCs para locais benéficos, estimulando o compartilhamento de recursos físicos, e permitindo que componentes do SN, como *switches* e servidores, sejam colocados em um estado de baixo consumo de energia. Porém, o trabalho de Bari et al. não define explicitamente quais capacidades são atribuídas às restrições, e não considera recursos em termos de memoria MB ou largura de banda em $Mbps$. Por fim, similarmente a Bari et al. (2015); Khebbache et al. (2017), Bari et al. (2019) propõem um algoritmo exato para resolver instâncias pequenas, e como é um problema de minimização, gerar um limite inferior para comparações. Uma estratégia semelhante é adotada nesta tese, mas em nosso caso, como procuramos maximizar, buscamos um limite superior.

Em um ambiente estático, Chen et al. (2022) propõem um algoritmo exato para minimizar os custos no problema de mapeamento de SFCs. O algoritmo proposto consiste em separar as redes virtuais e físicas em camadas, calculando a disponibilidade de recursos em cada camada, e implantando cada VNF na camada que oferecer um maior ganho. Os autores utilizam a decomposição de Benders para decompor o problema original em dois subproblemas: o de mapeamento hierárquico de VNFs e o de roteamento entre nós. Segundo os autores, o algoritmo pode reduzir o atraso de processamento de uma SFC, e

ao mesmo tempo, reduzir os custos envolvidos. O modo de geração das SFCs adotado por Chen et al. é usado nesta tese, mas, complementarmente, utilizamos parâmetros oriundos de SFCs reais de serviços de rede. Negativamente, e de modo diferente ao aplicado nesta tese, Chen et al. não diferenciam recursos de processamento e memória das SFCs, e nem definem unidades de medida para mensurar os parâmetros adotados.

2.2.2 Abordagens Estáticas e Dinâmicas

Luizelli et al. (2015) apresentam um algoritmo exato e uma heurística para resolver o VNF-PC de modo estático, com a função objetivo de reduzir o número de VNFs usadas. A heurística proposta no trabalho recorre ao próprio modelo em ILP para realizar as iterações. Nesse caso, cada iteração é feita com um limite de tempo. No final de cada iteração, a melhor solução é armazenada, e o número de funções instanciadas na iteração anterior torna-se uma restrição para a iteração atual. Apesar de atuar em um cenário com compartilhamento de recursos e roteamento das VNFs, os autores não tratam restrições como consumo de memória e não quantificam custos de operação, parâmetros inseridos na modelagem do problema tratado nesta tese. Similarmente a Luizelli et al., é assumido nesta tese que VNFs ativas podem ser compartilhadas entre diferentes SFCs. Em nível de pesquisa, o modelo computacional em ILP proposto por Luizelli et al. é bem definido, contemplando vários aspectos do problema real. Contudo, os algoritmos propostos, mesmo o heurístico, possuem um tempo computacional da ordem de minutos, sendo inviável sua aplicação para SFCs sensíveis ao tempo de processamento. De forma diferente, nesta tese, os algoritmos heurísticos são desenvolvidos para reduzir tal tempo de execução para a ordem de milissegundos.

Lange et al. (2017) abordam o problema VNF-PC em um ambiente *offline*. Os autores objetivam em seu trabalho responder heuristicamente às seguintes questões: quantas instâncias de VNF são necessárias para atender uma SFC? Em qual local as VNFs devem ser alocadas? Quais SFCs devem ser alocadas em cada instância de VNF? Qual política de roteamento estipular para encadear as VNFs através dos arcos do SN? Para esclarecer estas questões, os autores apresentam uma heurística baseada em *simulated annealing*. Referido algoritmo gera um conjunto de soluções, e usa a fronteira de *Pareto* para encontrar a melhor solução otimizando os objetivos: atraso total, número de saltos, número de instâncias de VNFs, e uso de CPU. Segundo os autores, o algoritmo proposto pode ser adaptado para efetuar reotimizações, tratando os aspectos dinâmicos do problema, mas o trabalho apresentado avalia somente aspectos *offline* e estáticos. Diferentemente, nesta tese, são propostas estratégias de reotimização, sendo analisadas suas vantagens e

desvantagens.

Houidi et al. (2017) propõem um algoritmo exato e uma heurística gulosa para resolver VNF-PC com escalonamento de SFCs em um ambiente *online*. Os autores exploram o *tradeoff* entre o custo de mapeamento e o desempenho ao redimensionar e migrar instâncias VNF de acordo com as demandas dos usuários. O objetivo do algoritmo é minimizar o tempo de interrupção do serviço para efetuar as migrações de SFCs. Para esse fim, a função objetivo é definida de forma a priorizar os mapeamentos das VNFs em servidores físicos com mais recursos disponíveis. A heurística proposta pelos autores reduz significativamente o tempo computacional em relação ao algoritmo exato, sendo de $23s$ para $11s$ no cenário menor, e de $46s$ para $15s$ no cenário maior. O modelo proposto pelos autores não considera aspectos de atraso e compartilhamento de VNFs, fatores que podem ser adversidades quando são propostas migrações de SFCs. Negativamente, o tempo computacional percebido na proposta de Houidi et al. (2017) é alto, e inviabiliza a aplicação do algoritmo no mapeamento de SFCs sensíveis ao atraso, que necessitam de um atendimento em tempo real. Diferentemente, nesta tese o foco é dado em algoritmos que possam ser executados em frações de segundos.

Sun et al. (2016) tratam o problema do mapeamento de SFCs através de um algoritmo exato e uma heurística para o cenário *offline*, sendo a heurística adaptada também para um tratamento *online*. A heurística proposta busca unir diferentes SFCs com mesmo nó de origem e destino em uma só SFC, ação chamada de afiliação. A heurística atua em duas etapas sequenciais: mapeamento de nós, feito de forma gulosa; e roteamento de arcos, efetuado através do caminho mais curto. Similarmente a Sun et al., a heurística proposta nesta tese atua de forma coordenada em fases distintas, uma para a instanciação de VNFs, e outra para o roteamento, com um algoritmo de busca em largura (*Breadth-First Search*, BFS).

Um dos desafios das redes baseadas em tecnologias NFV é aumentar lucro dos provedores (Chen et al., 2020). Consonantemente, nesta tese, o objetivo proposto é maximizar o lucro. De acordo com o exposto por Chen et al., o escalonamento de SFCs pode ser necessário caso os custos de um serviço sejam alterados. Chen et al. propõem um algoritmo guloso para prover o mapeamento de SFCs que reduz o custo de roteamento causado pelas operações de escalonamento. A ideia do algoritmo de escalonamento pode ser dividida em: *Scaling out*, nesta etapa as VNFs ativas são verificadas, e caso alguma VNF precisar ser escalonada, ela é reposicionada em um dos k caminhos mais curtos, em nós virtuais adjacentes a ela; e *Scaling in parts*, é uma etapa similar a uma busca local, neste caso, cada SFC ativa é percorrida, e caso seja possível escalonar alguma VNF, este é realizado. Os algoritmos propostos por Chen et al. inspiraram o desenvolvimento da busca local da heurística proposta nesta tese. Ademais, estudos que efetuam o escalonamento, chamado nesta tese de reotimização, são explorados no Capítulo 4.

Pham (2022) abordam o problema de mapeamento e migração de SFCs com o

objetivo de minimizar os custos de mapeamento com realocação em um ambiente *offline*, e efetuando um roteamento dinâmico entre as VNFs. No referido trabalho, são propostos dois algoritmos exatos, e dois heurísticos. Os algoritmos exatos são usados para obter as soluções ótimas de colocações e migrações de SFCs, obtendo um limite de valores para serem usados para avaliar as heurísticas. As heurísticas de migração são baseadas na metaheurística *Simulated Annealing* e em aprendizado por reforço. Segundo os autores, os resultados obtidos pelas heurísticas são próximos aos da solução ótima nos cenários avaliados, mas a heurística que utiliza aprendizado por reforço deve ser retreinada para se adequar ao comportamento dos novos dados, o que pode atrasar a execução.

São sumarizados na Tabela 2.2 os trabalhos relacionados à virtualização em redes, resolvidos com abordagens exatas ou heurísticas, e utilizados neste referencial teórico. Os artigos procedentes dos estudos envolvidos na construção desta tese foram inseridos e estão destacados em negrito. Complementarmente à Tabela 2.2, a Tabela 2.3 mostra um detalhamento comparativo entre alguns pontos trabalhados nesta tese, a alguns trabalhos da literatura.

Tabela 2.2: Trabalhos abordados com otimização exata e heurística

Problema	**Função Objetivo**	**Características**	**Referência**
Posicionamento e Roteamento de VMs	minimiza o custo do tráfego de dados	heurístico e *online*	(Jiang et al., 2012)
VNF-CC e VNF-PC tratados separados (VNE adaptado)	maximiza a banda residual, minimiza latência e nós usados	VNF-CC heurístico, VNF-PC exato e *offline*	(Mehraghdam et al., 2014)
VNF-PC	minimiza os custos de energia	exato, heurístico e *online*	(Bari et al., 2015)
VNF-PC	minimiza o número de instâncias de VNFs	exato, heurístico e *offline*	(Luizelli et al., 2015)
VNF-PC (VNE adaptado)	minimiza o uso de nós e o custo dos arcos	exato e *offline*	(Baumgartner et al., 2015)
VNF-CC e VNF-PC tratados juntos	minimiza o uso de banda do SN	heurístico e *offline*	(Beck and Botero, 2015)
VNF-P	minimiza o custo total da rede	exato e *offline*	(Cohen et al., 2015)
VNF-P com resiliência em nós e arcos	minimiza o número de instâncias utilizadas	exato e *offline*	(Hmaity et al., 2016)
VNF *placement and path selection*	minimiza o tamanho das rotas físicas usadas	heurístico e *online*	(Kuo et al., 2016)
SFC *deployment* (VNF-PC)	minimiza o custo total de mapeamento	exato, heurístico, *offline* e *online*	(Sun et al., 2016)
VNF-CC	minimiza a banda total demandada pela SFC	metaheurístico Busca Tabu	(Gil-Herrera and Botero, 2017)
VNF-PC	minimiza as interrupções e reconfigurações	exato, heurístico, *online* dinâmico	(Houidi et al., 2017)
VNF-CC	minimiza a banda total demandada pela SFC	exato	(Ocampo et al., 2017)
VNF-PC	minimiza o custo de migração e o atraso fim a fim	exato, heurístico, *offline* e dinâmico	(Zhang et al., 2017)
VNF-PC	minimiza saltos, atraso, cpu e instâncias usadas	heurístico e *offline*	(Lange et al., 2017)
VNF-PC	maximiza o lucro dos SPs	exato e *offline* com geração de colunas	(Liu et al., 2017)
VNF-PC	minimiza a utilização de arcos	heurístico *online*	(Khebbache et al., 2017)
VNF-PC	minimiza a alocação de recursos	exato, *math-heuristic* e *offline*	(Gao et al., 2018)
VNF-PC	minimiza o consumo de energia	busca Tabu, exato e *online*	(Bari et al., 2019)
VNF-PC com VNF-CC	**maximiza o lucro**	**exato e *online*** (*short paper*)	(Araújo et al., 2019a)

VNF-PC com VNF-CC	**maximiza o lucro**	**exato e *online*** (*full paper*)	(Araújo et al., 2019b)
VNF-PC com resiliência	maximiza o lucro do SPs	exato, heurístico *offline* e *online*	(Li et al., 2020)
VNF-PC	minimiza os custos	heurístico, *online* com escalonamento dinâmico	(Chen et al., 2020)
VNF-PC	**maximiza o lucro**	**heurístico e *online***	(Araújo et al., 2021a)
VNE	**maximiza o balanceamento de carga e o lucro**	**exato, *online*, *offline*, periódico, e com múltiplos domínios de rede**	(Araújo et al., 2021c)
VNF-PC	minimiza o consumo de recursos	heurístico e *offline*	(Wei et al., 2022b)
VNF-PC com migração	minimiza os custos de mapeamento e realocação	exato, aprendizado por reforço, *Simulated Annealing* e *offline*	(Pham, 2022)
VNF-PC (VNE adaptado)	maximiza receita	exato, relaxado, *online* em *slots* de tempo	(Maity et al., 2022)
VNF-PC	maximiza a confiabilidade e minimiza os custos	exato e com decomposição de Benders e *online*	(Chen et al., 2022)
VNF-PC	**maximiza o lucro**	**exato, *online*, e com estratégias de reotimização**	(Araujo et al., 2023)

Fonte: Elaborado pelo autor.

Tabela 2.3: Comparação desta tese com outros trabalhos da literatura

Característica	Este trabalho	Luizelli et al. (2015)	Gao et al. (2018)	Khebbache et al. (2017)	Khoshkholghi et al. (2019)	Bari et al. (2019)	Sharma et al. (2020)	Wei et al. (2022b)	Maity et al. (2022)	Chen et al. (2022)
Online (*on*)/ *offline* (*off*)	on/off	off	off	on	off	on	off	off	on	on
Número de VNFs por SFC	$[2, 6]$	$[2, 5]$	n/i	$[2, 10]$	$[4, 10]$	$[2, 6]$	$[4, 6]$	3	n/i	$[4, 6]$
Capacidade de servidor	*mem*/*cpu*	*cg*	*cpu*	*cpu*	*mem*/*cpu*	*cpu*	*cpu*	*mem*/*cpu*	*mem*/*cpu*	*cg*
Capacidade de enlace	*bw*	*cg*	*bw*	*bw*	*bw*	*cg*	*bw*	*bw*	*bw*	*bw*
Cálculo de atraso	nó/enlace	nó/enlace	nó/enlace	✗	✗	nó/enlace	✗	enlace	transmissão	nó/enlace
Topologia de rede real	✓	✗	✗	✓	✓	✓	✗	✗	✓	✗
Parâmetros reais de SFCs	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗
Algoritmo heurístico	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓
Algoritmo exato	✓	✓	✓	✓	✗	✓	✓	✗	✓	✓
Baseline	ILP	ILP	heurística	ILP	heurística	ILP	ILP	heurística	ILP VNE	heurística
Ordem do tempo de execução	*ms*	*min*	*seg*	*seg*	*min*	*ms*	*ms*	*ms*	n/i	*seg*
Reotimização/migração	✓	✗	✗	✗	✗	✓	✗	✓	✗	✗

cg = Capacidade genérica, não definida explicitamente em termos de recurso e unidade de medida
n/i = Não informado
min = Minutos
seg = Segundos
ms = Milissegundos
bw = Largura de banda
cpu = Capacidade de processamento
mem = Capacidade de memória

Fonte: Elaborado pelo autor.

2.3 Aprendizado de Máquina

Para Yi et al. (2018), o suporte à implantação de VNFs em larga escala é uma tarefa difícil e lenta, mas que pode ser inteligente e automatizada. As redes baseadas em NFV estão sendo amplamente adotadas em todo o mundo, e estão tornando-se cada

vez mais heterogêneas em recursos demandados. Nesse sentido, uma profusão de dados sobre o gerenciamento desses recursos de rede já está sendo criada e pode ser coletada. À vista disso, uma ação que pode facilitar o processo de mapeamento das redes virtuais é analisar os dados coletados de gerenciamentos de recursos anteriores, convertê-los em informações, e transformar essas informações em estratégias de gerenciamento de recursos (Gebremariam et al., 2019; Zhao et al., 2019). Neste caso, o AM é usado justamente para analisar os dados coletados, aprender comportamentos, e gerar estratégias, permitindo que a rede se comporte de uma forma autoadaptável.

De acordo com Xie et al. (2019), o aprendizado supervisionado é uma técnica de aprendizagem rotulada. Neste caso, os algoritmos recebem um conjunto de dados de treinamento para construir um modelo de sistema que represente a relação aprendida com os próprios dados de treino e seus respectivos rótulos. No caso das redes de computadores, esses dados podem ser características obtidas das SFCs ou do SN, fundamentos utilizado nesta tese. Após o treinamento do modelo, quando uma nova SFC chega para ser mapeada, o modelo pode ser usado para direcionar o processamento da nova SFC no espaço de soluções, e obter uma saída de modo mais automatizado e assertivo.

Quando é abordada a alocação de VNFs em nuvens, existe uma necessidade de resposta rápida em relação às instanciações de VNFs para manter níveis aceitáveis de QoS (Shi et al., 2015). Para resolver esse problema, Shi et al. aplicam o modelo de Markov para alocar dinamicamente os recursos oferecidos pelo SN. Adicionalmente, os autores utilizam um modelo de Redes Bayesianas para monitorar o uso de recursos demandados no passado e tentar prever demandas futuras. Neste algoritmo, à medida que as SFCs chegam e são mapeadas, o modelo é retreinado, e as probabilidades de previsão são atualizadas. Já a etapa de mapeamento é realizada minimizando o custo e buscando manter a viabilidade em relação às restrições de QoS pré-definidas.

Kim and Kim (2017) apresentam um algoritmo com aprendizado por reforço para resolver os problemas VNF-CC e VNF-PC em simultâneo. Para o aprendizado, são utilizadas características retiradas do SN e das VNFs já instanciadas. O algoritmo realiza o aprendizado aumentando ou diminuindo o valor da recompensa de cada ação conforme as condições de CPU, memória e banda do SN. Dessa forma, busca-se distribuir a demanda uniformemente por recursos do SN. Embora os resultados de trabalho de Kim and Kim sejam promissores, Sun et al. (2018) ressaltam que usar essa técnica em situações reais pode resultar em longos tempos de treinamento, e consequentes atrasos no processamento.

Kuang et al. (2018) abordam o problema de posicionamento dos controladores em redes com tecnologias SDN. Em tal problema, os controladores precisam ser colocados próximos aos locais onde os *switchs* de rede estão localizados. Para tal, o algoritmo *K-means* hierárquico é proposto para particionar a rede, através da geração de agrupamentos que consideram minimizar a latência entre os *switches* e controladores, e balancear a carga de cada controlador. Os resultados experimentais mostram que a partição feita com o

algoritmo proposto é mais balanceada do que com o algoritmo *K-means* tradicional.

Chen et al. (2019) apresentam um algoritmo para a alocação de recursos em redes de modo *offline* e estático. Tal algoritmo agrupa fluxos de NS similares com base na correlação das VNFs. Uma vez os fluxos agrupados, o algoritmo tenta mapear VNFs iguais de cada fluxo de um mesmo *cluster* em um único nó físico. Desse modo, reduzindo a quantidade de instâncias de VNFs, nós físicos e arcos usados. Apesar de introduzir conceitos de aprendizado não supervisionado para identificar características entre diferentes SFCs, Chen et al. limitam-se a avaliar um cenário pequeno, e com pouca variabilidade de recursos. Os autores consideram que cada fluxo pode demandar no máximo 2 VNFs diferentes, também não são considerados aspectos de chegadas e saídas de SFCs.

Subramanya and Riggio (2019) abordam o escalonamento VNFs com um modelo de rede neural, e o posicionamento das VNFs com ILP. No trabalho, o AM é aplicado para, com base em um histórico, propiciar que a rede aprenda e antecipe às necessidades de instanciações de VNFs que ainda vão ser demandadas. O modelo proposto minimiza o atraso fim a fim, utiliza restrições de capacidades de recursos, sendo resolvido pelo solver CPLEX. Negativamente, os autores assumem que todos os nós do SN oferecem uma quantidade igual de recursos, sendo que os recursos não são definidos separadamente em valores de CPU e memória, e também não são tratados aspectos de roteamento.

Uma abordagem apresentada no trabalho (Wei et al., 2022a) aborda a reconfiguração de *slices* de rede sob incerteza de demanda. Segundo os autores, a alocação de recursos para prover um *slice* pode ser feita de forma orientada ao modelo, ou aos dados. Os métodos orientados aos modelos podem causar superprovisionamento de recursos devido a uma falta de mecanismo de previsão de demanda. Por outro lado, os métodos orientados aos dados são pouco práticos, pois requerem diversas migração de VNFs fator que requer operações caras e demoradas. Para resolver este problema, os autores apresentam uma heurística híbrida que integra estes dois modelos, chamada *Hybrid Model-Data driven* (HMD). Neste caso um processo de predição é feito com RNN para gerar previsões de futuras demandas de tráfego. O HMD utiliza conjuntamente as previsões de demandas de tráfego e a otimização robusta proativa, para fazer a reconfiguração dos *slices* de rede.

Dalgkitsis et al. (2022) estudam o problema de mapeamento de SFCs visando reduções de latência e eficiência energética. O trabalho introduz um algoritmo para mapear SFCs em uma rede multidomínio, utilizando aprendizado por reforço. Tal algoritmo emprega agentes de aprendizado distribuídos em cada domínio que executa o mapeamento das VNFs localmente. Referido artigo utiliza de restrições de conservação de fluxo, atraso máximo tolerado, capacidades de processamento e memória, conceitos também adotados nesta tese. Contudo, trata-se de um algoritmo *offline* e não são feitas análises do tempo de processamento, fator fundamental para aplicações sensíveis ao atraso.

Alguns dos trabalhos citados no referencial teórico, são sumarizados, classificados, e mostrados na Tabela 2.4. Os artigos provenientes dos estudos relacionados a construção

desta tese foram introduzidos, e destacados em negrito.

Tabela 2.4: Trabalhos abordados com AM

	Objetivo	**Abordagem**	**Referência**
ATM	Controle de Admissão	*Neural Network*	(Cheng and Chang, 1997)
VNE	Atribuição de recursos	*Reinforcement Learning*	(Mijumbi et al., 2014)
NFV	Alocação de recursos	*Markov Decision Process*	(Shi et al., 2015)
NFV	Predição de recursos	*Neural Network*	(Shi et al., 2015)
SDN	Classificação de tráfego	*Random Forests*, *Gradient Boosting* e *Stochastic Gradient Boostin*	(Amaral et al., 2016)
VNE	Controle de Admissão	*Neural Network*	(Blenk et al., 2016)
NFV	Predição de recursos	*Neural Network*	(Mijumbi et al., 2017)
SDN	Detecção de ataque DDoS	*Support Vector Machines*	(Hu et al., 2017)
SDN	Planejamento dinâmico de SFCs	*Reinforcement Learning* e *Q-Learning*	(Kim and Kim, 2017)
WAN	Posicionamento de controladores	*Hierarchical* e *K-means*	(Kuang et al., 2018)
NFV	Predição de recursos	*K-means*, *Naive Bayes*, *Support Vector Machines*, *Neural Network*, *Random Forest*, e *Gradient Boosted Tree*	(Le et al., 2018)
NFV	Posicionamento de VNFs	*Kruskal Clustering*	(Chen et al., 2019)
NFV	Escalonamento de VNFs	*Neural Network*	(Subramanya and Riggio, 2019)
VNE	**Predição de método de resolução entre exato ou heurístico**	**AdaBoost, Naive Baye *Neural Network*, *Random Forest*, e *Gradient Boosted Tree***	(Araújo et al., 2020)
NFV	**Redução do espaço de solução**	**algoritmo exato e k-*Means***	(Araújo et al., 2021b)
NFV	**Redução do espaço de solução com retreinamento do modelo de AM**	**algoritmo exato, k-*Means* métrica de Kullback-Leibler**	(Araujo et al., 2022)
NFV	Predição de tráfego	*Recurrent Neural Networks*	(Wei et al., 2022a)
NFV	Posicionamento de VNFs	*Reinforcement Learning*	(Dalgkitsis et al., 2022)

Fonte: Elaborado pelo autor.

Capítulo 3

Modelos real e computacional para o problema VNF-PC

Neste Capítulo é apresentada uma formulação para o VNF-PC. Para tal, uma definição é proposta na Seção 3.1. Uma formulação computacional, descrita em ILP, com restrições que abordam os cenários *online* e *offline*, e diferentes estratégias de reotimização é apresentada na Seção 3.2. Por fim, são realizadas as considerações finais na Seção 3.3. Os papéis das entidades envolvidas no problema VNF-PC são definidos no Apêndice B.

3.1 Definição Formal do Problema

3.1.1 SN

Similarmente a Luizelli et al. (2017); Bari et al. (2019), o VNF-PC pode ser definido da seguinte forma. Seja o SN modelado como um grafo direcionado ponderado[1] $G = (N, L)$, em que N e L correspondem respectivamente ao conjunto de servidores (N-PoPs) e aos arcos do SN. Cada servidor $i \in N$ representa um local físico em que uma função de rede pode ser posicionada, e possui uma oferta de CPU (*core*) e de memória (*megabyte*, MB) diferentes, representadas por C_i e M_i respectivamente. Do mesmo modo Bari et al. (2015); Luizelli et al. (2017), é assumido que qualquer servidor $i \in N$ pode instanciar qualquer função de rede solicitada, desde que existam recursos residuais disponíveis.

Da mesma maneira que Jia et al. (2018), cada arco $(i, j) \in L$ possui uma capacidade de banda máxima BW_{ij} (Megabits por segundo, Mbps) e um atraso associado (milissegundos). O atraso é calculado pela função $atraso(i, j)$, gerado com base na distância calculada pela fórmula de Haversine (descrita no Apêndice B), levemente perturbada por

[1] Um grafo é ponderado quando suas arestas possuem um peso.

um número aleatório τ. Similarmente a Barbi et al. (2017), é utilizada a fórmula de Haversine para calcular precisamente a distância entre dois servidores físicos localizados no globo terrestre. Pelo SN ser representado como um grafo direcionado, não necessariamente $BW_{ij} = BW_{ji}$.

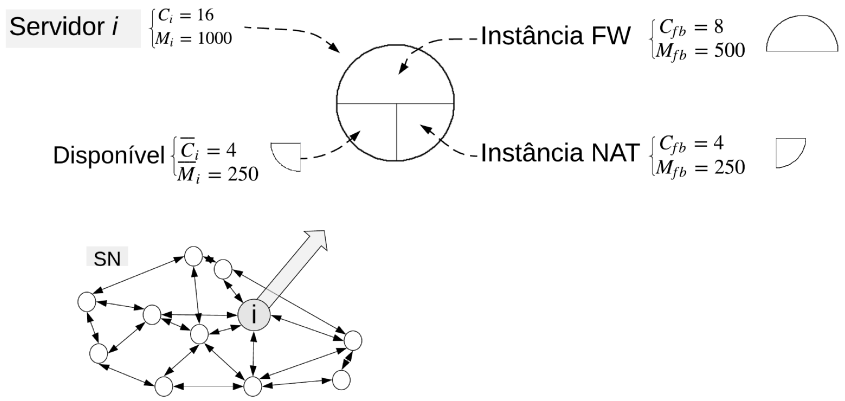
3.1.2 Funções Virtuais de Rede

Nesta tese, diferentes tipos NFs podem ser virtualizadas, *e.g.*, NAT, FW, IDS, etc. Cada NF a ser virtualizada deve ser atribuída a uma instância de VNF, que, idealmente, deve ser posicionada pelo SP em um ponto estratégico do SN para gerar menores custos, e cumprir com as restrições de QoS de cada SFC. Um dos desafios do VNF-PC é posicionar um número de instâncias de VNFs ideal para não haver recursos mal aproveitados, ociosos, ou ainda faltantes. Assim que uma instância de VNF é posicionada, ela pode ser compartilhada por diferentes SFCs simultaneamente.

Seja F o conjunto das NFs existentes, *i.e.*, $F = \{NAT, FW, TM, ...\}$. Cada função de rede $f \in F$ pode ser solicitada por uma SFC $v \in V$, atribuída e virtualizada sobre uma instância de VNF já posicionada. Seja B_f o conjunto de todas as instâncias de NFs de cada função $f \in F$, de diferentes capacidades, e podem ser posicionadas sobre um servidor genérico $i \in N$. Cada instância virtualizada $b \in B_f$ possui uma quantidade de recursos limitada, oferecida para o atendimento de uma ou mais VNFs, sendo C_{bf} a quantidade de CPUs (*core*, núcleos) e M_{bf} a quantidade de memória (MB) disponíveis. A Figura 3.1 mostra duas instâncias de funções de rede $f \in F$ (FW e NAT) posicionadas sobre um servidor $i \in N$ pertencente a um SN. É possível observar que, mesmo posicionadas as duas funções de rede, ainda existem recursos ociosos no servidor i. Tais recursos disponíveis podem ser usados caso seja necessário implantar outra função de rede $f \in F$, ou ainda aumentar (redimensionar) uma instância virtual de função de rede $b \in B_f$ já posicionada.

Quando uma instância de função de rede $b \in B_f$ é posicionada sobre um servidor físico do SN, além de oferecer fracionadamente quotas de C_{bf} e M_{bf} para atender às VNFs, ela consome uma parcela fixa de recursos C_{bf} e M_{bf} do respectivo servidor. Neste caso, ao virtualizar-se uma instância $b \in B_f$ sobre um servidor $i \in N$, deve garantir-se que os recursos físicos disponíveis no servidor $i \in N$ atenderão as demandas de todas as instâncias $b \in B_f$ de funções $f \in F$ posicionadas sobre ele.

Figura 3.1: Posicionamento de duas VNFs sobre um servidor $i \in N$ pertencente a um SN

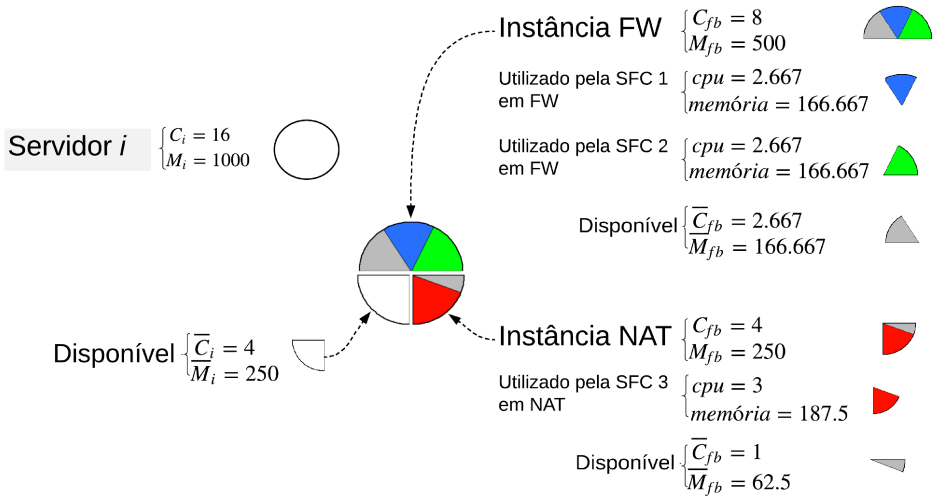


Fonte: Elaborado pelo autor.

Do mesmo modo que Bari et al. (2019), cada NF $f \in F$ possui um atraso de processamento, além de gerar um aumento ou redução de pacotes, ocasionado pelo processamento referente à função de rede $f \in F$ requerida sobre os dados trafegados. Esse aumento ou redução de fluxo é referenciado neste trabalho como η^f. Como exemplo, uma VNF FW pode potencialmente descartar alguns pacotes do fluxo de dados ($e.g.,\eta^{FW} = 0.9$), e reduzir a demanda por banda; ou ainda uma VPN pode aumentar o fluxo de dados ($e.g.,\eta^{VPN} = 1.2$) devido ao provimento de encapsulamento e segurança (Gu et al., 2019).

A Figura 3.2 mostra uma alocação de recursos do SN para atender a duas funções de rede $f \in F$ (FW e NAT) posicionadas e compartilhadas sobre o servidor $i \in N$ mostrado na Figura 3.1. No exemplo, as duas instâncias de NFs conseguem atender a 3 SFCs de SFCs diferentes, no caso, a instância da VNF FW atende às SFCs 1 e 2 simultaneamente, e a instância da VNF NAT atende à SFC 3. Em cada VNFs instanciada existe uma sobra de recursos não utilizados na própria instância $b \in B_f$ em questão, sendo que tal sobra pode ser usada caso alguma SFC aumente sua demanda no futuro.

Figura 3.2: Compartilhamento de instâncias de VNFs sobre um servidor



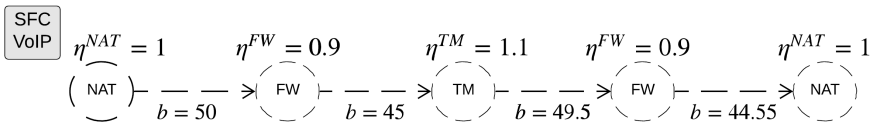
Fonte: Elaborado pelo autor.

3.1.3 SFCs

Em um ambiente de redes baseadas em NFV, um NS é oferecido pelo TSP através de uma sequência encadeada de VNFs. Cada sequência de VNFs possui um encadeamento específico para atender a um NS diferente. Segundo Pham et al. (2017); Askari et al. (2019), os clientes finais podem contratar diferentes configurações de SFCs, *e.g.*: RA - NAT → FW → TM → VOC → IDS; *Massive IoT* - NAT → FW → IDS; e Jogos *online* - NAT → FW → VOC → WOC → IDS.

Cada VNF de cada SFC deve ser posicionada e roteada sobre o SN de maneira a respeitar estritamente a ordem de encadeamento disposta pelo cliente. Nessa situação, o fluxo de dados (entrada/saída de bits) que percorre a SFC pode ser potencialmente alterado devido à ação de processamento que as VNFs encadeadas efetuam sobre os dados encaminhados da origem ao destino (Bari et al., 2019; Chen et al., 2022). A Figura 3.3 exemplifica um NS de VoIP que inicialmente requisitava uma largura de banda de 50 *Kbps*, mas que devido aos processamentos realizados pelas VNFs encadeadas, gera uma perda/ganho de pacotes (η^f), ação que diminui/aumenta a necessidade de reserva de banda.

Figura 3.3: Fluxo de dados de uma SFC para o provimento do NS VoIP

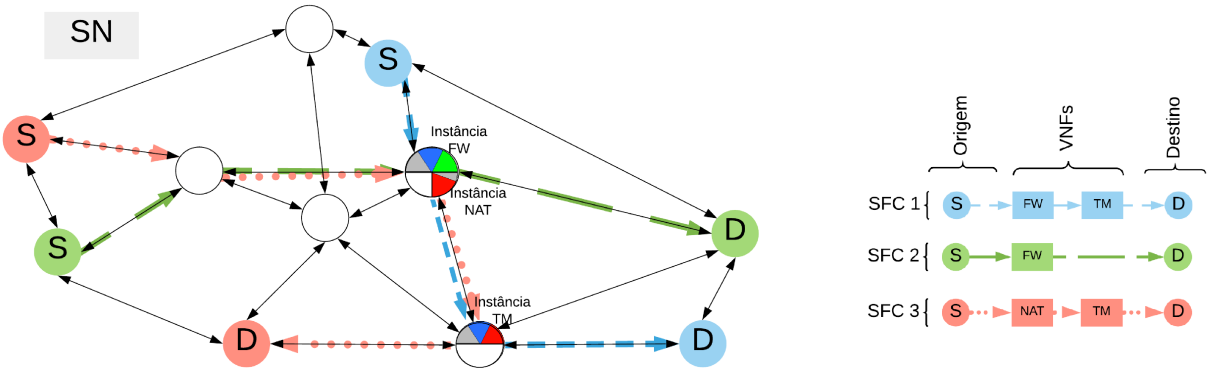


Fonte: Elaborado pelo autor.

Seja V o conjunto das SFCs. De modo similar a Bari et al. (2016); Fischer et al. (2013); Maity et al. (2022); Roig et al. (2019), cada SFC $v \in V$ possui um tempo de entrada t^v_{in}, um tempo de duração t^v_{dr}, e um atraso máximo de comunicação, dado por t^v_{dl}. Similarmente a Bari et al., cada SFC $v \in V$ é representada por um grafo direcionado acíclico $G^v = (N^v, L^v)$, em que N^v representa o conjunto de nós virtuais e L^v o conjunto de todos os arcos virtuais. Seja $N^v = \{s^v \cup d^v \cup F^v\}$ a união dos pontos de origem e destino s^v e d^v, e F^v o conjunto das funções virtuais demandadas ($F^v \subseteq F$). Os nós virtuais s^v e d^v não possuem demandas de memória, processamento ou geram um atraso, mas devem ser atribuídos estritamente sobre servidores $i \in N$ específicos, os quais representam os clientes finais. Em contrapartida, cada função virtual $f \in F^v$ possui uma demanda por CPU c^v_f e memória m^v_f, além de gerar um atraso de processamento dado pela função $atraso(k)$. Da mesma forma que Luizelli et al. (2017), assume-se que cada função de rede $f \in F^v$ pode ser atribuída a qualquer instância $b \in B_f$ posicionada sobre qualquer servidor $i \in N$ que possua recursos disponíveis. Cada arco virtual $(k, l) \in L^v$ possui uma demanda por banda bw^v_{kl} requerida para o mapeamento.

A Figura 3.4 exemplifica 3 SFCs, posicionadas e encadeadas com sucesso sobre um SN. Em redes baseadas em tecnologias NFV, o compartilhamento de recursos é estimulado para gerar melhorias relativas aos custos envolvidos. Neste sentido, as 3 SFCs estão dividindo um mesmo servidor. Nota-se que as SFCs 1 e 2 compartilham a mesma instância de VNF FW e as SFCs 1 e 3 compartilham a mesma instância da VNF TM, que também tem seus custos de implantação e operação divididos entre as partes envolvidas.

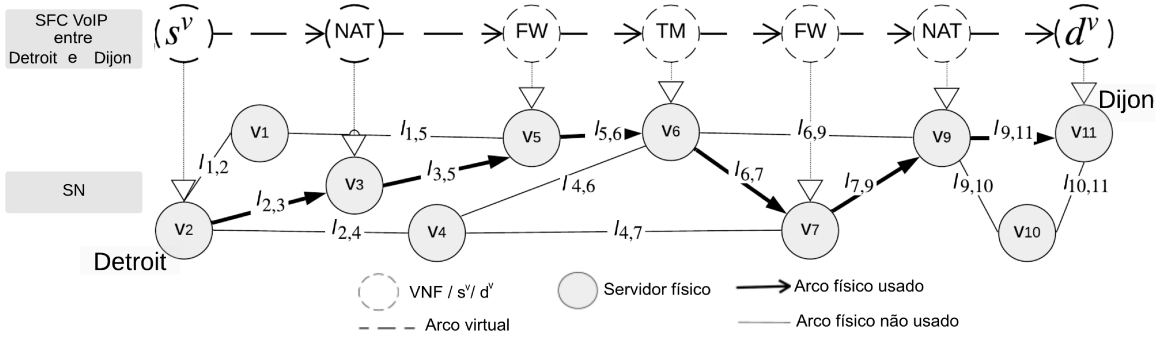
Figura 3.4: Compartilhamento de recursos entre 3 SFCs



Fonte: Elaborado pelo autor.

Cada SFC $v \in V$ mapeada sobre um SN gera um atraso fim a fim resultante dos recursos de arcos e servidores físicos utilizados. Porém, cada NS requisitado pelo cliente pode possuir um limite de atraso fim a fim máximo a ser respeitado para garantir uma qualidade de serviço mínima. A Figura 3.5 mostra o mapeamento de uma SFC, para fornecer o serviço de VoIP, mapeada sobre um SN composto por servidores NFV. No exemplo, posicionam-se os nós virtuais s^v, NAT, FW, TM, FW, NAT e d^v respectivamente sobre uma instância de função de rede alocada nos servidores v_2, v_3, v_5, v_6, v_7, v_9 e v_{11}. Neste caso o atraso de comunicação gerado é composto pelo atraso de processamento de cada VNF, mais o atraso de comunicação de cada arco físico utilizado, *i.e.*, $2 \cdot atraso(NAT) + 2 \cdot atraso(FW) + atraso(TM) + atraso(l_{2,3}) + atraso(l_{3,5}) + atraso(l_{5,6}) + atraso(l_{6,7}) + atraso(l_{7,9}) + atraso(l_{9,11}) \leq t^v_{dl}$.

Figura 3.5: Mapeamento de uma SFC (VoIP) sobre um SN



Fonte: Elaborado pelo autor.

3.1.4 Mapeamento de uma SFC

Uma solução para o mapeamento de uma SFC $v \in V$ no VNE-PC consiste em realizar com sucesso o mapeamento $f : G^v \rightarrow G$. Tal ação pode ser dividida em:
i) Atribuição de instâncias[2], posicionamento das VNFs e dos *endpoints* $k \in N^v$:

- $\forall f \in F^v,\ (\exists b \in B_f) \wedge (\exists i \in N) : (tipo(k) = tipo(b)) \wedge (c_f^v \leq \overline{C_{bf}}) \wedge (m_f^v \leq \overline{M_{bf}}) \wedge (C_{bf} \leq \overline{C_i}) \wedge (M_{bf} \leq \overline{M_i})$, no caso $\overline{C_{bf}}$ e $\overline{M_{bf}}$ representam a CPU e memória residual da instância de função de rede $b \in B_f$, e $\overline{C_i}$ e $\overline{M_i}$ representam a CPU e memória residual do servidor físico $i \in N$.
- $\exists i \in N : i = s^v$, *i.e.*, cada nó origem é associado a um servidor físico;
- $\exists i \in N : i = d^v$, *i.e.*, cada nó destino é associado a um servidor físico.

ii) Mapeamentos dos arcos entre as VNFs e pontos de origem e destino[3]:

- $\forall (k,l) \in L^v,\ \exists \{(i,j)^1, ..., (i,j)^m\} : bw_{kl}^v \leq \overline{BW_{ij}^1}, ..., bw_{kl}^v \leq \overline{BW_{ij}^m}$, no caso $\overline{BW_{ij}}$ representa a largura de banda residual dos arcos $(i,j) \in L$ que compõem o caminho físico que hospedará o arco virtual.

Por fim, o somatório dos atrasos gerados pelas VNFs $f \in F^v$ posicionadas e pelos caminhos físicos utilizados deve ser menor que t_{dl}^v. Caso as etapas definidas anteriormente, sejam atendidas com sucesso, a SFC pode ser mapeada; caso contrário o mapeamento não é possível nos recursos residuais existentes. Tal modelo de mapeamento é valido para o processamento de uma SFC por vez, caso necessite-se mapear mais de uma SFC simultaneamente, deve-se garantir que os recursos residuais do SN sejam suficientes para suprir os recursos demandados concomitantemente por tais SFCs.

3.2 Formulação Proposta

O VNF-PC pode ser modelado com ILP através de um conjunto de variáveis e restrições a serem respeitadas. De modo análogo à definição do problema (Seção 3.1), a formulação denotada pode ser aplicada tanto em um modelo *online* de mapeamento,

[2]Cada nó virtual $k \in N^v$ de uma mesma SFC deve ser posicionado em um nó físico $i \in N$ diferente. Essas restrições são chamadas *anti-affinity rules* (ETSI, 2014; Gao et al., 2018; Zoure et al., 2022).

[3]Cada arco virtual $(k,l) \in L^v$ pode ser posicionado em um arco físico único $(i,j) \in L$, ou em um caminho constituído por mais de um arco físico.

quanto *offline*. Admite-se V como conjunto de SFCs a serem mapeadas, e considera-se $|V| = 1$, no cenário *online*, no caso, mapeia-se uma SFC de cada vez; e $|V| > 1$ no cenário *offline*, no caso, mapeia-se várias SFCs juntas.

A modelagem apresentada nesta tese é baseada no modelo proposto por Luizelli et al. (2017). Contudo, de modo a melhor refletir conceitos do mundo real, as restrições de capacidade de nós físicos do SN (servidores) são definidas em termos de capacidade de processamento (*core*) e memória (MB). Outra diferença é que a formulação proposta pode ser aplicada para processar uma ou mais SFCs ao mesmo tempo. São mostradas na Tabela 3.1 os conjuntos, funções, parâmetros e variáveis presentes na modelagem proposta.

Tabela 3.1: Glossário de Conjuntos, Funções, Parâmetros e Variáveis

Conjuntos	**Definição**
V	Conjunto de todas SFCs a serem processadas
V^{at}	Conjunto de todas SFCs ativas no momento t
N	Conjuntos de servidores (N-PoPs) físicos i do SN
L	Conjuntos dos arcos físicos (i, j) do SN
F	Conjunto de todas funções de redes f a serem virtualizadas
B_f	Conjunto de todas as instâncias b de funções de redes do tipo $f \in F$
N^v	Conjuntos dos nós virtuais s^v, d^v, F^v demandados por uma SFC $v \in V$
L^v	Conjuntos dos arcos virtuais (k, l) demandados por uma SFC $v \in V$
F^v	Conjunto das NFs do tipo f demandados por uma SFC $v \in V$
Funções	**Definição**
$tipo(k) \rightarrow f \in F$	Retorna o tipo da função de rede $f \in F$ a que o nó K pertence
$atraso(f) \rightarrow \mathbb{R}_+$	Retorna o atraso que a função de rede $f \in F$ gera
$atraso(i, j) \rightarrow \mathbb{R}_+$	Retorna o atraso de ponta a ponta do arco físico $(i, j) \in L$
Parâmetros	**Definição**
$C_i \in \mathbb{R}_+$	Capacidade de CPU (core) total ofertado pelo servidor $i \in N$
$M_i \in \mathbb{R}_+$	Capacidade de memória (MB) total ofertada pelo servidor $i \in N$
$BW_{ij} \in \mathbb{R}_+$	Capacidade de banda (Mbps) total ofertada pelo arco $(i, j) \in L$
$C_{bf} \in \mathbb{R}_+$	Capacidade de CPU (core) total da instância b do tipo f
$M_{bf} \in \mathbb{R}_+$	Capacidade de Memória (MB) total da instância b do tipo f
$c_f^v \in \mathbb{R}_+$	CPU (core) demandado pela VNF $f \in F^v$
$m_f^v \in \mathbb{R}_+$	Memória (MB) demandado pela VNF $f \in F^v$
$bw_{kl}^v \in \mathbb{R}_+$	Banda (Mbps) demandado pelo arco $(k, l) \in L^v$
$t_{in}^v \in \mathbb{R}_+$	Tempo de chegada de uma SFC $v \in V$
$t_{dr}^v \in \mathbb{R}_+$	Tempo de duração de uma SFC $v \in V$
$t_{dl}^v \in \mathbb{R}_+$	atraso máximo suportado por uma SFC $v \in V$
$\eta^f \in \mathbb{R}_+$	Parâmetro de aumento/redução de fluxo ocasionado por uma função de rede $f \in F$
$\alpha \in \mathbb{R}_+$	Receita por $Mbps$ de banda utilizada
$\beta \in \mathbb{R}_+$	Receita por *core* de CPU utilizado
$\gamma \in \mathbb{R}_+$	Receita por MB de memória utilizada
$\delta \in \mathbb{R}_+$	Custo por $Mbps$ de banda alugada do InP
$\varepsilon \in \mathbb{R}_+$	Custo por core de CPU utilizado
$\tau \in \mathbb{R}_+$	Número randômico utilizado no cálculo do atraso de um arco físico $(i, j) \in L$
$\xi_{bf} \in \mathbb{R}_+$	Custo para instanciar uma VNF b do tipo f
$\zeta \in \mathbb{R}_+$	Custo por MB de memória utilizado
$\epsilon \in \mathbb{R}_+$	Custo para manter ativo determinado servidor (N-PoP)
Variáveis	**Definição**
$y^v \in \{0, 1\}$	Se igual a 1 indica que a SFC $v \in V$ foi mapeada com sucesso, 0 caso contrário
$w_{bf}^i \in \{0, 1\}$	Efetua o posicionamento (*placement*) de uma instância de função virtual sobre um servidor físico. Se igual a 1 indica que uma instância $b \in B_f$ da função de rede $f \in F$ do servidor físico $i \in N$ está sendo usada, 0 caso contrário

$z^v_{ki} \in \{0,1\}$	Efetua a atribuição (*assignment*) de uma VNF sobre uma instância de função virtual posicionada sobre um servidor físico. Se igual a 1 indica que a VNF ou *endpoint* $k \in N^v$ da SFC $v \in V$ foi atribuído sobre o servidor físico $i \in N$, 0 caso contrário
$x^{vkl}_{ij} \in \{0,1\}$	Efetua a função de roteamento orientado (*chaining*). Se igual a 1 indica que o arco virtual $(k,l) \in L^v$ da SFC $v \in V$ foi posicionado sobre o arco físico $(i,j) \in L$, 0 caso contrário

Fonte: Elaborado pelo autor.

3.2.1 Restrições do Modelo *Offline*

A formulação utilizada considera um conjunto V, com uma ou mais SFCs, sendo que estas são processadas utilizando os recursos do SN. No mapeamento *offline*, não é necessário utilizar/atualizar os recursos residuais do SN, visto que todas as SFCs existentes são processadas ao mesmo tempo. Tal formulação possui as seguintes restrições:

$$\sum_{f \in F} \sum_{b \in B_f} w^i_{bf} C_{bf} \leq C_i, \quad \forall i \in N \tag{3.1}$$

$$\sum_{f \in F} \sum_{b \in B_f} w^i_{bf} M_{bf} \leq M_i, \quad \forall i \in N \tag{3.2}$$

$$\sum_{v \in V} \sum_{\substack{f \in F^v : \\ tipo(f) = tipo(k)}} z^v_{ki} c^v_f \leq \sum_{b \in B_f} w^i_{bf} C_{bf}, \quad \forall i \in N, \forall f \in F \tag{3.3}$$

$$\sum_{v \in V} \sum_{\substack{f \in F^v : \\ tipo(f) = tipo(k)}} z^v_{ki} m^v_f \leq \sum_{b \in B_f} w^i_{bf} M_{bf}, \quad \forall i \in N, \forall f \in F \tag{3.4}$$

$$\sum_{i \in N} z^v_{ki} = y^v, \quad \forall f \in F^v, \forall v \in V \tag{3.5}$$

$$\sum_{i \in N : i = k} z^v_{ki} = y^v, \quad \forall k \in \{s^v \cup d^v\}, \forall v \in V \tag{3.6}$$

$$\sum_{b \in B_f} w^i_{bf} \leq 1, \quad \forall i \in N, \forall f \in F \tag{3.7}$$

$$\sum_{k \in N^v} z^v_{ki} \leq 1, \quad \forall i \in N, \forall v \in V \tag{3.8}$$

$$\sum_{k \in N^v} z^v_{ki} \leq S_i, \quad \forall i \in N, \forall v \in V \tag{3.9}$$

$$\sum_{v \in V} \sum_{(k,l) \in L^v} x^{vkl}_{ij} bw^v_{kl} \leq BW_{ij}, \quad \forall (i,j) \in L \tag{3.10}$$

$$\sum_{(i,j) \in L} \sum_{(k,l) \in L^v} atraso(i,j) x^{vkl}_{ij} + \sum_{i \in N} \sum_{f \in F^v} atraso(k) z^v_{ki} \leq t^v_{dl}, \quad \forall v \in V \tag{3.11}$$

$$\sum_{(i,j)\in L} x_{ij}^{vkl} - \sum_{(h,i)\in L} x_{hi}^{vkl} = z_{ki}^{v} - z_{li}^{v}, \quad \forall i \in N, \forall (k,l) \in E^v, \forall v \in V \tag{3.12}$$

As restrições 3.1 e 3.2 asseguram que todas as instâncias $b \in B_f$, de funções de rede $f \in F$, não extrapolem respectivamente a capacidade de processamento C_i e memória M_i existentes nos servidores físicos $i \in N$. Tais restrições integram a etapa de posicionamento (*placement*) de uma VNF. Os conjuntos de restrições 3.3 e 3.4 asseguram que o posicionamento de todas VNFs $f \in F^v$ sobre as instâncias $b \in B_f$ não extrapolem respectivamente a capacidade de CPU C_{bf} e memória M_{bf} total disponível. Tais restrições, com as 3.1 e 3.2 perfazem a etapa de posicionamento de uma instância VNF. As restrições 3.5 asseguram que cada VNFs $f \in F^v$ é mapeada em um único nó físico da rede para a SFC ser aceita. Tais restrições realizam a etapa de atribuição (*assignment*) de uma VNF sobre um servidor $i \in N$ do SN. Similarmente às restrições 3.5, mas tratando dos pontos de origem e destino, as restrições 3.6 asseguram o mapeamento de cada ponto de origem s^v e destino d^v para determinada SFC $v \in V$ ser aceita. Neste caso, para um ponto de origem ou destino $k \in \{s^v \cup d^v\}$ ser mapeado, o mesmo deve estar na mesma posição física do servidor $i \in N$ do SN ($i = k$). As restrições 3.7 asseguram que apenas uma instância de rede exclusiva é alocada para atender a determinado conjunto de VNFs do mesmo tipo. As restrições 3.8 asseguram que todos os nós virtuais $k \in N^v$ são atribuídos a nós físicos $i \in N$ diferentes para cada SFC $v \in V$. As restrições 3.9 estabelecem quais nós físicos $i \in N$ estão sendo utilizados (recebendo uma instância de VNF qualquer). Essas restrições são usadas para contabilizar quantos servidores estão ativos. O conjunto de restrições 3.10 asseguram que o mapeamento de todos os arcos virtuais $(k,l) \in L^v$ não extrapolarão a capacidade de banda BW_{ij} do SN. As restrições 3.11 asseguram que o mapeamento de todas as VNFs $f \in F^v$ e arcos virtuais $(k,l) \in N^v$ não extrapolarão o atraso t_{dl}^v máximo tolerado pela SFC $v \in V$. As restrições 3.12 são clássicas, chamadas na literatura de *Capacitated Multi-Commodity Flow*. Elas aplicam os princípios de conservação de fluxo para garantir o mapeamento de cada arco virtual $(k,l) \in L^v$ de cada SFC $v \in V$ sobre caminhos de arcos físicos $(i,j) \in L$.

3.2.2 Restrições do Modelo *Online*

Eventualmente podem existir SFCs ativas, posicionadas e encadeadas em um momento anterior sobre o SN. Neste caso, ao realizar-se o mapeamento de uma nova SFC, deve-se garantir que as SFCs ativas no momento t, representadas pelo conjunto $V^{at} \subset V$, continuem tendo seus serviços atendidos em um momento posterior t'. Deve-se processar

as novas SFCs $v \in V$ com as restrições 3.1 a 3.12, apresentadas no modelo *offline*; e, simultaneamente, garantir a viabilidade da solução com a reotimização das SFCs $v \in V^{at}$. Para esse fim, nesta tese assume-se três estratégias de reotimização diferentes. Ressalta-se que, ao se aplicar reconfigurações de mapeamentos em ambientes reais de internet, necessita-se de interromper os mapeamentos e fluxos de dados, remover as conexões ativas e reconfigurá-las (Fischer et al., 2013). Referida ação pode gastar muito tempo, ser custosa computacionalmente e causar problemas na qualidade do serviço prestado (Zhu and Ammar, 2006).

I Redimensionamento das instâncias de VNFs: processam-se as SFCs $v \in (V \cup V^{at})$ juntas, mas para as SFCs $v \in V^{at}$ permite-se somente o redimensionamento das instâncias de VNFs posicionadas, *i.e.*, reotimizam-se as restrições 3.1 a 3.4.

   A reotimização das instâncias de VNFs pode ser benéfica quando se necessita aumentar ou reduzir a capacidade de uma instância. Ao aumentar a capacidade de uma instância, permite-se que uma ou mais novas VNFs compartilhem a mesma instância de função de rede. Ao reduzir a capacidade de uma instância, busca-se um uso mais conciso e eficiente de recursos, e consequentemente um lucro maior, pois as instâncias VNFs de maior capacidade são potencialmente mais caras. Se efetuada sozinha, mantendo as variáveis de x_{ij}^{vkl} e z_{ki}^{v} fixas, a redefinição das variáveis w_{bf}^{i} é de complexidade polinomial. Repare que, nesta estratégia, são redimensionadas apenas as capacidades das instâncias de VMs para atender as VNFs que estão sendo processadas, não sendo necessário interromper o atendimento das demais SFCs que possam estar ativas simultaneamente no SN.

II Redimensionamento das instâncias de VNFs e reroteamento dos caminhos físicos: processam-se as SFCs $v \in (V \cup V^{at})$ juntas, mas para as SFCs $v \in V^{at}$ além do redimensionamento das instâncias de VNFs posicionadas, permite-se o reroteamento dos arcos utilizados, *i.e.*, reotimizam-se as restrições 3.1 a 3.4, e 3.10 a 3.12.

   A reotimização dos arcos virtuais pode ser benéfica quando existe uma alta fragmentação de recursos físicos do SN, inviabilizando o acesso a servidores potencialmente isolados por falta de banda nos arcos adjacentes. Ou ainda, caso seja necessário atender a uma nova SFC que possua aspectos de atraso fim a fim mais rígidos (baixos). Nesse caso, uma SFC antiga, com uma demanda por atraso menos rígida, pode ter seu roteamento alterado de modo a liberar os recursos antes alocados para a nova SFC utilizar. Mesmo se efetuada sozinha, mantendo as variáveis de w_{bf}^{i} e z_{ki}^{v} fixas, o cálculo das variáveis x_{ij}^{vkl} é de complexidade exponencial, e pode ser pensada como uma extensão do problema *Unsplittable Multicommodity Flow* (Paschos et al., 2018). Repare que, nesta estratégia, são redimensionadas as capacidades das instâncias de VMs para atender as VNFs que estão sendo processadas, e, eventualmente os rotea-

mentos podem ser alterados, sendo necessário interromper o atendimento das SFCs que poderão ser reotimizadas a nível de roteamento.

III Reposicionamento e redimensionamento das instâncias de VNFs, e reroteamento dos caminhos físicos: processam-se as SFCs $v \in (V \cup V^{at})$ juntas, mas para as SFCs $v \in V^{at}$ além do redimensionamento das instâncias de VNFs posicionadas e o reroteamento dos arcos utilizados, permite-se mudar a atribuição das instâncias de VNFs utilizadas, *i.e.*, reotimizam-se todas as restrições 3.1 a 3.12.

A reotimização das restrições de atribuição de instâncias de VNFs pode ser benéfica quando se necessita alterar a alocação de uma ou mais instâncias de VNFs sobre o SN. Essa ação é motivada por uma baixa capacidade de processamento ou memória residual de um servidor, inviabilizando o compartilhamento de recursos; ou para reduzir o atraso fim a fim de uma SFC. É possível observar que ao se efetuar uma reotimização da alocação das instâncias de VNFs, implica-se também no reroteamento dos arcos virtuais incidentes sobre ela, visto que a localização das instâncias mudou, *i.e.*, para recuperar-se a factibilidade da solução, deve-se necessariamente reotimizar todo o modelo. Neste caso, todas as restrições de todas as SFCs ativas são reotimizadas, logo, o posicionamento e roteamento de todas as SFCs pode ser alterado, implicando em uma potencial interrupção do serviço prestado para todos os NSs envolvidos.

3.2.3 Função Objetivo

Similarmente a Chowdhury et al. (2012), a equação 3.13 (*revenue*, R) fornece um indicador de quanto os provedores envolvidos recebem ao realizar o mapeamento de determinada SFC. Tal equação é expressa em função dos recursos virtuais demandados por componente de cada SFC mapeada. As constantes α, β e γ representam respectivamente as tarifas cobradas por *Mbps* de banda solicitada em cada arco virtual, por *core* de processamento, e por *MB* de memória solicitada em cada componente de cada SFC.

$$R = \sum_{v \in V} y^v \left(\sum_{(k,l) \in L^v} bw_{kl}^v \alpha + \sum_{k \in F^v} \left(c_f^v \beta + m_f^v \gamma \right) \right) \tag{3.13}$$

A equação 3.14 (*Link Cost*, LC) representa o custo dos provedores envolvidos ao efetuar uma reserva de banda e encaminhamento de tráfego para atender cada arco virtual de cada SFC $v \in V$. Conforme Bari et al. (2015), tal valor é calculado a cada

Mbps reservado para o tráfego de dados. A constante δ representa a tarifa cobrada pelo provedor, detentor da estrutura física, pelo uso (aluguel) dos arcos físicos do SN.

$$LC = \sum_{v \in V} \sum_{(k,l) \in L^v} \sum_{(i,j) \in L} x_{ij}^{vkl} bw_{kl}^{v} \delta \tag{3.14}$$

A equação 3.15 (*Server Cost*, *SC*) afere o custo de utilização de um servidor físico para fornecer atendimento às VNFs pertencentes às SFCs $v \in V$. Existem diferentes custos envolvidos: *i*) de manter ativo o servidor $i \in N$, valor que pode ser amortizado entre todas as SFCs $v \in V$ que o utilizam; *ii*) de licença de *software* e instanciação de diferentes funções de rede, valor que também pode ser amortizado entre todas as SFCs $v \in V$ que o utilizam; *iii*) de memória, e *iv*) de processamento, estimados por componentes requisitados por SFC entrante. As constantes ϵ, ε, ζ e ξ_{bf} representam respectivamente os valores gastos pelos provedores para manter ativo determinado servidor $i \in N$, por *core* de processamento, por *MB* de memória utilizado e por instância de VNF b do tipo f.

$$SC = \sum_{i \in N} \left(S_i \epsilon + \sum_{f \in F} \sum_{b \in B_f} w_{bf}^{i} \xi_{bf} + \sum_{v \in V} \sum_{k \in F^v} z_{ki}^{v} \left(c_f^v \varepsilon + m_f^v \zeta \right) \right) \tag{3.15}$$

A modelagem proposta induz a variável y^v de cada SFC $v \in V$ a assumir o valor 1, garantindo o mapeamento de todos os nós de origem (s^v), destino (d^v) e VNFs (F^v) (restrições 3.5 e 3.6). Na premissa de maximizar um ganho financeiro por SFC atendida com sucesso, representado pela função R, e minimizar os custos gerados pelos servidores ativos, instâncias de VNFs utilizadas e arcos demandados, representados pelas equações SC e LC, a função objetivo é definida como mostrado na equação 3.16.

$$\text{Maximizar } R - LC - SC \tag{3.16}$$

3.3 Considerações Finais

Neste Capítulo foi introduzida a definição formal do problema VNF-PC e um modelo matemático baseado em ILP para resolvê-lo em um contexto *offline* e *online*. Foram apresentadas diferentes estratégias de reotimização, o que implica uma potencial migração dos componentes antes mapeados para atender a uma nova SFC. A principal motivação da abordagem apresentada é gerar boas soluções para o VNF-PC respeitando os aspectos de QoS dos clientes, como, atraso fim a fim, memória, processamento e largura de banda.

Capítulo 4

Ambiente de simulação e experimentos de reotimização

O VNF-PC pode ser abordado sob várias perspectivas de migração/reconfiguração de SFCs (reotimização das restrições). O objetivo deste capítulo é definir o ambiente de simulação adotado e realizar alguns experimentos aplicando diferentes estratégias de reconfiguração. Para tal, uma definição detalhada do ambiente de simulação é mostrada na Seção 4.1. Na Seção 4.2 são mostrados alguns experimentos computacionais. Por fim, são realizadas as considerações finais na Seção 4.3.

4.1 Configuração do Ambiente de Simulação

Para confrontar os algoritmos propostos, é necessário estabelecer um conjunto de métricas quantitativas para a avaliação dos resultados obtidos. Esta seção formaliza referidas métricas, bem como o ambiente de simulação.

4.1.1 Métricas

As métricas utilizadas são embasadas nos trabalhos de Chowdhury et al. (2012); Laghrissi and Taleb (2018); Jia et al. (2018); Bari et al. (2019), sendo:

Taxa de Aceitação (TA): consiste na razão entre o número de SFCs mapeadas e o total de SFS que chegaram até o momento t. Seja V^t o conjunto de SFCs que chegaram até t, e y^v a variável binária que indica se a SFC v é atendida ou não, tem-se:

$$\frac{1}{|V^t|}\sum_{v\in V^t} y^v \tag{4.1}$$

Atraso médio fim a fim no momento t (AM^t): indica como as SFCs dos usuários estão sendo atendidas. Um baixo atraso fim a fim é importante para garantir aspectos de qualidade de conexão para determinadas aplicações. Tal métrica é calculada pela razão entre a soma dos *atrasos*, demandados pelos arcos e funções virtuais utilizadas de todas as SFCs, dividida pelo número de SFCs ativas na unidade de tempo t. Seja $|V^{at}|$ o conjunto de todas SFCs $v \in V$ ativas no momento t, referida métrica pode ser expressa pela equação:

$$\frac{1}{|V^{at}|}\sum_{v\in V^{at}}\left(\sum_{(i,j)\in L}\sum_{(k,l)\in L^v} atraso(i,j)x_{ij}^{vkl} + \sum_{i\in N}\sum_{k\in N_f^v} atraso(k)z_{ki}^v\right) \tag{4.2}$$

Número de Servidores Ativos no momento t (SA^t): reporta o número absoluto de servidores ativos em determinado momento t. Quanto menor o número de servidores sendo usados, maior a economia com os custos operacionais. Seja S_i^t a variável que indica se o servidor físico $i \in N$ do SN está ativo no momento t, tem-se:

$$\sum_{i\in N} S_i^t \tag{4.3}$$

Espalhamento dos Nós Virtuais no momento t (ENV^t): indica o quão conciso ou difuso, em relação aos servidores usados, é o posicionamento médio dos nós virtuais $k \in N_f^v$ das diferentes SFCs ativas sobre o SN no momento t. Valores próximos a 1 indicam que para cada VNF demandada, um servidor é alocado exclusivamente para servi-la. Quanto mais esses valores decrescem, mais servidores físicos estão sendo compartilhados. Neste caso, procura-se minimizar essa métrica, dada pela equação:

$$\frac{SA^t}{\sum_{v\in V^{at}} |N_f^v|} \tag{4.4}$$

Espalhamento dos Arcos Virtuais no momento t (EAV^t): indica o quão conciso ou difuso, em percentuais de arcos físicos utilizados, é o roteamento médio de um conjunto de SFCs ativas sobre a estrutura do SN no momento t. Similarmente ao espalhamento dos nós virtuais, essa métrica deve ser minimizada para promover um uso mais conciso de arcos do SN, e uma economia nos custos relativos ao uso dos arcos. Tal métrica é dada pela equação:

$$\frac{1}{|V^{at}|}\sum_{v\in V^{at}}\frac{1}{|L|}\sum_{(i,j)\in L}\sum_{(k,l)\in L^v} x_{ij}^{vkl} \tag{4.5}$$

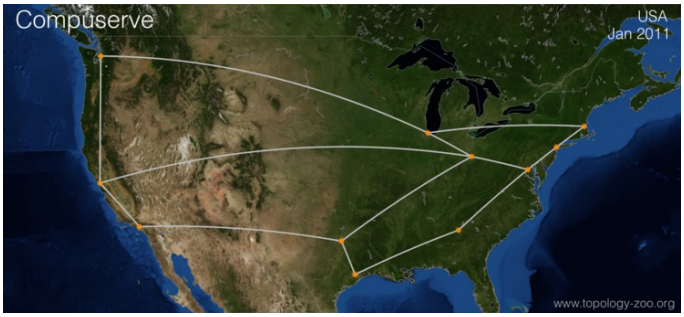
Tempo de processamento: o tempo real de processamento é um aspecto significativo a ser avaliado quando se tratam de problemas que exigem uma resposta em um curto tempo. Sendo o simulador implementado em C++, a aferição do tempo de processamento de cada algoritmo é feita através da biblioteca padrão *time.h*. A contagem de tempo é feita sempre antes e após se iniciar o algoritmo proposto, e aferida em segundos.

Lucro dos provedores: métrica importante do ponto de vista dos provedores envolvidos. Neste caso, quanto maior a receita (Equação 3.13) e menor os custos envolvidos (Equações 3.14 e 3.15), maior é o lucro gerado, demonstrado na equação 3.16.

4.1.2 Substrato de Rede

Da mesma forma que Sahhaf et al. (2015); Cohen et al. (2015); Kuo et al. (2016); Jia et al. (2018), neste trabalho consideram-se topologias reais fornecidas por Knight et al. (2011). Devido à complexidade do problema, nos experimentos realizados com o algoritmo exato, optou-se por trabalhar com uma topologia menor, chamada *CompuServe Network*. Tal escolha se dá por o algoritmo exato possuir o tempo execução diretamente ligado ao número de componentes presentes na topologia a ser processada, fator que, conforme experimentos preliminares, inviabiliza a execução em um tempo viável em topologias maiores. A topologia da *CompuServe Network* possui 11 nós e 14 arcos distribuídos pelos Estados Unidos da América, conforme mostrado na Figura 4.1.

Figura 4.1: Distribuição topológica dos nós e arcos físicos da rede *CompuServe*



Fonte: http://www.topology-zoo.org/dataset.html.

Assume-se que o SN utilizado é constituído por *hardware* COTS, e similarmente a Baumgartner et al. (2015); Luizelli et al. (2017), supõe-se que cada nó (servidor) consiga hospedar qualquer VNF requisitada. Do mesmo modo que Luizelli et al. (2017); Fischer et al. (2019), considera-se que cada nó físico pode possuir uma configuração diferente de recursos ofertados. Similarmente a Jia et al. (2018), para cada nó físico é atribuído alea-

toriamente uma máquina com os recursos utilizados pela rede *Amazon Elastic Compute Cloud* (Amazon EC2[1], na Tabela 4.1).

Tabela 4.1: Configurações dos servidores físicos adotados

Processador	**CPU** C_i (vCPU)	**memória** M_i (GB)
Xeon Scalable	48	1800
AWS Graviton2	64	3600
AWS Graviton2	64	3800
Xeon Scalable	96	1800
Intel Xeon Platinum	96	3600
AMD EPYC 7002	96	3800

Fonte: https://aws.amazon.com/pt/ec2/instance-explorer

Do mesmo modo que Hmaity et al. (2016); Luizelli et al. (2017), são adotados arcos bidirecionais entre os servidores físicos $i \in N$. Similarmente a Fischer et al. (2019), cada arco $(i, j) \in L$ do SN possui uma capacidade de banda escolhida aleatoriamente, no qual $BW_{ij} \in \{20, 40, 60, 80, 100\}$ $Mbps$. Do mesmo modo que Sahhaf et al. (2015); Jia et al. (2018), o cálculo do atraso de cada arco físico é ajustado proporcionalmente a seu comprimento geográfico e perturbado por um número randômico τ. Nesta tese, tal número é definido como $\tau \in [0.015; 0.025]$. Essa escolha de variação do valor de τ é feita com base nos valores apresentados por Jia et al. (2018), e aproximados para à distância de Haversine.

4.1.3 Funções de Redes

Seguindo os conceitos dos autores de Tanenbaum and Wetherall (2011); Alleg et al. (2017), as NFs empregadas conseguem atender a serviços de diferentes classes, como: aplicativos de transferência de arquivos, aplicações interativas e em tempo real. Tais classes de serviços englobam NF completas como: RA, *Narrowband Internet of Things* (NB-IoT), *Massive Internet of Things* (MIoT), serviços *web*, agricultura de precisão (Abdelhamid, 2019; Askari et al., 2019; Bari et al., 2019; Ghaznavi et al., 2017; H. et al., 2012; Ruiz-Perez et al., 2018). Cada VNF possui demandas de CPU e memória com base na funcionalidade requerida. As NFs $f \in F$ adotadas neste trabalho são comuns na literatura, fundamentadas e caracterizadas nos trabalhos de Bari et al. (2016); Zhang et al. (2017); Pham et al. (2017); Askari et al. (2019); Bari et al. (2019); Fischer et al. (2019); Gu et al. (2019); Chen and Zhao (2022), e mostradas na Tabela 4.2.

Existem diferentes sistemas comerciais de computação em nuvem *online* gerenciados por provedores como a *Amazon*, *Google*, *IBM*, *Microsoft*, *Joynet*, *OpSource*, etc.

[1]Disponível em *https://aws.amazon.com/pt/ec2/instance-explorer*, acessado dia 15/07/2021

Além disso, diversos fatores estão envolvidos nos custos de implantação e operação de cada componente do SN, como marcas de *hardware*, licenças de *software*, e custos relativos ao consumo de energia. Abstraindo-se dessa variação de custos, similarmente a Bhamare et al. (2017); Jia et al. (2018); Leivadeas et al. (2017, 2019), considera-se um modelo simplificado, baseado em configurações presentes nas instâncias da Amazon EC2.

Tabela 4.2: Funções de rede utilizadas

Tipo	c_f^v (vCPU)	m_f^v (GB)	fluxo η^f	atraso (ms)
Proxy	4	200	0.9	0.25
Firewall (FW)	4	400	0.9	0.8
Traffic Manager (TM)	10	300	0.9	0.1
Intrusion Detection System (IDS)	8	800	0.8	0.01
Network Address Translation (NAT)	16	400	1.0	0.1
WAN Optimization Controller (WOC)	2	800	1.1	0.2
Video Optimization Controller (VOC)	8	1000	1.2	0.25

Fonte: Elaborado pelo autor.

Dependendo das VNFs demandadas para serem instanciadas (Tabela 4.2), uma determinada instância é escolhida de modo que os requisitos de recursos possam ser mapeados em uma configuração disponível mais adequada. Para o mapeamento de cada VNF $f \in F$, assume-se a disponibilidade de 5 VMs com capacidades de memória, core e custos diferentes, mostrados na Tabela 4.3.

Tabela 4.3: Configurações de VMs de uso geral da Amazon EC2, com o sistema operacional *Linux*, e precificação da região leste dos EUA.

b	**Tipo de instância (servidor virtual)**	**CPU** C_{bf} (vCPU)	**memória** M_{bf} (GB)	**custo*** ξ_{bf} ($)
1	*M6 Double Extra Large*	8	475	0.361
2	*M6 Quadruple Extra Large*	16	950	0.723
3	*M5 Eight Extra Large*	32	1200	1.648
4	*M6 16 Extra Large*	64	1900	2.892
5	*M5 24 Extra Large*	96	3600	5.424

*preço definido pela relação instância-hora consumida

Fonte: https://aws.amazon.com/pt/ec2/instance-explorer

Os valores adotados para o cálculo da receita dos provedores é definido pela relação requisição-hora, e cobrados proporcionalmente pela quantidade de banda, CPU e memória demandadas. Tais valores são representados pelas constantes α, β, γ, e definidas respectivamente, como $\{0.0036, 0.018, 0.036\}$. Adotam-se também os valores $\{0.00072, 0.0036, 0.0072, 0.36\}$ como custos atribuídos respectivamente às constantes δ, ε, ζ e ϵ. Na prática, esses valores não são reais, mas definidos após uma avaliação preliminar da relação de valores de mercado e da literatura, e aproximados ao modelo. Além disso, segundo Luizelli et al. (2015), o estabelecimento de valores fixos para esses parâmetros facilita a avaliação da proposta em detrimento a outros fatores secundários. Caso tal modelo seja adotado em ambientes reais, as funções de receita (Equação 3.13) e custo (Equações 3.14 e 3.15), bem como os valores de precificação podem ser alterados.

4.1.4 Classes de SFCs

Com base nas classes de serviços definidas em Tanenbaum and Wetherall (2011), e nas VNFs mostradas na Tabela 4.2, as SFCs utilizadas neste trabalho são sumarizadas em classes, baseadas nos trabalhos de Savi et al. (2015); Savi et al. (2016); Alleg et al. (2017); Hmaity et al. (2017); Abdelhamid (2019). Apesar de, para fins de exemplificação, cada NS ser atribuído a uma classe diferente (Tabelas 4.4, 4.5 e 4.6), tais NSs podem ser recategorizados e/ou a parametrização das classes alteradas caso seja necessário.

Aplicativos em tempo real (Classe 1): inclui NSs que normalmente possuem um alto volume de tráfego (telefonia sobre IP a parte) e que são extremante sensíveis a atrasos, *e.g.* aplicações em 5G, alguns dispositivos de IoT, videoconferência e carros autônomos. Alguns dos NSs desta classe são mostrados na Tabela 4.4. Tais NSs podem tolerar atrasos fim a fim de até $150ms$ (Alleg et al., 2017). Para definir tal classe, adote t_{dr}^{v} como tempo de duração de uma SFC e bw a reserva de banda em $Mbps$ necessária para atender às restrições de QoS.

Tabela 4.4: NSs que são muito sensíveis à métricas de QoS

Tipo	Exemplo de serviço	Sequência de VNFs	bw $(Mbps)$	t_{dr}^{v}(s)	Referência
1	Conferência de video	NAT → FW → TM → VOC → IDS	5	3000	(Savi et al., 2016)
2	Jogos *online*	NAT→FW→VOC→WOC→IDS	20	10800	(Hmaity et al., 2017) (Google, 2021a)
3	*Smart Factory*	NAT → FW	20	900	(Askari et al., 2019)
4	*VoIP*	NAT→FW→TM→FW→NAT	0.064	180	(Hmaity et al., 2017) (Holub et al., 2018)
5	RA	NAT → FW → TM → VOC → IDS	5	1800	(Askari et al., 2019) (Savi et al., 2016) (Li and Tu, 2020)

Fonte: Elaborado pelo autor.

Aplicativos interativos (Classe 2): inclui NSs que são sensíveis à métricas de QoS e possuem características de tráfego em tempo real, *e.g. web services* ou *login* remoto em um sistema industrial. Segundo Alleg et al. (2017), nesta classe de serviços podem ser tolerados atrasos fim a fim de até $300ms$ por NS, como mostrado na Tabela 4.5.

Tabela 4.5: NSs que são sensíveis à métricas de QoS

Tipo	Exemplo de serviço	Sequência de VNFs	bw $(Mbps)$	t_{dr}^{v}(s)	Referência
1	*Local A2A* web service*	NAT→FW→TM→WOC→IDS	0.2	1	(Nguyen et al., 2008) (Hmaity et al., 2017) (Ouni et al., 2017)
2	*Internacional A2A* web service*	NAT→FW→TM→WOC→IDS	10	1	(Nguyen et al., 2008) (Hmaity et al., 2017) (Ouni et al., 2017)
3	Requisitos de segurança	*proxy* → *FW* → IDS	0.1	1	(Jia et al., 2018)

(**Application-to-Application*)

Fonte: Elaborado pelo autor.

Aplicativos de transferência de arquivos (Classe 3): inclui NSs que são menos sensíveis à métricas de QoS e/ou que possuem características de melhor esforço (entrega de pacotes não garantida). Segundo Brown (2017), as aplicações do tipo mIoT e NB-IoT são caracterizadas pela necessidade do suporte à transferência de arquivos entre um grande número de dispositivos IoT, mas são menos sensíveis à restrições como banda e atraso fim a fim, sendo inseridas nesta classe. Outros exemplos são aplicativos com conexão ponto a ponto, como a transmissão de *streaming* de vídeo. Nestes casos, se todos os pacotes forem atrasados uniformemente por alguns segundos, nenhum dano será causado. Nesta classe de serviços, conforme Abdelhamid (2019), podem ser tolerados atrasos fim a fim de até $30s$ por NS, como mostrado na Tabela 4.6.

Tabela 4.6: NSs que são menos sensíveis à métricas de QoS

Tipo	**Exemplo de serviço**	**Sequência de VNFs**	bw $(Mbps)$	t^v_{dr}(s)	Referência
1	Video *streaming* UHD	NAT→FW→TM→VOC→IDS	25	7200	(Google, 2021b) (Hmaity et al., 2017)
2	Áudio *streaming*	NAT→FW→TM→IDS	0.32	120	(Deezer, 2021)
3	mIoT	NAT → FW → IDS	1	90	(Bhamare et al., 2017) (Askari et al., 2019)
4	NB-IoT	NAT → FW → IDS	0.25	60	(Li and Tu, 2020)

Fonte: Elaborado pelo autor.

Similarmente a Chowdhury et al. (2009); Bari et al. (2019), adota-se a distribuição *Poisson* para modelar a taxa de chegada das SFCs em todas as classes, justamente por tal escolha melhor retratar o comportamento de NS em computação em nuvem, como *Video Streaming On-Demand* e VoIP. Para modelar a duração de cada SFC (t^v_{dr}) e sua respectiva demanda de largura banda (bw), similarmente a Chowdhury et al., é utilizada uma distribuição exponencial com a média μ diferente para cada tipo de SFC, em cada classe de serviço (Tabelas 4.4, 4.5 e 4.6). De modo a caracterizar as aplicações de redes processadas neste trabalho, pode-se resumir as características de tais classes de serviços na Tabela 4.7.

Tabela 4.7: Características das classes de serviços mostradas nas Tabelas 4.4, 4.5 e 4.6

Classe1			
Tipo	t^v_{dr}	t^v_{dl}	bw
1	3000	150	5
2	10800	150	20
3	900	150	20
4	180	150	0.064
5	1800	150	5

Classe 2			
Tipo	t^v_{dr}	t^v_{dl}	bw
1	1	300	0.2
2	1	300	10
3	1	300	0.1

Classe 3			
Tipo	t^v_{dr}	t^v_{dl}	bw
1	7200	30000	25
2	120	30000	0.32
3	90	30000	1
4	60	30000	0.25

Fonte: Elaborado pelo autor.

As SFCs utilizadas neste trabalho são geradas similarmente a Sun et al. (2016); Hmaity et al. (2016); Lange et al. (2017). Para a geração de cada SFC, dois servidores da rede são escolhidos aleatoriamente e definidos como pontos de origem (s^v) e destino (d^v) de cada SFC. Bari et al. (2016); Pham et al. (2017) adotam que cada SFC é composta por

3 VNFs, por outro lado Zhang et al. (2017) adotam que uma SFC pode conter entre 2 e 5 VNFs. Wang et al. (2021b) propõem que uma SFC contenha entre 3 e 5 VNFs. Wei et al. (2022b) e Chen et al. (2022) adotam respectivamente que uma SFC pode conter entre 3 e 6, e 4 e 6 VNFs. Neste trabalho, utiliza-se a mesma escolha de Bari et al. (2019), no caso, uma SFC pode conter entre 2 e 6 VNFs, justamente para as combinações de VNFs gerarem uma ampla gama de SFCs, aptas a atender a um maior número de diferentes NSs. São associadas aleatoriamente para cada SFC, VNFs da Tabela 4.2. As demais características de cada SFC são geradas conforme as classes e tipos em questão.

É assumido que, devido aos NSs atrelados, o intervalo de chegadas das SFCs segue uma distribuição *Poisson* de média $\lambda = 600$ na Classe 1, neste caso, em média, 10 SFCs chegam a cada $6.000t$; de média $\lambda = 400$ na Classe 2, neste caso, em média, 15 SFCs chegam a cada $6.000t$; e por fim de média $\lambda = 500$ na Classe 3, neste caso, em média, 12 SFCs chegam a cada $6.000t$. Como consequência, na Classe de SFCs 2, os NSs são caracterizados por uma baixa duração, mas com uma taxa de entrada um pouco mais intensa, o que é característico de *web services* (Sobh and Fakhry, 2014). Por outro lado, na Classe de SFCs 3, as SFCs são caracterizadas por uma duração maior que na Classe 2, mas com uma taxa de entrada menos intensa. Por fim, a Classe de SFCs 1 é caracterizada por uma taxa de entrada bem menos intensa que as classes 2 e 3, mas as SFCs tendem a ter uma duração maior na classe 1. Referidos valores de λ foram adotados de forma que a rede fique com um nível de trabalho alto, mas com taxa e rejeição baixas. Caso seja adotada uma topologia de rede física maior/menor, tal taxa de chegada pode ser ajustada; ou ainda podem ser adotadas taxas de chegadas de SFCs inerentes a serviços reais de um provedor de rede específico. Com base em tais classes, são definidos os seguintes 4 cenários:

- Cenário 1, assume-se ser constituído por SFCs da classe 1, sendo $\frac{1}{5}$ de SFCs de cada tipo definido na Tabela 4.7. Possui uma taxa de entrada com $\lambda = 600$;
- Cenário 2, assume-se ser constituído por SFCs da classe 2, sendo $\frac{1}{3}$ de SFCs de cada tipo definido na Tabela 4.7. Possui uma taxa de entrada com $\lambda = 400$;
- Cenário 3, assume-se ser constituído por SFCs da classe 3, sendo $\frac{1}{4}$ de SFCs de cada tipo definido na Tabela 4.7. Possui uma taxa de entrada com $\lambda = 500$;
- Cenário 4, constituído pela união dos três cenários anteriores, *i.e.*, as SFCs chegam com diferentes intervalos e características.

Os valores de λ e a composição de cada classe de serviço, foram estipulados para compor sinteticamente os experimentos desta Seção. Pondera-se que, caso a modelagem seja empregada em um cenário real de rede, tais valores podem ser alterados conforme os serviços demandados no ambiente em questão.

4.2 Experimentos Computacionais

Os experimentos foram realizados em um computador *Intel* Core i3-8300 3.7GHz, com 16 GB de DDR4 2400MHz em *dual channel*, utilizando o sistema operacional Ubuntu 20.04. O simulador é implementado em C++ e a abordagem desenvolvida através da API CPLEX 12.6. Referidos experimentos consideram um cenário *online* em relação à chegada de SFCs (Seção 3.2). Para os experimentos são consideradas 20.000 unidades de tempo t, originando os cenários: *i)* $\lambda = 600$ e $20.000t$ com 33 SFCs processadas; *ii*) $\lambda = 400$ e $20.000t$ com 49 SFCs processadas; *iii*) $\lambda = 500$ e $20.000t$ com 40 SFCs processadas; e *iv*) união dos cenários anteriores com 122 SFCs processadas.

A respeito do tempo de processamento de cada SFC, alguns trabalhos como Bari et al. (2016); Bhamare et al. (2017); Bari et al. (2019); Sharma et al. (2020) levam frações de segundos para a execução, outros geram uma tomada de decisão na casa de segundos, como Khebbache et al. (2017), e até minutos, como Luizelli et al. (2015); Khoshkholghi et al. (2019). Estas restrições não são consideradas nestes experimentos, pois fogem ao escopo da experimentação, e serão abordadas nos capítulos seguintes. Entre os objetivos destes experimentos estão elucidar as seguintes questões: qual o impacto em termos de lucro as reconfigurações podem gerar? Qual o tempo atrelado a aplicação de cada modelo de reconfiguração? É necessário reconfigurar todo o modelo para se gerar boas soluções? Deste modo, nestes experimentos procura-se mostrar a eficiência das diferentes estratégias de mapeamento dinâmico (reotimização), e comparar os *trade-off*s entre os algoritmos e as métricas avaliadas em diferentes cenários.

4.2.1 Algoritmos Avaliados

Cada algoritmo realiza o mapeamento com base nos recursos residuais do SN, sendo que o *solver* resolve cada modelo em sua otimalidade. São avaliados os algoritmos:

- ILP^{i} - reotimizam-se somente as restrições de dimensionamento de instâncias (*images*, i) de VNFs. Como não são reotimizados os posicionamentos e encadeamentos das VNFs ativas, não se pode afirmar que a solução gerada é ótima;
- ILP^{ic} - reotimizam-se as restrições de dimensionamento de instâncias (i) e as de encadeamento (*chain*, c) das SFCs ativas. Como não são reotimizados os posicionamentos de VNFs, não se pode afirmar que a solução gerada é ótima;

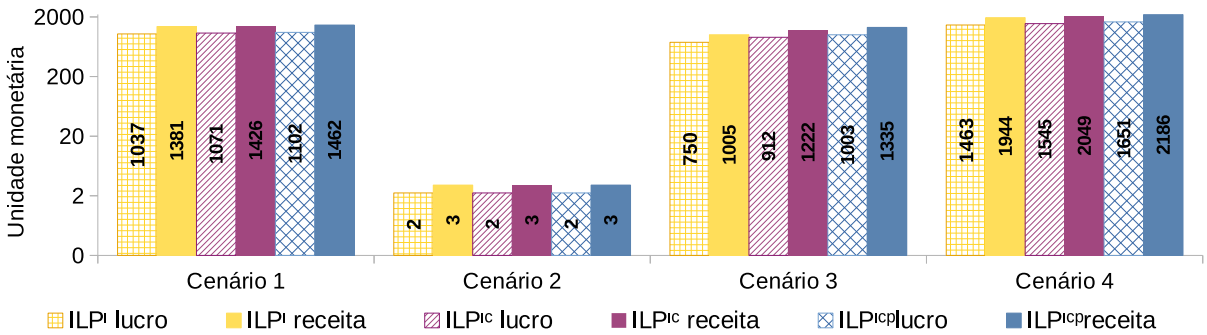
- ILP^{icp} - reotimizam-se as restrições de dimensionamento de instâncias (i) de VNFs, as de encadeamento (c), e as de atribuição de posicionamento de VNFs (*placement*, p) das SFCs ativas. A solução obtida possui garantias de ser ótima globalmente.

4.2.2 Comparação entre as Estratégias de Reotimização

Nas Figuras 4.2 a 4.4 são mostrados respectivamente o lucro total gerado, o tempo de processamento médio por SFC mapeada, e a taxa de aceitação final. As Figuras 4.5 a 4.7 mostram em janelas de tempo os resultados aferidos com determinada métrica no instante t. Variam-se os cenários entre 1, 2, 3 e 4, com as janelas de tempo mostradas no eixo x, e a métrica em questão no eixo y. Para cada um dos gráficos mostrados nas Figuras 4.5 a 4.7, é gerada uma tabela que resume o comportamento de cada algoritmo (Apêndice C).

Análise da relação lucro e receita (Figura 4.2): devido à função objetivo do modelo proposto ser delineada para maximizar o lucro, iniciam-se estes estudos pela referida métrica. Como definido na função objetivo (Equação 3.16), a receita é constituída do valor cobrado pelos provedores para atender a um serviço; e o lucro pela receita descontada das despesas para atender um serviço. Neste sentido, é importante que as SFCs sejam mapeadas para gerar um maior lucro pelo aumento da receita e redução do custo. A receita gerada pelos algoritmos (Figura 4.2) é influenciada pela taxa de aceitação (Figura 4.4). Neste caso, quanto mais SFCs forem mapeadas com sucesso, maior tende a ser a receita (Equação 3.13), e potencialmente, maior o lucro resultante (Equação 3.16). Contudo, outros fatores também são relevantes no lucro gerado, como o compartilhamento de servidores físicos (Figura 4.5) e espalhamento dos arcos virtuais de cada SFC (Figura 4.6).

Figura 4.2: Lucro e receita total dos provedores



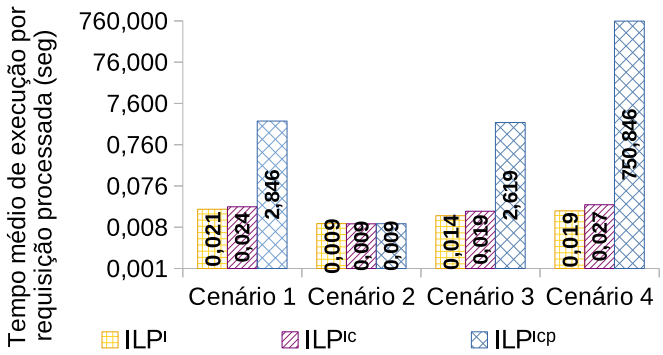
Fonte: Elaborado pelo autor.

Nesta análise, o ideal que as receitas e os lucros sejam próximos, resultando em custos menores. Deve-se destacar que a função objetivo maximiza o lucro, e não a taxa de aceitação. Assim, e em alguns casos, o algoritmo pode priorizar o mapeamento de alguma SFC com um posicionamento sobre o SN que gere inicialmente um maior lucro, mas que por se tratar de um ambiente *online*, afete o mapeamento de outras SFCs que possam ser processadas em momentos posteriores.

Na Figura 4.2, o algoritmo ILP^{icp} destaca-se com maiores lucros em relação aos outros algoritmos, justamente por gerar as maiores receitas advindas de uma maior taxa de aceitação; e um mapeamento mais conciso, com mais servidores físicos compartilhados (Figura 4.5a). Comparando a variação percentual de tais métricas entre os algoritmos ILP^{i} e ILP^{icp}, no cenário 4, o algoritmo ILP^{i} lucra $\approx 11.36\%$ a menos (Figura 4.2), possui uma taxa de aceitação $\approx 5.36\%$ menor (Figura 4.4), um espalhamento de nós $\approx 17.51\%$ menor (Tabela C.4), mas ao benefício de uma redução no tempo de execução de $\approx 99.99\%$ (Figura 4.3). O algoritmo ILP^{icp}, mesmo com o aumento no lucro percebido em todos os experimentos em relação aos outros algoritmos, possui um aumento significativo no tempo de execução, sugerindo uma inviabilidade prática de sua aplicação nos cenários avaliados. Outras explanações sobre estes experimentos são explicadas no Apêndice C.

Análise do tempo de execução total (Figura 4.3): percebe-se que os cenários com menos SFCs ativas simultaneamente, por possuírem menos componentes físicos para serem processados, possuem um número menor de variáveis e restrições no modelo matemático a ser resolvido, implicando um tempo de execução mais baixo.

Figura 4.3: Tempo médio de processamento por SFC



Fonte: Elaborado pelo autor.

Corroborando a observação do parágrafo anterior, percebe-se que no Cenário 2, o tempo de execução é o mesmo entre os diferentes algoritmos, justamente por não haver SFCs ativas em simultâneo. Comparando os cenários 1 e 4, esse aumento no tempo de processamento é de $\approx 747\%$ por SFC processada com o algoritmo ILP^{icp}. Isto acontece devido à complexidade do problema, em que um aumento linear no número de componentes a serem processados implica em um aumento exponencial no número de variáveis a serem processadas. Dada esta característica, o tempo de processamento computacional se torna

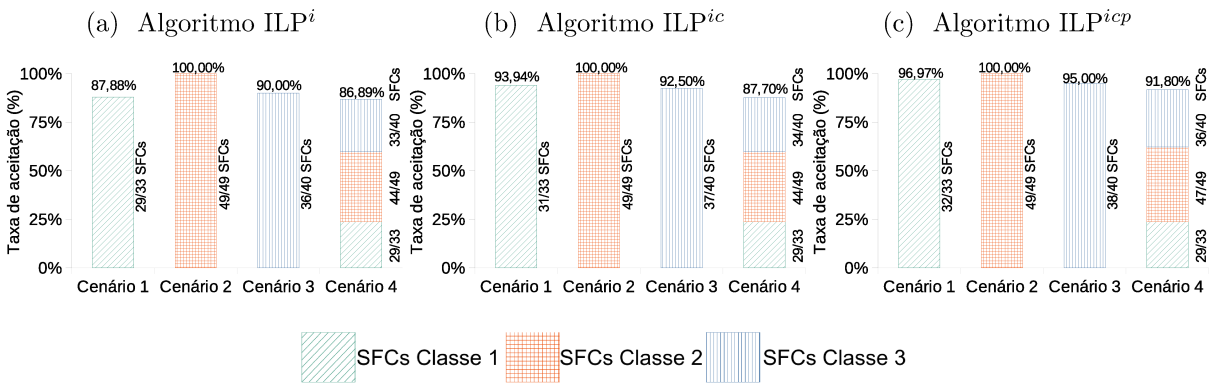
um fator crítico em redes com um elevado número de componentes a serem processados, impedindo que soluções ótimas globais sejam geradas.

No cenário 2, em que não existem SFCs ativas simultaneamente, todos os algoritmos geraram uma taxa de aceitação igual, um lucro igual e um tempo de execução próximo. Percepção que corrobora a premissa de que a reconfiguração só é efetiva quando existem várias SFCs ativas concomitantemente. Nos outros casos, percebe-se que a reconfiguração completa gera melhoras nos lucros (Figura 4.2), mas essa melhora nos lucros gera um aumento substancial no tempo de execução (Figura 4.3). Demais análises sobre estes experimentos podem ser observadas no Apêndice C.

Análise da taxa de aceitação (Figura 4.4): além do percentual de aceitação de cada algoritmo, ao lado de cada barra de cada gráfico plotado também é mostrado em valores absolutos quantas SFCs de cada classe foram mapeadas. Os experimentos com o cenário 2 são caracterizados por uma alta taxa de chegada, e um baixo tempo de vida de cada SFC. Em tal cenário não existem SFCs ativas simultaneamente, logo não existe uma disputa por recursos do SN, e a taxa de aceitação se mantém máxima (100%). Neste caso, como cada SFC tem um baixo tempo de duração, e como o intervalo de chegadas é maior, quando uma nova SFC chegar, várias outras terão provavelmente encerrado seu ciclo de vida, reintegrando e disponibilizando os recursos do SN antes alocados para o atendimento de novas SFCs.

Observando os percentuais de mapeamentos de cada algoritmo no cenário 4 (Figura 4.4), percebe-se que os algoritmos analisados geram um comportamento igualitário não priorizando o mapeamento das SFCs de uma classe em detrimento a outra. Este fato reflete uma política de mapeamento neutra, mas, caso necessário, pode-se alterar a função objetivo para priorizar o mapeamento de SFCs de uma classe.

Figura 4.4: Taxa de aceitação final



Fonte: Elaborado pelo autor.

Nos cenários 1 e 4 é percebida uma taxa de aceitação de respectivamente $87,88\%$ e 96.97% com o algoritmo ILP^{i}, e de 86.89% e 91.8% com o algoritmo ILP^{icp} (Figura 4.4). Tal percepção mostra que ambos os algoritmos conseguem processar eficientemente a carga de chegada de tais SFCs sem grandes perdas, sendo tal diferença de $\approx 9.37\%$ e

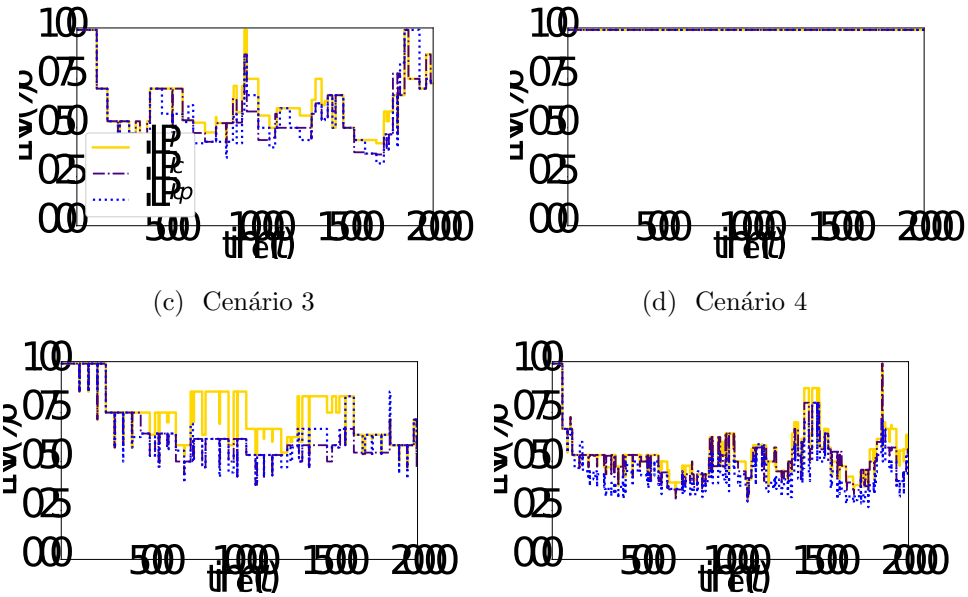
$\approx 5.36\%$ respectivamente em ambos os casos (Figura 4.4). Comparando esses números com o lucro gerado, percebe-se que este é mais impactante, neste caso, o algoritmo ILP^{icp} consegue lucrar até $\approx 25\%$ a mais que o algoritmo ILP^{i}. Ou seja, em termos de maximizar o lucro, reconfigurar todo o modelo pode ser viável, mas como discutido anteriormente, tal ação acarreta um aumento significativo no tempo de processamento (Figura 4.3).

Análise do espalhamento dos nós virtuais (Figura 4.5): as oscilações percebidas nos gráficos da Figura 4.5, em especial no cenário 4, são devidas ao baixo tempo de duração que cada SFC pode possuir. Como o cenário 4 é composto por SFCs das 3 classes de serviços, e como na classe 2 as SFCs são caracterizadas por uma alta taxa de chegada de SFCs e um baixo tempo de vida, tais entradas e saídas geram tal efeito.

No cenário 4 com o algoritmo ILP^{icp}, esse espalhamento chega à média de $\approx 46.6\%$ (Tabela C.4), *i.e.*, cada servidor físico ativo é utilizado na média por cerca de 2 SFCs diferentes. Além do custo de manter um servidor ligado ser maior que o custo de uso de um arco, um dos fatores que influencia este comportamento é fundamentado na dimensão da topologia adotada. Por se tratar de uma topologia pequena, e como as solicitações de SFCs podem originar-se dos variados *endpoints* presentes nos nós físicos, a tendência é que o compartilhamento de servidores seja maior, visto que as alternativas de servidores físicos para instanciar uma VNF são menores. Outro fator que afeta esse compartilhamento de servidores são as restrições de atraso (Figura 4.7), pouco significativas neste cenário, mas que podem ser mais significativas em cenários maiores, devido a uma maior distância geográfica dos N-PoPs (Figura 4.1).

Figura 4.5: Espalhamento dos nós virtuais (ENV)

(a) Cenário 1 (b) Cenário 2



(c) Cenário 3 (d) Cenário 4

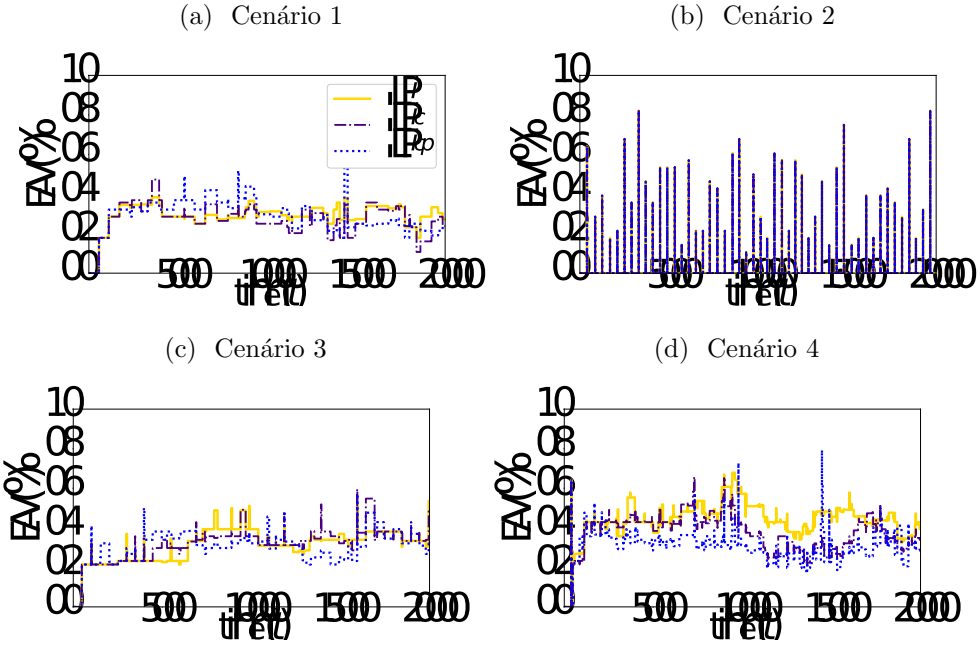
Fonte: Elaborado pelo autor.

Na Figura 4.5d, o algoritmo ILP^{icp} se destaca por produzir um espalhamento de nós

mais significativo em relação aos outros algoritmos, sendo de até $\approx 26\%$ (*e.g.* $t = 5000$). Fator que influenciou positivamente no lucro gerado, conforme mostrado na Figura 4.2. No cenário 2, a referida métrica de espalhamento é de 100%, para qualquer algoritmo avaliado (Tabela C.2), pois para cada VNF demandada, ela é mapeada sobre um servidor físico do SN exclusivo. Tal fato acontece por não haver SFCs ativas simultaneamente, e por não ser permitido o mapeamento de VNFs diferentes de uma mesma SFC sobre o mesmo servidor físico. Nos demais casos, o algoritmo ILP^{ipc} gerou um espalhamento de nós melhor que os outros algoritmos, ação que aumentou o compartilhamento de recursos, e consequentemente, reduziu o custo relativo ao uso de servidores do SN. Como exemplo, no cenário 1 o algoritmo ILP^{icp} gerou um compartilhamento de nós $\approx 7\%$ menor que o do algoritmo ILP^{i}, e $\approx 5.8\%$ menor que o do algoritmo ILP^{ic} (Tabela C.4). Percepção que, dentre outros fatores, explica o lucro gerado pelo algoritmo ILP^{icp} (Figura 4.2). Demais análises sobre estes experimentos podem ser observadas no Apêndice C.

Análise do espalhamento dos arcos virtuais (Figura 4.6): constata-se que o nível de espalhamento dos arcos virtuais é moderadamente elevado em tais experimentos (Tabelas C.5 a C.8). Como tal métrica é definida computando o número de arcos físicos do SN, logo o esperado é que o espalhamento seja pior em topologias que possuem menos arcos, como é o caso da *Compuserve*. Tal potencial escassez pode ser prejudicial, pois aumenta a concorrência por bons roteamentos entre as requisições a serem mapeadas.

Figura 4.6: Espalhamento dos arcos virtuais



Fonte: Elaborado pelo autor.

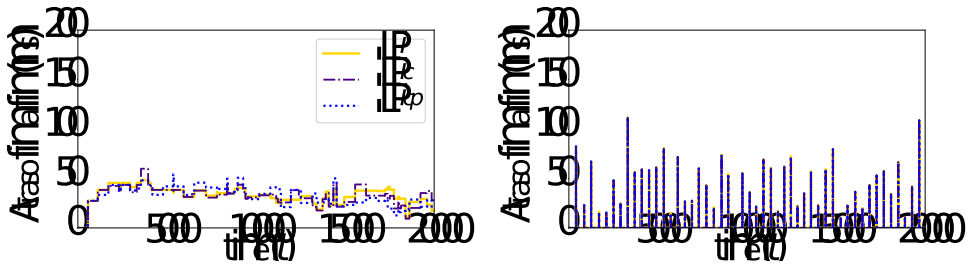
Como no cenário 2 não existem duas ou mais SFCs mapeadas simultaneamente, logo, não existe uma necessidade de reconfiguração dos mapeamentos, e o espalhamento dos arcos virtuais é o mesmo entre as três abordagens. Observando o algoritmo ILP^{icp},

em todos os cenários avaliados, percebe-se que o lucro está envolvido com os roteamentos gerados. Como exemplo, no cenário 4, que possui uma alta taxa de chegadas e saídas de SFCs, por reconfigurar todo modelo, o algoritmo ILP^{icp} gera um espalhamento $\approx 44.6\%$ menor que o algoritmo ILP^{i}, e $\approx 22.50\%$ menor que o algoritmo ILP^{ic} (Tabela C.8). Tal fator contribuiu no aumento da taxa de aceitação (Figura 4.4), por haverem mais recursos residuais para serem usados por outras requisições; e no lucro gerado (Figura 4.2), por ter um custo menor com arcos (Equação 3.16).

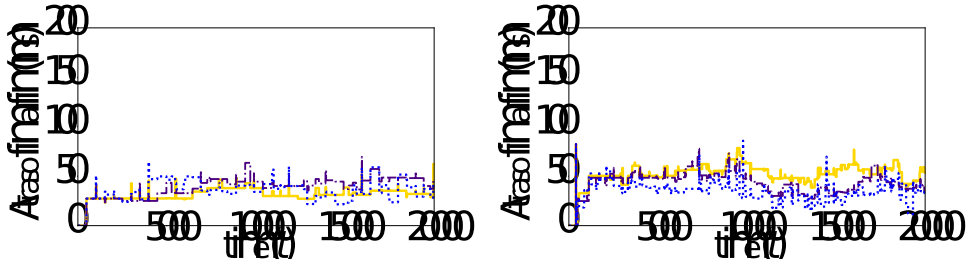
Análise do atraso médio fim a fim (Figura 4.7): deve-se ponderar nesta análise que, conforme a modelagem adotada, o custo de utilização de cada arco de um caminho físico é fixo, e o atraso fim a fim de cada arco físico pode variar de acordo com sua extensão e com o parâmetro τ. Como a função objetivo minimiza o custo e não o atraso, algumas vezes as modelagens podem utilizar um caminho mais longo, com um atraso maior, mas com um custo final menor. Assim, o atraso fim a fim não reflete identicamente a métrica de espalhamento médio dos caminhos virtuais (Figura 4.6).

Figura 4.7: Atraso fim a fim na topologia *CompuServe*

(a) Cenário 1 (b) Cenário 2



(c) Cenário 3 (d) Cenário 4



Fonte: Elaborado pelo autor.

Como as SFCs mapeadas sobre a topologia *CompuServe* são limitadas a arcos curtos, o atraso fim a fim se mostra uma restrição folgada, mantido geralmente abaixo de $\approx 80ms$, independente do algoritmo adotado. No cenário 2 em específico, por este não possuir SFCs ativas simultaneamente, o algoritmo realiza alguns mapeamentos mais longos sobre o SN, o que implica alguns picos de até $120ms$. Esses picos são gerados por reportarem o espalhamento de arcos de cada SFC individualmente, visto que não existem outras ativas em simultâneo. Nos outros gráficos, como existem várias SFCs ativas, tais picos são mitigados em valores médios. Percebe-se que as restrições de atraso fim a fim

não são um fator crítico nestes experimentos, mas, por outro lado, caso seja adotada uma topologia maior, essas restrições podem ficar mais críticas.

4.3 Considerações Finais

Neste Capítulo é introduzido o cenário de experimentação utilizado, algumas métricas de rede, e definidas classes que ajudam a sumarizar e entender os tipos de NS atendidos. Ressalta-se que, apesar de serem definidas classes de serviços e parâmetros, tais valores são considerados para fins específicos de experimentação nesta tese, e devem ser redefinidos caso o algoritmo seja empregado em outros ambientes.

Devido à complexidade do problema, os experimentos foram realizados com a topologia *CompuServe*, justamente por esta ser menor em número de componentes que outras topologias usadas na literatura. Uma topologia com menos componentes físicos, implica em menos variáveis e restrições para serem processadas pelo algoritmo exato, e um menor tempo de execução. Foram realizados experimentos computacionais segmentados em diferentes classes de serviço para verificar o desempenho dos algoritmos propostos com os diferentes métodos de reotimização.

Em tratamentos *online*, soluções ótimas globais dificilmente serão encontradas com os algoritmos ILP^{i} e ILP^{ic}. O ótimo global só é garantidamente alcançado com a aplicação da estratégia de reotimização completa junto à abordagem exata, algoritmo chamado de ILP^{icp}. Neste caso, a cada nova SFC entrante, o mapeamento de todas as SFC ativas sobre o SN é reconfigurado. Dessa maneira, a nova SFC entrante irá fazer parte da nova solução ótima gerada. Porém, deve se levar em conta que o número de variáveis analisadas pelo modelo irá aumentar, o que demanda mais tempo de execução para se gerar o resultado ótimo. Apesar de não contabilizado pelas métricas nos experimentos realizados, outro aspecto fundamental é a interrupção do serviço que uma reotimização pode gerar. Neste caso, em uma reotimização com o algoritmo ILP^{icp} , todas as SFCs devem ter seu atendimento suspenso, para a migração dos componentes físicos alocados para seu atendimento, ação que pode gerar problemas na qualidade do serviço.

Com base nos experimentos computacionais, percebe-se que o algoritmo exato com uma reconfiguração completa dos mapeamentos das SFCs ativas (ILP^{icp}), mesmo em topologias pequenas, como a *CompuServe*, é custoso computacionalmente e pode levar horas para ser executado, ação que certamente inviabiliza sua aplicação prática. Os algoritmos ILP^{i} e ILP^{ic} apesar de acarretarem em um tempo de execução mais baixo em relação ao algoritmo ILP^{icp}, podem ser considerados lentos, pois tal processamento é dependente da quantidade de variáveis e restrições presentes no modelo. Tais experimentos motivam a

aplicação de heurísticas e técnicas de aprendizado de máquina para a geração de boas soluções para o VNF-PC em um tempo praticável, independentemente do número de dispositivos do substrato de rede, SFCs ativas ou do intervalo de chegada das SFCs. O algoritmo que efetua a reconfiguração completa dos roteamentos e instâncias de VNFs, não se mostrou promissor em relação ao algoritmo ILP^i devido a um maior tempo computacional observado.

Devido ao baixo número de componentes a serem processados, as restrições de atraso fim a fim não se mostraram rígidas, e a topologia de rede *Compuserve* não se mostra promissora para ser estudada nos capítulos seguintes deste trabalho. Em tais capítulos, por serem aplicadas técnicas heurísticas, pretende-se adotar uma topologia maior em números de componentes físicos, aspectos que se assemelham mais a ambientes reais de redes baseadas em tecnologias NFV. Dentre as estratégias de reconfiguração, percebe-se que reotimizar todas as restrições do modelo é uma ação extremamente custosa computacionalmente. Percebe-se também que os algoritmos ILP^{i}e ILP^{ic} geram resultados muito próximos, justamente por ambos não reposicionarem (*(re)placement*) as VNFs instanciadas.

Capítulo 5

Heurística Semigulosa com Busca Local

Neste Capítulo é proposta uma heurística para resolver o VNF-PC em um baixo tempo computacional, em um ambiente *online*, sem migrações de mapeamentos, sendo a reotimização aplicada somente em termos de redimensionamento das capacidades atreladas às instâncias de VMs. Inicialmente, no Apêndice D, são apresentados alguns conceitos introdutórios sobre algoritmo gulosos, heurísticas e metaheurísticas, com foco na VNS. Na Seção 5.1, uma adaptação do algoritmo semiguloso e uma busca local são apresentados para resolver o VNF-PC. Na Subseção 5.1.1 é apresentada a estrutura construtiva da heurística proposta, e na Subseção 5.1.2 as estruturas de vizinhança desenvolvidas. Na Subseção 5.1.3 é mostrada a busca local. Alguns experimentos computacionais que comparam a heurística proposta com um algoritmo exato são mostrados na Seção 5.2, e por fim, na Seção 5.3 são realizadas as considerações finais.

5.1 Heurística Semigulosa com Busca Local para o VNF-PC

A adaptação heurística apresentada nesta Seção, chamada Semigulosa com Busca Local (SGBL), visa prover soluções rápidas para o VNF-PC. O algoritmo proposto é baseado em uma heurística semigulosa, e a busca local na metaheurística VNS. Tal algoritmo busca a criação de soluções diversificadas para o problema VNF-PC, explorando diferentes pontos do espaço de solução. O algoritmo semiguloso é fundamentado em um processo iterativo, multipartida, que constrói uma solução factível de cada vez. Propõe-se neste algoritmo, após a criação de n soluções, refinar a melhor solução encontrada dentre as n geradas com a busca local. O algoritmo proposto, mostrado no pseudocódigo do Algoritmo 1, é uma variação do algoritmo Semiguloso proposto por Hart and Shogan (1987) e

compartilha conceitos da metaheurística *Greedy Randomized Adaptive Search Procedure* (GRASP), proposta por Feo and Resende (1995) (ver detalhes no Apêndice D). O Algoritmo 1 recebe como parâmetro uma SFC v a ser mapeada, e o parâmetro α, usado no algoritmo construtivo.

A principal diferença da implementação mostrada no Algoritmo 1 para a metaheurística GRASP, está no ponto que não é aplicada a busca local em cada solução s' gerada. Neste caso, um conjunto de soluções factíveis s' é gerado por um número limitado de iterações (linhas 2 a 9) com o algoritmo construtivo semiguloso (Algoritmo 2), sendo a melhor solução armazenada (linha 5). Posteriormente, somente a melhor solução encontrada, caso exista, é então submetida a uma busca local (linha 13). Caso nenhuma solução factível seja encontrada, a SFC é rejeitada (linha 11).

Em experimentos preliminares, realizados para parametrizar a heurística proposta, percebe-se que a aplicação da busca local a cada solução gerada, como feito na metaheurística GRASP, elevou consideravelmente o tempo de processamento, sem gerar resultados significativos. Contudo, foi notado que, com a aplicação da busca local somente na melhor solução, poderia se investir mais tempo nesta etapa, fator que melhorou o desempenho da abordagem. Assim, a aplicação da busca local somente na melhor solução se dá pela necessidade de uma tomada de decisão rápida quanto ao mapeamento ou não da SFC.

Algoritmo 1 SGBL (v, α)

```
1:  f*, i ← 0, 0
2:  enquanto i < MAX_1 faça
3:      s' ← Construtivo Semiguloso (v, α)      {fase de construção semigulosa}
4:      se f(s') > f* então
5:          s*, f*, i ← s', f(s'), 0 {armazena melhor solução e reinicia i}
6:      senão
7:          i ← i + 1
8:      fim se
9:  fim enquanto      {MAX_1 iterações sem melhoras atingidas}
10: se f* = 0 então
11:     devolve false      {falha de mapeamento}
12: senão
13:     v = RVNS (s*, α)      {busca local somente na melhor solução factível}
14: fim se
15: devolve v {mapeamento otimizado}
```

O número máximo de iterações realizadas pelo algoritmo SGBL é definido por um limite de iterações sem melhora, chamado de MAX_1 (linha 2). Esse método de parada é usado devido à grande variabilidade de SFCs que podem ser geradas e aos inúmeros diferentes estados que o SN residual pode admitir. Dessa forma, em alguns casos, uma SFC pode ser mais facilmente mapeada, com poucas alterações em sua vizinhança; e em outros, uma SFC mais complexa pode estar sujeita a mais trocas de vizinhanças, requerendo um

número de iterações maior. Caso nenhuma solução factível seja encontrada até MAX_1, a SFC é rejeitada (linha 12).

5.1.1 Heurística Construtiva Semigulosa

Nesta Seção é apresentada a heurística construtiva semigulosa para criar uma solução inicial factível para algoritmo SGBL (linha 3, Algoritmo 1). Tal heurística é motivada pelo bom desempenho da heurística proposta no trabalho Araujo et al. (2018), mas ajustada para resolver o problema VNF-PC. Alguns dos pontos que diferenciam a heurística proposta nesta Seção, para a heurística proposta no trabalho Araujo et al. (2018), consistem na equação de formação da LRC (Equação 5.1), na inserção da etapa de posicionamento das instâncias de funções de redes, e na adição das restrições de atraso fim a fim do problema.

No algoritmo SGBL, a construção de uma solução inicial factível é indispensável, pois é essencial haver uma solução viável para a aplicação da busca local, e geração de possíveis melhorias em termos da função objetivo. O pseudocódigo do algoritmo de geração da solução inicial é simples (Algoritmo 2), e executado para cada SFC entrante.

Algoritmo 2 Construtivo Semiguloso(v, α)

1: $s^v \leftarrow i \in N$ tal que $s^v = i$
2: $d^v \leftarrow j \in N$ tal que $d^v = j$
3: **para cada** $k \in N_f^v$ **faça**
4: $\quad$ LRC$_k \leftarrow$ calculaLRC(k, α)
5: $\quad$ $i \leftarrow$ elementoAleatorio(LRC$_k$)
6: $\quad$ **se** $mapeia(k, i) ==$ **falso então**
7: $\quad\quad$ **devolve** false $\quad$ {falha no posicionamento de instância ou mapeamento de VNF}
8: $\quad$ **fim se**
9: **fim para**
10: **para cada** $(k, l) \in N^v$ **faça**
11: $\quad$ $(i, j) \leftarrow$ nó físico de (k, l)
12: $\quad$ **se** $BFS(i, j) ==$ **falso então**
13: $\quad\quad$ **devolve** false $\quad$ {falha no roteamento}
14: $\quad$ **fim se**
15: **fim para**
16: **se** $calculaDelay(v) > t_{dl}^v$ **então**
17: $\quad$ **devolve** false $\quad$ {atraso fim a fim inviável, extrapola limite do cliente}
18: **fim se**
19: **devolve verdadeiro**

O algoritmo 2 recebe como entrada o SN residual, a SFC a ser processada, e o parâmetro α usado para controle de admissibilidade da LRC. Ao final, o algoritmo gera um mapeamento viável da SFC, ou uma rejeição. Tal algoritmo divide-se em duas estruturas:

I Linhas 3 a 8: relativo ao posicionamento das instâncias de VNFs (*Placement*) e a atribuição dos nós virtuais. Dada uma SFC $v \in V$, deve-se encontrar um conjunto de nós físicos que possuam uma capacidade de CPU e memória residuais suficientes para a instanciação das instâncias de VMs solicitadas e para o mapeamento de cada VNFs $k \in N_f^v$ demandada. Para cada nó virtual deve-se fazer o mapeamento sobre um nó físico, sendo que um nó físico nunca deve hospedar mais de um nó virtual de uma mesma SFC. Nesta etapa, caso necessário, a atribuição de uma VNF demandada pode ser feita em uma instância de VM já posicionada, sendo necessário apenas ajustar as capacidades de memória e CPU utilizadas;

II Linhas 10 a 15: relativo ao encadeamento das funções de rede. Dado um mapeamento válido de nós virtuais, e considerando a capacidade residual de largura de banda de cada enlace físico, para cada arco virtual $(k, l) \in L^v$ deve-se encontrar um encadeamento (roteamento) no substrato que conecte os nós físicos cujos nós do arco virtual são mapeados.

As duas estruturas de repetição descritas anteriormente são tratadas sequencialmente de forma que se o mapeamento de nós virtuais não obtiver sucesso (laço I), a SFC é rejeitada sem que esta chegue na etapa de mapeamento de arcos (laço II). São realizados nas linhas 1 e 2 os mapeamentos dos nós virtuais de origem (s^v) e destino (d^v). O posicionamento e dimensionamento das instâncias de VNFs $b \in B_f$ é realizado iterando-se por todas VNFs $k \in N_f^v$ da SFC v (linhas 3 a 8). A função $mapeia(k, i)$ consiste na verificação se o nó i, escolhido arbitrariamente dentro da LRC_k, possui uma instância aberta do tipo da função requerida por k, e se essa instância possui recursos para hospedar a VNF k. Caso não exista uma instância posicionada da função requerida, o algoritmo deve instanciar uma VNF $b \in B_f$ para atender à função de rede virtual $k \in N_f^v$ demandada. Essas ações consistem na aplicação dos conjuntos de Restrições 4.1 até 4.4, e buscam dimensionar (ou redimensionar) o tamanho da instância de VNF instanciada conforme a demanda.

Caso os mapeamentos das VNFs demandadas aconteçam com sucesso, a etapa de encadeamento é realizada iterando-se por todos os arcos virtuais (linhas 10 a 15). Neste caso, para efetuar-se o roteamento, busca-se minimizar o número de saltos efetuados nos arcos do SN, nessas condições, utiliza-se o algoritmo BFS. Caso o atraso fim a fim gerado pelo mapeamento seja superior ao limite estabelecido pelos clientes (Restrições 4.11), o mapeamento não é aceito (linha 17). Caso todas as etapas aconteçam com sucesso, o algoritmo retorna com sucesso uma solução factível na linha 19.

LRC: seguindo os preceitos de Hart and Shogan (1987); Feo and Resende (1995), para cada VNF $k \in N_f^v$ existe uma Lista de Candidatos (LC) para hospedar a função de rede k demandada (LC_k) (ver Apêndice D). A partir da LC_k e de um parâmetro de admissibilidade $\alpha \in [0, 1]$, é criada uma Lista Restrita de Candidatos (LRC) para cada $k \in N_f^v$ (LRC_k, linha 4, Algoritmo 2) com elementos que atendam os critérios abaixo:

$$\{i \in LRC_k \mid h(i) \leq h_{max} - \alpha(h_{max} - h_{min})\} \tag{5.1}$$

$$h(i) = hops(s^v, i) + hops(i, d^v) \tag{5.2}$$

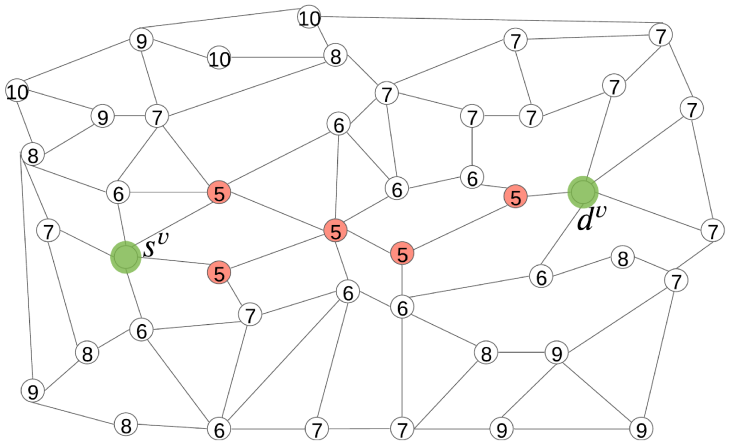
Neste caso, $hops(i, j)$ é uma função que retorna o número de saltos (arcos físicos intermediários) presentes no menor caminho físico entre os nós i e j, sendo calculada com uma BFS. Essa função é usada para controlar o mapeamento das VNFs demandadas em servidores mais próximos das intermediações dos nós de origem (s^v) e destino (d^v), e consequentemente controlar o atraso fim a fim dentro de um limite aceitável para o usuário final (Restrições 3.11). Como proposto por Feo and Resende (1995), cada LRC_k gerada para cada elemento $k \in N_f^v$ é composta de elementos que podem ser inseridos na solução parcial sem gerar inviabilidades nas restrições do problema.

Na formulação clássica da LRC, a escolha de um valor adequado de α é decisiva, pois controla o seu tamanho e a diversificação de elementos presentes no espaço de solução. No algoritmo apresentado neste trabalho, por tratar-se de um problema de maximização, um valor de α próximo a 1 tende a gerar uma LRC mais seletiva, permitindo o mapeamento de cada VNF $k \in N_f^v$ nos servidores mais próximos dos *endpoints* s^v e d^v (Figura 5.1a). Por outro lado, um valor de α próximo a 0 tende a gerar uma LRC menos seletiva, deixando o algoritmo aleatorizado (Figura 5.1c), gerando mais diversificação no espaço de solução. As Figuras 5.1a até 5.1c exemplificam a função $h(x)$ aplicada a alguns roteadores $i \in N$ de um SN qualquer (valor interno de cada nó). Para cada valor de α, uma quantidade de elementos diferente é inserida na LRC_k de cada VNF $k \in N_f^v$ (nós coloridos). É possível observar que quanto mais próximos os roteadores forem dos pontos de origem e destino (pontos verdes), menor o valor de $h(x)$ (cores mais fortes), do contrário, quanto mais longe estiverem, maior o valor de $h(x)$ (cores mais fracas).

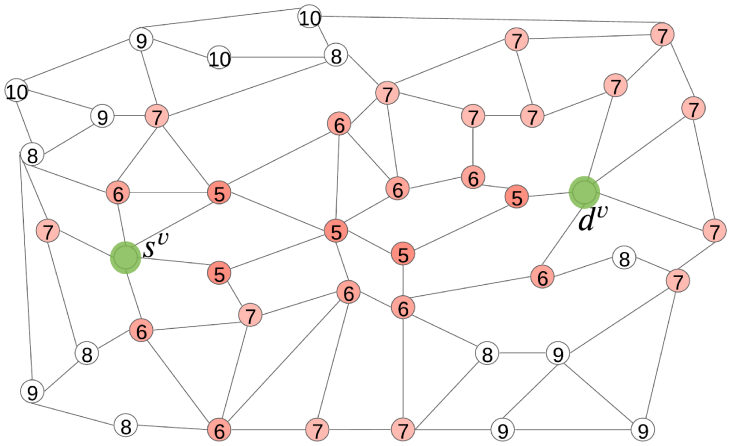
Na Figura 5.1a, são inicialmente verificados os valores de $h(x)$ para cada nó físico do SN. Após estes cálculos, os valores 10 e 5 são encontrados e atribuídos respectivamente para h_{max} e h_{min}. Aplicando $h_{max} = 10$, $h_{min} = 5$ e $\alpha = 0.9$ na Equação 5.1, gera-se um limite de 5.5. Ou seja, se a função $h(x)$ de determinado nó obtiver um valor menor que 5.5 (nós destacados na Figura 5.1a), tal nó pode ser usado no processamento da SFC, do contrário ele é desprezado. Percebe-se que, como esperado, um alto valor de α gera poucas opções para os mapeamentos de VNFs.

Figura 5.1: Cálculo da Função $h(x)$ para diferentes valores de α

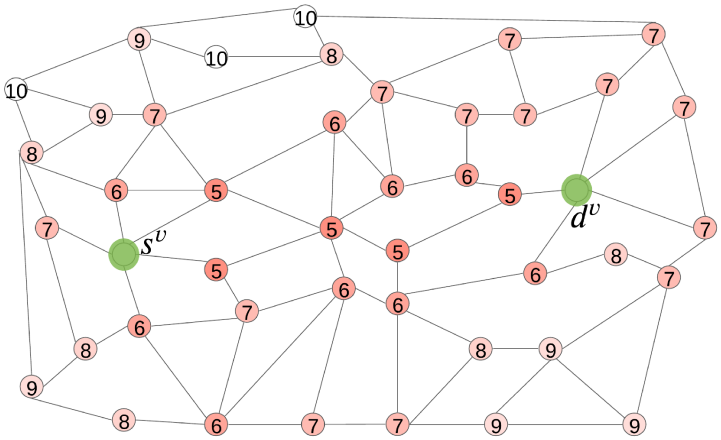
(a) LRC seletiva, $\alpha = 0.9$



(b) LRC moderada, $\alpha = 0.5$



(c) LRC aleatorizada, $\alpha = 0.1$



Fonte: Elaborado pelo autor.

No exemplo da Figura 5.1b, aplicando $h_{max} = 10$, $h_{min} = 5$ e $\alpha = 0.5$ na Equação 5.1, gera-se um limite de 7.5. Assim, se a função $h(x)$ de determinado nó físico obtiver um valor menor que 7.5 (nós destacados na Figura 5.1b), tal nó pode ser usado no pro-

cessamento da SFC. Percebe-se que, diferentemente do caso anterior ($\alpha = 0.9$), um valor moderado de α gera mais opções para os mapeamentos de VNFs, deixando alguns nós mais afastados dos pontos de origem e destino fora da LRC.

Na Figura 5.1c, aplicando $h_{max} = 10$, $h_{min} = 5$ e $\alpha = 0.1$ na Equação 5.1, gera-se um limite alto, sendo de 9.5. Deste modo, se a função $h(x)$ de determinado nó físico obtiver um valor menor que 9.5 (nós destacados na Figura 5.1c), tal nó pode ser usado no processamento da SFC. Percebe-se que, diferentemente dos casos anteriores ($\alpha = 0.9$ e $\alpha = 0.5$), um valor baixo de α gera muitas opções para os mapeamentos de VNFs, deixando apenas os poucos nós mais afastados fora da LRC. Percebe-se que com a redução do valor α, a LRC vai aumentando sua cardinalidade de elementos, deixando o algoritmo mais aleatorizado em relação a um valor de α alto.

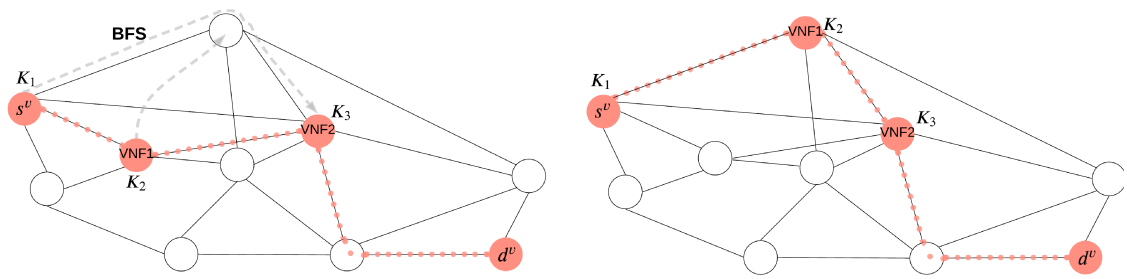
5.1.2 Vizinhanças

A vizinhança de uma solução s é chamada conjunto $N(s) \in X$, em que X consiste no espaço de todas as diferentes soluções factíveis existentes para determinada instância (Resende and Ribeiro, 2010). Cada solução $s' \in N(s)$ pode diferir-se de s por poucos ou muitos elementos. As estruturas de vizinhança apresentadas neste trabalho são fundamentadas em dois movimentos de trocas: vizinhanças $N_1(s)$ e $N_2^{\alpha}(s)$.

Vizinhança $N_1(s)$: baseada no reposicionamento de uma VNF em uma região próxima aos nós físicos em que ela está conectada. Na implementação proposta, uma VNF $k \in N_f^v$ é escolhida aleatoriamente. Tal reposicionamento é feito em um nó físico presente no caminho mínimo entre o nó posterior e anterior a k. O referido caminho é encontrado com uma BFS modificada. Tal modificação garante que o novo roteamento se difira do anterior, e garanta recursos para o processamento das demandas de recursos do nó k (Figura 5.2).

Figura 5.2: Movimento de vizinhança N_1

(a) Solução inicial s (b) Solução vizinha $s' \in N(s)$

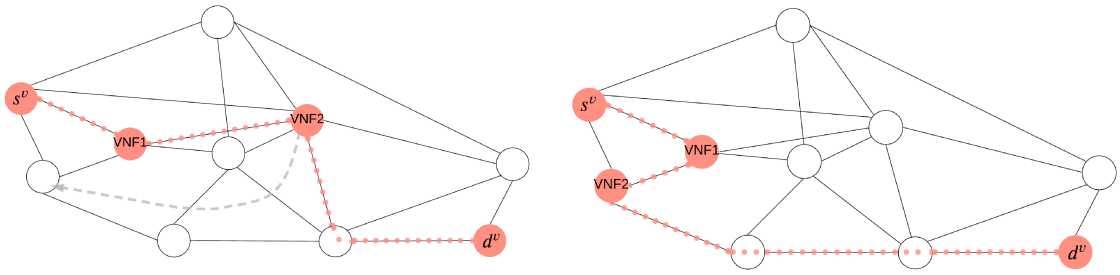


Fonte: Elaborado pelo autor.

Vizinhança $N_2^\alpha(s)$: baseada no reposicionamento de uma VNF em uma região do SN definida pelo parâmetro α (originário da LRC). A vizinhança $N_2^\alpha(s)$ gera uma modificação na solução corrente que pode ser atenuada ou acentuada pelo valor do parâmetro α. Nesta proposta, uma VNF $k \in N_f^v$ é escolhida aleatoriamente. Após, é feita uma tentativa de mapeamento da VNF k em algum nó físico aleatório de sua respectiva LRC_k (definida com o valor de α). Caso o mapeamento ocorra com sucesso, um novo roteamento é feito para reconectar a VNF k ao encadeamento de funções $(k, l) \in L^v$ (Figura 5.3).

Figura 5.3: Movimento de vizinhança N_2^α

(a) Solução inicial s (b) Solução vizinha $s' \in N(s)$



Fonte: Elaborado pelo autor.

5.1.3 Busca Local

Uma das características do VNP-PC é que o tamanho do espaço de soluções aumenta exponencialmente em relação ao número de componentes a serem processados. Tal característica deixa o número de vizinhanças a serem investigadas excepcionalmente alto, o que pode deixar o algoritmo de refinamento exaustivo, lento e inviável de ser aplicado. Assim, do mesmo modo que Xiao et al. (2011); Araujo et al. (2018), nesta tese é utilizado um algoritmo modificado da metaheurística VNS, chamado *Reduced Variable Neighborhood Search* (RVNS). O algoritmo RVNS é indicado para instâncias nas quais a busca local é cara. Neste caso a busca local determinística é alterada para melhorar sua eficiência em tempo computacional (Hansen and Mladenović, 2001).

Na implementação proposta, mostrada no Algoritmo 3, a cada iteração da busca local, parte-se da solução corrente factível s para obter uma solução vizinha aleatória dentro de uma vizinhança $N_k(s) \in X$. A cada iteração, a solução corrente (s^*) é submetida a uma busca local (linha 6 ou 8) originando uma nova solução, chamada s'. Se a solução gerada s' for melhor que a solução s^*, a busca continua a partir de s', retornando-se à primeira estrutura de vizinhança (linha 6). Caso contrário, a busca continua a partir da

estrutura de vizinhança $k+1$ (linha 8). A condição de parada é definida como MAX_2 iterações sem melhora.

Algoritmo 3 RVNS(s^*, α)

```
1:  i ← 0
2:  enquanto i ≤ MAX_2 faça
3:     k ← 1
4:     enquanto k ≤ 2 faça
5:        se k = 1 então
6:           s' = vizinhançaN_1(s*)
7:        senão
8:           s' = vizinhançaN_2^α(s*)
9:        fim se
10:       se f(s') > f(s*) então
11:          s*, k, i ← s', 1, 0
12:       senão
13:          k ← k + 1, i ← i + 1
14:       fim se
15:    fim enquanto
16: fim enquanto
17: devolve s*
```

5.2 Experimentos Computacionais

Devido à inviabilidade computacional de se executar algoritmos exatos em cenários médios ou grandes, no Capítulo 4 os experimentos foram realizados em uma topologia de rede menor, denominada *Compuserve*. Neste capítulo propomos um tratamento heurístico, o que viabiliza a utilização de uma topologia de rede maior, sendo escolhida a *Cogent Network*. Tal topologia possui 186 nós e 214 arcos distribuídos fisicamente por países da América do Norte, Europa e Ásia, conforme mostrado na Figura 5.4.

Figura 5.4: Distribuição topológica dos nós e arcos físicos da rede *Cogent*

Fonte: http://www.topology-zoo.org/dataset.html

Os experimentos computacionais exibidos nesta Seção são realizados em um cenário *online* em relação à chegada das SFCs, e executados com os mesmos parâmetros de *hardware* e métricas apresentadas na Seção 4.1. Os experimentos desta Seção buscam responder às questões: quais os impactos as políticas de vizinhança causam na taxa de aceitação e no lucro? A heurística proposta é competitiva em termos de lucro? Para esse fim, simulações são realizadas para mostrar a eficiência da heurística e das vizinhanças propostas, além de definir o melhor valor do parâmetro α. Para tal, é proposto o uso do algoritmo ILP^i como um *baseline*. Tal escolha é baseada nos experimentos do Capítulo 4, em que foi mostrado que o algoritmo ILP^i, apesar de não garantir um resultado ótimo global, consegue bons resultados, e em um tempo inferior aos outros algoritmos exatos.

Devido à escolha de uma topologia de rede maior, com mais nós e arcos físicos, propõem-se realizar os experimentos em dois cenários de parametrização:

I O mesmo apresentado na Seção 5.1, com as mesmas classes de serviços e valores de λ, mas considerando um horizonte temporal de $50000t$ na análise da chegada de SFCs. Este cenário é referenciado nesta Seção como esparso em relação à chegada de SFCs;

II O mesmo apresentado na Seção 5.1, com as mesmas classes de serviços, mas com uma taxa de chegada de SFCs intensificada em 80% em relação aos valores de λ apresentados anteriormente. Como a entrada de SFCs é mais acentuada, o horizonte de análise de processamento é menor, sendo este de $10000t$. Este cenário é referenciado nesta Seção como denso em relação à chegada de SFCs.

5.2.1 Algoritmos Avaliados

Cada algoritmo avaliado realiza o mapeamento de uma nova SFC com base nos recursos residuais do SN no momento de mapeamento t. O valor do parâmetro α presente na heurística SGBL é variado em $\alpha \in \{0.5, 0.7, 0.9, 1.0\}$. Como definido na Subseção 5.1.1, tal parâmetro determina a quantidade de componentes presentes da LRC de cada no virtual de cada SFC processada. Neste caso, quanto maior o valor de α (próximo a 1), menos elementos estarão na LRC, e o algoritmo tende a ter um comportamento guloso; do contrário, a LRC conterá mais elementos, e o algoritmo tende a ficar aleatorizado. Os algoritmos avaliados são:

- ILP^i - algoritmo exato, no qual reotimizam-se as restrições de dimensionamento de instâncias de VNFs (Capítulo 4). Usado como *baseline*;

- SG - heurística semigulosa (sem busca local). Neste caso é variado o valor do parâmetro α;
- SGBL - heurística semigulosa com a busca local. Neste caso é variado o valor do parâmetro α;
- SGBLILP - heurística semigulosa com a busca local e o algoritmo ILP^i. Neste caso é variado o valor de α, e, caso o algoritmo SGBL não consiga gerar uma solução factível, o algoritmo ILP^i é acionado para tentar recuperar a falha de mapeamento.

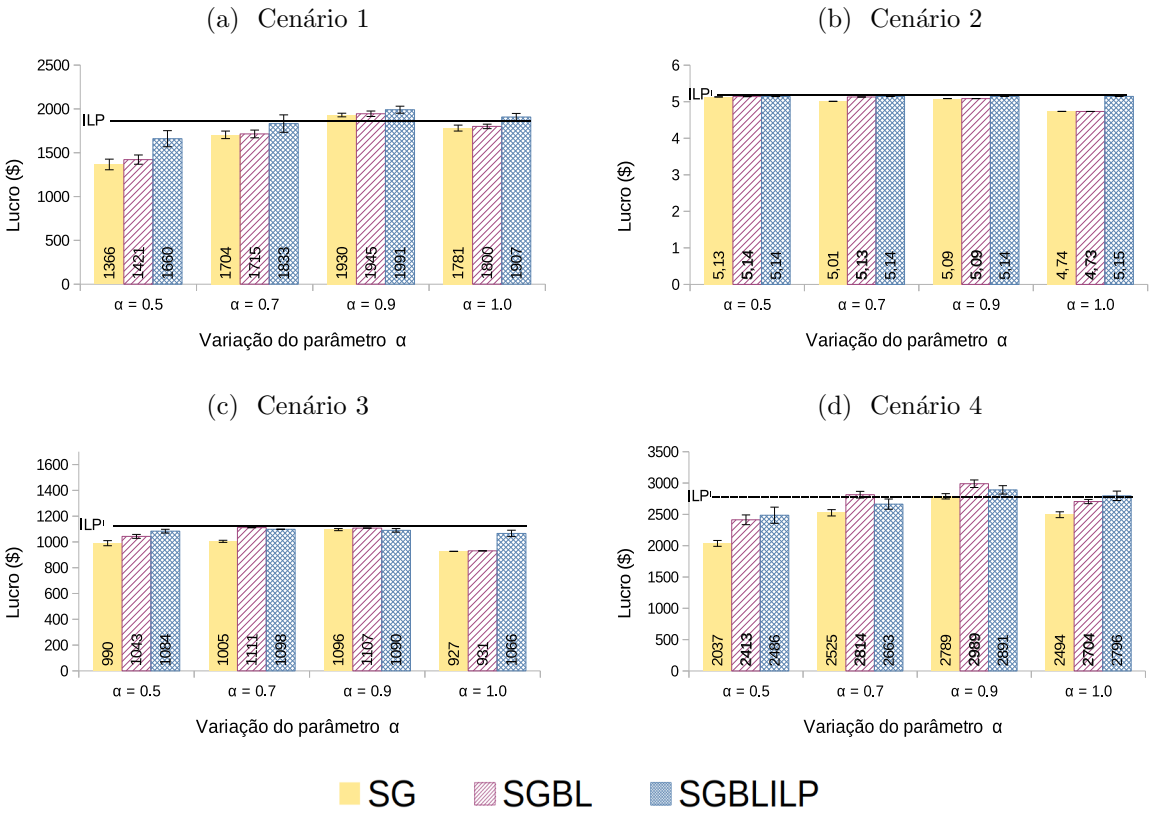
5.2.2 Comparação entre a Heurística e o Algoritmo Exato

Em cada cenário são realizados os mapeamentos de mais de 300 SFCs. Os experimentos heurísticos, por serem não determinísticos, são repetidos 15 vezes e mostrados com um IC de 95% (Figuras 5.5 a 5.7). A baixa dispersão dos dados com o IC proposto demonstra a robustez da heurística. Os limites de iterações do algoritmo SGBL são definidos como $MAX_1 = 50$ e $MAX_2 = 300$. Tais valores são definidos, com base em experimentos preliminares, para que a heurística seja competitiva em relação ao ILP^i, em tempo de execução, até nos cenários mais simples. Para fins de simplicidade de visualização, as Figuras referentes ao Cenário esparso foram inseridas separadamente no Apêndice E. As Figuras 5.8 a 5.10 ilustram as métricas descritas na Subseção 4.1.1 para uma única execução dos algoritmos em uma instância escolhida aleatoriamente. Nestes casos, o eixo x apresenta o tempo de simulação em unidades de tempo t e o eixo y a métrica observada. Na geração destes gráficos, exportaram-se às métricas em questão a cada unidade de tempo t, correspondendo a entrada da primeira até a última SFC de cada conjunto de dados. Para cada um dos gráficos mostrados nas Figuras 5.8 a 5.10, é gerada uma tabela que resume o comportamento de cada algoritmo (Apêndice E).

Análise do lucro (Figuras 5.5 e E.1): ao se aumentar a taxa de chegada de SFCs, passando do cenário esparso, para o denso, o lucro final diminuiu. Um fator que corroborou para essa queda foi a diminuição da taxa de aceitação. Com uma chegada de SFCs intensificada, cenário 2 a parte (Figuras 5.5b e E.1b), a disputa entre SFCs por recursos do SN aumenta, produzindo uma queda na taxa de aceitação, afetando negativamente a receita, e consequentemente reduzindo o lucro. Como exemplo, no Cenário 4 esparso e o algoritmo ILP^i, a taxa de aceitação é de $\approx$ 88% (Figura E.3d) e o lucro é de $\approx$ 5280 (Figura E.3d). Com uma taxa de chegadas intensificada, a taxa de aceitação cai para $\approx$ 82% (Figura 5.7d) e o lucro para $\approx$ 2800 (Figura 5.7d). Um ponto que fundamenta essa queda no lucro é relacionado às SFCs que deixaram de ser atendidas nos

experimentos densos. Ao se intensificar a chegada de SFCs, o atendimento das SFCs da Classe 2 é diminuído, e como a receita das SFCs desta classe é baixa (Tabela E.1 e Figura E.1b), este fator não afeta significativamente o lucro. Por outro lado, as Classes 1 e 3 tiveram sua aceitação reduzida em ≈ 13% cada, e como a receita gerada por estas SFCs é bem mais expressiva, devido a elas terem um tempo de vida bem mais significativo, houve referida diminuição no lucro (Figuras E.1a e E.1c).

Figura 5.5: Lucro total dos provedores no cenário denso



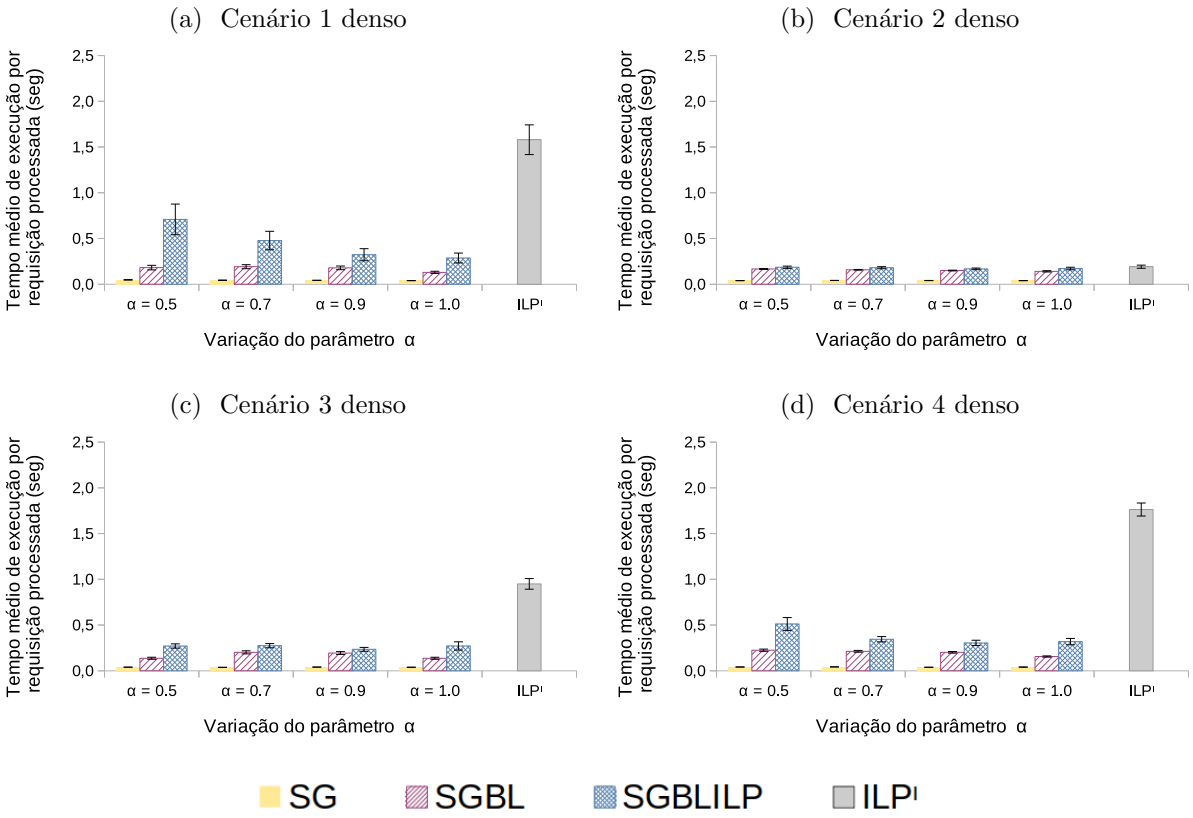
Fonte: Elaborado pelo autor.

Observando o algoritmo SGBLILP, percebe-se que o mapeamento das SFCs que não foram aceitas com a SGBL não afeta estatisticamente o lucro. Esse fato se dá por a heurística possuir um número baixo de rejeições, e pelo algoritmo ILP não conseguir reverter significativamente essa taxa de rejeição (Figura 5.7a). Modelo exato à parte, o maior lucro é obtido pelas variações que utilizam um α maior, como 0.9, e com a busca local. Neste caso, o parâmetro $\alpha = 0.9$ gera uma exploração racional do espaço de soluções, com uma maior seletividade de elementos presentes na LRC de cada nó virtual, propiciando uma economia relativa a um maior compartilhamento de servidores. Como exemplo, nos experimentos com o cenário 4 e denso, o algoritmo SGBL consegue diminuir o espalhamento dos nós virtuais em até ≈ 42% em comparação à variação sem busca local (Tabela E.5). Outro ponto é que, visto que o atraso fim a fim é uma restrição apertada nesta topologia (Figura 5.10), com α alto, a heurística consegue avaliar mais soluções factíveis em

um tempo menor em relação a um α baixo. Um α muito baixo deixa a geração de soluções inicias aleatorizada, em que várias soluções podem ser construídas e descartadas, pois, extrapolam o atraso fim a fim. Um valor de $\alpha = 1$ deixa o algoritmo guloso, e este tende a não gerar boas soluções, corroborando com as premissas das metaheurística GRASP, apresentada em Feo and Resende (1995). Demais análises sobre estes experimentos podem ser observadas no Apêndice E.

Análise do tempo de execução total (Figuras 5.6 e E.2): no cenário 2, por não haver SFCs ativas simultaneamente, foi usado para fundamentar a definição do número de iterações da heurística. Como apenas uma SFC é processada, não existe uma disputa entre diferentes SFCs por recursos do SN, nem VNFs já instanciadas ou servidores ativos para serem compartilhados.

Figura 5.6: Tempo de execução médio por SFC processada no cenário denso



Fonte: Elaborado pelo autor.

A parametrização referente ao número de iterações é definida de modo que o algoritmo construtivo semiguloso possua um tempo de execução competitivo com o algoritmo ILP^i até nos cenários mais elementares em números de variáveis e restrições (Figuras 5.6b e E.2b). Assim, define-se $MAX_1 = 50$. Em casos que poucas melhoras na solução corrente são observadas, o algoritmo possui um tempo de execução menor; e em caso de muitas melhoras, o tempo de execução tende a ser maior. De modo a priorizar a busca local, atribui-se um número maior de iterações para tal etapa em detrimento ao algoritmo

construtivo, neste caso, adota-se $MAX_2 = 300$. Observando a Figura 5.6d, o tempo gasto pela heurística é menor com $\alpha = 0.9$ em relação a $\alpha = 0.5$, acontecimento devido ao menor número de soluções vizinhas existentes para serem investigadas.

Especificamente, nos experimentos do cenário 2, como somente uma SFC é mapeada por vez, e não existem SFCs ativas, o algoritmo ILP^i consegue gerar uma solução ótima global. Devido à extrema simplicidade proposital (e irreal) do cenário avaliado, em que não existem disputas de recursos entre diferentes SFCs, fragmentação do SN, e instâncias de VNFs para serem compartilhadas, a heurística SGBL produz um lucro próximo ao ILP^i e a busca local não se mostra expressiva. Mas, à medida que o cenário de simulação se torna mais complexo, como no cenário 4, percebe-se que a busca local gera melhoras no lucro. O algoritmo SGBL gera um tempo de execução regular e baixo para todos os cenários avaliados (menos de 220 milissegundos no pior caso, Figura E.2d). Outras observações sobre estes experimentos são apresentadas no Apêndice E.

Análise da taxa de aceitação (Figuras E.3 e 5.7): mesmo a busca local não tendo um efeito significativo para melhorar taxa de aceitação em alguns experimentos (*e.g.,* $\approx 1.9\%$ para a heurística com $\alpha = 0.9$ no Cenário 4 esparso, Figura E.3d), ela é mais significativa em aumentar o lucro ($\approx 7.4\%$ no mesmo experimento, Figura E.1d), melhorar o compartilhamento de servidores físicos ($\approx 30.5\%$ no mesmo experimento, Tabela E.2), e reduzir o atraso fim a fim ($\approx 27.8\%$ no mesmo experimento, Tabela E.6).

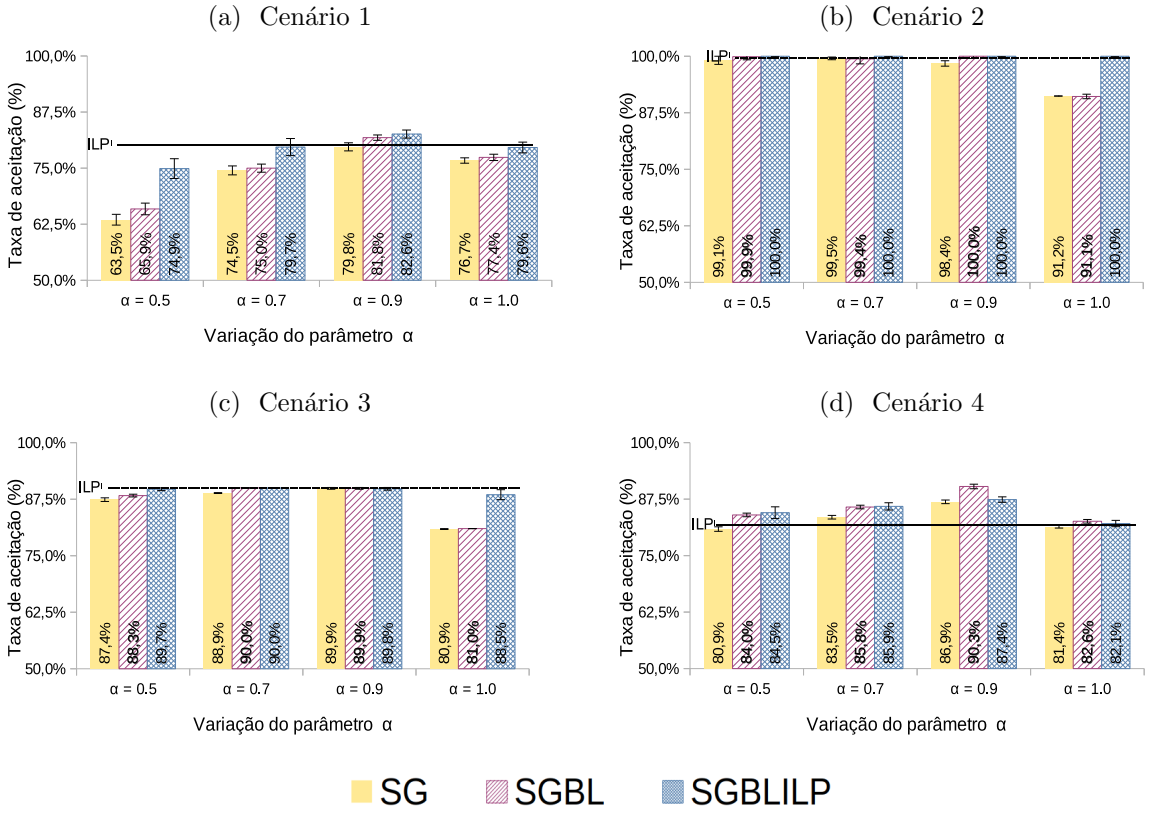
Em alguns experimentos mostrados nas Figuras E.3 e 5.7, o algoritmo SGBL gera uma taxa de aceitação bem próxima e/ou superior ao algoritmo ILP^i. Vale ressaltar que ambos os algoritmos maximizam o lucro, e não a taxa de aceitação, e que algoritmo ILP^i, por não efetuar a reconfiguração das restrições de posicionamento e encadeamento de VNFs, gera um resultado exato, potencialmente de boa qualidade, mas não garantidamente um ótimo global (mais detalhes no Capítulo 4). Como explicado anteriormente, o resultado ótimo global só é garantidamente obtido reconfigurando todas as restrições do modelo. E, apesar de uma alta taxa de aceitação ser crucial para a geração de lucros, outros fatores influenciam esse comportamento, como um bom posicionamento e espalhamento dos nós e roteamento dos caminhos virtuais.

Em todos os experimentos realizados foi percebida uma baixa variação no IC com uma alta taxa de aceitação e lucro para o algoritmo SGBLα9. Tal comportamento demonstra que as diferentes estratégias de vizinhança propostas (Subseção 5.1.2) conseguem conduzir a heurística para bons ótimos locais até mesmo nos cenários mais difíceis, como no cenário com $\lambda = 62.5$. Observando a variação heurística sem busca local, e com $\alpha = 1$ a parte, percebe-se que a escolha do parâmetro α com um valor baixo falha mais ao gerar soluções factíveis em relação a um valor alto. Por outro lado, o parâmetro α com valores maiores gera uma LRC mais seletiva, com opções de mapeamento para serem investigadas mais próximas dos pontos de origem s^v e destino d^v de cada SFC.

O parâmetro α é distintamente superior com o valor de 0.9. Essa ação potencial-

mente reduz o número de soluções infactíveis para serem investigadas, devido ao elevado atraso fim a fim; e implicitamente gera um balanceamento de carga sobre o SN. Neste caso, as SFCs são mapeadas sobre o SN sempre perto dos seus respectivos pontos de origem e destino, distribuindo a alocação de recursos físicos homogeneamente em regiões próximas no SN. Ação que potencialmente gera menos gargalos e fragmentações. Em contrapartida, tal valor de α gera um espaço de soluções mais restrito, ação que pode refletir em um espalhamento de VNFs ruim (baixo compartilhamento de servidores), sendo até 19.8% pior se comparado ao algoritmo ILP^i para o cenário esparso (Tabela E.2). Algumas outras observações relevantes sobre estes experimentos são explicadas no Apêndice E.

Figura 5.7: Taxa de aceitação final no cenário denso



Fonte: Elaborado pelo autor.

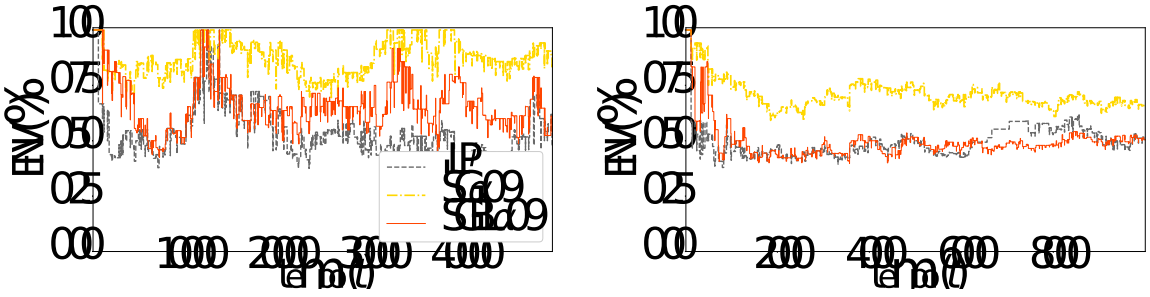
Análise do espalhamento dos nós virtuais: de forma geral, a variação SGBL com $\alpha = 0.9$ obteve os resultados mais estáveis e promissores em tempo de execução, taxa de aceitação e lucro, mostrando que uma LRC seletiva conduz a soluções de boa qualidade. Deste modo, para aumentar a legibilidade dos gráficos sequentes, estes são plotados somente com os experimentos realizados com os algoritmos SG com $\alpha = 0.9$, SGBL com $\alpha = 0.9$ e ILP^i. Devido ao grande desdobramento de abordagens e cenários realizados, são mostrados nesta Seção somente os experimentos com o cenário 4, visto que este é mais completo e significante, com uma maior heterogeneidade de SFCs processadas.

Em um cenário mais denso (Figura 5.8b), o algoritmo SGBL conseguiu melhorar

mais o compartilhamento em relação ao esparso, ficando com um valor muito próximo ao do algoritmo ILP^i (Tabela E.3). Em um cenário mais denso, com um baixo intervalo na taxa de chegada e saída de novas SFCs (Figura 5.8b), existem mais SFCs para serem mapeadas. Como os recursos de *hardware* dos dispositivos físicos são limitados (memória, processamento e largura de banda), existe uma necessidade de se ativar mais servidores físicos para hospedar as VNFs. Em tal cenário, o algoritmo também aumentou a sua taxa de aceitação (Figuras 5.7 e E.3), assim, com uma taxa de aceitação maior, e mais servidores ativos, maior a chance de haver um compartilhamento de servidores. Neste caso, a heurística SGBL conseguiu aumentar o nível de compartilhamento em $\approx 35\%$ em relação ao cenário esparso, e promoveu um compartilhamento de servidores apenas 0.05% menor que o algoritmo ILP^i. Como visto nos experimentos, vale ressaltar que o algoritmo ILP^i promoveu uma taxa de aceitação menor quando a taxa de entrada das SFCs foi intensificada.

Figura 5.8: Espalhamento dos nós virtuais no Cenário 4

(a) Chegada esparsa de SFCs (b) Chegada densa de SFCs



Fonte: Elaborado pelo autor.

Dentre os cenários analisados até o momento, a heurística SGBL é promissora em relação ao baixo tempo computacional, e à constante, e alta taxa de aceitação. Mas, em alguns casos, a heurística SGBL tende a não gerar um compartilhamento de servidor tão promitente quanto o algoritmo ILP^i. Comportamento que pode ser prejudicial em casos específicos, como em situações que o custo $\epsilon \in \mathbb{R}_+$ de manter determinado servidor ativo seja muito elevado, o que afetaria negativamente o lucro final. Demais análises podem ser observadas no Apêndice E.

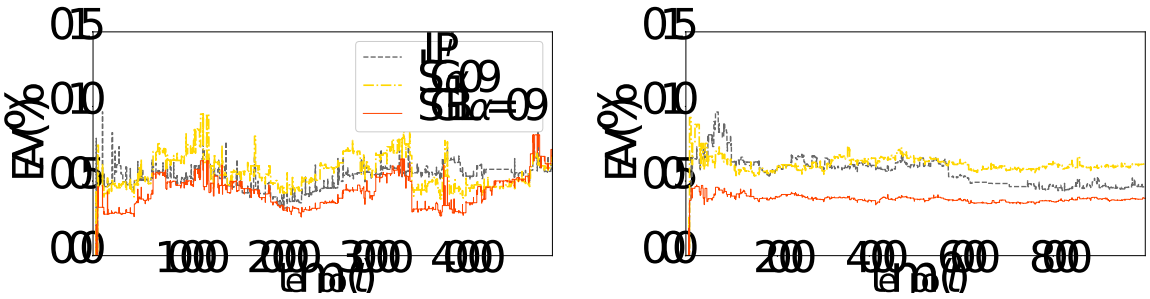
Análise do espalhamento dos arcos virtuais (Figura 5.9): a aplicação do algoritmo BFS para efetuar o roteamento nas abordagens heurísticas, além da baixa complexidade computacional, tem como propósito minimizar o número de saltos (*hops*) no encadeamento das VNFs sobre a rede física. Tal ação gera roteamentos de VNFs com caminhos físicos concisos (menor número de saltos). A premissa é que ao gerar-se um consumo menor de recursos do SN, em termos de arcos físicos, mais recursos residuais vão existir para a orquestração de uma nova SFC entrante. Ou seja, tal ação pode impactar beneficamente na taxa de aceitação. Premissa confirmada nos cenários

investigados (Figuras 5.9a e 5.9b), se for observada uma maior taxa de aceitação do algoritmo SGBL com $\alpha = 0.9$ em relação ao algoritmo ILP^i (Figuras E.3d e 5.7d).

O espalhamento dos arcos virtuais é $\approx 5.76\%$ pior no algoritmo ILP^i em relação à heurística com busca local no cenário esparso (Tabela E.4), e $\approx 5.51\%$ no cenário denso (Tabela E.5). Deve-se considerar que, por o ILP^i possuir uma visão global do SN, em alguns momentos tal algoritmo pode priorizar o compartilhamento de algum servidor em detrimento a um caminho com menos saltos. Percebe-se um *trade-off* entre o espalhamento de servidores e arcos. Quanto mais servidores forem compartilhados (Figura 5.8), mais os encadeamentos entre as VNFs vão ser potencialmente espalhados sobre SN para fornecer uma conexão para determinada SFC. Demais análises no Apêndice E.

Figura 5.9: Espalhamento dos arcos virtuais no Cenário 4

(a) Chegada esparsa de SFCs (b) Chegada densa de SFCs



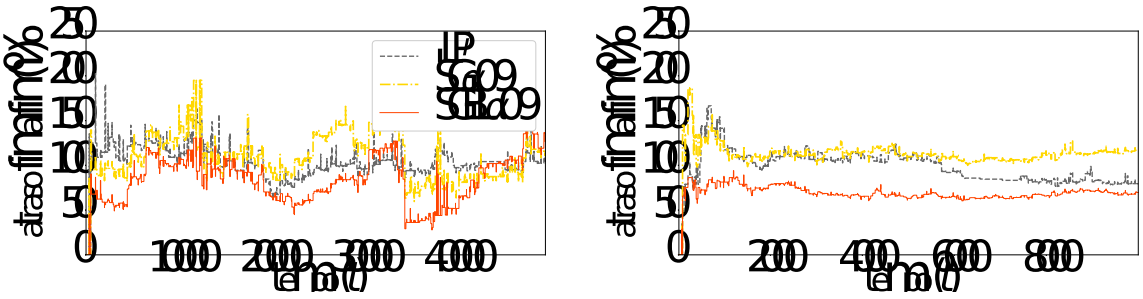
Fonte: Elaborado pelo autor.

Análise do atraso médio fim a fim (Figura 5.10): em ambos os experimentos, o algoritmo ILP^i, por maximizar o lucro, gera na maioria dos mapeamentos, roteamentos mais longos, que implicam em um atraso fim a fim maior. Este acontecimento é devido ao custo dos arcos serem menores em relação aos custos de se manter um servidor ativo, assim, tal modelo prioriza o compartilhamento de servidores em detrimento a roteamentos mais curtos. Por outro lado, a heurística, por gerar mapeamentos mais próximos dos nós s^v e d^v de cada SFC, tende a gerar uma melhor QoE para o usuário final. Neste caso, os roteamentos possuem menos saltos, ação que reduziu o atraso fim a fim em até $\approx 40\%$ em comparação o algoritmo exato (Tabela E.7).

Comparando as Figuras 5.10a com 5.10b e 5.9a com 5.9b, percebe-se que existe um comportamento similar em ambos casos. Com base em tais comportamentos, infere-se que, dentre outros fatores, para gerar-se um baixo atraso fim a fim, é necessário haver um bom roteamento dos arcos virtuais $(k, l) \in L^v$, além de um bom posicionamento das VNFs $k \in N_f^v$. Neste sentido, um roteamento que gera um menor atraso fim a fim, também é benéfico por utilizar menos arcos do SN, e deixar mais recursos residuais livres para serem utilizados nos mapeamentos de outras SFCs. Outras análises são descritas no Apêndice E.

Figura 5.10: Atraso fim a fim no Cenário 4

(a) Chegada esparsa de SFCs (b) Chegada densa de SFCs



Fonte: Elaborado pelo autor.

5.3 Considerações Finais

Neste Capítulo foi apresentada uma heurística para resolver o problema VNF-PC em um ambiente *online* e baixo tempo computacional, respeitando os aspectos de QoS dos clientes. Foram realizados experimentos para verificar o desempenho do algoritmo proposto. Alguns experimentos foram realizados em cenários mais simples, como o cenário 2, e utilizados para fundamentar o tempo de execução de cada etapa da heurística proposta (parâmetros MAX_1 e MAX_2). Tais valores são determinados para que a heurística seja competitiva em tempo de execução perante o modelo exato, executado no CPLEX, até nos casos mais triviais do referido problema. Outros experimentos computacionais foram realizados para definir o melhor valor do parâmetro α a ser utilizado, sendo, de modo geral, o melhor comportamento percebido para valores que geram uma LRC mais seletiva, como $\alpha = 0.9$. Por outro lado, percebeu-se que uma LRC com $\alpha = 1$ implica em resultados de baixa qualidade, devido ao mapeamento guloso.

Com base nos experimentos computacionais, é percebido que um algoritmo exato é custoso computacionalmente, e mesmo reconfigurando-se somente as restrições de instâncias de VNFs nos servidores (algoritmo ILP^i), tende a ser impraticável para cenários mais complexos. Por outro lado, o algoritmo heurístico, apesar de renunciar uma garantia de otimalidade da solução, gera bons resultados, com altos lucros e taxa de aceitação, além de poder ser aplicado em um reduzido tempo computacional para cenários complexos e com várias e diferentes SFCs ativas simultaneamente.

Foi proposto e avaliado um algoritmo híbrido, chamado SGBLILP, em que, caso o algoritmo SGBL rejeitasse um mapeamento, tal algoritmo tenta-se reverter a rejeição com a resolução pelo ILP^i. Percebe-se que a heurística SGBLILP não é eficaz, na prática, ela não gera melhorias na taxa de aceitação em relação à heurística SGBL parametrizada

com $\alpha = 0.9$. Logo se conclui que, se uma SFC é rejeitada pelo algoritmo SGBL, é porque existe uma possibilidade alta de ela ser realmente infactível.

Devido ao elevado número de componentes a serem processados, e a distância entre os servidores físicos da topologia de rede *Cogent* ser maior, ação que deixa as restrições de atraso fim a fim terem mais importância, tal topologia se mostra promissora para ser estudada nos capítulos seguintes deste trabalho. Em outra análise, pelo cenário 4 combinar as SFCs de todas as classes de serviços de rede, este se mostra mais real, o deixando mais relevante de ser analisado. Propõe-se nos próximos capítulos gerar um algoritmo híbrido, que combine as vantagens das técnicas de aprendizado de máquina com os algoritmos exatos e heurísticos. Infere-se que, com a utilização de técnicas de aprendizado de máquina, pode-se reduzir o espaço de busca, e melhorar o tempo de execução dos algoritmos.

Capítulo 6

Redução do espaço de solução: Abordagem de Clusterização por Localização Espacial

Em um modelo matemático de otimização, o número de variáveis e restrições do VNF-PC está diretamente ligado com o número de dispositivos ofertados pelo SN e demandados por cada SFC. Em ambientes reais, o espaço de busca é muito grande, não sendo possível explorá-lo em sua totalidade em um tempo computacionalmente viável. A percepção empregada no algoritmo apresentado neste Capítulo, parte do princípio que, no VNF-PC, à medida que o número de componentes da rede cresce linearmente, o espaço de busca torna-se maior, e, consequentemente, o tempo de processamento computacional aumenta exponencialmente. O mesmo princípio deve ser válido inversamente. Para reduzir o espaço de busca, e consequentemente o tempo de resolução, ou diminui-se o número de componentes demandados por cada SFC, ou dos componentes ofertados pelo SN.

Para tratar referido problema, é proposto o redimensionamento do espaço de busca através da aplicação de técnicas de clusterização. O emprego de tais técnicas busca identificar regiões localmente promissoras de dispositivos do SN para atender à determinada SFC. Como cada componente do SN implica em um conjunto de variáveis e restrições na modelagem do problema, reduz-se assim o número de soluções a serem investigadas no espaço de soluções. O intuito é gerar um algoritmo híbrido, combinando vantagens de um modelo de otimização exato com a redução do espaço de busca com clusterização. Neste sentido, a redução do espaço de busca é feita em uma etapa anterior a aplicação do algoritmo exato de mapeamento, *i.e.*, o algoritmo de mapeamento exato ILP^i irá considerar no processamento das SFCs somente as variáveis aptas a compor a solução do problema.

Inicialmente, no Apêndice F, são apresentados alguns conceitos introdutórios sobre métricas e algoritmos de clusterização, dentre eles, o K-*Means*, o *Hierarchical Clustering* e o *Density-Based Spatial Clustering of Applications with Noise* (*DBSCAN*). Uma vez conhecidos os algoritmos de clusterização, pode-se definir qual será adotado. Tal escolha será aplicada em um algoritmo baseado em aspectos geográficos dos servidores, conceito referido como Clusterização por Localização Espacial, explicado na Seção 6.1. Na Seção

6.2 são reportados alguns experimentos computacionais com tal proposta. Por fim, na Seção 6.3 são realizadas as considerações finais deste Capítulo.

6.1 Clusterização por Localização Espacial

As técnicas de clusterização propõem a categorização de elementos com atributos similares dentro de um mesmo conjunto. O K-*Means* é um algoritmo de clusterização, popular e de fácil implementação, baseado em um modelo de centroides e noções de similaridade (MacQueen, 1967). O algoritmo de clusterização chamado *Hierarchical Clustering* também é experimentado nesta tese. Tal algoritmo é baseado em um modelo de conectividade para gerar uma hierarquia de *clusters* (Driver and Kroeber, 1932). Outro algoritmo explorado é o DBSCAN. Este é baseado em um modelo de densidade que isola diferentes regiões dos dados e atribui cada dado dentro dessas regiões ao mesmo *cluster* (Ester et al., 1996). Os algoritmos supracitados são detalhadamente explicados no Apêndice F.

Na abordagem proposta, o princípio empregado na aplicação da clusterização é o de que a localização geográfica de cada servidor do SN pode ser utilizada como um fator relevante no processo de otimização do VNF-PC. Neste caso, cria-se um SN clusterizado, baseado em técnicas de AM não supervisionado, que determine geograficamente bons subconjuntos de dispositivos do SN a serem utilizados no processamento de cada SFC. Essa ação potencialmente reduziria o número de variáveis e restrições em um modelo matemático exato de otimização, e consequentemente o espaço de soluções e o tempo de processamento. O conceito de localização espacial proposto neste trabalho é caracterizado pela posição física dos servidores existentes no SN, sendo que tal localização pode ser mensurada como coordenadas (x,y) ou ainda em latitude e longitude.

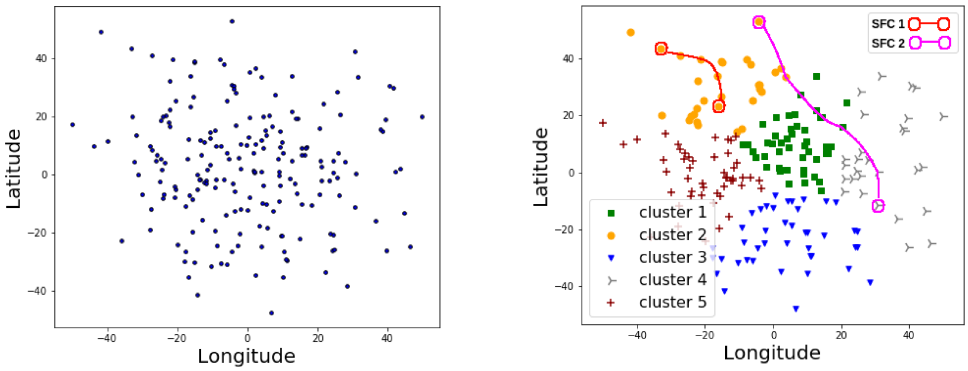
O objetivo da clusterização neste algoritmo é induzir uma política assertiva de padrões de posicionamento de VNFs sobre o SN, e, por conseguinte, conduzir o processo de busca no espaço de soluções. Assim, as SFCs podem ser potencialmente atendidas localmente, beneficiando-se da localização espacial, e reduzindo o número de dispositivos físicos a serem processados. O algoritmo de clusterização proposto agrupa os nós físicos do SN utilizando informações retiradas de uma análise da localização espacial geográfica de cada servidor. Como exemplo, a Figura 6.1a mostra os nós físicos de um SN com suas respectivas localizações geográficas (latitude e longitude) plotadas no plano. A Figura 6.1b mostra o SN da Figura 6.1a clusterizado pelo algoritmo K-*Means*, sendo $k = 5$.

A hipótese investigada neste algoritmo sugere que o SN original (Figura 6.1a) possa ser dividido em k diferentes *clusters* (c_k) de componentes disjuntos, em que $SN = \{c_0 \cup c_1 \cup c_2 \cup ... \cup c_k\}$ (Figura 6.1b). Deste modo, uma SFC com os nós de origem

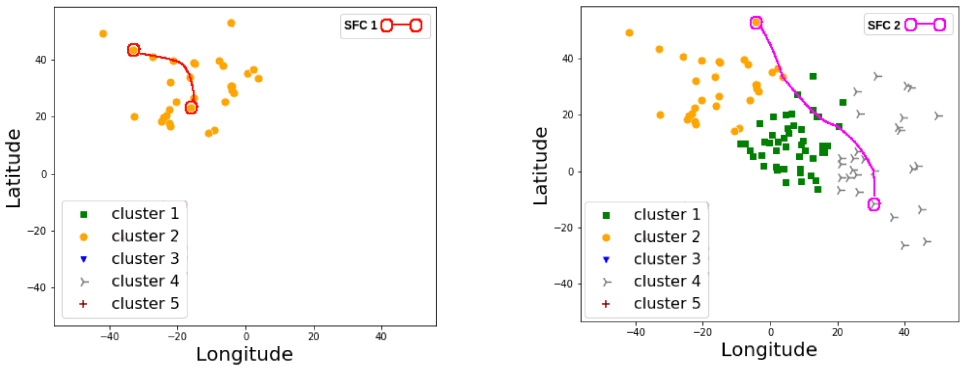
(s^v) e destino (d^v) em um mesmo cluster, como exemplo a SFC1 (Figura 6.1b) pode ser posicionada e encadeada com sucesso em um substrato de rede induzido ($\overline{SN}$) pelos nós físicos pertencentes ao seu respectivo cluster, no caso, o *cluster* amarelo (Figura 6.1c, $\overline{SN} = \{c_2\}$). Outra situação pode acontecer se uma SFC possuir os nós de origem (s^v) em um *cluster* e destino (d^v) em outro, como exemplo a SFC2 (Figura 6.1b). Neste caso, tal SFC pode ser posicionada e encadeada com sucesso em um $\overline{SN}$ induzido pelos nós físicos pertencentes à junção dos dispositivos físicos dos *clusters* presentes no caminho mínimo entre os *endpoints* s^v e d^v, *i.e.*, $\overline{SN} = \{c_1 \cup c_2 \cup c_4\}$ (Figura 6.1d). Um algoritmo de caminho mais curto, para minimizar o número de saltos efetuados (nós intermediários), é usado de modo a definir os *clusters* que serão processados para mapear cada SFC. Como o objetivo é minimizar o número de saltos de cada caminho, o algoritmo BFS é aplicado.

Figura 6.1: Modelo operacional da Abordagem de Clusterização por Localização Espacial

(a) SN completo (b) SN clusterizado (k=5)



(c) $\overline{SN} = c^2$ (d) $\overline{SN} = c^1 \cup c^2 \cup c^4$



Fonte: Elaborado pelo autor.

Em referida abordagem, cada SFC entrante é processada em um $\overline{SN}$ induzido, gerado por uma clusterização. Tal algoritmo recebe como entrada o SN original e o conjunto V_t^a de SFCs ativas, e pode ser descrita em duas etapas:

I Treinamento do modelo de AM (Algoritmo 4): nesta etapa o SN é clusterizado,

dando origem a um vetor de k $\overline{SN}s$ induzidos ($\overline{SN}[k]$, linha 1), em que k representa o número de *clusters* encontrado. Este agrupamento é feito apenas uma vez e usado para todas as SFCs que possam ser demandadas e mapeadas com o Algoritmo 5.

Algoritmo 4 VNF-PC por Localização Espacial (SN, V_t^a)

1: $\overline{SN}[k] \leftarrow$ clusterização(SN)
2: Mapeamento por Localização Espacial $(\overline{SN}[k], V_t^a)$

II Processamento das SFCs (Algoritmo 5), dividido nas seguintes etapas:

a) O algoritmo espera até que uma SFC v entre para ser processada (linhas 1 a 8). Tal estrutura de repetição é responsável por gerenciar a chegada de cada SFC. Em um cenário *online*, pode ser considerado que esta estrutura é executada durante todo o período de processamento das SFCs;

b) Para cada SFC v entrante, o algoritmo define através do algoritmo BFS qual será o $\overline{SN}$ induzido a ser usado no seu processamento (linha 2);

c) Após definir o $\overline{SN}$ induzido a ser usado no processamento da SFC $v \in V$, tal SFC é então processada com as outras SFCs ativas (se $|V_t^a| > 0$, linha 3);

d) Se a SFC $v \in V$ for mapeada, ela será adicionada ao conjunto de SFCs ativas (linha 4) e atualizada a rede residual, ou rejeitada caso contrário (linha 6).

Algoritmo 5 Mapeamento por Localização Espacial $(\overline{SN}[k], V_t^a)$

1: **para** `cada SFC` v `entrante` **faça**
2: $v.\overline{SN} \leftarrow BFS(\overline{SN}[k], s^v, d^v)$
3: **se** $ILP(v, V_t^a) = true$ **então**
4: $V_t^a \leftarrow V_t^a + \{v\}$
5: **senão**
6: rejeita v
7: **fim se**
8: **fim para**

Restrições de integração ao modelo matemático: ao agregar o modelo de clusterização (Seção 6.1) à formulação do problema (Seção 3.1), algumas restrições devem ser adicionadas, são elas:

$$z_{ki}^v = 0, \quad \forall k \in N^v, \forall v \in V, \forall i \in N^c \tag{6.1}$$

$$x_{ij}^{vkl} = 0, \quad \forall (k,l) \in L^v, \forall v \in V, \forall (i,j) \in L^c \tag{6.2}$$

O conjunto de restrições 6.1 garante que cada variável z_{ki}^v referente ao nó físico $i \in N$ que não pertença ao substrato induzido de rede receba obrigatoriamente o valor 0,

não sendo permitido seu uso como variável básica no posicionamento das VNFs $k \in N^v$ de cada SFC $v \in V$ entrante, assim, limitando o espaço de soluções. Neste caso N^c representa o conjunto de nós físicos no conjunto N, mas não pertencem ao *cluster* induzido, *i.e.*, $N^c = \{i | i \in N \wedge i \notin \overline{SN}^k\}$. O conjunto de restrições 6.2 garante que cada variável x_{ki}^{vkl} referente ao arco físico $(i, j) \in L$ que não pertença ao substrato induzido de rede, receba obrigatoriamente o valor 0, não sendo permitido seu uso como variável básica no roteamento entre as VNFs $(k, l) \in L^v$ de cada SFC $v \in V$ entrante, limitando assim o espaço de soluções. Neste caso L^c representa o conjunto de arcos físicos no conjunto L, mas não pertencem ao *cluster* induzido, *i.e.*, $L^c = \{(i, j) | (i, j) \in L \wedge (i, j) \notin \overline{SN}^k\}$.

6.2 Experimentos Computacionais

Os experimentos computacionais exibidos nesta Seção são realizados em um cenário *online* em relação à chegada de SFCs. Dentre os objetivos destes experimentos estão mostrar a eficiência do algoritmo de Mapeamento por Localização Espacial, e definir valores apropriados para a quantidade de *clusters* a serem gerados. Para tal, é usado o algoritmo exato apresentado no Capítulo 3. A importância da adoção do algoritmo exato neste experimento está em uma análise mais precisa dos resultados, visto que não há erros de tomada de decisão e vieses que podem potencialmente ocorrer com heurísticas.

As SFCs, o SN, as métricas, e as demais parametrizações são as mesmas apresentadas nas Seções 4.1 e 5.2. De modo a verificar a generalização das abordagens para os diferentes cenários, são analisados 10 conjuntos diferentes de SFCs do Cenário 4, usando a mesma taxa de chegada do cenário denso apresentado na Seção 5. Dessa forma, mesmo sendo exatas as abordagens utilizadas, os resultados geram um intervalo de confiança. Os experimentos são mostrados com um intervalo de confiança de 95%, e divididos em duas Subseções: na Subseção 6.2.1 são mostrados alguns experimentos e interpretações realizados com os diferentes algoritmos de clusterização; e na Subseção 6.2.3 são apresentados os experimentos explorando as métricas apresentadas na Subseção 4.1.1.

6.2.1 Interpretação dos Modelos de Clusterização

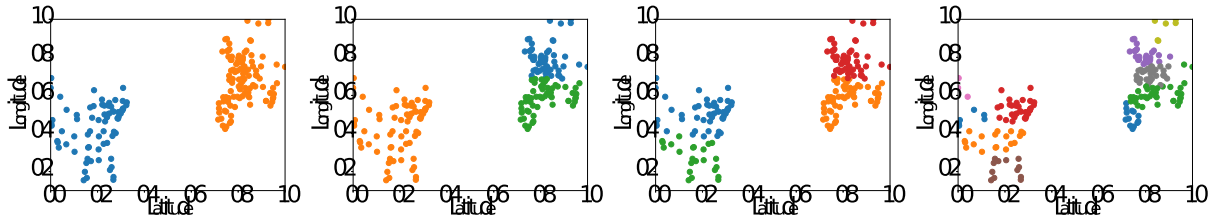
Um dos pontos investigados nesta Seção é avaliar o quão os algoritmos K-*Means* e *Hierarchical Clustering* conseguem agrupar os roteadores do SN de maneira uniforme,

gerando uma clusterização promissora. Neste sentido, os roteadores físicos da topologia *Cogent* são clusterizados com tais abordagens, sendo usadas como características a localização geográfica normalizada de cada nó da rede. Os experimentos realizados com o algoritmo DBSCAN, por serem menos promissores, foram retirados desta Seção, e estão disponíveis no Apêndice G.

K-*Means*: neste algoritmo é necessário informar o valor de k, para tal, são realizados vários experimentos em que o valor de K é variado em $k = 2, 3, ..., 24$. Na aplicação em questão, o valor máximo que k pode assumir é o número máximo de servidores físicos do SN. Logo, no pior caso, pode-se associar cada servidor físico a um *cluster* diferente, *i.e.*, $|N| = k$. Observando a Figura 6.2, percebe-se que, como esperado, em todas as clusterizações realizadas, cada roteador (no físico) é automaticamente atribuído a um cluster. Isso porque o K-*Means* não processa aspectos de ruídos, como o DBSCAN. De modo a melhorar a visualização, em tais figuras são mostrados somente alguns valores de k.

Figura 6.2: Visualização da clusterização com o K-*Means*, variando k

(a) $k = 2$ (b) $k = 3$ (c) $k = 4$ (d) $k = 13$



Fonte: Elaborado pelo autor.

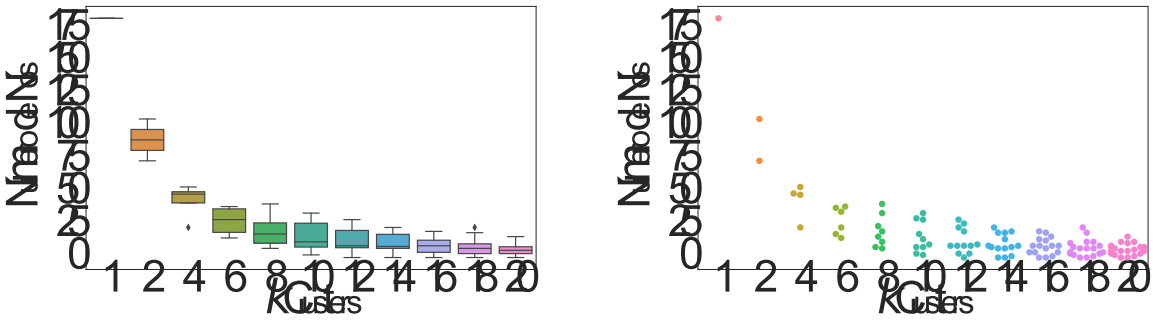
Em uma análise visual da Figura 6.2, percebe-se que para todos os valores de k experimentados, o algoritmo K-*Means* gerou *clusters* regulares e balanceados em números de componentes. Comportamento que em termos da aplicação é benéfico, pois, uma geração de *clusters* desbalanceados pode não reduzir o tempo de execução em alguns casos, ou um *cluster* pode não ser abrangente o suficiente para mapear uma SFC. Complementarmente à tal análise, no Apêndice G, as Figuras G.1 e G.2 apresentam o *Elbow Method* e o *Silhouette Coefficient* resultante das clusterizações apresentadas na Figura 6.2, onde foi constatado que bons valores adotados são $k = 2$ e $k = 3$.

Mesmo com as técnicas *Elbow Method* e *Silhouette Coefficient* indicando que bons valores de k são 2 e 3, deve-se considerar qual impacto que essas escolhas causam na redução de componentes a serem processados no mapeamento de cada SFC. Nesse sentido, a Figura 6.3 indica o quão relacionado está o valor de k, com a redução do número de componentes de cada cluster. Processando todos os componentes do SN juntos, existem 186 nós físicos a serem usados no mapeamento de uma nova SFC ($k = 1$, Figura 6.3b). Adotando $k = 2$, formam-se dois grupos de nós levemente desequilibrados, um com 77 e outro com 109 (Figuras 6.2a e 6.3b). Para $k = 4$ nota-se uma geração de mais *clusters* em desequilíbrio, sendo cada *cluster* com 26, 52, 57 e 51 nós respectivamente (Figuras

6.2c e 6.3b). Esse desequilíbrio gera uma forte alteração da medida de centralidade do respectivo *box plot* (Figura 6.3a), e pode ser prejudicial, pois um *cluster* grande pode não reduzir o tempo de execução e um *cluster* pequeno pode não ser amplo o bastante para mapear uma SFC. Analisando o retângulo de cada *box plot* da Figura 6.3a, infere-se que o ideal é que eles sejam regulares em relação à composição, e achatados, conforme percebidos para $k = 16$.

Figura 6.3: Variação no número de nós dos *clusters* com o algoritmo K-*Means*

(a) Box plot (b) Swarm plot



Fonte: Elaborado pelo autor.

Percebe-se que o número de componentes em cada *cluster* diminui rapidamente quando se aumenta o valor de k (Figura 6.3). Porém, quando um valor acima de $k = 18$ é usado, os *clusters* gerados começam a ser muito pequenos em termos do número de componentes, restringindo significativamente a quantidade de variáveis do modelo matemático, e, consequentemente, o espaço de solução. Neste caso, boas soluções provavelmente não seriam geradas. Infere-se que a clusterização gerada pelo K-*Means* é promissora, mas com os experimentos realizados ainda é difícil definir qual o melhor valor de k a ser utilizado. Assim, são propostos nas próximas seções outros experimentos para analisar o impacto desta redução de componentes pela aplicação em questão.

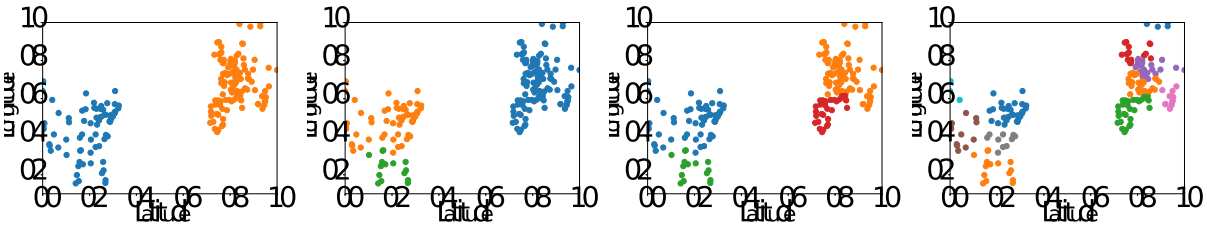
***Hierarchical Clustering*:** referido algoritmo não exige inicialmente que se informe previamente o número de *clusters* a serem gerados. Tal etapa pode ser efetuada através da análise visual de um dendrograma. O dendrograma gerado na clusterização da posição geográfica dos roteadores da topologia *Cogent* é mostrado no Apêndice G (Figura G.3), no qual é possível inferir que um bom valor é $k = 2$.

Comparando visualmente os *clusters* gerados com o K-*Means* e o *Hierarchical Clustering* (Figuras 6.2 e 6.4), percebe-se que ambas as abordagens, tendem a gerar *clusters* diferentes. Como exemplo, considerando $k = 3$, a clusterização com K-*Means* gerou *clusters* com um número de componente mais equilibrado que o algoritmo *Hierarchical Clustering* (Figuras 6.2b e 6.4b). E o mesmo comportamento se repete para outros valores de k. Essa situação pode ser prejudicial no desempenho da abordagem, pois um

cluster grande pode não reduzir significativamente o tempo de execução. Percebe-se que o algoritmo *Hierarchical Clustering* tende a não gerar *clusters* muito regulares.

Figura 6.4: Visualização da clusterização com o *Hierarchical Clustering*, variando k

(a) $k = 2$ (b) $k = 3$ (c) $k = 4$ (d) $k = 13$

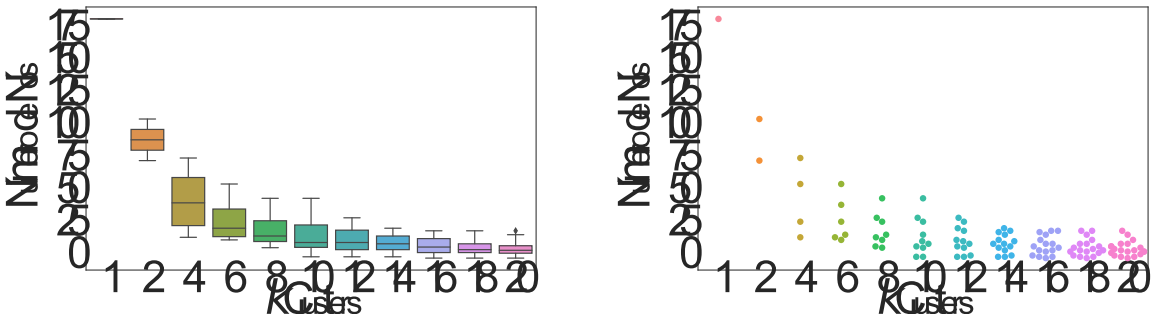


Fonte: Elaborado pelo autor.

Mesmo com a análise visual dos *clusters* gerados, deve-se considerar qual impacto que o valor de k causa na redução de componentes a serem processados no mapeamento das SFCs. Nesse sentido, a Figura 6.5 indica o quão relacionado está o valor de k, com a redução do número de componentes de cada cluster.

Figura 6.5: Variação no número de nós dos *clusters* com o *Hierarchical Clustering*

(a) Box plot (b) Swarm plot



Fonte: Elaborado pelo autor.

Diferentemente dos experimentos realizados com o algoritmo K-*Means*, a dispersão dos componentes de cada *cluster* tende a ser maior com o algoritmo *Hierarchical Clustering*. Como exemplo, para $k = 4$, o K-*Means* gera os *clusters* com 26, 52, 57 e 51 nós respectivamente (Figura 6.3a). Por outro lado, com o algoritmo *Hierarchical Clustering*, para $k = 4$, geram-se os *clusters* com 59, 79, 18 e 30 nós respectivamente, gerando uma maior dispersão em seu respectivo *box plot* (Figura 6.5a).

Com os experimentos realizados ainda é difícil definir qual o melhor valor de k a ser utilizado, e se os *clusters* gerados são melhores ou piores, para a aplicação, em relação aos gerados com o algoritmo K-*Means*. Assim, são propostos outros experimentos para analisar o impacto desta redução de componentes a serem processados pela abordagem em questão, fazendo uma comparação com os algoritmos K-*Means* e *Hierarchical Clustering*.

6.2.2 Algoritmos Avaliados

Devido à inviabilidade computacional de reotimizar todas as restrições de todas as SFCs ativas, percebida nos experimentos da Subseção 4.2, a variação do algoritmo ILP utilizado realiza o mapeamento de uma nova SFC com base nos recursos residuais do SN no instante t, e reotimiza as restrições de *placement* relativas às instâncias de VNFs ativas. Variam-se também os valores de k, em que:

- ILP - algoritmo exato, sem redução do espaço de busca, no qual reotimizam-se as restrições de dimensionamento de instâncias de VNFs (Capítulo 4). Usado como *baseline*;
- ILPsk - utiliza o $\overline{SN}$ residual induzido pela Clusterização por Localização Espacial para efetuar o processamento de cada SFC. Neste caso, o espaço de busca é reduzido pela clusterização. Esta abordagem pode ser executada com o algoritmo K-*Means* (KM), ou com o *Hierarchical Clustering* (HC). Como exemplo, o algoritmo ILPs12KM antes de mapear uma SFC utiliza a técnica de Clusterização Espacial do SN em 12 *clusters* gerados com o algoritmo K-*Means*.

6.2.3 Análise das Estratégias de Clusterização

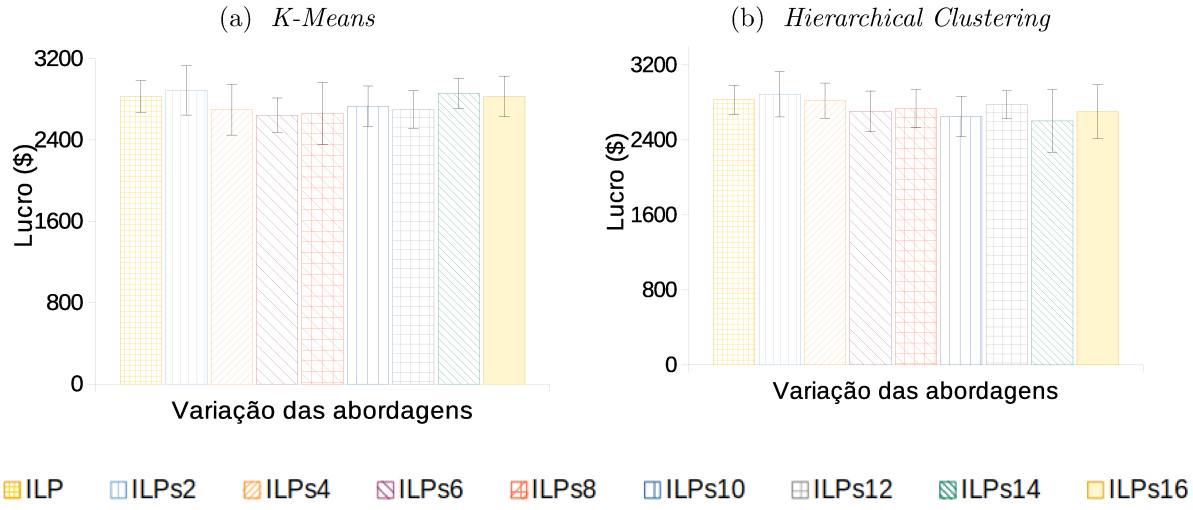
Com base na interpretação inicial dos modelos de clusterização apresentada anteriormente, definem-se os valores de k para serem investigados como $k = \{1, 2, 4, ..., 14, 16\}$. Da Figura 6.6 até a Figura 6.8 são mostrados em colunas o acumulado final do processamento de todas as SFCs no cenário avaliado, sendo no eixo x variado o algoritmo de clusterização, e no eixo y mostrados os valores das métricas em questão. Das Figuras 6.9 a 6.11, os resultados são mostrados em janelas de tempo que correspondem à entrada da primeira até a última SFC do conjunto de dados. Em tais Figuras são mostradas as métricas descritas na Subseção 4.1.1 para uma única execução dos algoritmos em uma instância escolhida aleatoriamente. Nestes casos, o eixo x apresenta o tempo de simulação em unidades de tempo t e o eixo y a métrica observada. Para cada um dos gráficos gerados, mostrados nas Figuras 6.9 a 6.11, é gerada uma tabela que resume o comportamento de cada algoritmo (Apêndice H).

Análise do lucro (Figura 6.6): neste caso, percebe-se uma oscilação em relação às médias aferidas nos experimentos, mas com base no intervalo de confiança, pode-se afirmar com 95% de confiança que essa variação não é estatisticamente significativa

para ambas as abordagens de clusterização. Observa-se que ambas as abordagens de agrupamento, embora não afetem estatisticamente o lucro dos fornecedores, também não geram perdas financeiras em relação ao algoritmo ILP.

O algoritmo ILPsk , por realizar os mapeamentos em subconjuntos de dispositivos do SN original, processa regiões disjuntas do SN para o mapeamento de cada SFC. Essa ação, implicitamente, promove um balanceamento de carga das SFCs sobre o SN, causando o efeito de roteamentos com menor número de saltos (Tabela H.4), atraso fim a fim menores (Tabela H.5), mas, como percebido, um menor compartilhamento de servidores (Tabela H.3). Outras análises sobre o lucro podem ser vistas no Apêndice H.

Figura 6.6: Lucro final dos provedores



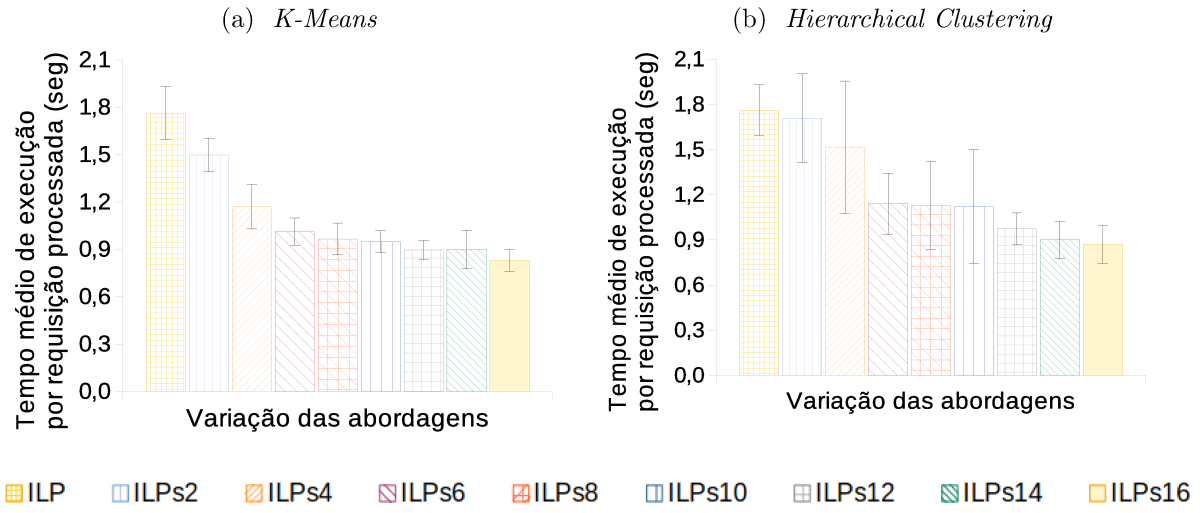
Fonte: Elaborado pelo autor.

Análise do tempo de execução total (Figura 6.7): como premissa, se por um lado o número de componentes em um *cluster* for relativamente pequeno (alta quantidade de *clusters*), uma SFC específica pode ter um mapeamento ruim, devido às reduzidas opções de mapeamento, e menor espaço para otimização. Por outro lado, se o número de componentes em um *cluster* for relativamente grande (baixo número de *clusters*), pode não haver melhorias no tempo de execução, o processo de otimização ser lento, mas podendo gerar soluções de maior qualidade.

Conforme observado na Figura 6.7, em ambas as abordagens de clusterização (k-*Means* e *Hierarchical Clustering*) houve uma diferença significativa no tempo de execução em relação ao modelo de otimização tradicional, em alguns casos sendo até de $\approx 112\%$ (ILP com $\approx 1.76ms$ e ILPs16Km com $\approx 0.83ms$, Figura 6.7). Tal redução no tempo de execução está diretamente relacionada ao tamanho dos *clusters* que formam os SNs induzidos. Comparando as Figuras 6.3, 6.5 e 6.7, nota-se que ao aumentar o valor de k o número médio de componentes dos $\overline{SNs}$ induzidos é menor, conforme mostrado no segundo quartil de cada *box plot* apresentado nas Figuras 6.3a e 6.5a. Fator que, como

esperado, afeta diretamente o tempo de execução, confirmando a premissa de que uma redução do número de componentes processados, reduz o espaço de soluções a ser investigado, e consequentemente, reduz o tempo de execução.

Figura 6.7: Tempo médio por SFC



Fonte: Elaborado pelo autor.

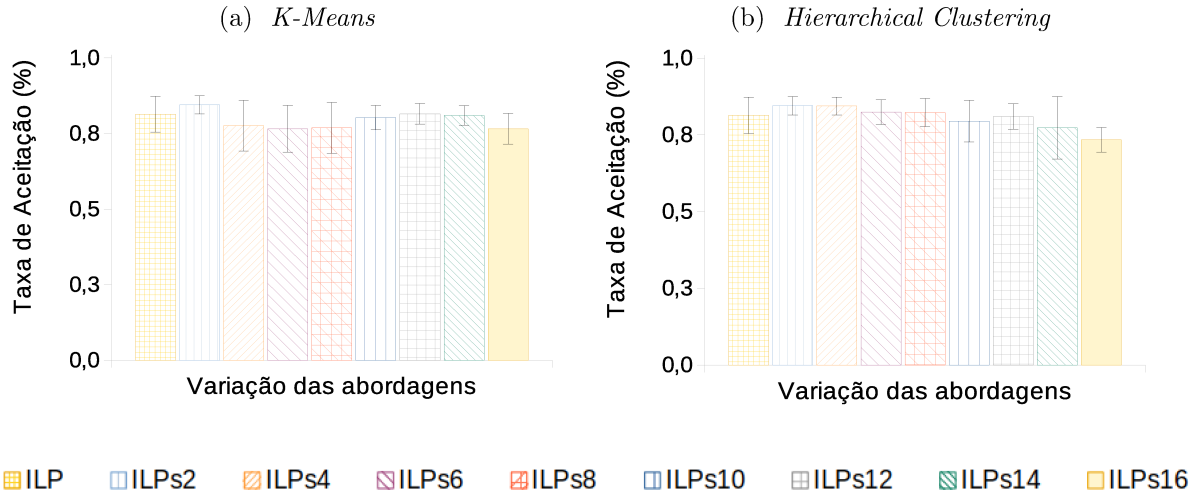
O intervalo de confiança presente no tempo de processamento médio com o algoritmo *Hierarchical Clustering* (Figura 6.7), se mostrou esparsado e irregular, comportamento atribuído à geração de *clusters* disformes em número de componentes. Conforme percebido nas Figuras 6.3 e 6.5, os *clusters* gerados pelo algoritmo *Hierarchical Clustering* possuem uma dispersão na clusterização de componentes mais alta que o gerado com o algoritmo K-*Means*. Neste ponto, o algoritmo K-*Means* se mostra mais promissor de ser empregado, justamente por gerar *clusters* mais homogêneos em quantidade de elementos, fator que refletiu positivamente no tempo de processamento do respectivo algoritmo.

Análise da taxa de aceitação (Figura 6.8): em concordância com os experimentos mostrados na Figura 5.7d, todas as abordagens propostas obtiveram uma taxa de aceitação constante e alta, mas um pouco distante do 100% de SFCs aceitas. Acontecimento devido à taxa de chegada de SFCs ser elevada neste cenário (explicado nos experimentos da Seção 5). Percebe-se que o desempenho do algoritmo ILP*sk* é positivo, pois reduz notadamente o tempo de execução, sem gerar perdas estatisticamente relevantes nos lucros, nem reduzir a taxa de aceitação.

Conforme pode ser observado nas Tabelas H.1 e H.2, presentes no Apêndice H, percebe-se que, similarmente aos experimentos dos Capítulos 4 e 5, ambas as abordagens propostas não priorizam o mapeamento de nenhuma classe em detrimento a outra, independentemente do valor de k adotado. Esse fator é positivo e deve ser ressaltado, pois, a taxa de aceitação é mantida estável de maneira igualitária. Não aumentando o mapeamento de SFCs mais fáceis (classe 2, por exemplo) e diminuindo o de SFCs mais

complexas (classe 3, por exemplo).

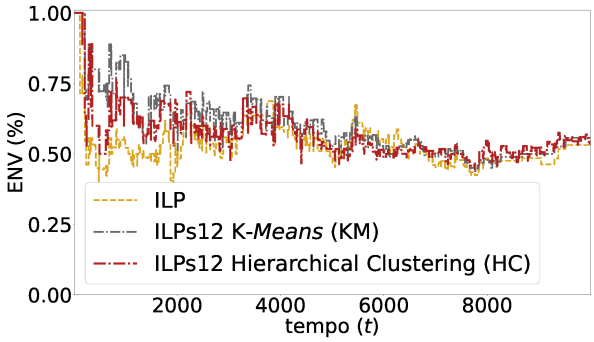
Figura 6.8: Taxa de aceitação final



Fonte: Elaborado pelo autor.

Análise do espalhamento dos nós virtuais (Figura 6.9): com o intuito de facilitar a leitura dos gráficos mostrados na sequência desta Seção, exportam-se somente os resultados dos algoritmos: ILP, ILPs12KM e ILPs12HC. Elegem-se tais abordagens como promissoras por reduzirem significativamente o tempo de execução, e não afetarem estatisticamente a taxa de aceitação e lucro dos provedores. O próximo gráfico mostra o quão centralizado ou distribuído é o espalhamento médio das VNFs demandadas sobre os nós físicos do SN. Dessa forma, o objetivo é minimizar essa métrica, no caso, valores próximos a 1 mostram um baixo compartilhamento de nós físicos.

Figura 6.9: Espalhamento dos nós virtuais



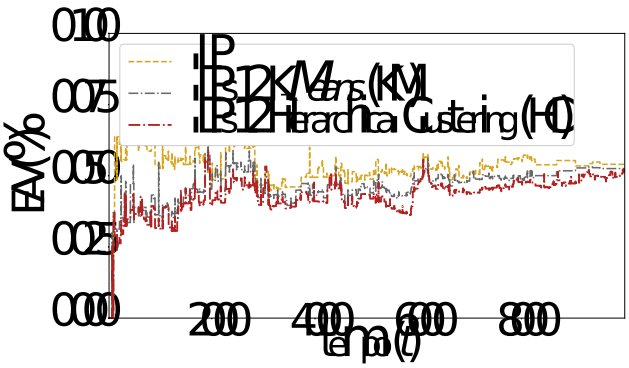
Fonte: Elaborado pelo autor.

No experimento, o algoritmo ILP gerou um espalhamento de nós virtuais baixo, de $\approx 53.58\%$, *i.e.*, na média, cada servidor físico ativo está sendo usado por duas SFCs diferentes (Tabela H.3). Similarmente aos experimentos já analisados no Capítulo 5, dentre outros fatores, este comportamento ocorre devido a um intenso fluxo de chegadas

e saídas de SFCs. Neste caso, as abordagens demandam a ativação de mais servidores (nós físicos) do SN para atender às demandas das SFCs entrantes e não gerar rejeições. Com mais nós físicos ativos, maior a probabilidade de ocorrerem compartilhamentos de recursos. Repare que, em relação ao espalhamento dos nós virtuais, o comportamento foi próximo entre todos os algoritmos analisados, sendo o algoritmo ILP ligeiramente melhor. Outros comentários sobre estes experimentos podem ser vistos no Apêndice H.

Análise do espalhamento dos arcos virtuais (Figura 6.10): nesta métrica, percebe-se que o algoritmo ILP*sk* induz um roteamento de arcos mais conciso (menos saltos). Como exemplo, a Figura 6.10 reporta que o mapeamento das requisições ocupa na média ≈ 5.31% dos arcos físicos do SN com o algoritmo ILP, enquanto no algoritmo ILPs12KM este valor é de ≈ 4.33%, e no algoritmo ILPs12HC este valor é de ≈ 4.66% (Tabela H.4). Assim, referido algoritmo utiliza até ≈ 22.57% menos de arcos físicos do SN em relação ao algoritmo tradicional (Tabela H.4). Este fato ocorre devido ao algoritmo baseado no modelo de otimização tradicional possuir uma visão completa do espaço de busca, maximizando o compartilhamento de servidores para reduzir os custos de operação, *i.e.,* o mapeamento com uma visão completa do SN centraliza/concentra o consumo de recursos em alguns servidores devido aos custos operacionais de arcos serem menores que os custos dos servidores físicos. Outros comentários sobre estes experimentos podem ser vistos no Apêndice H.

Figura 6.10: Espalhamento dos arcos virtuais



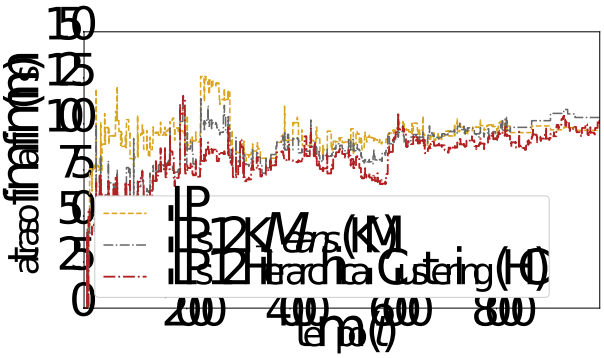
Fonte: Elaborado pelo autor.

Análise do atraso médio fim a fim (Figura 6.11): conforme a Equação 3.11, quanto mais arcos físicos forem utilizados no mapeamento, maior será o atraso fim a fim gerado em cada SFC. Nos experimentos realizados, semelhantemente ao espalhamento de arcos (Figura 6.10), o algoritmo baseado no modelo de otimização tradicional além de realizar um mapeamento de arcos mais espalhado sobre os arcos físicos do SN, gera um atraso fim-a-fim mais alto para o usuário final.

O algoritmo ILP*sk* com o K-*Means* gera um espalhamento de arcos mais conciso. Esse comportamento é previsto devido à natureza da proposta de redução dos componentes processados do SN, e ao K-*Means* gerar *clusters* mais balanceados em relação ao

Hierarchical Clustering. Dessa forma é improvável que uma SFC seja mapeada de forma espalhada (estendida) sobre os arcos físicos. A mesma argumentação é verificada para o algoritmo ILP tradicional, mas com um efeito oposto. Percebe-se que, tanto para o espalhamento de arcos, quanto na geração do atraso fim a fim, o algoritmo baseado no clusterizador K-*Means*, por gerar agrupamentos mais uniformes (Figuras 6.3 e 6.5), obteve um comportamento melhor, com um menor atraso fim a fim, que o algoritmo que utiliza o *Hierarchical Clustering*. Outras informações podem ser verificadas no Apêndice H.

Figura 6.11: Atraso fim a fim



Fonte: Elaborado pelo autor.

6.3 Considerações Finais

Neste Capítulo foi apesentada uma abordagem baseada em agrupamentos, chamada Clusterização por Localização Espacial. Tal abordagem, integradamente ao modelo matemático, resolve o problema VNF-PC em um ambiente *online* e com um tempo computacional reduzido em relação ao algoritmo exato tradicional.

Experimentos computacionais foram realizados para verificar o desempenho da abordagem proposta e definir o algoritmo de clusterização mais indicado. O algoritmo DBSCAN não se mostrou promissor e os experimentos com ele foram descontinuados. Outros experimentos foram realizados para definir bons valores para o parâmetro k presente no algoritmo de clusterização. Por fim, foram realizados experimentos para comparar os ganhos e perdas dos algoritmos de clusterização propostos em relação ao modelo de otimização exato. Os resultados validam a hipótese de que, para mapear determinadas SFCs, alguns componentes do SN são subutilizados e não precisam ser processados.

Observa-se em todos os cenários avaliados uma diferença significativa de até $\approx$ 112% do tempo de execução das abordagens de clusterização em comparação com o modelo de otimização tradicional. Essa redução ocorreu por existirem menos restrições e

variáveis para serem processadas, ação que reduz o espaço de busca processado. Outro ponto é que, apesar de reduzir o tempo computacional, o algoritmo proposto não reduziu estatisticamente o lucro e a taxa de aceitação dos provedores em relação ao algoritmo tradicional. Uma vez validada a abordagem de redução de dimensionalidade junto ao algoritmo exato, propõe-se no próximo Capítulo amadurecer o conceito aplicado. Infere-se que, explorando conceitos de IA, e de um histórico de SFCs mapeadas no passado, seja possível gerar agrupamentos de recursos para mapear novas SFCs que sejam demandadas no futuro.

Capítulo 7

Redução do espaço de solução: Clusterização por Consumo de Recursos e Retreinamento

Neste Capítulo é proposta uma abordagem aprimorada em relação à abordagem de Clusterização por Localização Espacial apresentada no Capítulo 6. Tal aprimoramento é concebido com base nos resultados promissores percebidos por referida abordagem, agregado de conceitos particulares do algoritmo K-*Means*, e um histórico de mapeamentos, tratamento chamado de Clusterização por Consumo de Recursos. Ao se empregar conceitos de histórico de consumo e treinamento, depara-se com as questões: quando e qual a importância de atualizar os grupos? Neste Capítulo, adota-se o retreinamento orientado por métricas, no caso, os dados processados são monitorados, e caso as demandas se desviem significativamente das anteriores, tal etapa deve ser realizada (Najafzadeh et al., 2021). Para definir estes momentos, optou-se por utilizar a métrica de entropia relativa, oriunda da teoria da informação, chamada divergência de Kullback and Leibler (1951) (KL). Tal modificação adiciona adaptabilidade de aplicação a cenários dinâmicos quanto a chegada de SFCs, características inerentes a ambientes de redes.

Neste Capítulo, a abordagem proposta é definida na Seção 7.1. Na Seção 7.2 são feitos alguns experimentos computacionais preliminares. Na Seção 7.3 são exploradas mudanças que podem acontecer nas demandas, e como o retreinamento dos modelos foi conduzido. Na Seção 7.4 são feitos experimentos computacionais que abordam mudanças nas demandas das SFCs. E, por fim, na Seção 7.5 são realizadas as considerações finais.

7.1 Clusterização por Consumo de Recursos

A abordagem de Clusterização por Consumo de Recursos se fundamenta nos mesmos princípios de redução de dimensionalidade do SN (Capítulo 6), e utiliza dos conceitos

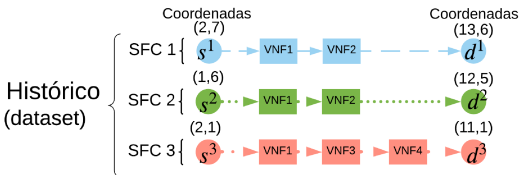
de centroides e similaridades presentes no algoritmo K-*Means* (Apêndice F).Contudo, diferentemente da abordagem de Clusterização por Localização Espacial, propõe-se um aprimoramento na geração dos agrupamentos, sendo agora fundamentada em um histórico otimizado de mapeamentos de SFCs. Tal histórico é referido como conjunto H, constituído de inúmeras SFCs $v \in V \subset H$ mapeadas com sucesso em um momento passado.

Abordagem Proposta: o princípio empregado é o de que SFCs com características similares possam ser mapeadas em recursos físicos similares do SN. Desta forma, por conceitos de similaridade, se em um momento passado uma SFC $\mathbf{x}$ é mapeada sobre os recursos $\mathbf{r}$ de um substrato de rede residual $\mathbf{s}$, em um momento futuro uma nova SFC $\mathbf{y}$, que seja semelhante a $\mathbf{x}$, pode potencialmente também ser mapeada com sucesso sobre os mesmos recursos $\mathbf{r}$, desde que o estado do substrato de rede residual seja similar a $\mathbf{s}$.

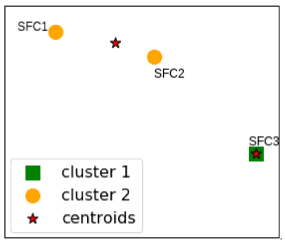
Com base no histórico H de mapeamentos, pode-se criar um modelo de clusterização, que determine bons $\overline{SNs}$ induzidos de dispositivos do SN original para serem utilizados no processamento de uma nova SFC ($\overline{SN} \subset SN$). Essa ação potencialmente reduziria o número de variáveis e restrições a serem processadas pelo algoritmo, e consequentemente o tempo de processamento. Como exemplo, a Figura 7.1a mostra 3 SFCs pertencentes ao histórico H, com seus respectivos pontos de origem (s^v) e destino (d^v) referenciados com coordenadas geográficas (x, y). A Figura 7.1b mostra as 3 SFCs do histórico H clusterizadas pelo algoritmo K-*Means* com $k = 2$.

Figura 7.1: Modelo operacional da Abordagem de Clusterização por Consumo de Recursos

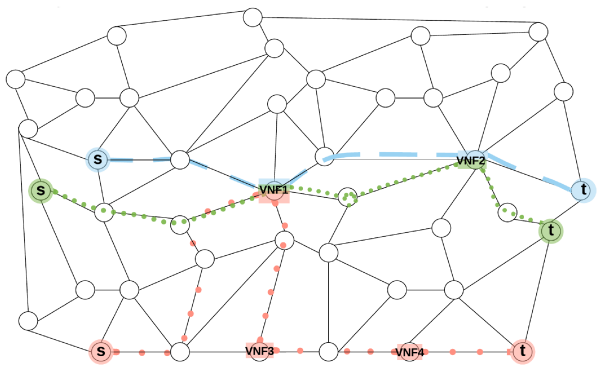
(a) SFCs $v \in V \subset H$



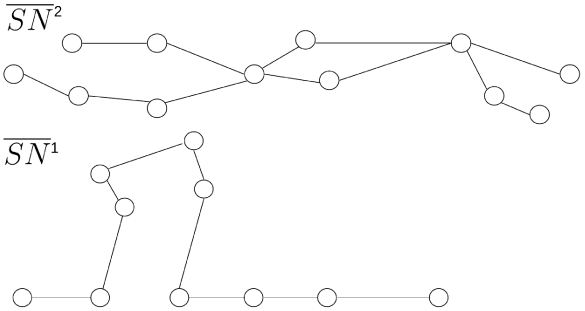
(b) Histórico clusterizado



(c) Mapeamento das SFCs sobre o SN no passado



(d) Exemplo de $\overline{SN}$



Fonte: Elaborado pelo autor.

Neste exemplo de clusterização são usadas como características as coordenadas

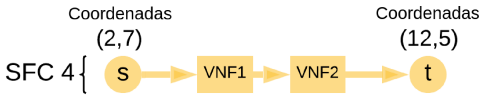
geográficas dos pontos de origem e destino, além das VNFs demandadas por cada SFC. Apesar de a dificuldade de visualizar todas as características do exemplo no espaço acima de $\mathbb{R}^2$, ilustrativamente, imagine que esta clusterização possa ser visualizada no plano, como mostrado na Figura 7.1b. Repare que, na Figura 7.1b, as SFCs 1 e 2 por serem similares formam um cluster, ficando a SFC 3 separada em outro.

Uma vez clusterizado o histórico H (Figura 7.1b), podem então ser sumarizados os recursos físicos demandados no passado pelas próprias SFCs que pertencem ao mesmo *cluster* (Figura 7.1c). Neste caso específico, como as SFCs 1 e 2 estão no mesmo cluster, os recursos físicos utilizados por ambas no passado (Figura 7.1c), são agrupados, formando um SN induzido de dispositivos ($\overline{SN}^2$), como mostrado na Figura 7.1d. Por outro lado, como a SFC 3 é única em seu cluster, os recursos físicos utilizados por ela no passado, mostrado na Figura 7.1c, são agrupados formando outro SN induzido de dispositivos ($\overline{SN}^1$), como mostrado na Figura 7.1d.

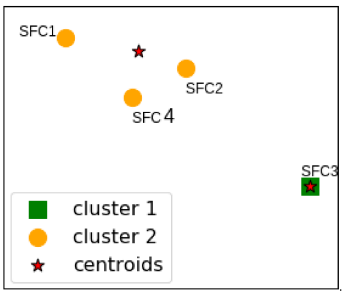
Após a clusterização do histórico (Figura 7.1b), e definidos os k $\overline{SNs}$ induzidos (Figura 7.1d), podem-se inferir os recursos a serem utilizados no processamento de uma nova SFC. Suponha que o mapeamento de uma nova SFC é demandado (SFC 4, Figura 7.2a), seguindo o algoritmo de Clusterização por Consumo de Recursos deve-se: definir à qual *cluster* a nova SFC pertence, para em seguida processa-lá no $\overline{SN}$ induzido pelo seu respectivo cluster. Maximizando a similaridade entre a nova SFC (SFC4, Figura 7.2a) e os *clusters* já computados, atribui-se a SFC4 ao *cluster* 2, (Figura 7.2b), e, neste caso, a SFC4 pode então ser processada no seu respectivo substrato de rede induzido ($\overline{SN}^2$)(Figura 7.2c).

Figura 7.2: Exemplo do mapeamento de uma nova SFC

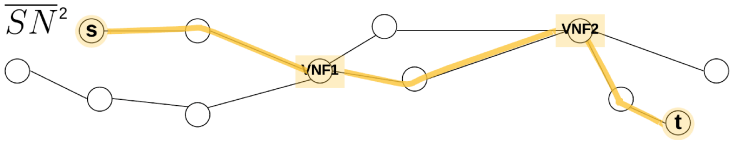
(a) SFC a ser mapeada



(b) Cluster



(c) $\overline{SN}^2$ induzido com o mapeamento da nova SFC



Fonte: Elaborado pelo autor.

O princípio fundamental desta estratégia é que cada *cluster* gere um SN induzido que consiga hospedar uma determinada SFC por completo. Neste sentido, a hipótese a ser investigada é: quanto mais diverso o histórico H de SFCs for, maior a possibilidade de o algoritmo generalizar corretamente para novos mapeamentos. Caso o histórico H não seja suficientemente grande, uma nova SFC pode ser agrupada incorretamente em uma região infactível, e rejeitada por falta de recursos ou incompatibilidades com a região de mapeamento. Outro ponto questionado é: quão grande este histórico de mapeamento deve ser. A escolha de trabalhar com o histórico H otimizado agrega valor à abordagem no sentido de gerar bons $\overline{SNs}$ induzidos para o mapeamento de uma nova SFC. De outro modo, a utilização de um histórico H sem garantias de boas soluções, poderia induzir à replicação de mapeamentos distantes do ótimo e/ou ruins. As demais explicações sobre o conjunto de SFCs $v \in V \subset H$, suas respectivas características, e normalizações são detalhadas na Subseção 7.2.2.

Neste trabalho é assumido que o provedor deva gerar em uma atividade de simulação paralela ao algoritmo de mapeamento, um histórico otimizado de SFCs mapeadas. Outra opção é, caso o provedor já possua tal histórico, ele pode ser utilizado como entrada para o algoritmo de clusterização proposto. O histórico usado neste trabalho é dinâmico, *i.e.*, são adicionados novos mapeamentos a ele. Dessa forma, se em algum momento a topologia do SN mudar, ou uma nova demanda por serviços de rede surgir, basta retreinar por simulações o algoritmo de clusterização para o modelo se adequar a essas novas demandas. Referido retreinamento pode ser executado em uma atividade paralela ao algoritmo de mapeamento, não afetando o processamento das SFCs entrantes. Nesta Seção, pretende-se inicialmente validar o método de clusterização proposto, e em seguida, analisar os efeitos da realimentação do histórico e o retreinamento do modelo de AM.

Algoritmicamente, tal abordagem recebe como entrada o SN original, o histórico H e um conjunto V_t^a de SFCs ativas; e funciona em etapas, descritas como:

I Algoritmo 6, treinamento do modelo de AM:

Algoritmo 6 VNF-PC por Consumo de Recursos $(SN, H, k, V_t^a, centroids[k])$

1: $V_t^a \leftarrow \varnothing$
2: $\overline{SN}[k] \leftarrow \varnothing$
3: $centroids[k] \leftarrow K\text{-}means(H, k)$
4: **para** `cada SFC` $v \in H$ **faça**
5: $\quad v.cluster \leftarrow minimizaDistanciaEuclidiana(v, centroids[k])$
6: $\quad \overline{SN}[v.cluster] \leftarrow \overline{SN}[v.cluster] + recursosUsados(v)$
7: **fim para**
8: Mapeamento por Consumo de Recursos $(\overline{SN}[k], V_t^a)$

a) O histórico de mapeamentos H é agrupado pelo algoritmo K-*Means* para gerar um vetor com k *clusters*, no qual cada *cluster* possui seu respectivo centroide

(linha 3). Esse agrupamento é feito uma vez e mantido em todas as etapas subsequentes, podendo ser atualizado caso necessário (retreinamento);

b) Cada SFC $v \in H$ é atribuída a um cluster, ação determinada pela minimização da distância euclidiana, calculada usando um conjunto de características pertencentes a VNF $v \in H$ e cada elemento do vetor $centroids[k]$ (linha 5);

c) Com base nos recursos usados no passado por cada SFC $v \in H$, um vetor de $\overline{SNs}$ induzidos é gerado ($\overline{SN}[k]$, linha 6). Esta geração de $\overline{SNs}$ induzidos é feita uma vez, atualizada em caso de retreinamento, e utilizada no Algoritmo 7.

II Algoritmo 7, processamento das SFCs:

Algoritmo 7 Mapeamento por Consumo de Recursos ($\overline{SN}[k], centroids[k], V_t^a$)

1: **para** `cada SFC` v `entrante` **faça**
2: $cluster \leftarrow minimizaDistanciaEuclidiana(v.features, centroids[k])$
3: $v.\overline{SN} \leftarrow \overline{SN}[cluster]$
4: **se** $ILP(v, V_t^a) = true$ **então**
5: $V_t^a \leftarrow V_t^a + \{v\}$
6: **senão**
7: rejeita v
8: **fim se**
9: **fim para**

a) O algoritmo espera até que uma SFC entre para ser processada (linhas 1 a 9). Tal estrutura de repetição é responsável por gerenciar a chegada e o processamento de cada SFC. Em um cenário *online*, pode ser considerado que este laço é executado durante todo o período de mapeamento das SFCs;

b) Para cada SFC v que chega para ser mapeada, o $\overline{SN}$ induzido a ser usado no processamento de tal SFC (linha 2) é definido através da minimização da distância euclidiana;

c) Após definir o $\overline{SN}$ induzido a ser usado no mapeamento da SFC $v \in V$, tal SFC é então processada com as outras SFCs ativas (se $|V_t^a| > 0$, linha 4);

d) Se a SFC $v \in V$ for mapeada com sucesso, ela será adicionada ao conjunto de SFCs ativas (linha 5), ou rejeitada caso contrário (linha 7).

Como mostrado na linha 3 do Algoritmo 7, um dos principais pontos deste algoritmo é a atribuição de uma nova SFC a um *cluster* já calculado. O algoritmo K-*Means*, possui uma maneira natural de realizar tal passo, que consiste em realizar a etapa de atribuição de um novo dado, sem atualizar os *clusters*, e isto é feito justamente minimizando a distância adotada (Equação F.1). Essa classificação não é possível nos algoritmos DBSCAN e *Hierarchical Clustering*, pois eles geram os *clusters* com outras métricas, e não

possuem o princípio de centroides para a classificação de um novo dado. O funcionamento dos algoritmos *Hierarchical Clustering* e do DBSCAN pode ser visto no Apêndice F.

No *Hierarchical Clustering* é minimizada a variância dos *clusters* que estão sendo aninhados. Neste caso, o objetivo é o menor incremento na variância total em torno da média. O *Hierarchical Clustering* constrói iterativamente o *cluster* começando a partir dos dados existentes, e os aninha de acordo com uma métrica. Assim, a adição de um novo dado a ser classificado (SFC a ser mapeada) implica em treinar o modelo desde o início. Ação inviável no cenário adotado, devido à complexidade cúbica do algoritmo, ao tamanho do conjunto de dados a ser clusterizado, e por se tratar de um ambiente *online*, que exige uma resposta rápida para se efetuar o mapeamento. Outro ponto que inviabiliza a aplicação do *Hierarchical Clustering* nesta abordagem é sua complexidade de espaço. O *Hierarchical Clustering* não pode ser escalado para grandes volumes de dados. Neste caso, devido à complexidade de espaço ser da ordem de $\Omega(n^2)$, em experimentos preliminares com um *Dataset* contendo 40.000 SFCs, o gasto de memória é da ordem de $1.6\text{x}10^9$, em termos práticos, em $\approx$ 10 segundos de treinamento do modelo, houve um consumo de mais de 16GB de memória RAM.

No algoritmo DBSCAN a adição de um novo dado a ser classificado implica em treinar o modelo desde o início, pois este novo dado pode modificar estruturalmente os *clusters* já formados. Neste algoritmo, dois pontos pertencem ao mesmo *cluster* se a distância mínima entre os dados dos *clusters* estiver abaixo do parâmetro EPS. Ao se inserir um novo dado, pode-se demandar uma junção entre dois *clusters* separados. Logo, a classificação de um novo dado também requer o novo cálculo dos *clusters*, o que é inviável, devido ao tempo de processamento.

Outro ponto que inviabiliza a aplicação do algoritmo DBSCAN é ele ser voltado para aplicações onde são necessárias a detecção de *outliers*. No caso deste problema, por definição, não existem *outliers*. Caso algum *outliers* fosse detectado, seria uma clusterização incorreta, pois, tratam-se de servidores físicos do SN e recursos, que não podem ser desprezados. Assim, por o *K-Means* se mostrar mais adequado em uma análise algorítmica, com uma complexidade mais baixa de tempo e memória, e uma forma natural de classificação de novos dados, a aplicação do DBSCAN e do *Hierarchical Clustering* não foram continuadas nestes experimentos.

7.2 Experimentos Computacionais

Os experimentos exibidos nesta Seção são realizados em um cenário *online* e utilizam os mesmos cenários e parâmetros utilizados nos experimentos do Capítulo 6. Entre

os objetivos destes experimentos estão: mostrar a eficiência da abordagem chamada Clusterização por Consumo de Recursos junto ao K-*Means*; e mostrar os efeitos da adição de novos dados no histórico com retreinamento do modelo.

7.2.1 Algoritmos Avaliados

São realizados experimentos com um algoritmo exato, sem limitação de tempo de execução, baseada na variação do modelo que reotimiza as restrições de *placement* relativas às instâncias de VNFs ativas. Variam-se os valores de k, onde:

- ILP - algoritmo exato, sem redução do espaço de busca, no qual reotimizam-se as restrições de dimensionamento de instâncias de VNFs (Capítulo 4). Usado como *baseline*;
- ILPsk - algoritmo que utiliza o $\overline{SN}$ residual induzido pela Clusterização por Localização Espacial para processar cada SFC (Seção 6.1). Neste caso, o espaço de busca é reduzido pela clusterização. Como exemplo, o algoritmo $ILPs10$ antes de mapear uma SFC utiliza a referida técnica com $k = 10$;
- ILPrk - algoritmo que utiliza o $\overline{SN}$ residual induzido pela Clusterização por Consumo de Recursos para processar cada SFC (Seção 7.1). Neste caso, o espaço de busca é reduzido pela clusterização. Como exemplo, o algoritmo $ILPr500$ antes de mapear uma SFC, utiliza referida técnica com $k = 500$.

7.2.2 *Dataset*

O histórico foi gerado a partir de simulações controladas, em *offline*, realizadas exclusivamente para treinar os modelos de AM. Nestas simulações, o mapeamento deve ser realizado forma otimizada. No entanto, em alguns casos, o provedor pode possuir um *log* do sistema incluindo esse histórico de mapeamentos já otimizado, e reutilizá-lo.

Em muitos *datasets* podem existir dados faltantes *e.g.*, oriundos de falhas no sistema de coleta ou problemas de integração entre diferentes sistemas. Este é um ponto positivo em se trabalhar com *datasets* gerados por simulações. Como os dados foram gerados em um ambiente controlado, existe uma certeza de não haver informações ausentes ou distorcidas, que poderiam afetar os resultados das análises.

Neste trabalho, o *dataset* utilizado é gerado a partir de simulações usando o SN apresentado na Seção 5.2, o algoritmo exato ILP sem clusterização (Capítulo 3), os mesmos parâmetros de SFCs e configurações propostas na Subseção 4.1. A adoção do algoritmo exato para gerar o *dataset* é justificada pelos bons resultados gerados, visto que não há erros de tomada de decisão e vieses que podem potencialmente ocorrer com heurísticas. Caso contrário, um mapeamento de baixa qualidade presente no histórico poderia acarretar uma replicação de mapeamentos ruins, e comprometer o resultado final.

A geração das SFCs para serem mapeadas possui uma aleatoriedade referente à quantidade demandada de recursos (Seção 4.1). A frequência de chegada das SFCs é dada conforme as classes de serviços tratadas, sendo adotado o Cenário 4 de SFCs (Seção 4.1). Foram geradas e processadas 60000 SFCs, sendo computadas somente as mapeadas com sucesso (47.538 SFCs). Em uma análise de possíveis características a serem utilizadas, conhecidas a *priori* pelos provedores, adotou-se as mostradas na Tabela 7.1.

Tabela 7.1: Descrição das características analisadas

Descrição	característica
Latitude do nó de origem (s^v) da SFC	F1
Longitude do nó de origem (s^v) da SFC	F2
Latitude do nó de destino (d^v) da SFC	F3
Longitude do nó de destino (d^v) da SFC	F4
Largura de banda inicial demandada pela SFC (bw^v)	F5
Capacidade de memória inicial demandada pela SFC (m_f^v)	F6
Capacidade de processamento inicial demandado pela SFC (c_f^v)	F7
Atraso fim a fim máximo tolerado por uma SFC (t_{dl}^v)	F8
Número de VNFs demandadas pela SFC ($\|F^v\|$)	F9
Média da capacidade de memória residual no instante t	F10
Média da capacidade de processamento residual no instante t	F11
Média da capacidade de largura de banda residual no instante t	F12
Somatório do número de arcos ativos do SN no instante t	F13
Somatório do número de servidores ativos no instante t	F14

Fonte: Elaborado pelo autor.

7.2.3 Pré-processamento

A normalização de dados consiste em transformar os valores das características para um intervalo específico conhecido. Esta técnica é usada para evitar que o modelo resultante seja enviesado para as características com maior ordem de grandeza. A aplicação da normalização consiste em uma transformação afim, onde os valores são modificados por um transformador linear. Algumas abordagens de AM não variam o resultado mesmo havendo uma transformação afim dos dados de entrada, *e.g.,* uma rede neural. Contudo, toda normalização gera uma ponderação de dados, fator que pode ser problemático no algoritmo K-*means*, sendo este sensível à transformações afins.

Uma transformação afim nos dados para a execução do algoritmo K-*means* implica uma mudança no espaço métrico, afetando diretamente a distância euclidiana entre dois dados (Kaufman and Rousseeuw, 1990). Segundo os autores, por um lado, deixar as variações de dados desiguais pode ser interpretado como aplicar mais peso nas variáveis com menor variação. Por outro, normalizar os dados pode ser pensado como uma geração de pesos relativos aos dados, ocasionando perda de relação entre valores, o que pode causar efeitos adversos nos *clusters* concebidos. Um exemplo de tal problema é descrito no trabalho Singhal et al. (1996). Os autores observaram que esquemas de normalização tendem a favorecer características de maior amplitude, em detrimento as de menor. Para resolver tal problema, eles propõem um esquema de normalização com base em um fator, chamado pivô. A ideia é que o pivô penalize algumas características. Neste trabalho é utilizada a normalização Min-Max definida como:

$$x = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{7.1}$$

A aplicação da Equação 7.1 modifica cada campo de cada coluna para um valor entre $[0; 1]$. Ressalta-se que, a normalização tende a reduzir a compreensão do modelo analisado, perdendo-se as referências dos valores reais de cada campo. Ao se normalizar, é atribuído a todas as variáveis um peso igual, na esperança de alcançar a chamada objetividade (Kaufman and Rousseeuw, 1990). Contudo, a normalização não pode ser usada indiscriminadamente. Sem um conhecimento prévio do problema pode-se gerar uma descaraterização da grandeza dos valores. Em aplicações específicas, certas variáveis podem ser intrinsecamente mais importantes que outras, e uma atribuição de pesos deve ser ponderada para não se gerar resultados inesperados.

O problema tratado neste Capítulo é um caso parecido com o descrito por Kaufman and Rousseeuw (1990), pois as caraterísticas relacionadas à localização geográfica devem ser estritamente satisfeitas para que o mapeamento seja factível. Por outro lado, características como demanda e consumo de recursos estão relacionadas à métricas de lucros e custos à serem otimizados, e não diretamente à viabilidade do problema. Portanto, deve-se gerar uma normalização, ponderando os dados para que algumas características não sobreponham a viabilidade do mapeamento a ser gerado no SN induzido em questão.

7.2.4 Seleção de características

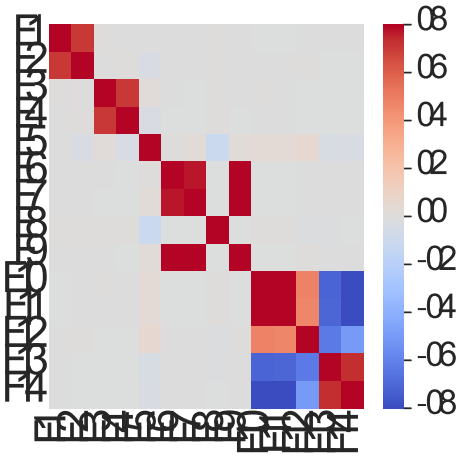
A variância é uma métrica que descreve a variabilidade das observações em relação à sua média, e usada na seleção de características para tentar identificar características mais relevantes. Referido valor é denominado como média dos desvios quadrados e repre-

sentado com o símbolo σ^2. Efetuando uma análise exploratória dos dados, é percebido que algumas características não variam significativamente durante o processo de mapeamento das SFCs. Do ponto de vista prático, se a variação de uma característica for baixa, trata-se de um recurso de comportamento quase constante, que gera poucas informações significativas, e não melhorará o desempenho do modelo de aprendizado de máquina resultante. Assim, tal característica pode ser omitida do conjunto de dados, visto que não contribui para a capacidade de clusterização e previsão da aplicação. Uma visualização da variância das características analisadas deste *dataset* é apresentada no Apêndice I.

Outra etapa realizada na seleção de características é a análise de correlação dos dados utilizados na geração do modelo de clusterização. A análise de correlação é uma etapa fundamental para definir quais características usar ou não. Tal técnica informa como as informações em um conjunto de dados estão relacionadas entre si. Nesta análise, o valor da correlação deve ficar entre -1 e $+1$, e pode ser interpretado da seguinte forma: se o valor se aproxima de 0, às duas características são independentes uma da outra; se o valor se aproxima de $+1$, trata-se de uma correlação positiva e indica que as características se movem na mesma direção; e se o valor se aproxima de -1, trata-se de uma correlação negativa e indica que as características se movem na direção oposta.

Para tal análise, é criado um mapa de calor de correlação (Figura 7.3). Com base em tal correlação, pode-se tratar o problema da presença de uma relação linear aproximada entre as características, problema chamado intercorrelação. Quando as variáveis estão correlacionadas entre si, pode-se afirmar que existe intercorrelação entre elas (Kutner et al., 2004). A existência da intercorrelação pode afetar a variância, dando mais peso a uma característica em relação a outra, o que a princípio não é indicado.

Figura 7.3: Mapa de calor da correlação entre as características analisadas



Fonte: Elaborado pelo autor.

Um ponto colocado por Holland (1986), é que duas características correlacionadas não implicam, necessariamente, no fato que existe alguma relação entre elas. Analisando

o mapa de calor, percebem-se correlações entre:

- $F10, F11, F12$, $F13$ e $F14$: observando a variância de cada caraterística (Apêndice I),optou-se por manter a $F12$, pois infere-se que ela é relacionada ao uso de arcos, fator que pode fragmentar o SN, e gerar rejeições de mapeamentos de SFCs;
- $F6$, $F7$ e $F9$: são caraterísticas com correlação próximas de 1, e em experimentos preliminares, percebeu-se que apesar de as características possuírem uma variância significativa, a manutenção de qualquer uma delas no modelo gerou uma queda de desempenho na abordagem, assim as três características foram removidas. Este fato é relacionado ao discutido por Kaufman and Rousseeuw (1990), ao se utilizar várias caraterísticas diferentes no *K-means*, são atenuadas as importâncias de algumas características, em prol de outras. Nesta aplicação, acaba-se tirando importância das restrições de localidade, fundamentais para um mapeamento ser aceito, e colocando importância em outros aspectos, neste caso, relacionado as capacidades demandadas de memória, processamento e quantidade VNFs, fundamentais para se gerar bons lucros, mas não são fatores escassos ao ponto de gerar rejeições (neste cenário);
- $F5$ e $F8$: em experimentos preliminares, percebeu-se que manter a característica $F8$ e remover a $F5$ gerou uma melhora no desempenho do algoritmo. Outro ponto que motivou esta escolha fundamenta-se na variância de $F8$ ser maior que a de $F5$ (Apêndice I);
- $F1$ e $F2$ e $F3$ e $F4$: apesar de existir uma correlação entre esses pares de características, elas apresentam uma alta variância (Apêndice I),e do ponto de vista da aplicação, são fundamentais, pois determinam os pontos de mapeamento S^v e D^v de uma SFC. Assim, tais características são mantidas no modelo;

Por fim, mantiveram-se as características $F1, F2, F3, F4, F8, F12$ para o restante dos experimentos (Tabela 7.1).

7.2.5 Interpretação dos Modelos de Clusterização

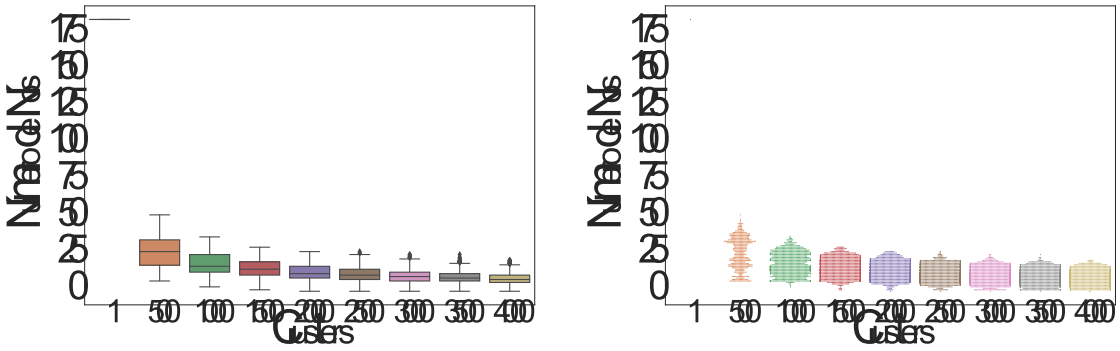
O objetivo desta Seção é avaliar o quanto o algoritmo K-*Means* consegue agrupar as caraterísticas selecionadas de maneira uniforme, gerando uma clusterização promissora. Preliminarmente, uma explicação sobre o K-*Means* é suas métricas de avaliação podem ser vistas no Apêndice F. No algoritmo K-*Means* é necessário informar quantos k *clusters* serão gerados. Para tal, são realizadas várias execuções do algoritmo, onde o valor de k é variado entre $k = 1$ e $k = 4000$ (valores maiores não geraram alterações proeminentes).

Em uma análise prévia, mostrada no Apêndice I, as Figura I.3 e I.4 apresentam o *Elbow Method* e o *Silhouette Coefficient* resultante das clusterizações do *dataset*, no qual foi constatado que o melhor valor é $k = 5$. Contudo, trata-se de uma análise simples, e outros experimentos devem ser conduzidos para serem definidos bons valores de k.

Como percebido nos experimentos com a abordagem de Clusterização por Localização Espacial, se o número de componentes em um *cluster* for relativamente pequeno (alto valor de k), uma SFC específica pode ser rejeitada por haver poucos recursos físicos aptos para seu mapeamento. Por outro lado, se o número de componentes em um *cluster* for relativamente grande (baixo valor de k), pode não haver melhorias no tempo de execução. Percepção que também é induzida para a abordagem de Clusterização por Consumo de Recursos. A hipótese, mais uma vez é que a escolha de bons valores de k é fundamental para o desempenho deste algoritmo. A Figura 7.4 mostra o quão relacionado está o valor de k, com a redução do número de componentes de cada cluster.

Figura 7.4: Variação no número de nós dos *clusters* com o algoritmo K-*Means*

(a) Box plot (b) Swarm plot



Fonte: Elaborado pelo autor.

Analisando as Figuras 7.4a e 7.4b, nota-se que o número de componentes em cada *cluster* diminui mais lentamente em relação ao aumento do valor de k. Se por um lado, na abordagem Clusterização por Localização Espacial, uma dúzia de *clusters* afeta significativamente a quantidade de componentes em cada $\overline{SN}$ induzido, por outro, na abordagem de Clusterização por Consumo de Recursos esse valor é bem maior. Na abordagem Clusterização por Consumo de Recursos, necessita-se de algumas centenas de *clusters* para perceber o efeito da redução do número de componentes. Esse comportamento é devido ao aumento no valor de k possuir um limite (teto) diferente em cada algoritmo. Neste caso, o valor máximo de k é limitado pelo número de SFCs presentes no histórico H.

As análises anteriores são introdutórias, e deixam uma lacuna em relação à escolha do valor k junto às métricas de rede, fator que motiva a realização dos experimentos mostrados na Subseção 7.2.6. Contudo, percebe-se boas faixas de valores de k a serem investigados, sendo $k = \{500, 1000, ..., 4000\}$.

7.2.6 Análise da Estratégia de Clusterização

Análise da ponderação dos valores das características normalizadas: algumas restrições que devem ser satisfeitas para a SFC ser mapeada são as localizações geográficas dos pontos de origem e destino. As outras características utilizadas, são relacionadas a aspectos de otimização. Neste ponto percebe-se um impasse: tais características devem ser tratadas com o mesmo peso na hora de calcular o centroide de cada *cluster* que irá dar origem aos SNs induzidos? Ao se tratar tais características de modo normalizado, todas devem ficar no intervalo $[0; 1]$, alterando a relação de valor das características.

Uma questão a ser investigada é analisar se tais caraterísticas devem ser ponderadas ou não, de modo a não se criar inviabilidades, e até que ponto essa ponderação afeta a função objetivo. Para tal, os experimentos mostrados nesta Seção são realizados com um parâmetro Υ, utilizado para ponderar o peso das características no momento de se realizar as clusterizações. Foi experimentalmente adotado $\Upsilon \in \{1, 10, 20, ..., 100\}$. No caso, $\Upsilon = 1$ indica que as caraterísticas $F8$ e $F12$ possuem o mesmo peso, *i.e.* $\frac{F8}{1}$ e $\frac{F12}{1}$; e $\Upsilon = 100$ indica que as caraterísticas $F8$ e $F12$ possuem um peso amortizado, *i.e.* $\frac{F8}{100}$ e $\frac{F12}{100}$, afetando diretamente o cálculo das distâncias no algoritmo K-*Means*. A Tabela 7.2 mostra o impacto da variação deste parâmetro no mapeamento das SFCs entrantes.

Tabela 7.2: Impacto da variação do parâmetro Υ com k fixado em 2000 (IC= 95%)

Variação de parâmetro	Taxa de aceitação (%)	IC (%)	Tempo de processamento por SFC (seg)	IC (seg)	Lucro ($)	IC
ILPr2000 sem $f8$ e $F12$	79,99	0,98	1,024	0,091	2758,072	253,040
ILPr2000 e Υ=90	84,29	1,21	1,063	0,073	2839,628	218,421
ILPr2000 e Υ=70	84,19	0,83	1,083	0,077	2820,203	178,202
ILPr2000 e Υ=**50**	**84,58**	**0,73**	**1,062**	**0,103**	**2885,648**	**196,765**
ILPr2000 e Υ=30	83,38	1,25	1,115	0,162	2792,027	187,742
ILPr2000 e Υ=10	80,23	0,89	1,391	0,764	2726,598	153,586
ILPr2000 e Υ=1	6,46	1,80	0,092	0,020	171,452	77,488
Algoritmo ILP^i	81,40	5,87	1,763	0,167	2828,250	155,592

Fonte: Elaborado pelo autor.

Ao se realizarem os mapeamentos sem a utilização das características $F8$ e $F12$, a taxa de aceitação dos mapeamentos diminuiu em relação a outros casos (Tabela 7.2). Como exemplo, ao adotar $\Upsilon = 50$, a utilização das características relacionadas ao atraso fim a fim e banda residual afetaram os *clusters* gerados, e consequentemente na distribuição dos mapeamentos sobre o SN, fator que elevou a taxa de aceitação. Por outro lado, percebe-se que os lucros gerados nestes casos oscilaram dentro de um IC, não mostrando serem estatisticamente relevantes com 95%. Comportamento que demonstra um possível *trade-off*. No caso, mesmo mapeando menos SFCs, o modelo sem as caraterísticas $F8$ e $F12$ consegue gerar um maior compartilhamento de recursos, mantendo uma boa geração de lucros. Por outro lado, com as características $F8$ e $F12$, o modelo tende a gerar uma

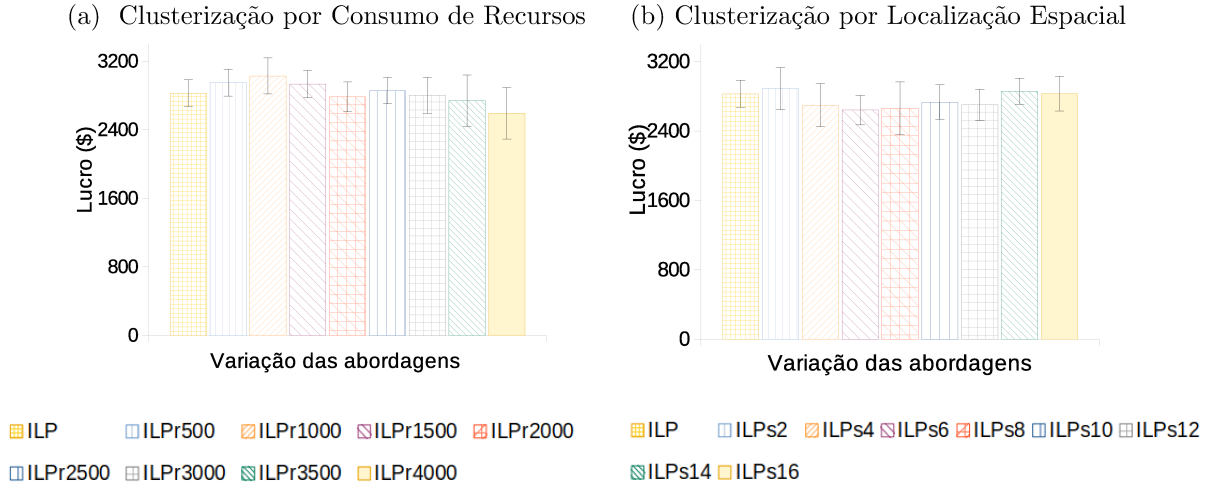
taxa de aceitação maior, mas com uma queda no compartilhamento de recursos, fator que pode elevar os custos de atender um determinado serviço. Aspectos relacionados a este *trade-off* serão investigados com mais profundidade ao decorrer deste Capítulo. Uma versão completa da Tabela 7.2, e outras análises podem ser encontradas no Apêndice I.

Ressalta-se que os valores de Υ e as caraterísticas utilizadas são experimentais, e trata-se de um tema que pode ainda ser amplamente explorado em trabalhos futuros, principalmente em relação à engenharia de características e parametrização do algoritmo de AM. Por fim, percebe-se que, na abordagem Clusterização por Consumo de Recursos, bons valores de k para serem investigados, são mais altos que na abordagem de Clusterização por Localização, sendo $k = \{500, 1000, ..., 3000\}$ e $\Upsilon = 50$.

Da Figura 7.5 até a Figura 7.7 mostram-se em colunas o acumulado final do processamento de todas as SFCs do cenário avaliado. Das Figuras 7.8 a 7.10, os resultados são mostrados em janelas de tempo que correspondem à entrada da primeira até a última SFC do conjunto de dados. Em tais Figuras são mostradas as métricas descritas na Subseção 4.1.1 para uma única execução dos algoritmos em uma instância escolhida aleatoriamente. Nestes casos, o eixo x apresenta o tempo de simulação em unidades de tempo t e o eixo y a métrica observada. Para cada um dos gráficos mostrados nas Figuras 7.8 a 7.10, é gerada uma tabela que resume o comportamento de cada algoritmo (Apêndice I).

Análise do lucro (Figura 7.5): o lucro é influenciado diretamente pela taxa de aceitação (Figura 7.7). Percebe-se que ambas as abordagens analisadas, independentemente do valor de k adotado, não geram uma diferença estatisticamente significativa com 95% de confiança no lucro dos provedores. Tal percepção demonstra que os SN induzidos conseguem atender às SFCs, e incluem dispositivos similares aos que seriam utilizados caso o mapeamento fosse realizado com o algoritmo tradicional ILP.

Figura 7.5: Lucro final dos provedores

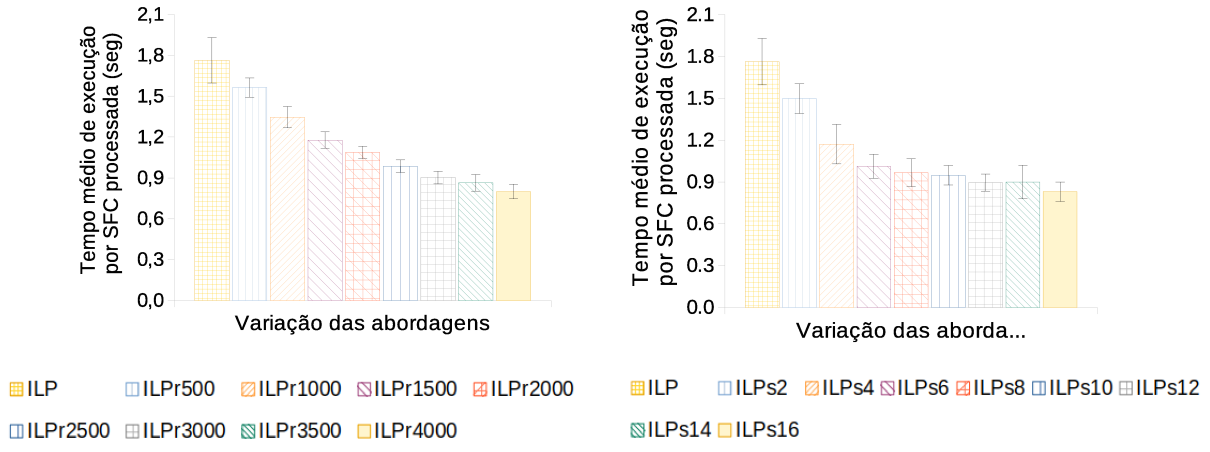


Fonte: Elaborado pelo autor.

Análise do tempo de execução (Figura 7.6): o algoritmo ILPrk possui uma diferença no tempo de execução de até $\approx$ 120% em relação ao algoritmo ILP, sendo o ILPrk o mais rápido (Figura 7.6a com $k = 4000$). No algoritmo ILPrk , existe uma inclusão de novas características a serem clusterizadas em relação ao algoritmo ILPsk. Percebe-se que essa adição não afetou significativamente o tempo de execução, mas gerou variâncias menores, como percebido se comparados os ICs das Figuras 7.6a e 7.6b.

Figura 7.6: Tempo médio por SFC total

(a) Clusterização por Consumo de Recursos (b) Clusterização por Localização Espacial



Fonte: Elaborado pelo autor.

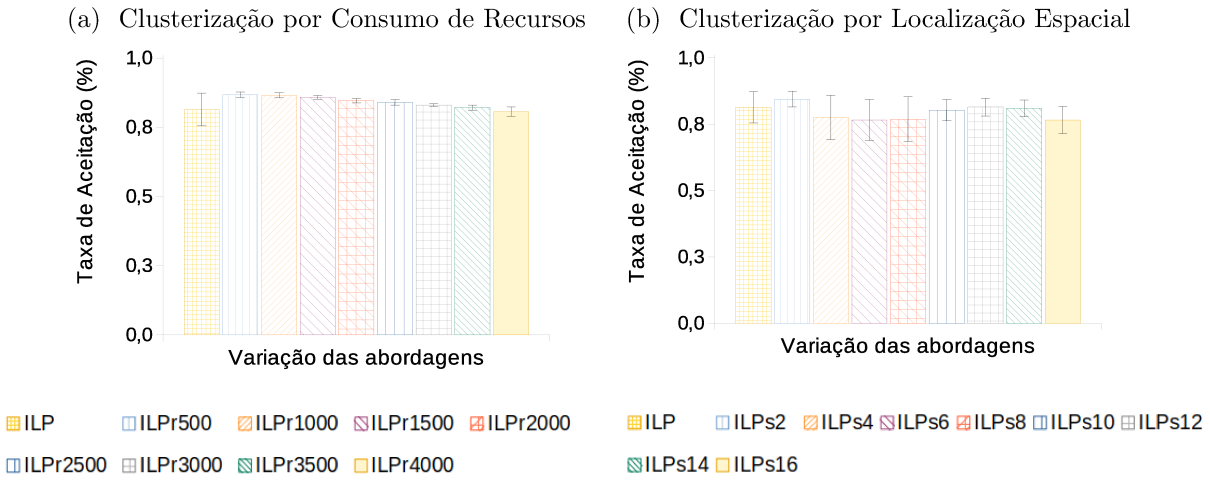
Em ambas as abordagens houve uma redução no tempo de execução em relação ao algoritmo ILP. Tal redução está diretamente relacionada ao tamanho dos *clusters* que formam os SNs induzidos. Observando a Figura 7.4, nota-se que ao aumentar o valor de k o número médio de componentes dos $\overline{SNs}$ induzidos é menor, conforme mostrado no segundo quartil de cada *box plot* (Figura 7.4a). Fator que afeta diretamente o tempo de execução. É mostrado na Figura 7.6 que a redução do tempo de execução decresce rapidamente para os incrementos iniciais no valor de k, mas esse decrescimento se estabiliza em determinado momento. Tais experimentos demonstram que aumentar o valor de k acima de um limite de $k = 12$ no algoritmo ILPsk, e $k = 3000$ no algoritmo ILPrk, não causam impactos expressivos na redução do tempo de execução.

Análise da taxa de aceitação (Figura 7.7): todos os algoritmos obtiveram uma taxa de aceitação constantemente alta, não sendo estatisticamente diferentes em relação ao algoritmo ILP com uma confiança de 95%. Tal percepção corrobora a premissa da abordagem, que bons SNs induzidos reduzem o tempo de execução sem afetar negativamente a taxa de aceitação, podendo ser até melhor em alguns casos pontuais, como para baixos valores de k.

Percebe-se na Figura 7.7 que as variâncias são bem menores no algoritmo ILPrk em relação ao ILPsk. Resultado de certa forma prenunciado, uma vez que as características

adicionadas no algoritmo ILPrk acarreta uma maior regularidade também nos lucros, como mostrado anteriormente na Figura 7.5. Outras análises sobre a taxa de aceitação são mostradas no Apêndice I.

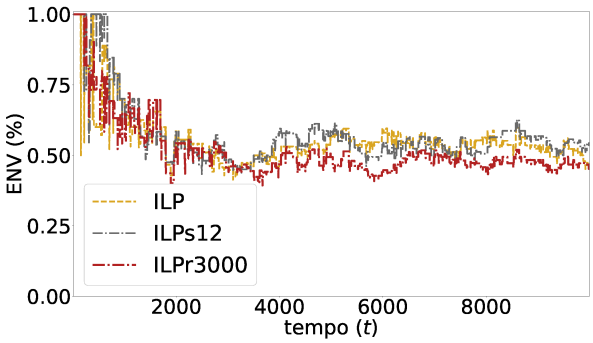
Figura 7.7: Taxa de aceitação final



Fonte: Elaborado pelo autor.

Análise do espalhamento dos nós virtuais (Figura 7.8): com o intuito de facilitar a leitura dos gráficos mostrados na sequência desta Seção (Figuras 7.8 a 7.10), exportam-se somente os resultados de algumas variações de algoritmos promissoras, sendo ILP, ILP$^i s12$ e ILP$^i r3000$. Um resumo numérico da Figura 7.8 pode ser visto no Apêndice I. A Figura 7.8 mostra o quão centralizado ou distribuído é o espalhamento médio das VNFs demandadas sobre os nós físicos do SN. Os 3 algoritmos mostram um espalhamento de nós virtuais próximo de 50%, *i.e.,* na média, cada servidor físico ativo está sendo usado por duas SFCs diferentes (Tabela I.3).

Figura 7.8: Espalhamento dos nós virtuais



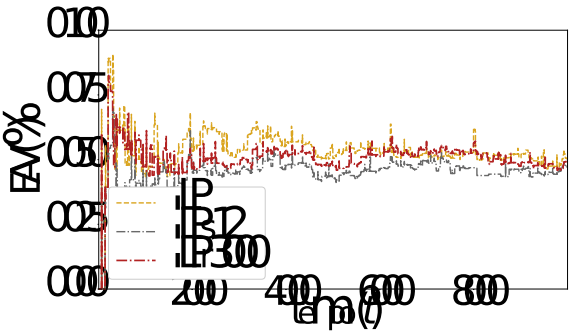
Fonte: Elaborado pelo autor.

Em tal experimento, o algoritmo ILP$s12$ mostra um espalhamento de $\approx$ 57.34%, sendo $\approx$ 11.57% pior que o algoritmo ILP$r3000$. Fator devido ao algoritmo ILP$r3000$

gerar $\overline{SNs}$ induzidos com um posicionamento de VNFs que incentiva o compartilhamento de nós físicos. Devido ao algoritmo ILPrk ser baseado em informações otimizadas, aprendidas do histórico H, ele consegue prever assertivamente quais nós físicos do SN são mais qualificados para serem compartilhados ou não. Percebe-se que o espalhamento do algoritmo ILPr3000 é melhor que o do algoritmo ILP, sendo esta diferença média de $\approx 8.19\%$ (Tabela I.3). Tal ocorrência é devido ao extenso histórico de mapeamentos presente na base de dados, possibilitando uma política de mapeamentos de servidores mais assertiva em distribuição, e compartilhamento de nós físicos.

Análise do espalhamento dos arcos virtuais (Figura 7.9): para se reduzir o número de arcos físicos utilizados, esta métrica deve ser minimizada. Um resumo numérico da Figura 7.9 pode ser visto no Apêndice I. Constata-se que o algoritmo ILPsk induz um roteamento médio de arcos mais conciso (menos saltos), e consequentemente, usa menos arcos físicos do SN em relação ao modelo ILP tradicional em todos os experimentos realizados (Tabela I.4), sendo um comportamento similar percebido em relação ao algoritmo ILPrk. Como exemplo, o mapeamento das SFCs ocupa na média $\approx 5.29\%$ dos arcos físicos do SN com o algoritmo ILP, enquanto no algoritmo ILPs12 este valor é de $\approx 4.50\%$, e no algoritmo ILPr3000 este valor é de $\approx 5.02\%$. Este fato ocorre devido ao modelo de otimização tradicional possuir uma visão completa do espaço de busca, maximizando o compartilhamento de servidores para reduzir os custos de operação (Figura 7.8), *i.e.,* o mapeamento com uma visão completa do SN centraliza o consumo de recursos em alguns servidores devido aos custos operacionais de arcos físicos serem menores que os custos dos servidores.

Figura 7.9: Espalhamento dos arcos virtuais



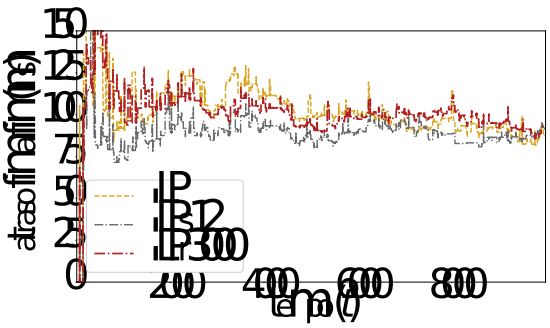
Fonte: Elaborado pelo autor.

Devido aos *clusters* gerados pelo algoritmo ILPrk serem baseados em um histórico gerado pelo algoritmo ILP, é percebido que tal algoritmo possui um espalhamento de caminhos virtuais maiores em até $\approx 10.32\%$ que o algoritmo ILPsk. Percebe-se também que, além de ter uma taxa de aceitação alta, ao algoritmo ILPrk consegue reduzir o espalhamento de arcos virtuais em relação ao algoritmo tradicional baseado em ILP em $\approx 5.44\%$. Tal aspecto acontece por o algoritmo ILP tradicional não possuir informações

deste histórico para gerar os roteamentos, por outro lado, o ILPrk é baseado em um longo histórico de mapeamentos otimizados realizados no passado.

Análise do atraso médio fim a fim (Figura 7.10): um resumo numérico da Figura 7.10 pode ser visto no Apêndice I. De modo semelhantemente ao espalhamento de arcos (Tabela I.4), o modelo de otimização tradicional e o algoritmo ILPrk, além de causarem um mapeamento de arcos mais espalhado sobre os recursos físicos do SN (Tabela I.5), originaram um atraso fim-a-fim mais alto para o usuário final. No algoritmo ILPsk, esse comportamento é previsto devido à natureza da proposta de redução dos componentes processados do SN. Dessa forma é improvável que uma SFC seja mapeada de forma espalhada (estendida) sobre os arcos físicos e servidores distantes no SN. A mesma argumentação é verificada para as abordagens ILP e ILPrk, mas com um efeito contraposto. O algoritmo ILP possui uma visão global do SN, e centraliza os mapeamentos sobre os nós físicos já ativos de modo a evitar maiores custos de servidores (menor espalhamento de nós, Tabela I.4). Assim o algoritmo ILP pode mapear uma SFC de forma espalhada sobre os arcos do SN, gerando um alto atraso fim a fim, o que pode ser prejudicial para o usuário final. Como o algoritmo ILPrk utiliza SNs induzidos gerados por mapeamentos realizados simuladamente com o algoritmo ILP, tal abordagem possui um comportamento similar, gerando um atraso fim a fim mais alto.

Figura 7.10: Atraso fim a fim



Fonte: Elaborado pelo autor.

7.3 Alterações do *Dataset*

O objetivo desta Seção é avaliar quais impactos que possíveis alterações no *Dataset* podem gerar, e qual a importância do retreinamento do modelo, *i.e.*, da geração de novos $\overline{SNs}$ induzidos a serem usados pelo algoritmo ILPrk. Explorando tais questões, alguns pontos devem ser trabalhados, dentre eles, é dado foco no ponto: como saber quando retreinar e atualizar os $\overline{SNs}$ induzidos? Neste trabalho adota-se o chamado retreinamento

orientado por métricas, no caso, os dados processados são monitorados, e caso as demandas processadas se desviem significativamente dos dados de treinamento originais, tal etapa deve ser efetuada (*Drift Detection*) (Sakamoto et al., 2015; Najafzadeh et al., 2021).

7.3.1 Divergência de KL

Um modo de se realizar o retreinamento orientado por métricas é calcular a distância entre as duas distribuições. Uma alternativa que vem sendo usada na literatura de AM é o cálculo da divergência entre duas distribuições de probabilidade (Karampasi et al., 2020; Gong et al., 2021). O tal conceito pode ser interpretado como uma medida não simétrica do quão essas distribuições são diferentes. Para a detecção destes desvios, optou-se por utilizar a divergência KL (Kullback and Leibler, 1951). Alguns conceitos introdutórios sobre entropia e a divergência KL são mostrados no Apêndice I.

Nesta tese, ao invés de retreinar o modelo e gerar os novos SN induzidos com base na quantidade de novas instâncias, ou em períodos pré-determinados, a divergência de KL será utilizada. Tal aspecto é embasado na premissa que, de posse de uma *dataset* considerável, muitas SFCs têm um comportamento de demandas parecido, e não trazem ganho de informação. Logo, neste caso, não existe a necessidade de se fazer um retreinamento. Deste modo, a divergência de KL irá caracterizar essa necessidade de retreinamento. Caso os valores da métrica de KL destoem abruptamente de 0, o modelo deve ser retreinado; caso os valores sejam próximos de 0, não existe a necessidade de retreinamento, pois as SFCs mapeadas não geram diferenças significativas na distribuição de probabilidade.

7.4 Experimentos Computacionais

7.4.1 Cenário de Experimentos e Tomada de Decisão

Se os dados do *dataset* que originaram o modelo no passado começarem a divergir dos dados observados no momento atual, existe a necessidade de retreinar o modelo. Para simular tal comportamento, é definido um cenário onde são assumidos os mesmos parâmetros dos experimentos da Seção 7.2, considerados $110000t$ unidades de tempo na

simulação, e proposta uma situação hipotética relacionada à variação do atendimento das classes de serviços apresentadas na Subseção 4.1.4. Por fim, são assumidos, para fins de exemplificação, alguns momentos distintos de simulação:

I Nesta etapa, o provedor acaba de iniciar suas operações, logo ele não possui um histórico das SFCs processadas. Neste momento, o provedor presta atendimento somente às SFCs de classe 3. É assumido que o provedor deva realizar o mapeamento das SFCs com o algoritmo exato apresentado nas seções anteriores. Posteriormente, em uma atividade paralela, quando o provedor conseguir gerar uma caracterização das SFCs que estão sendo demandadas, similarmente à Subseção 7.2.2, é possível gerar um *dataset* de SFCs, processá-lo de maneira otimizada por simulação, e gerar os SNs induzidos para serem utilizados pelo algoritmo ILPrk. Uma questão surge neste ponto: quantas SFCs o histórico deve possuir para gerar um *dataset* amplo, que abarque a variabilidade de SFCs demandadas?

II Nesta etapa o provedor possui uma experiência com o atendimento das SFCs da classe 3, e começa atender a uma nova parcela de mercado, processando agora as SFCs da classe 2 concomitantemente à classe 3. Ressalta-se que à classe 2 é mais desafiadora de ser processada, no sentido das SFCs possuírem restrições de atraso fim a fim mais rígidas. Neste ponto geram-se as seguintes questões: *i*) o histórico otimizado que o provedor possui, que contém somente SFCs da classe 3, irá gerar um desempenho aceitável no mapeamento das novas SFCs? *ii*) como afirmar que o retreinamento do modelo é necessário?

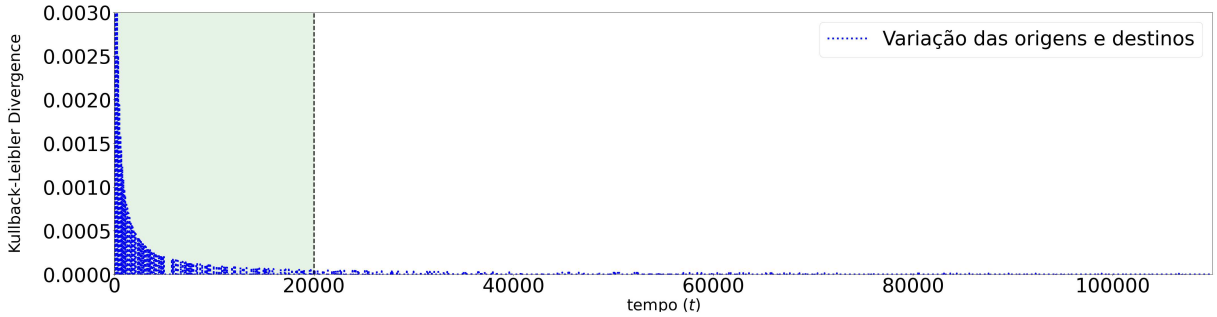
III Visando atender a uma nova e crescente parcela de mercado, o provedor inicia o atendimento das SFCs da classe 1. Tais SFCs são muito sensíveis à métricas de QoS, requerendo atrasos fim a fim de menos de 150ms por SFC, exemplos Jogos *online* e aplicações de RA. Essa situação é similar ao acontecido no momento 2, quando o provedor iniciou o atendimento das SFCs de uma classe que não conhecia. Dessa forma, são geradas as perguntas: *i*) o histórico otimizado que o provedor possui, que contém somente SFCs das classes 2 e 3 e não contempla demandas da classe 1, irá gerar um desempenho aceitável no mapeamento das novas SFCs? *ii*) como afirmar que o retreinamento do modelo é necessário?

Todas as questões levantadas nos itens anteriores serão respondidas com uma tomada de decisão embasada na divergência de KL. Neste caso, a divergência de KL é usada para auxiliar o cientista de dados a perceber os momentos de mudança de comportamento, e a entender os pontos que devem ser retrabalhados para a geração de novos modelos, e consequentemente, novos SN induzidos.

7.4.2 Análise dos Momentos de Retreinamento

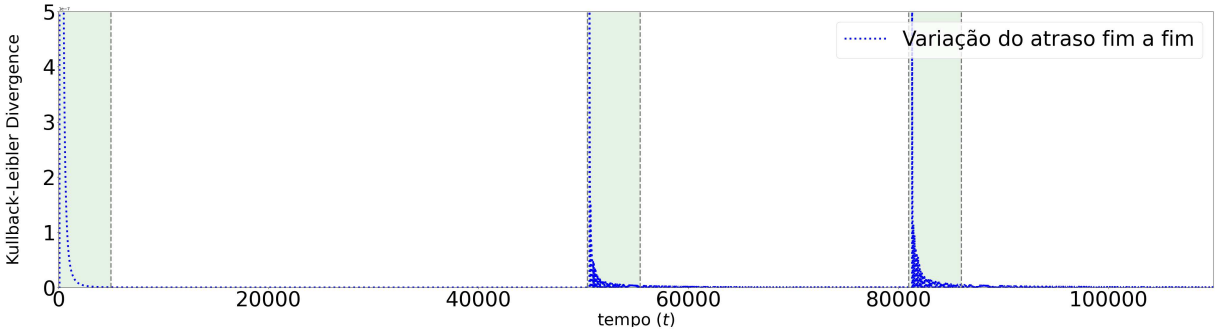
Momento 1 (de 0t a 50000t): o provedor desconhece as características das SFCs que vão ser processadas. Adota-se que os mapeamentos iniciais são feitos com o algoritmo ILP^i. Com o mapeamento das SFCs, o provedor começa a conhecer as características que estão sendo demandas, e, em determinado momento ele já pode gerar um *dataset*, e começar a aplicar o algoritmo ILPrk. Neste ponto, a divergência de KL deve ser usada. Desta forma, todas as caraterísticas do processamento das SFCs, devem ser monitoradas, e caso exista uma estabilização da métrica de KL, tal *dataset* pode ser gerado. Para uma melhor visualização, a divergência de KL é plotada para as caraterísticas: variações dos nós de origem e destino (Figura 7.11), do atraso fim a fim (Figura 7.12), e a largura de banda residual dos arcos SN no momento de mapeamento t (Figura 7.13).

Figura 7.11: Monitoramento da variação dos nós s^v e d^v demandados por cada SFC



Fonte: Elaborado pelo autor.

Figura 7.12: Monitoramento da variação do t^v_{dl} demandado por cada SFC

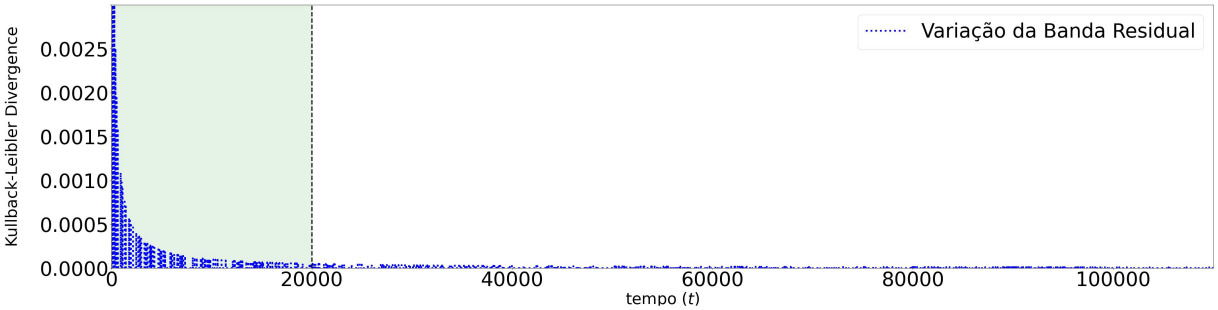


Fonte: Elaborado pelo autor.

Nas figuras, a parte sombreada representa o momento em que houve diferenças nas distribuições analisadas. A variação dos nós de origem e destino tende a ser bem diferente no começo dos mapeamentos, mas em um curto intervalo t, tal métrica decai e se aproxima de 0 (Figura 7.11). A variação da métrica KL para o t^v_{dl} decai bem mais rapidamente,

justamente por as SFCs serem da mesma classe, e essa métrica não variar neste momento (Figura 7.12). Similarmente, no momento 20000t já é conhecida toda variação de banda residual demandada por aqueles serviços atendidos (Figura 7.13). Infere-se que um bom momento para geração do *dataset*, e treinamento do modelo para a aplicação do algoritmo ILPrk, é justamente a partir do instante 20000, pois é o ponto máximo onde a métrica KL se estabilizou próxima de 0 para todas as características analisadas.

Figura 7.13: Monitoramento da variação da banda residual do SN

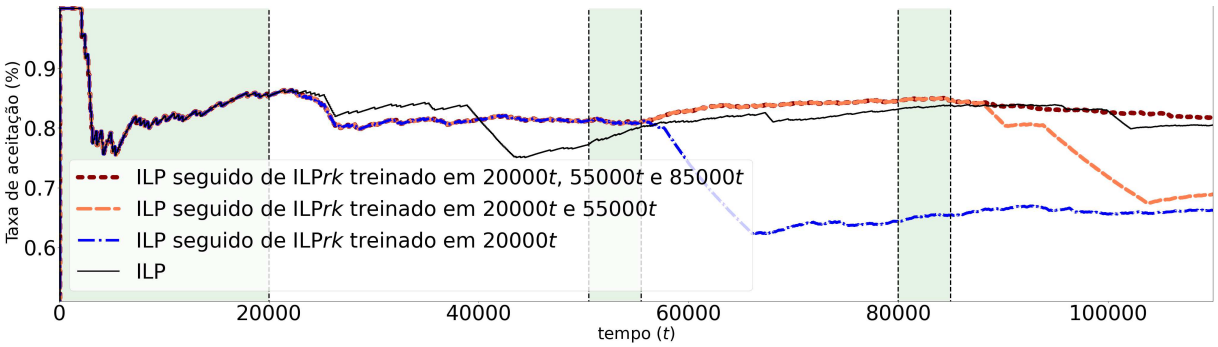


Fonte: Elaborado pelo autor.

Deste modo, no instante 20000t, em uma atividade paralela e independente do algoritmo de mapeamento, é gerado um *dataset* similarmente ao apresentado na Seção 7.2.2. Para tal, toda uma análise de características é feita, o modelo de clusterização gerado, os SNs induzidos formados, e finalmente, o algoritmo ILPrk pode ser então aplicado.

Análise da taxa de aceitação do momento 1 (Figura 7.14 de 0t a 50000t): considere que a linha preta representa os mapeamentos durantes os 50000t com o algoritmo ILP; e a linha azul o experimento em que até 20000t os mapeamentos foram realizados com o algoritmo ILP, e após 20000t, com o algoritmo ILPkr.

Figura 7.14: Impacto do retreinamento na taxa de aceitação



Fonte: Elaborado pelo autor.

Percebe-se que, do mesmo modo que os experimentos realizados na Subseção 7.2, o algoritmo ILPrk possui uma taxa de aceitação próxima ao algoritmo ILP. Mais especificamente, no cenário avaliado, o algoritmo ILPrk gerou uma taxa de aceitação maior

que o algoritmo exato (Tabela I.6, Apêndice I). Como já discutido na Seção 6.2, neste caso, por o algoritmo ILP processar todos os nós físicos para gerar um mapeamento, ele possui uma visão global do substrato de rede, e visando minimizar os custos, realiza um mapeamento com um maior compartilhamento de servidores em detrimento a roteamentos de arcos virtuais mais alongados sobre arcos físicos SN. Por outro lado, o algoritmo que realiza os mapeamentos nos SNs induzidos, pode em algum momento não compartilhar um servidor já ativo, que está sendo usado por outra SFC, justamente por ele não pertencer ao $\overline{SNs}$ induzido processado. Citada característica do algoritmo ILPkr, de realizar mapeamentos mais concisos em termos de arcos usados, mostra-se ser uma política promissora, pois ao se realizar mapeamentos com menos arcos, potencialmente abre-se mão do compartilhamento de alguns servidores, mas, por outro lado, utiliza-se menos arcos físicos, deixando mais recursos residuais disponíveis para serem utilizados por outras SFCs, comportamentos que afetaram a taxa de aceitação. Demais observações sobre estes experimentos, incluindo uma análise de lucro, são mostradas no Apêndice I.

Momento 2 (de 50000t a 80000t): o provedor já processou e conhece as características das SFCs demandadas até o momento 20000t. Dessa forma, pode-se aplicar o algoritmo ILPrk. Contudo, as demandas das SFCs eventualmente podem mudar. Se as características das SFCs processadas se alterarem significativamente, o esperado é que a distribuição de probabilidade que rege essa nova distribuição difira da distribuição anterior. Assume-se que referidas características devem estar em constante monitoramento, e caso exista alguma alteração significativa, o provedor deve então efetuar um novo treinamento do modelo, de modo que o algoritmo se adapte aos novos dados.

Observando as Figuras 7.11 e 7.13, mas considerando o intervalo de 50000t a 80000t, percebe-se que tal divergência não oscilou, indicando que, mesmo atendendo mais SFCs, sendo algumas pertencentes à classe 2, não houve variabilidade das origens e destinos das SFCs, nem na banda residual, logo, a *priori*, o modelo ainda continua sendo válido. Por outro lado, percebe-se que a variação da métrica KL para o t^v_{dl} volta a oscilar, mostrando um crescimento em 50000t (Figura 7.12). Esse crescimento caracteriza uma alteração nas demandas de t^v_{dl} das SFCs processadas, e reflete justamente o momento que o provedor começa a atender as SFCs da classe 2. A hipótese é que neste ponto, devido à alteração das demandas relacionadas, um novo *dataset* deve ser criado e novos SNs induzidos derivados.

Infere-se que um bom momento para geração de um novo *dataset*, e treinamento do modelo para a aplicação atualizada do algoritmo ILPrk, é justamente a partir do instante 55000, pois é o ponto máximo onde a métrica KL se estabilizou próxima de 0 para todas as características analisadas. Este novo *dataset*, deve então conter as características das novas demandas, em conjunto com as que antes estavam no *dataset* inicial.

Análise da taxa de aceitação do momento 2 (Figura 7.14 de 50000t a 80000t): considere que a linha preta representa os mapeamentos das SFCs com o algoritmo ILP; a linha azul representa o experimento em que até 20000t os mapeamentos

foram realizados com o algoritmo ILP, e após 20000t, com o algoritmo ILPkr; e por fim, a linha laranja representando o experimento em que até 20000t os mapeamentos foram realizados com o algoritmo ILP, e após, com o algoritmo ILPkr treinado em 20000t e retreinado em 55000t.

Percebe-se que, o algoritmo ILPrk, treinado somente em 20000t, possui uma queda forte na taxa de aceitação após 55000t. Essa queda é decorrente de tal algoritmo utilizar um modelo treinado com um *dataset* que só conhecia SFCs da classe 3. Neste caso, as SFCs da classe 2 se diferem muito da classe 3, e possuem restrições de t_{dl}^{v} mais rígidas. Neste ponto, percebe-se que o modelo precisa ser construído mais uma vez, mas agora, abarcando também as novas características das SFCs que estão sendo processadas na atualidade. Analisando os mapeamentos representados pela linha laranja, em que até 20000t o mapeamento foi realizado com o algoritmo ILP, e após, com o algoritmo ILPkr treinado em 20000t e 55000t, percebe-se que o novo treinamento do modelo é efetivo, e o algoritmo ILPkr retreinado consegue uma taxa de aceitação próxima ao algoritmo exato. Mais especificamente, e similarmente aos experimentos mostrados na Figura 7.14, o algoritmo ILPrk retreinado gerou uma taxa de aceitação maior que o algoritmo exato (Tabela I.7, Apêndice I), tendo uma maior taxa de aceitação no mapeamento das SFCs de ambas as classes envolvidas. Outras explicações sobre estes experimentos e uma análise de lucro são mostradas no Apêndice I.

Momento 3 (de 80000t a 110000t): nesta situação o provedor já processou algumas SFCs, e conhece as características das demandas que estão sendo processadas até o momento 80000t (classes 2 e 3). Dessa forma, pode-se aplicar o algoritmo ILPrk com o histórico já conhecido e atualizado em 55000.

Observando a Figura 7.11, mas considerando o intervalo de 80000t a 110000t, percebe-se que a após os 80000t, tal divergência não oscilou, indicando que, mesmo atendendo SFCs das 3 classes diferentes, não houve variabilidade das origens e destinos que efetuam as SFCs. Infere-se que após o período de 20000t, todos os nós de origem e destino já são conhecidos, e, a *priori*, o modelo ainda continua sendo válido. Entretanto, observando o atraso fim a fim, a métrica KL volta a oscilar, mostrando um crescimento em 80000t (Figura 7.12). Esse crescimento caracteriza alterações nas demandas de t_{dl}^{v} das SFCs processadas, refletindo o momento que o provedor começa a atender SFCs pertencentes a classe 1. A hipótese é que devido à alteração das demandas, outro novo *dataset* deve ser criado, o modelo de clusterização refeito, e novos SNs induzidos gerados. Deste modo, um bom momento para geração de um novo *dataset*, que contenha as características demandadas anteriormente, acrescido das atuais, é o instante 85000.

Análise da taxa de aceitação do momento 3 (Figura 7.14 de 80000t a 110000t): considere que a linha preta representa o mapeamento das SFCs durantes os 110000t com o algoritmo exato; a linha azul o experimento em que até 20000t os mapeamentos foram realizados com o algoritmo ILP, e após 20000t, com o algoritmo

ILPkr; a linha laranja representando o experimento em que até 20000t os mapeamentos foram realizados com o algoritmo ILP, e após, com o algoritmo ILPkr treinado em 20000t e 55000t; e por fim, a linha vermelha representando o experimento em que até 20000t os mapeamentos foram realizados com o algoritmo ILP, e após, com o algoritmo ILPkr treinado em 20000t, 55000t e 85000t.

Confrontando a Figura 7.14 e a Tabela I.8 (Apêndice I), percebe-se que, o algoritmo ILPrk, mesmo treinado em 20000 e retreinado em 55000, possui uma queda forte na taxa de aceitação após 85000. Essa queda é decorrente de tal abordagem utilizar um modelo treinado com um *dataset* que só possuía amostras de SFCs das classes 2 e 3. Neste caso, as SFCs das classes 2 e 3 se diferem muito da classe 1. Percebe-se mais uma vez que um novo modelo precisa ser gerado.

Analisando os mapeamentos representados pelo algoritmo ILPkr (linha vermelha), percebe-se que os retreinamentos são promissores, e o algoritmo consegue uma taxa de aceitação próxima ao *baseline*. Demais análises sobre estes experimentos, compreendendo uma análise de lucro, são expostas no Apêndice I.

7.5 Considerações Finais

O VNF-PC é um problema de virtualização de funções de rede cuja solução impacta diretamente nos lucros dos provedores e na qualidade da conexão para o usuário final. Como o VNF-PC é um problema NP-difícil, e as instâncias reais geralmente são grandes, sua resolução exata tradicional torna-se impraticável devido ao elevado tempo de execução. Neste Capítulo, apresentou-se uma abordagem baseada em IA, chamada Clusterização por Consumo de Recursos (ILPrk). Referida abordagem, integradamente à modelagem em ILP, resolve o problema VNF-PC em um ambiente *online* e com um baixo tempo computacional. Além das abordagens, apresentou-se uma técnica baseada na Divergência de Kullback-Leibler para se definir os momentos necessários de se aplicar uma reconfiguração do modelo de clusterização. Os experimentos mostraram que o retreinamento é necessário quando alguma mudança na distribuição de probabilidade das características demandadas é percebida.

Foram realizados extensos experimentos computacionais para verificar o desempenho das abordagens propostas. Inicialmente, foram propostos experimentos para definir bons valores para o parâmetro k presente no algoritmo K-*Means*. Foram realizados outros experimentos para comparar os ganhos e perdas da abordagem de clusterização proposta em relação ao modelo de otimização exato.

Similarmente aos experimentos do Capítulo 6, observa-se em todos os cenários

avaliados uma redução significativa do tempo de execução das abordagens de clusterização em comparação com o algoritmo que utiliza modelo de otimização tradicional. Outro ponto é que, apesar de reduzir o tempo computacional, as abordagens propostas não reduziram estatisticamente o lucro e a taxa de aceitação dos provedores em relação ao *baseline*. Ainda foram realizados experimentos para mostrar a importância de se manter o histórico de mapeamentos (base de dados) atualizado, neste caso, um histórico defasado pode comprometer o desempenho do referido algoritmo.

Uma vez validadas as abordagens de redução de dimensionalidade junto ao algoritmo de resolução exata, propõe-se no próximo Capítulo integrar estes estudos junto a heurística SGBL proposta no Capítulo 5. A pressuposição a ser investigada é de que em alguns momentos um tratamento com o algoritmo ILPrk pode ser aplicado em um tempo praticável, mas em outros não. Neste caso, um classificador baseado em AM supervisionado pode ser usado para definir em quais momentos pode-se aplicar um tratamento heurístico, e em quais, deve-se aplicar um tratamento exato.

Capítulo 8

Conclusões e Trabalhos Futuros

Neste Capítulo são feitas as considerações finais sobre este trabalho, e, posteriormente, apontadas direções de pesquisa que dão continuidade ao trabalho atual. Na Seção 8.1, são resumidas as hipóteses que foram investigadas, bem como os resultados obtidos. Na sequência, na Seção 8.2, são apontadas evoluções convergentes com aspectos relacionados ao conteúdo desta tese.

8.1 Observações Finais

Neste trabalho realizamos um estudo do problema de alocação de recursos em redes baseadas em tecnologias NFV, chamado NFV-RA. Em especial, foi dado foco na proposição de algoritmos para solucionar o problema VNF-PC. Referido estudo apresentou formulações, restrições clássicas da literatura, classes de serviços, algoritmos exatos, heurísticos, baseados em IA e hibridizações. Todos os experimentos/simulações computacionais envolvendo os algoritmos propostos foram realizados no mesmo ambiente computacional, fator que permitiu uma comparação direta entre as diversas abordagens.

Inicialmente foi feita uma contextualização da área de estudo com foco no problema VNF-PC, e em seguida foi feito um levantamento de trabalhos relacionados para nortear o desenvolvimento desta tese. Após estes estudos, foi proposta uma definição do problema, junto a uma formulação em ILP, que aborda restrições reais de mercado, como: largura de banda nos enlaces, capacidades de processamento e memória nos e servidores físicos e nas instâncias de NFs, roteamento encadeado, conservação de fluxo, atraso fim a fim gerado pelo processamento das NFs e pelo encadeamento. Tudo sob uma ótica de maximizar os lucros dos provedores.

Na sequência, a partir do modelo definido em ILP para o problema, foram estipuladas estratégias de reconfiguração. No Capítulo 4, referidas estratégias foram experimentadas com um algoritmo baseado na formulação do problema. As simulações realizadas buscaram responder às seguintes questões:

I Qual o impacto em termos de lucro as reconfigurações podem gerar?

O algoritmo ILP^{icp}, por reconfigurar todo o modelo, destaca-se com maiores lucros em relação aos outros algoritmos, justamente por gerar as maiores receitas advindas de uma maior taxa de aceitação; e um mapeamento mais conciso, com mais servidores físicos compartilhados. O algoritmo ILP^{i}, por efetuar somente a reconfiguração do tamanho das instâncias alocadas, consegue gerar uma boa relação lucro/receita, mas com um lucro final ligeiramente abaixo ao do algoritmo ILP^{ic}.

II Qual o tempo atrelado à aplicação de cada modelo de reconfiguração?

Percebe-se que o algoritmo ILP^{icp}, com uma reconfiguração completa dos mapeamentos das SFCs ativas, mesmo em topologias pequenas, é custoso computacionalmente e pode levar horas para ser executado, ação que certamente inviabiliza sua aplicação prática. Os algoritmos que efetuam uma reconfiguração parcial (ILP^{i} e ILP^{ic}) apesar de acarretarem em um tempo de execução mais baixo em até 99.99% em relação ao algoritmo ILP^{icp}, podem ser considerados lentos em alguns casos, pois tal processamento é dependente da quantidade de variáveis e restrições presentes no modelo.

III É necessário reconfigurar todo o modelo para se gerar boas soluções?

Apesar de garantidamente gerar uma solução ótima global, o algoritmo que efetua a reconfiguração completa dos roteamentos e instâncias de VNFs, não se mostrou promissor em relação ao algoritmo ILP^{i}, devido a um maior tempo computacional observado. Por outro lado, os demais algoritmos com reconfigurações parciais, apesar de uma queda no lucro de até 28.59%, são extremamente mais rápidos que o algoritmo ILP^{icp}, podendo ser aplicados em um tratamento *online* em alguns cenários específicos.

Visto a inviabilidade computacional de aplicar um tratamento exato em cenários maiores de rede, no Capítulo 5 foi proposta uma heurística, de rápida execução. As simulações com a heurística proposta buscam responder às questões:

I Qual o efeito gerado pela política de realizar os mapeamentos das NFs demandadas em regiões próximas à SFC?

Percebe-se que o movimento de trazer a instanciação das VNFs para uma região próxima à origem e destino de uma SFC, apesar de reduzir o compartilhamento de servidores, é benéfico tanto para os provedores, quanto para os usuários. Para os usuários, o mapeamento de redes tende a ficar mais conciso em uma região do SN, com menos saltos, e um atraso fim a fim menor, potencialmente melhorando os aspectos de QoE. Pelo lado dos provedores, existe uma redução do uso de enlaces físicos, aspecto que tende a mitigar problemas de fragmentação de recursos do SN.

II A heurística proposta é competitiva em termos de lucro gerado?

Sim, adotando como *baseline* o algoritmo exato ILP^i, devido a um menor uso de enlaces físicos, e consequente menor fragmentação do SN, a heurística proposta consegue uma taxa de aceitação melhor em alguns casos, e, assim, aumentar o lucro.

Na sequência deste trabalho, no Capítulo 6, foram propostas aplicações de métodos de clusterização, advindos de conceitos de IA, com o objetivo de reduzir o espaço de soluções processado pelos algoritmos. Nas simulações realizadas, procurou-se responder as questões:

I Com base na localização geográfica, é possível identificar a *priori clusters* de dispositivos de um SN aptos para mapear uma SFC?

Os experimentos mostraram que sim, que a política de geração de *clusters* utilizada conseguiu definir bons agrupamentos de recursos para mapear uma SFC. Os experimentos mostraram que a utilização desta abordagem não reduziu a taxa de aceitação, nem o lucro.

II Ao aplicar um algoritmo exato em um espaço de soluções diminuído, o tempo de tomada de decisão sobre o mapeamento de uma SFC no VNF-PC pode ser reduzido?

Sim, o pressuposto adotado, foi confirmado. O tempo de execução é diretamente relacionado à quantidade de dispositivos processados pelo algoritmo. Neste caso, como a clusterização agrupou os componentes do SN, o mapeamento das SFCs entrantes foi processado somente em certos grupos de dispositivos. Como exemplo, o algoritmo $ILPs_{12}$, implementado com o K-*Means* reduziu o tempo de processamento em 50% se comparado com o algoritmo sem clusterização.

III Dentre os algoritmos de IA que efetuam a clusterização, qual seria a melhor opção?

Foram experimentados os algoritmos: K-*Means*, *Hierarchical Clustering* e o DBSCAN. Dentre outros motivos, o uso do DBSCAN não é indicado por ele caracterizar alguns servidores físicos do SN, que estão longe fisicamente de outros servidores, como *outliers*. O algoritmo *Hierarchical Clustering* mostrou-se promissor, mas, o algoritmo K-*Means*, por gerar agrupamentos mais uniformes, e obteve um comportamento melhor, com um menor atraso fim a fim, sendo o mais indicado.

De forma a agregar valor ao algoritmo de clusterização apresentado no Capítulo 6, no Capítulo 7 foi proposto outro algoritmo que, além de clusterizar as regiões geográficas dos pontos de origem e destino de uma SFC, considera outros componentes nos agrupamentos, incluindo características do SN residual. Também foram investigados aspectos de mudanças que venham a ocorrer nas demandas das SFCs, e como essas mudanças podem alterar o desempenho do algoritmo. Dentre outros pontos, as simulações realizadas buscaram responder às seguintes questões:

I É possível clusterizar um histórico de recursos utilizados no passado, para prever quais recursos serão utilizados no mapeamento de uma nova SFC?

Sim, a abordagem aplicada é promissora, gerando bons *clusters* de recursos, chamados de $\overline{SNs}$ induzidos, para serem utilizados no mapeamento de uma SFC.

II SFCs com características parecidas tendem a utilizar recursos similares?

Com base nos experimentos, conclui-se que sim, SFCs similares podem utilizar recursos similares do SN para serem processadas, desde que o SN residual atual seja parecido com o utilizado na hora da formação dos $\overline{SNs}$ induzidos. Repare que, não é afirmado que esta é a melhor opção de mapeamento, visto que tal algoritmo processa somente uma parte do SN, mas sim que, os recursos disponibilizados pelos $\overline{SNs}$ induzidos são uma boa opção de componentes para atender a determinada SFC.

III Adicionar novas características na clusterização melhora o desempenho da abordagem proposta?

Similarmente aos experimentos do Capítulo 6, a abordagem de Clusterização por Consumo de Recursos não reduziu com significância estatística a um nível de 95% a taxa de aceitação, nem o lucro se comparado com o baseline (ILP^i). Conclui-se também que a abordagem de Clusterização por Localização Espacial gera um intervalo de confiança com uma maior amplitude e oscilação em relação ao lucro gerado pela abordagem de Clusterização por Consumo de Recursos. À vista disso, a adição de novas características na clusterização adicionou robustez ao algoritmo.

IV Ao se clusterizar as várias características, qual a importância da aplicação de técnicas de normalização e ponderação dos dados no ambiente avaliado?

Estes são aspectos fundamentais em algoritmos baseados em distâncias. Elas não são técnicas obrigatórias, contudo, impactam no desempenho do modelo. Neste caso, o objetivo é colocar os dados em uma mesma escala. Tal ação pode ser prejudicial se não bem avaliada, pois ao se normalizar, está sendo passado para o algoritmo que todas as características têm peso igual. O que em parte não é verdade, na hora da clusterização existem características relacionadas à viabilidade do problema, logo, estas devem ser ponderadas, evitando a geração de soluções inviáveis.

V Quantas SFCs o histórico deve possuir para gerar um *dataset* amplo, que abarque a variabilidade de recursos processados?

Não existe um número fixo, o que existe são indícios que o *dataset* contempla uma ampla gama de características que podem ser demandadas no momento do mapeamento. Para capturar estes indícios, foi proposto o uso da divergência de Kullback-Leibler. Com base nessa métrica é possível estimar se os recursos que estão sendo demandados no presente se diferenciam dos demandados no passado, ou não.

VI É necessário atualizar os *clusters* gerados?

Sim, mostrou-se que se o *dataset* que dá origem aos $\overline{SNs}$ induzidos não contemplar as características necessárias no momento de mapeamento, a taxa de rejeição pode subir. Neste caso, o *dataset* deve ser atualizado e o modelo retreinado.

VII Como saber quando retreinar e atualizar os $\overline{SNs}$ induzidos?

Neste aspecto, mais uma vez a divergência de Kullback-Leibler pode ser utilizada, agora para verificar quando as distribuições de probabilidades entre os recursos já processados, e os novos, são significativamente diferentes.

Sendo realizados e interpretados diversos experimentos, e geradas as respectivas e intrínsecas contribuições, são atingidos os objetivos propostos. Aproveitamos a experiência adquirida na realização deste trabalho para sugerir uma opção de continuidade para trabalhos futuros. Ela é apresentada na Seção 8.2. Por fim, esperamos que este trabalho possa contribuir com a comunidade acadêmica, especialmente com aqueles interessados na aplicação de métodos híbridos na resolução de problemas de alocação de recurso em redes baseadas em tecnologia NFV.

8.2 Pesquisas Futuras

8.2.1 Contextualização

Uma das métricas avaliadas e trabalhadas nesta tese é o atraso. Um dos elementos que influencia no aumento do atraso fim a fim é a distância física entre a origem e o destino do dado. A latência percebida com alguns experimentos realizados com a topologia de rede *Cogent* (Capítulo 5), e outras verificadas no mundo real com a ferramenta YuIP[1] (Tabela 8.1), sugere a necessidade de um estudo ao optar pelo lançamento de um novo NS. Por exemplo, ao hospedar um NS em um servidor virtual que esteja com o preço mais acessível, deve-se ponderar o tempo que o usuário levará para ter a sua requisição atendida. Tal atraso pode ser determinante na viabilidade do serviço.

[1]Ferramenta *online* para verificar a latência de uma origem para diversos servidores espalhados pelo mundo. Disponível em https://yuip.org/ma/test-latency

Tabela 8.1: Atraso fim a fim com origem em Belo Horizonte para diversos destinos

País	Região do servidor	Latência (ms)
Brasil	São Paulo	50
Estados Unidos	Califórnia	319
Reino Unido	Inglaterra	330
Alemanha	Hesse	340
Japão	Tokyo	446
Austrália	Nova Gales do Sul	485
Índia	Maarastra	506

Fonte: Elaborado pelo autor.

Adotando um NS oriundo de uma IoT atual, como carros autônomos ou sensores de cidades inteligentes, este é extremamente sensível ao atraso, e potencialmente, um processamento de dados em nuvens distribuídas globalmente pode inviabilizar sua aplicação prática (Chang et al., 2021; Ndikumana et al., 2021). Nesta perspectiva, um dos mitigadores do problema de alto atraso fim a fim é a implementação da chamada Computação de borda (*Edge Computing*, EC) (Li et al., 2020). EC se refere à inserção de uma camada de servidores intermediária entre o usuário e o *data center* de computação virtualizado em nuvem (Khan et al., 2019; Foukalas and Tziouvaras, 2022).

Em termos de aplicações, a EC vem sendo utilizada para oferecer suporte a NSs como: veículos autônomos, transmissão de vídeo, RV e RA (Chang et al., 2021; Grande and Beltrán, 2020; Kokkinos, 2022). O objetivo não é romper com o modelo de computação virtualizada em nuvem, mas sim agregar valor à estrutura, trabalhando com uma camada de servidores intermediária, próxima a origem dos dados, e caso necessário, o processamento em servidores da nuvem continuaria sendo realizado. A EC direciona os NS, antes atendidos exclusivamente em servidores em nuvem, para serem processados também na borda da rede.

A EC é caracterizada por uma alta largura de banda, latência ultrabaixa e acesso em tempo real às informações da rede (Khan et al., 2019). Neste contexto, a integração de novas tecnologias como EC e IoT são linhas de atuação recente, tanto no mercado, quanto na academia (Kokkinos, 2022; Tärneberg et al., 2022; Nayak et al., 2022). Em especial, o paradigma de EC vem ganhando espaço por atuar como um habilitador na implementação de tecnologias como 5G/6G e IoT, conectando instalações e serviços de computação virtualizada aos usuários finais (Khan et al., 2019). Por fim, a EC reduz os atrasos fim a fim, ação essencial em aplicações de IoT atuais. Ressalta-se que os dispositivos de borda normalmente possuem recursos de processamento e memória inferiores aos demais, *e.g.*, FPGA, GPU, e dispositivos de computação baseada em IoT como: *Arduino* e *Raspberry Pi* (Kokkinos, 2022).

Adicionalmente à EC, a *Computing Continuum* (CC) refere-se a uma infraestrutura usada por cargas de trabalho longas, com fluxos interruptos de aplicativos complexos, combinando processamento e computação de dados em tempo real (Tärneberg et al., 2022). Neste sentido, computação de alto desempenho, IA e redes 5G/6G são exemplos

de tecnologias que fazem uso do conceito de CC para prover aplicações como RA e RV. Essas tecnologias exigem uma integração fluida de recursos na borda e no núcleo da rede, abrindo caminho para dar suporte a fluxos de trabalho de NSs dinâmicos e orientados a dados (Rosendo et al., 2022; Tärneberg et al., 2022).

8.2.2 *Background*

Utilizando o EC, à medida que a nuvem se estende à borda, os serviços de computação podem ser espalhados por um conjunto de recursos computacionais que abrangem os dispositivos dos usuários, nuvem e infraestrutura intermediária (Chang et al., 2021; Kokkinos, 2022). Em comparação com a computação em nuvem, a computação de borda implanta serviços próximo ao usuário, resultando em latência ultrabaixa (Xie et al., 2020). Além disso, o aumento da capacidade de rede promete menores atrasos nas transferências de dados, com capacidade que pode ser usada para processar grandes quantidades de dados com tempos de resposta reduzidos (Kokkinos, 2022; Tärneberg et al., 2022). Essas grandes quantidades de dados são frequentemente processadas por meio de abordagens de IA, buscando extrair conhecimento a partir de dados brutos gerados por um conjunto heterogêneo de aplicativos (Chang et al., 2021). A IA tem evoluído como uma ferramenta para executar tarefas de aprendizado também na borda, usada logo após a produção dos dados, em vez de transferir dados para a nuvem. Embora existam estudos nessa direção, a literatura é relativamente escassa no tratamento eficiente de problemas de otimização em redes, sendo um campo de pesquisa a ser explorado em trabalhos futuros.

No contexto de aplicações de IoT, estas são baseadas na presença de sensores utilizados para coletar dados em um ambiente, como por exemplo: máquinas em um parque industrial, ou sensores em uma cidade inteligente (Babun et al., 2021; Kokkinos, 2022). Esses dados são normalmente enviados para servidores virtualizados em uma nuvem para serem processados. No entanto, um conjunto de requisitos e restrições deve ser atendido e o modelo exclusivo de computação em nuvem pode não ser o mais conveniente por vários fatores, como: restrições de disponibilidade de largura de banda, custos elevados, e atrasos, principalmente no contexto de aplicações de tempo real.

Algumas das limitações citadas anteriormente podem ser mitigadas com uma infraestrutura de EC (Khan et al., 2019). Neste caso é inserida uma camada de servidores localizados na borda da rede, alocada em uma posição intermediária entre os sensores e os servidores da nuvem, de modo a reduzir o volume de dados transferidos para a nuvem, atrasos e tempos de resposta para os diversos aplicativos. Neste meio, a virtualização das NFs e a EC são tecnologias promissoras para acelerar a implantação das redes 5G

e 6G, sendo fortemente exploradas pela IoT (Kokkinos, 2022). Definir a alocação de recursos para atender às requisições, surge como um desafio em redes habilitadas para EC por fatores como: estrutura hierárquica e geo-distribuída, tornando o posicionamento das funções dependente da localização; e processamento hierárquico, incorrendo em diferentes requisitos de atraso.

8.2.3 Problema

A integração de diferentes tecnologias e aplicações apresentam desafios diversos, por exemplo, uma integração IoT-EC-SN. Neste caso, o desenvolvimento o desenvolvimento de algoritmos de mapeamento inteligentes se torna fundamental para otimizar a utilização de recursos para cargas de trabalho atuais e futuras. No entanto, nestes ambientes, a quantidade de recursos disponíveis na borda da rede é escassa e, portanto, é necessário gerenciá-los de forma eficiente para lidar com as demandas cada vez maiores (Wang et al., 2021a). No problema de integração exemplificado, um conjunto de sensores é distribuído para coletar e disseminar dados aos usuários. O fluxo gerado por cada sensor tem como destino um nó de EC, que pode enviar para outro nó EC ou para um nó do SN (e vice-versa). Dessa forma, cria-se uma hierarquia *edge-cloud*, onde as tarefas de processamento e os dados podem se mover tanto horizontalmente, através da *edge*, quanto verticalmente, da *edge* para a *cloud*, de forma a se adaptar aos recursos residuais da rede. Neste sentido, o problema consiste em planejar e/ou operar esta rede em tempo real, para dar suporte aos aplicativos de IoT ou a algum outro NS suportado por esta rede, e atender restrições de capacidade, buscando garantir referências de QoS e de QoE.

Para esse fim, a otimização, simulação, técnicas de IA e heurísticas precisam ser exploradas para projetar um conjunto de algoritmos com o objetivo de fornecer diferentes compensações entre otimização e complexidade. Para tal, devem ser desenvolvidas formulações com objetivos que busquem otimizar a receita, custos, atrasos e energia. Nestes cenários, a IA também pode ser utilizada para reduzir as dimensões de um problema, viabilizando a sua solução em um tempo de execução menor. Ademais, os problemas de otimização podem explorar técnicas específicas para reconhecimento de propriedades significativas de um problema, e de acordo com os dados de entrada decidir a forma como o algoritmo será aplicado. Outra vertente é utilizar técnicas de análise de dados para auxiliar a tomada de decisão durante o processo de busca no espaço de soluções, evitando que uma busca fique retida em regiões de ótimos locais.

Referências

Abdelhamid, A. (2019). *Service Function Placement and Chaining in Network Function Virtualization Environments*. PhD thesis, Université de Bordeau.

Alleg, A., Ahmed, T., Mosbah, M., Riggio, R., and Boutaba, R. (2017). Delay-aware vnf placement and chaining based on a flexible resource allocation approach. In *2017 13th International Conference on Network and Service Management (CNSM)*, pages 1–7.

Amaral, P., Dinis, J., Pinto, P., Bernardo, L., Tavares, J., and Mamede, H. S. (2016). Machine learning in software defined networks: Data collection and traffic classification. In *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, pages 1–5.

Araujo, S., de Souza, F. S. H., and Mateus, G. R. (2018). Abordagens exata e heurística para o mapeamento de redes virtuais. Master's thesis, Universidade Federal de Minas Gerais, Departamento de Ciência da Computação - Instituto de Ciências Exatas, Belo Horizonte, Brasil.

Araujo, S., Souza, F., and Mateus, G. (2023). Modeling and analysis of different reconfiguration strategies for virtual network function placement and chaining with service classes identification. *IEEE Latin America Transactions*, 100.

Araujo, S. M. A., de Souza, F. S. H., and Mateus, G. R. (2022). A demand aware strategy for a machine learning approach to vnf-pc problem. In *2022 IEEE 11th International Conference on Cloud Networking (CloudNet)*, pages 211–219.

Araújo, S., Souza, F., and Mateus, G. (2021a). Uma abordagem heurística para o posicionamento e encadeamento de funções virtuais de rede em ambientes online. In *Anais do XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 784–797, Porto Alegre, RS, Brasil. SBC.

Araújo, S., Souza, F. S., and Mateus, G. R. (2020). Alocação de recursos para redes virtuais com seleção de método de resolução via aprendizado de máquina. In *Anais Estendidos do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 153–160, Porto Alegre, RS, Brasil. SBC.

Araújo, S. M., de Souza, F. S., and Mateus, G. R. (2021b). A hybrid optimization-machine learning approach for the vnf placement and chaining problem. *Computer Networks*, 199:108474.

Araújo, S. M. A., de Souza, F. S. H., and Mateus, G. R. (2019a). A composition selection mechanism for chaining and placement of virtual network functions. In *2019 15th International Conference on Network and Service Management (CNSM)*, pages 1–5.

Araújo, S. M. A., de Souza, F. S. H., and Mateus, G. R. (2019b). Flexible compositions for the virtual network function chain placement in online environments. In *31th International Teletraffic Congress (ITC 31)*, Budapest, Hungary.

Araújo, S. M. A., Souza, F. S. H. D., and Mateus, G. R. (2021c). On the analysis of online and periodic virtual network embedding in multi-domain environments. *International Journal of Networking and Virtual Organisations*, 24(1):1.

Arthur, D. and Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA. Society for Industrial and Applied Mathematics.

Askari, L., Musumeci, F., and Tornatore, M. (2019). Latency-aware traffic grooming for dynamic service chaining in metro networks. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pages 1–6.

Babbar, H., Rani, S., AlZubi, A. A., Singh, A., Nasser, N., and Ali, A. (2022). Role of network slicing in software defined networking for 5g: Use cases and future directions. *IEEE Wireless Communications*, 29(1):112–118.

Babun, L., Denney, K., Celik, Z. B., McDaniel, P., and Uluagac, A. S. (2021). A survey on iot platforms: Communication, security, and privacy perspectives. *Computer Networks*, 192:108040.

Barbi, R., Buravlev, V., Mezzina, C. A., and Schiavoni, V. (2017). Block placement strategies for fault-resilient distributed tuple spaces: An experimental study. In Chen, L. Y. and Reiser, H. P., editors, *Distributed Applications and Interoperable Systems*, pages 67–82, Cham. Springer International Publishing.

Bari, F., Chowdhury, S. R., Ahmed, R., Boutaba, R., and Duarte, O. C. M. B. (2016). Orchestrating virtualized network functions. *IEEE Transactions on Network and Service Management*, 13(4):725–739.

Bari, M. F., Chowdhury, S. R., Ahmed, R., and Boutaba, R. (2015). On orchestrating virtual network functions. In *2015 11th International Conference on Network and Service Management (CNSM)*, pages 50–56.

Bari, M. F., Chowdhury, S. R., and Boutaba, R. (2019). Esso: An energy smart service function chain orchestrator. *IEEE Transactions on Network and Service Management*, 16(4):1345–1359.

Baumgartner, A., Reddy, V. S., and Bauschert, T. (2015). Mobile core network virtualization: A model for combined virtual core network function placement and topology optimization. In *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, pages 1–9.

Beck, M. T. and Botero, J. F. (2015). Coordinated Allocation of Service Function Chains. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6.

Bhamare, D., Jain, R., Samaka, M., and Erbad, A. (2016). A survey on service function chaining. *J. Netw. Comput. Appl.*, 75(C):138–155.

Bhamare, D., Samaka, M., Erbad, A., Jain, R., Gupta, L., and Chan, H. A. (2017). Optimal virtual network function placement in multi-cloud service function chaining architecture. *Computer Communications*, 102:1–16.

Blenk, A., Kalmbach, P., van der Smagt, P., and Kellerer, W. (2016). Boost online virtual network embedding: Using neural networks for admission control. In *2016 12th International Conference on Network and Service Management (CNSM)*, pages 10–18.

Boutaba, R., Salahuddin, M., Limam, N., Ayoubi, S., Shahriar, N., Estrada-Solano, F., and Caicedo Rendon, O. (2018). A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities. *Journal of Internet Services and Applications*, 9.

Breitgand, D., Eisenberg, V., Naaman, N., Rozenbaum, N., and Weit, A. (2021). Toward true cloud native nfv mano. In *2021 12th International Conference on Network of the Future (NoF)*, pages 1–5.

Brown, G. (2017). 4G & 5G-Ready Network Slicing: A Heavy Reading white paper produced for Affirmed Networks. Relatório técnico.

Chang, Z., Liu, S., Xiong, X., Cai, Z., and Tu, G. (2021). A survey of recent advances in edge-computing-powered artificial intelligence of things. *IEEE Internet of Things Journal*, 8(18):13849–13875.

Chen, J., Chen, J., Hu, R., and Zhang, H. (2019). Clusvnfi: A hierarchical clustering-based approach for solving vnfi dilemma in nfv orchestration. *IEEE Access*, 7:173257–173272.

Chen, R. and Zhao, J. (2022). Scalable and flexible traffic steering for service function chains. *IEEE Transactions on Network and Service Management*, 19(3):2048–2062.

Chen, W., Wang, Z., Zhang, H., Yin, X., Shi, X., and Sun, L. (2020). Poster abstract: Joint optimization of service function chain elastic scaling and routing. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1306–1307.

Chen, X., Zhou, J., and Wei, S. (2022). Sfc-ho: Reliable layered service function chaining. *IEEE Access*, 10:106352–106368.

Cheng, R. . and Chang, C. . (1997). Neural-network connection-admission control for atm networks. *IEE Proceedings - Communications*, 144(2):93–98.

Chowdhury, M., Rahman, M. R., and Boutaba, R. (2012). Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Transactions on Networking*, 20(1):206–219.

Chowdhury, N. M. M. K., Rahman, M. R., and Boutaba, R. (2009). Virtual network embedding with coordinated node and link mapping. In *IEEE INFOCOM 2009*, pages 783–791.

Cohen, R., Lewin-Eytan, L., Naor, J. S., and Raz, D. (2015). Near optimal placement of virtual network functions. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 1346–1354.

Comaneci, D. and Dobre, C. (2018). Securing networks using sdn and machine learning. In *2018 IEEE International Conference on Computational Science and Engineering (CSE)*, pages 194–200.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*. The MIT Press, 2 edition.

Dalgkitsis, A., Garrido, L. A., Rezazadeh, F., Chergui, H., Ramantas, K., Vardakas, J. S., and Verikoukis, C. (2022). Sche2ma: Scalable, energy-aware, multidomain orchestration for beyond-5g ignoreurllc services. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–11.

Das, S. and Nene, M. J. (2017). A survey on types of machine learning techniques in intrusion prevention systems. In *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 2296–2299.

Deezer (2021). Deezer audio quality. [Online];Disponível em https://support.deezer.com/hc/en-gb/articles/115003865685-Deezer-Audio-Quality.

Driver, H. and Kroeber, A. (1932). Quantitative expression of cultural relationships. *University of California Publications in American Archaeology and Ethnology*, 31(4):211–256.

Elfatih, N. M., Hasan, M. K., Kamal, Z., Gupta, D., Saeed, R. A., Ali, E. S., and Hosain, M. S. (2022). Internet of vehicle's resource management in 5g networks using ai technologies: Current status and trends. *IET Communications*, 16(5):400–420.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press.

ETSI (2014). Network Functions Virtualisation (NFV); Service Quality Metrics. Relatório técnico.

Feng, H., Llorca, J., Tulino, A. M., Raz, D., and Molisch, A. F. (2017). Approximation algorithms for the nfv service distribution problem. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9.

Feo, T. A. and Resende, M. G. (1995). Greedy randomized adaptive search procedure. *Journal of Global Optimization*, 6:109–133.

Fischer, A., Bhamare, D., and Kassler, A. (2019). On the construction of optimal embedding problems for delay-sensitive service function chains. In *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–10.

Fischer, A., Botero, J. F., Beck, M. T., de Meer, H., and Hesselbach, X. (2013). Virtual Network Embedding: A Survey. *IEEE Communications Surveys Tutorials*, 15(4):1888–1906.

Foukalas, F. and Tziouvaras, A. (2022). Qoe-aware edge computing through service function chaining. *IEEE Internet Computing*, 26(02):53–60.

Foukas, X., Patounas, G., Elmokashfi, A., and Marina, M. K. (2017). Network Slicing in 5G: Survey and Challenges. *IEEE Communications Magazine*, 55(5):94–100.

Gao, M., Addis, B., Bouet, M., and Secci, S. (2018). Optimal orchestration of virtual network functions. *Computer Networks*, 142:108 – 127.

Gebremariam, A. A., Usman, M., and Qaraqe, M. (2019). Applications of artificial intelligence and machine learning in the area of sdn and nfv: A survey. In *2019 16th International Multi-Conference on Systems, Signals & Devices (SSD)*, pages 545–549.

Ghaznavi, M., Shahriar, N., Kamali, S., Ahmed, R., and Boutaba, R. (2017). Distributed service function chaining. *IEEE Journal on Selected Areas in Communications*, 35(11):2479–2489.

Gil-Herrera, J. and Botero, J. F. (2017). A scalable metaheuristic for service function chain composition. In *2017 IEEE 9th Latin-American Conference on Communications (LATINCOM)*, pages 1–6.

Goemans, M. X. and Williamson, D. P. (1996). *The Primal-Dual Method for Approximation Algorithms and Its Application to Network Design Problems*, page 144–191. PWS Publishing Co., USA.

Gong, J., Simeone, O., and Kang, J. (2021). Bayesian variational federated learning and unlearning in decentralized networks. *CoRR*, abs/2104.03834.

Google (2021a). Bandwidth, data usage, and stream quality: Google Support. [online] Disponível em https://support.google.com/stadia/answer/9607891?hl=en.

Google (2021b). Requisitos do sistema. [Online]; accessed 01-Julho-2021,Disponível em https://support.google.com/youtube/answer/78358?hl=pt-BR.

Grande, E. and Beltrán, M. (2020). Edge-centric delegation of authorization for constrained devices in the internet of things. *Computer Communications*, 160.

Gu, Y., Hu, Y., Ding, Y., Lu, J., and Xie, J. (2019). Elastic virtual network function orchestration policy based on workload prediction. *IEEE Access*, 7:96868–96878.

Gupta, A., Kleinberg, J., Kumar, A., Rastogi, R., and Yener, B. (2001). Provisioning a virtual private network: A network design problem for multicommodity flow. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, STOC '01, page 389–398, New York, NY, USA. Association for Computing Machinery.

Gupta, L., Samaka, M., Jain, R., Erbad, A., Bhamare, D., and Chan, H. A. (2017). Fault and performance management in multi-cloud based nfv using shallow and deep predictive structures. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–8.

H., A. J., Pathan, M. P. M. R. A.-S. K., and Deris, M. M. (2012). *Internet and distributed computing advancements : theoretical frameworks and practical applications*. Information Science Reference Hershey, PA.

Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449 – 467.

Hart, J. and Shogan, A. W. (1987). Semi-greedy heuristics: An empirical study. *Operations Research Letters*, 6(3):107–114.

He, Y., Zhang, X., Xia, Z., Liu, Y., Sood, K., and Yu, S. (2022). Joint optimization of service chain graph design and mapping in nfv-enabled networks. *Computer Networks*, 202:108626.

Herrera, J. G. and Botero, J. F. (2016). Resource Allocation in NFV: A Comprehensive Survey. *IEEE Transactions on Network and Service Management*, 13(3):518–532.

Hmaity, A., Savi, M., Musumeci, F., Tornatore, M., and Pattavina, A. (2016). Virtual network function placement for resilient service chain provisioning. In *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pages 245–252.

Hmaity, A., Savi, M., Musumeci, F., Tornatore, M., and Pattavina, A. (2017). Protection strategies for virtual network functions placement and service chains provisioning. *Networks*, 70(4):373–387.

Holland, P. W. (1986). Statistics and causal inference. *Journal of the American Statistical Association*, 81(396):945–960.

Holub, J., Wallbaum, M., Smith, N., and Avetisyan, H. (2018). Analysis of the dependency of call duration on the quality of voip calls. *IEEE Wireless Communications Letters*, 7(4):638–641.

Houidi, O., Soualah, O., Louati, W., Mechtri, M., Zeghlache, D., and Kamoun, F. (2017). An efficient algorithm for virtual network function scaling. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–7.

Hu, D., Hong, P., and Chen, Y. (2017). Fadm: Ddos flooding attack detection and mitigation system in software-defined networking. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–7.

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323.

Jia, Y., Wu, C., Li, Z., Le, F., and Liu, A. (2018). Online scaling of nfv service chains across geo-distributed datacenters. *IEEE/ACM Transactions on Networking*, 26(2):699–710.

Jiang, J., Lan, T., Ha, S., Chen, M., and Chiang, M. (2012). Joint vm placement and routing for data center traffic engineering. *Proceedings - IEEE INFOCOM*, pages 2876–2880.

Jin, X. and Han, J. (2010). *K-Means Clustering*, pages 563–564. Springer US, Boston, MA.

Karampasi, A., Kakkos, I., Miloulis, S.-T., Zorzos, I., Dimitrakopoulos, G. N., Gkiatis, K., Asvestas, P., and Matsopoulos, G. (2020). A machine learning fmri approach in the diagnosis of autism. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 3628–3631.

Karp, R. M. (1972). *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA.

Kaufman, L. and Rousseeuw, P. J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, Inc.

Khan, W. Z., Ahmed, E., Hakak, S., Yaqoob, I., and Ahmed, A. (2019). Edge computing: A survey. *Future Generation Computer Systems*, 97:219–235.

Khebbache, S., Hadji, M., and Zeghlache, D. (2017). Virtualized network functions chaining and routing algorithms. *Computer Networks*, 114:95 – 110.

Khoshkholghi, M. A., Taheri, J., Bhamare, D., and Kassler, A. (2019). Optimized service chain placement using genetic algorithm. In *2019 IEEE Conference on Network Softwarization (NetSoft)*, pages 472–479.

Kim, S. I. and Kim, H. S. (2017). A research on dynamic service function chaining based on reinforcement learning using resource usage. In *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 582–586.

Knight, S., Nguyen, H., Falkner, N., Bowden, R., and Roughan, M. (2011). The internet topology zoo. *Selected Areas in Communications, IEEE Journal on*, 29(9):1765 –1775.

Kokkinos, P. (2022). Towards the realization of converged cloud, edge and networking infrastructures in smart megacities. In *IEEE Symposium on Computers and Communications*.

Kuang, H., Qiu, Y., Li, R., and Liu, X. (2018). A hierarchical k-means algorithm for controller placement in sdn-based wan architecture. In *2018 10th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, pages 263–267.

Kullback, S. and Leibler, R. A. (1951). On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86.

Kuo, T., Liou, B., Lin, K. C., and Tsai, M. (2016). Deploying chains of virtual network functions: On the relation between link and server usage. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9.

Kutner, M., Nachtsheim, C., Neter, J., and Li, W. (2004). *Applied Linear Statistical Models*. Information Science Reference Hershey, PA.

Laghrissi, A. and Taleb, T. (2018). A Survey on the Placement of Virtual Resources and Virtual Network Functions. *IEEE Communications Surveys Tutorials*, pages 1–1.

Lange, S., Grigorjew, A., Zinner, T., Tran-Gia, P., and Jarschel, M. (2017). A multi-objective heuristic for the optimization of virtual network function chain placement. In *2017 29th International Teletraffic Congress (ITC 29)*, volume 1, pages 152–160.

Le, L., Lin, B. P., Tung, L., and Sinh, D. (2018). Sdn/nfv, machine learning, and big data driven network slicing for 5g. In *2018 IEEE 5G World Forum (5GWF)*, pages 20–25.

Leivadeas, A., Falkner, M., Lambadaris, I., and Kesidis, G. (2017). Optimal virtualized network function allocation for an sdn enabled cloud. *Computer Standards & Interfaces*, 54:266 – 278. SI: Standardization SDN&NFV.

Leivadeas, A., Kesidis, G., Ibnkahla, M., and Lambadaris, I. (2019). VNF Placement Optimization at the Edge and Cloud. *Future Internet*, 11(3).

Li, J., Liang, W., Huang, M., and Jia, X. (2020). Reliability-aware network service provisioning in mobile edge-cloud networks. *IEEE Transactions on Parallel and Distributed Systems*, pages 1–1.

Li, Y. and Tu, W. (2020). Traffic modelling for iot networks: A survey. In *Proceedings of the 2020 10th International Conference on Information Communication and Management*, ICICM 2020, page 4–9, New York, NY, USA. Association for Computing Machinery.

Liu, J., Lu, W., Zhou, F., Lu, P., and Zhu, Z. (2017). On dynamic service function chain deployment and readjustment. *IEEE Transactions on Network and Service Management*, 14(3):543–553.

Luizelli, M. C., Bays, L. R., Buriol, L. S., Barcellos, M. P., and Gaspary, L. P. (2015). Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 98–106.

Luizelli, M. C., da Costa Cordeiro, W. L., Buriol, L. S., and Gaspary, L. P. (2017). A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining. *Comput. Commun.*, 102(C):67–77.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif. University of California Press.

Maity, I., Vu, T. X., Chatzinotas, S., and Minardi, M. (2022). D-vine: Dynamic virtual network embedding in non-terrestrial networks. In *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 166–171.

Mehraghdam, S., Keller, M., and Karl, H. (2014). Specifying and placing chains of virtual network functions. In *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*, pages 7–13.

Mijumbi, R., Gorricho, J., Serrat, J., Claeys, M., De Turck, F., and Latré, S. (2014). Design and evaluation of learning algorithms for dynamic resource management in virtual networks. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–9.

Mijumbi, R., Hasija, S., Davy, S., Davy, A., Jennings, B., and Boutaba, R. (2017). Topology-Aware Prediction of Virtual Network Function Resource Requirements. *IEEE Transactions on Network and Service Management*, 14(1):106–120.

Mijumbi, R., Serrat, J., Gorricho, J. L., Bouten, N., Turck, F. D., and Boutaba, R. (2016). Network Function Virtualization: State-of-the-Art and Research Challenges. *IEEE Communications Surveys Tutorials*, 18(1):236–262.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computer and Operations Research*, 24(11):1097–1100.

Najafzadeh, H., Roberts, P., Penmatcha, R., Clark, J., Soward, E., Boyd, R., Lulla, A., Arsanjani, A., Porcelli, G. A., Muppala, S., Eigenbrode, S., Nguyen, P., Carlson, B., Potter, B., Leirer, A., and Sobek, J. (2021). Machine Learning Lens: AWS Well-Architected Framework. Relatório técnico. [online] Disponível em https://docs.aws.amazon.com/wellarchitected/latest/machine-learning-lens/wellarchitected-machine-learning-lens.pdf#machine-learning-lens.

Nalewajski, R. F. (2006). 3 - entropy, information and communication channels. In Nalewajski, R. F., editor, *Information Theory of Molecular Systems*, pages 56–90. Elsevier Science, Amsterdam.

Nayak, S., Patgiri, R., Waikhom, L., and Ahmed, A. (2022). A review on edge analytics: Issues, challenges, opportunities, promises, future directions, and applications. *Digital Communications and Networks*.

Ndikumana, A., Tran, N. H., Kim, D. H., Kim, K. T., and Hong, C. S. (2021). Deep learning based caching for self-driving cars in multi-access edge computing. *IEEE Transactions on Intelligent Transportation Systems*, 22(5):2862–2877.

Nguyen, D. T., Pham, C., Nguyen, K. K., and Cheriet, M. (2020). Placement and chaining for run-time iot service deployment in edge-cloud. *IEEE Transactions on Network and Service Management*, 17(1):459–472.

Nguyen, P., Chiu, T., Vedati, B. R. J. C., and Nalik, J. (2008). Blue coat and sap: Network infrastructure enhancements for successful enterprise soa deployments. [Online] Disponível em https://archive.sap.com/documents/docs/DOC-17548.

Nguyen, V., Brunstrom, A., Grinnemo, K., and Taheri, J. (2017). Sdn/nfv-based mobile packet core network architectures: A survey. *IEEE Communications Surveys Tutorials*, 19(3):1567–1602.

Nokia (2016). White Paper: Dynamic end-to-end network slicing for 5G. Relatório técnico. [online] Disponível em http://www.hit.bme.hu/~jakab/edu/litr/5G/NOKIA_dynamic_network_slicing_WP.pdf.

Ocampo, A. F., Gil-Herrera, J., Isolani, P. H., Neves, M. C., Botero, J. F., Latré, S., Zambenedetti, L., Barcellos, M. P., and Gaspary, L. P. (2017). Optimal Service Function Chain Composition in Network Functions Virtualization. In Tuncer, D., Koch, R., Badonnel, R., and Stiller, B., editors, *Security of Networks and Services in an All-Connected World*, pages 62–76, Cham. Springer International Publishing.

Open Networking Foundation (2019). [online] Disponível em https://www.opennetworking.org.

Ouni, A., Kessentini, M., Inoue, K., and Cinneide, M. O. (2017). Search-based web service antipatterns detection. *IEEE Transactions on Services Computing*, 10(4):603–617.

Paschos, G. S., Abdullah, M. A., and Vassilaras, S. (2018). Network slicing with splittable flows is hard. In *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1788–1793.

Pearson, R., Zylkin, T., Schwaber, J., and Gonye, G. (2004). Quantitative evaluation of clustering results using computational negative controls. *Proceedings of the 2004 SIAM International Conference on Data Mining*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikitlearn-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pham, C., Tran, N. H., Ren, S., Saad, W., and Hong, C. S. (2017). Traffic-aware and energy-efficient vnf placement for service chaining: Joint sampling and matching approach. *IEEE Transactions on Services Computing*, pages 1–1.

Pham, T.-M. (2022). Optimizing service function chaining migration with explicit dynamic path. *IEEE Access*, 10:16992–17002.

Research And Markets (2020). Network function virtualization (nfv) market by component, virtualized network function, application, end user—global forecast to 2027. [online] Disponível em https://www.researchandmarkets.com/reports/5157301/network-function-virtualization-nfv-market-by.

Resende, M. and Ribeiro, C. C. (2016). *Optimization by GRASP*. Springer-Verlag New York, 1 edition.

Resende, M. C. and Ribeiro, C. C. (2010). Grasp : Greedy randomized adaptive search procedures. In *Search Methodologies - Introductory Tutorials in Optimization and Decision Support Techniques*, pages 287–312. Edmund K. BurkeGraham Kendall.

Ricci, R., Alfeld, C., and Lepreau, J. (2003). A solver for the network testbed mapping problem. *SIGCOMM Comput. Commun. Rev.*, 33(2):65–81.

Roig, J. S. P., Gutierrez-Estevez, D. M., and Gündüz, D. (2019). Management and orchestration of virtual network functions via deep reinforcement learning. *IEEE Journal on Selected Areas in Communications*, pages 1–1.

Rosendo, D., Costan, A., Valduriez, P., and Antoniu, G. (2022). Distributed intelligence on the edge-to-cloud continuum: A systematic literature review. *Journal of Parallel and Distributed Computing*, 166.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

Ruiz-Perez, L., Durán, R. J., de Miguel, I., Khodashenas, P. S., Pedreno-Manresa, J., Merayo, N., Aguado, J. C., Pavón-Mariño, P., Siddiqui, S., Mata, J., Fernández, P., Lorenzo, R. M., and Abril, E. J. (2018). Genetic algorithm for effective service mapping in the optical backhaul of 5g networks. In *2018 20th International Conference on Transparent Optical Networks (ICTON), Bucharest, Romania, July 1-5, 2018*, pages 1–4. IEEE.

Sahhaf, S., Tavernier, W., Rost, M., Schmid, S., Colle, D., Pickavet, M., and Demeester, P. (2015). Network service chaining with optimized network function embedding supporting service decompositions. *Computer Networks*, 93:492–505. Cloud Networking and Communications II.

Sakamoto, Y., Fukui, K.-I., Gama, J., Nicklas, D., Moriyama, K., and Numao, M. (2015). Concept drift detection with clustering via statistical change detection methods. In *2015 Seventh International Conference on Knowledge and Systems Engineering (KSE)*, pages 37–42.

Savi, M., Hmaity, A., Verticale, G., Höst, S., and Tornatore, M. (2016). To distribute or not to distribute? impact of latency on virtual network function distribution at the edge of fmc networks. In *2016 18th International Conference on Transparent Optical Networks (ICTON)*, pages 1–4.

Savi, M., Tornatore, M., and Verticale, G. (2015). Impact of processing costs on service chain placement in network functions virtualization. In *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, pages 191–197.

Sharma, G. P., Tavernier, W., Colle, D., and Pickavet, M. (2020). Vnf-aapc: Accelerator-aware vnf placement and chaining. *Computer Networks*, 177:107329.

Shi, R., Zhang, J., Chu, W., Bao, Q., Jin, X., Gong, C., Zhu, Q., Yu, C., and Rosenberg, S. (2015). Mdp and machine learning-based cost-optimization of dynamic resource allocation for network function virtualization. In *2015 IEEE International Conference on Services Computing*, pages 65–73.

Shinde, H. and Sankhe, A. (2017). Comparison of enhanced dbscan algorithms: A review. In *International Journal of Engineering Research & technology (IJERT)*, volume 5.

Singh, P. and Meshram, P. A. (2017). Survey of density based clustering algorithms and its variants. In *2017 International Conference on Inventive Computing and Informatics (ICICI)*, pages 920–926.

Singhal, A., Buckley, C., and Mitra, M. (1996). Pivoted document length normalization. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '96, page 21–29, New York, NY, USA. Association for Computing Machinery.

Sinnott, R. W. (1984). Virtues of the haversine. *Sky and Telescope*, 68(2):158–159.

Sobh, T. and Fakhry, M. (2014). Evaluating web services functionality and performance. *International Journal of Information Technology and Computer Science*, 6:18–27.

Song, X., Li, W., Ma, D., Wu, Y., and Ji, D. (2018). An enhanced clustering-based method for determining time-of-day breakpoints through process optimization. *IEEE Access*, 6:29241–29253.

Subramanya, T. and Riggio, R. (2019). Machine learning-driven scaling and placement of virtual network functions at the network edges. In *2019 IEEE Conference on Network Softwarization (NetSoft)*, pages 414–422.

Sun, J., Huang, G., Sun, G., Yu, H., Kumar, A., and Chang, V. (2018). A q-learning-based approach for deploying dynamic service function chains. *Symmetry*, 10:646.

Sun, J., Zhang, Y., Liu, F., Wang, H., Xu, X., and Li, Y. (2022). A survey on the placement of virtual network functions. *Journal of Network and Computer Applications*, 202:103361.

Sun, Q., Lu, P., Lu, W., and Zhu, Z. (2016). Forecast-assisted nfv service chain deployment based on affiliation-aware vnf placement. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6.

Swami, R., Dave, M., and Ranga, V. (2019). Software-defined networking-based ddos defense mechanisms. *ACM Computing Surveys*, 52(2):28:1–28:36.

Tanenbaum, A. S. and Wetherall, D. (2011). *Computer networks, 5th Edition*. Pearson.

Tariq, M. A., Farooq, M. U., Zeeshan, M., Hassan, A., and Akhunzada, A. (2022). Fipam: Fuzzy inference based placement and migration approach for nfv-based iots. *IEEE Transactions on Network and Service Management*, pages 1–1.

Thorndike, R. L. (1953). Who belongs in the family? *Psychometrika*, 18:267–276.

Tärneberg, W., Fitzgerald, E., Bhuyan, M., Townend, P., Årzén, K.-E., Östberg, P.-O., Elmroth, E., Eker, J., Tufvesson, F., and Kihl, M. (2022). The 6g computing continuum (6gcc): Meeting the 6g computing challenges. In *2022 1st International Conference on 6G Networking (6GNet)*, pages 1–5.

Vassilaras, S., Gkatzikis, L., Liakopoulos, N., Stiakogiannakis, I. N., Qi, M., Shi, L., Liu, L., Debbah, M., and Paschos, G. S. (2017). The algorithmic aspects of network slicing. *IEEE Communications Magazine*, 55(8):112–119.

Wang, L., Dolati, M., and Ghaderi, M. (2021a). Change: Delay-aware service function chain orchestration at the edge. In *2021 IEEE 5th International Conference on Fog and Edge Computing (ICFEC)*, pages 19–28.

Wang, M., Cheng, B., Li, B., and Chen, J. (2019). Service function chain composition and mapping in nfv-enabled networks. In *2019 IEEE World Congress on Services (SERVICES)*, volume 2642-939X, pages 331–334.

Wang, X., Xu, B., and and, F. J. (2021b). An efficient service function chains orchestration algorithm for mobile edge computing. *KSII Transactions on Internet and Information Systems*, 15(12):4364–4384.

Wei, F., Qin, S., Feng, G., Sun, Y., Wang, J., and Liang, Y.-C. (2022a). Hybrid model-data driven network slice reconfiguration by exploiting prediction interval and robust optimization. *IEEE Transactions on Network and Service Management*, 19(2):1426–1441.

Wei, S., Zhou, J., and Chen, S. (2022b). Delay-aware multipath parallel sfc orchestration. *IEEE Access*, 10:120035–120055.

White Paper NFV (2012). Network functions virtualisation: An introduction, benefits, enablers, challenges & call for action. issue 1. Relatório técnico. [online] Disponível em https://www.bibsonomy.org/bibtex/2af88fcf004d54bbcc23d951d64b3f82d/bazza030.

Xiao, Y., Kaku, I., Zhao, Q., and Zhang, R. (2011). A reduced variable neighborhood search algorithm for uncapacitated multilevel lot-sizing problems. *European Journal of Operational Research*, 214(2):223 – 231.

Xie, A., Huang, H., Wang, X., Qian, Z., and Lu, S. (2020). Online vnf chain deployment on resource-limited edges by exploiting peer edge devices. *Computer Networks*, 170:107069.

Xie, J., Yu, F. R., Huang, T., Xie, R., Liu, J., Wang, C., and Liu, Y. (2019). A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges. *IEEE Communications Surveys Tutorials*, 21(1):393–430.

Yi, B., Wang, X., Li, K., k. Das, S., and Huang, M. (2018). A comprehensive survey of network function virtualization. *Computer Networks*, 133:212 – 262.

Yousaf, F. Z., Bredel, M., Schaller, S., and Schneider, F. (2017). Nfv and sdn—key technology enablers for 5g networks. *IEEE Journal on Selected Areas in Communications*, 35(11):2468–2478.

Zhang, J., Zeng, D., Gu, L., Yao, H., and Xiong, M. (2017). Joint optimization of virtual function migration and rule update in software defined nfv networks. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–5.

Zhang, X., Wu, C., Li, Z., and Lau, F. C. (2017). Proactive vnf provisioning with multi-timescale cloud resources: Fusing online learning and online optimization. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9.

Zhao, Y., Li, Y., Zhang, X., Geng, G., Zhang, W., and Sun, Y. (2019). A survey of networking applications applying the software defined networking concept based on machine learning. *IEEE Access*, 7:95397–95417.

Zhu, Y. and Ammar, M. (2006). Algorithms for assigning substrate network resources to virtual network components. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, pages 1–12.

Zoure, M., Ahmed, T., and Réveillère, L. (2022). Network services anomalies in nfv: Survey, taxonomy, and verification methods. *IEEE Transactions on Network and Service Management*, 19(2):1567–1584.

Apêndice A

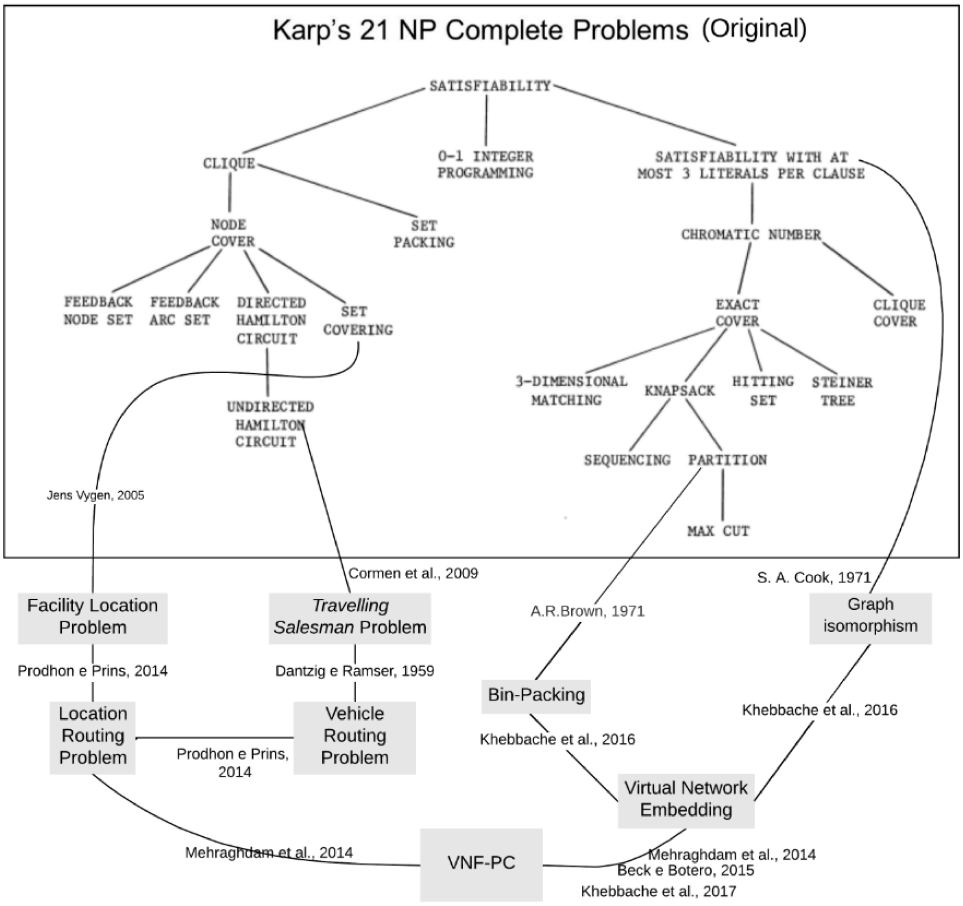
Reduções Polinomiais: VNF-PC

Muitos problemas de otimização combinatória são improváveis de serem resolvidos eficientemente em um tempo computacional baixo, sendo pertencentes a uma classe de problemas computacionais chamada NP-difícil (Goemans and Williamson, 1996). Cormen et al. (2001) corroboram e completam tal declaração, afirmando que em um problema de otimização podem existir várias soluções factíveis, cada uma com um valor associado. Neste caso, deseja-se encontrar a solução factível com o melhor valor dentre as soluções investigadas. Por outro lado, em problemas de decisão, não se está interessado na melhor solução factível, e sim em uma simples resposta do tipo sim ou não. Pode-se transformar um determinado problema de otimização em um problema de decisão, impondo-lhe um limite de valor a ser otimizado. No VFC-PC, por exemplo, ao invés de minimizar o custo de atendimento de uma SFC em um problema de otimização, pode-se propor a pergunta: é possível mapear determinada SFC com um determinado custo? Transformando-o assim em um problema de decisão. Deste modo, a relação entre um problema de otimização e seu respectivo problema de decisão pode ser usada para mostrar a dificuldade de sua resolução, *i.e.*, se um problema de otimização é de fácil resolução, seu respectivo problema de decisão também o é; e se um problema de otimização é de difícil resolução, seu respectivo problema de decisão também o é.

As transformações (reduções) de tempo polinomial, referidas pelo símbolo $\propto$, servem para mostrar formalmente que um problema é pelo menos tão difícil quanto algum outro (Cormen et al., 2001). Assim, um problema pertence à classe NP-difícil[1] se ele é pelo menos tão difícil quanto qualquer outro problema de tempo polinomial não determinístico (*Nondeterministic Polynomial time*, NP). A Figura A.1 mostra as 21 transformações iniciais para problemas NP-completos, proposta originalmente por Karp (1972), acrescentada das indicações de reduções polinomiais que mostram que o problema VNF-PC também o é.

[1]Um problema é NP-difícil se e somente se o problema de satisfabilidade, chamado SAT, é redutível em tempo polinomial a ele (Cormen et al., 2001), no caso $SAT \propto VNF\text{-}PC$.

Figura A.1: Transformações dos 21 problemas NP-completos original de Karp (1972), acrescentada das sucessivas reduções polinomiais até o problema VNF-PC



Fonte: Karp (1972).

Khebbache et al. (2017) provam que o VNF-PC é um problema NP-difícil por sua semelhança com o VNE. A prova que o VNF-PC pertence à classe NP-difícil pode ser resumida em dois casos:

I Caso simplificado: o mapeamento de uma requisição sem arestas é uma ação equivalente à alocação ótima de um nó (virtual), com demandas de recursos sobre os nós físicos do SN, que possuem limitações de recursos ofertados. Esse problema é uma extensão do *Bin-Packing*, sendo este conhecidamente NP-difícil, logo o VNF-PC também é;

II Caso usual: o mapeamento de uma requisição com arestas é uma extensão do problema NP-difícil de isomorfismo de subgrafos, em que um grafo virtual menor (requisição) é mapeado em um grafo maior (SN). Tanto no VNE quanto no VNF-PC, as restrições adicionais de CPU, memória e banda podem ser entendidas como uma agregação do *Bin-Packing* ao isomorfismo de subgrafos, conduzindo a problemas combinatoriais mais complexos.

Apêndice B

Conceitos adicionais do VNF-PC

B.1 Entidades Envolvidas

Similarmente ao modelo de negócios proposto por Mijumbi et al. (2016); Laghrissi and Taleb (2018), os papéis das entidades de rede envolvidas no ambiente do VNF-PC são definidas como:

- Os Provedores de VNFs (*VNF Providers*, VNFPs) e Provedores de Serviços (*Server Providers*, SPs): são as entidades que ocupam o lugar das fornecedoras de equipamentos de rede em um modelo legado. Tais entidades implantam e gerenciam as VNFs sobre os recursos administrados pelos InPs, conduzindo a instanciação de máquinas virtuais conforme os NS demandados pelos clientes. Tanto os VNFPs, quanto os SPs, podem possuir ou alugar recursos dos InPs para oferecer essas funções;

- Os Provedores de Infraestrutura (*Infrastructure Providers*, InPs): são empresas responsáveis por administrar a estrutura física do SN que será usada para acomodar a infraestrutura das redes baseadas em tecnologias NFV, *e.g.*, recursos de processamento e memória. Os InPs são as entidades responsáveis pelo gerenciamento e distribuição de recursos físicos para os Provedores de Serviços de Telecomunicação utilizarem.

- Os Provedores de Serviços de Telecomunicação (*Telecommunications Service Provider*, TSPs): contratam os recursos físicos dos InPs e as VNFs posicionadas pelos SPs, para então encadeá-las, e oferecê-las como serviço ao usuário final. Em alguns casos, os TSPs podem possuir uma infraestrutura própria de rede sem a necessidade de utilizar InPs externos. Nesse modelo, os SPs ficam a cargo de alocar os recursos residuais de servidor do substrato físico de rede e gerir as VNFs, e os TSPs são responsáveis por operar o encadeamento desas VNFs.

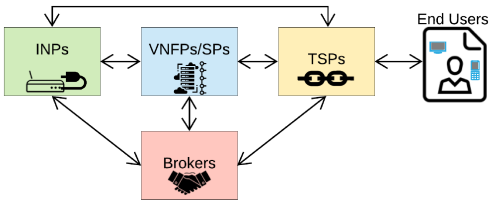
- Os clientes (*End Users*): são os consumidores finais, e contratam os recursos dos TSPs de acordo com uma demanda específica. Cada cliente pode possuir uma

demanda exclusiva de recursos, aspectos diferentes de QoS, e alugar os recursos dos TSPs por uma fração diferente de tempo (duração da SFC).

- Corretores (*Brokers*): são entidades que intermediam a contratação de recursos entre os SPs, VNFPs, TSPs e InPs. Em alguns casos o TSP pode necessitar de recursos de InPs diferentes, ou um conjunto de VNFs oferecido por diferentes SPs para atender a um cliente. Neste caso o corretor de rede pode auxiliar na integração entre os recursos de infraestrutura demandados e os ofertados.

A Figura B.1 exemplifica os atores envolvidos no modelo de negócio das redes baseadas em tecnologias NFV. No exemplo, insere-se os *Brokers* para completude da ilustração, mas sua presença não é obrigatória em redes baseadas em tecnologias NFV.

Figura B.1: Modelo de Negócio das redes NVF



Fonte: Mijumbi et al. (2016).

B.2 Fórmula de Haversine

Devido às topologias de redes adotadas serem de amplitude continental/global, determinar a distância entre dois pontos quaisquer não pode ser ser feito com o cálculo de uma linha reta, pois existe uma curvatura atrelada. Para mitigar este problema, adota-se a fórmula de Haversine (Sinnott, 1984). A fórmula de Haversine é uma equação, baseada na lei dos cossenos, que considera a curvatura da Terra para calcular a distância entre dois pontos, dada por:

$$a = \sin^2 \frac{(latOri - latDest)}{2} + \cos(latOri) \cdot \cos(latDest) \cdot \sin^2 \frac{(longOri - longDest)}{2} \tag{B.1}$$

$$\text{distância de Haversine} = R \cdot 2 \cdot atan2\left(\sqrt{a}, \sqrt{1-a}\right) \tag{B.2}$$

Considere $R = 6371$ (raio do Planeta Terra); $latOri$ e $longOri$ a latitude e longitude dos pontos de origem, e $latDest$ e $longDest$ dos pontos de destino.

Apêndice C

Análise Complementar dos Experimentos do Capítulo 4

C.1 Relação Lucro e Receita

O algoritmo ILP^{i}, apesar de efetuar somente a reconfiguração do tamanho das instâncias alocadas, consegue gerar uma boa relação lucro/receita (Figura 4.2), mas com um lucro final abaixo ao do algoritmo ILP^{ic} (Figura 4.2). Como exemplo, nos experimentos com o cenário 1, essa diferença é de $\approx 3.2\%$; para o cenário 3 é de $\approx 17\%$; e para o cenário 4 de $\approx 5.28\%$. Esse fato ocorre não só devido à receita do algoritmo ILP^{i} ser inferior ao do algoritmo ILP^{ic} (Figura 4.4), mas também pelo uso mais racional dos arcos físicos, o que potencialmente fragmenta menos o substrato de rede, e permite o mapeamento de outras SFCs que podem gerar uma receita maior. Depreende-se que, apesar de não aumentar significativamente o tempo de processamento (Figura 4.3), reconfigurar os roteamentos pode gerar um impacto promissor em alguns cenários, como o cenário 3, mas em outros, como os cenários 1, 2 e 4, tal efeito não é tão significativo.

Percebe-se uma diminuição do efeito da reconfiguração entre os algoritmos nos cenários com menos SFCs ativas simultaneamente. Este fenômeno pode ser observado nitidamente no cenário de experimentos 2. Visto que em tal cenário o intervalo de chegadas é maior que o tempo de vida das SFCs, não existem SFCs ativas simultaneamente, e o efeito da reconfiguração é nulo. Com menos SFCs ativas, menor a disputa por recursos, e menor a necessidade de reotimização das restrições.

Na Figura 4.2, para todos os algoritmos avaliados, percebe-se que no cenário 2 o lucro e a receita tendem a ser baixos. Isto acontece por tal cenário ser caracterizado por SFCs com uma duração muito baixa. Neste caso, uma SFC com um baixo tempo de vida, *e.g.*, *web service*, gera uma baixa receita para o provedor. Por outro lado, os cenários em que as SFCs possuem uma duração maior, *e.g.*, o Cenário 4, que possuem SFCs como de jogos *online* e *streaming* vídeo, tendem a gerar receitas e lucros maiores.

C.2 Tempo de Execução

Por processar todas as restrições e variáveis de todas as SFCs ativas e entrantes simultaneamente, o algoritmo ILP^{ipc} possui um tempo de execução elevado e impraticável chegando a $\approx$ 25 horas para processar 122 SFCs e gerar um resultado ótimo global (Figura 4.3, cenário 4). Por outro lado, reotimizando (redimensionando) apenas as instâncias de VNFs ativas e/ou encadeamentos, os demais algoritmos são mais rápidos que o algoritmo ILP^{icp}, mas, implicam em um menor lucro gerado (Figura 4.2). Como exemplo no cenário 4, o algoritmo ILP^{i} possui um tempo de execução $99,94\%$ menor que o algoritmo ILP^{ipc} (Figura 4.3), mas ao custo de uma queda no lucro de $\approx$ 12%. Percebe-se que a reconfiguração total realizada pelo algoritmo ILP^{icp} pode aumentar o lucro, contudo, por se tratar de um cenário *online*, em que existe a necessidade de uma tomada de decisão rápida sobre o mapeamento ou não, de uma SFC, o tempo de execução de tal algoritmo pode ser considerado impraticável (Figura 4.3).

C.3 Espalhamento dos Nós Virtuais

Nos experimentos mostrados na Figura 4.5, nos casos em que existem várias SFCs ativas simultaneamente (cenários 1 e 4), a pressuposição (confirmada) é que o compartilhamento de nós físicos é maior em relação aos casos em que não existem, ou existem poucas SFCs ativas simultaneamente (cenários 2 e 3). Como observado na Tabela, 4.7, no cenário 3, o tempo de duração das SFCs tende a ser menor em relação ao cenário 1. Com uma duração menor, menos SFCs vão estar ativas simultaneamente, prejudicando o compartilhamento de nós. Repare que este tempo é de $\approx$ 65.8% com o algoritmo ILP^{icp} no cenário de 3, e de $\approx$ 57.9% e $\approx$ 46.6% respectivamente nos cenários 1 e 4 (Tabelas C.1, C.3 e C.4). Outro ponto que deve ser abordado é que os servidores possuem restrições de capacidade de memória e processamento, assim, se a quantidade de SFCs ativas simultaneamente for alta, mais servidores devem ser ativados para o atendimento desta demanda. Como há mais servidores ativos, e a chega de novas SFCs é intensa (como exemplo no cenário 4), a propensão é que o compartilhamento de servidores seja maior. Um resumo numérico das Figuras 4.5a a 4.5d pode ser visto nas Tabelas C.1 a C.4.

Tabela C.1: Resumo do Gráfico 4.5a

Média ($\overline{x}$)		
ILP^{i}	ILP^{ic}	ILP^{icp}
61.98%	58.56%	57.94%
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP^{i}-ILP^{ic}	ILP^{ic}-ILP^{icp}	ILP^{i}-ILP^{icp}
5.84%	1.07%	6.98%

Tabela C.2: Resumo do Gráfico 4.5b

Média ($\overline{x}$)		
ILP^{i}	ILP^{ic}	ILP^{icp}
100.00%	100.00%	100.00%
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP^{i}-ILP^{ic}	ILP^{ic}-ILP^{icp}	ILP^{i}-ILP^{icp}
0.00%	0.00%	0.00%

Tabela C.3: Resumo do Gráfico 4.5c

Média ($\overline{x}$)		
ILP^{i}	ILP^{ic}	ILP^{icp}
74.87%	64.31%	65.84%
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP^{i}-ILP^{ic}	ILP^{ic}-ILP^{icp}	ILP^{i}-ILP^{icp}
16.42%	−2.33%	13.71%

Tabela C.4: Resumo do Gráfico 4.5d

Média ($\overline{x}$)		
ILP^{i}	ILP^{ic}	ILP^{icp}
54.78%	52.81%	46.61%
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP^{i}-ILP^{ic}	ILP^{ic}-ILP^{icp}	ILP^{i}-ILP^{icp}
3.72%	13.30%	17.51%

Fonte: Elaborado pelo autor.

C.4 Espalhamento dos Arcos Virtuais

Um resumo numérico das Figuras 4.6a a 4.6d pode ser visto nas Tabelas C.5 a C.8.

Tabela C.5: Resumo do Gráfico 4.6a

Média ($\overline{x}$)		
ILP^{i}	ILP^{ic}	ILP^{icp}
29.00%	27.25%	28.78%
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP^{i}-ILP^{ic}	ILP^{ic}-ILP^{icp}	ILP^{i}-ILP^{icp}
6.42%	−5.31%	0.77%

Tabela C.6: Resumo do Gráfico 4.6b

Média ($\overline{x}$)		
ILP^{i}	ILP^{ic}	ILP^{icp}
0.26%	0.26%	0.26%
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP^{i}-ILP^{ic}	ILP^{ic}-ILP^{icp}	ILP^{i}-ILP^{icp}
0.00%	0.00%	0.00%

Tabela C.7: Resumo do Gráfico 4.6c

Média ($\overline{x}$)		
ILP^{i}	ILP^{ic}	ILP^{icp}
30.79%	31.66%	30.29%
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP^{i}-ILP^{ic}	ILP^{ic}-ILP^{icp}	ILP^{i}-ILP^{icp}
−2.76%	4.52%	1.64%

Tabela C.8: Resumo do Gráfico 4.6d

Média ($\overline{x}$)		
ILP^{i}	ILP^{ic}	ILP^{icp}
43.69%	37.01%	30.21%
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP^{i}-ILP^{ic}	ILP^{ic}-ILP^{icp}	ILP^{i}-ILP^{icp}
18.04%	22.50%	44.60%

Fonte: Elaborado pelo autor.

C.5 Atraso Médio Fim a Fim

Um resumo numérico das Figuras C.9 a 4.7d pode ser visto nas Tabelas C.5 a C.12.

Tabela C.9: Resumo do Gráfico 4.7a

Média ($\overline{x}$)		
ILP^{i}	ILP^{ic}	ILP^{icp}
$32.96ms$	$32.29ms$	$32.07ms$
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP^{i}-ILP^{ic}	ILP^{ic}-ILP^{icp}	ILP^{i}-ILP^{icp}
2.07%	0.69%	2.77%

Tabela C.10: Resumo do Gráfico 4.7b

Média ($\overline{x}$)		
ILP^{i}	ILP^{ic}	ILP^{icp}
$0.30ms$	$0.30ms$	$0.30ms$
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP^{i}-ILP^{ic}	ILP^{ic}-ILP^{icp}	ILP^{i}-ILP^{icp}
0.00%	0.00%	0.00%

Tabela C.11: Resumo do Gráfico 4.7c

Média ($\overline{x}$)		
ILP^{i}	ILP^{ic}	ILP^{icp}
$31.31ms$	$38.76ms$	$35.29ms$
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP^{i}-ILP^{ic}	ILP^{ic}-ILP^{icp}	ILP^{i}-ILP^{icp}
−19.23%	9.83%	−11.29%

Tabela C.12: Resumo do Gráfico 4.7d

Média ($\overline{x}$)		
ILP^{i}	ILP^{ic}	ILP^{icp}
$51.80ms$	$42.97ms$	$34.00ms$
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP^{i}-ILP^{ic}	ILP^{ic}-ILP^{icp}	ILP^{i}-ILP^{icp}
20.55%	26.38%	52.35%

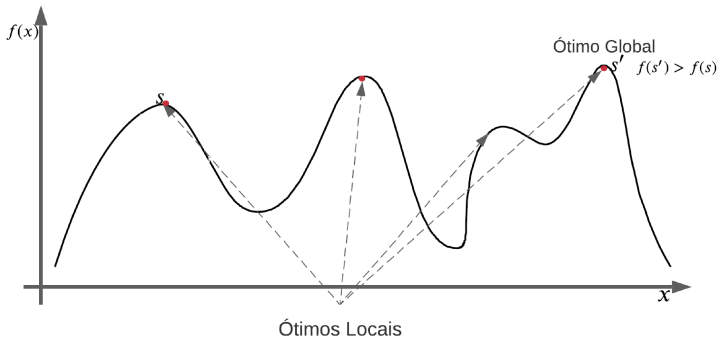
Fonte: Elaborado pelo autor.

Apêndice D

Conceitos de Algoritmo Guloso, Heurística Semigulosa e Metaheurística VNS

Um problema de otimização é definido como um problema no qual se procura determinar um valor de máximo/mínimo de uma função (Cormen et al., 2001). Os algoritmos desenvolvidos para solucionar problemas de otimização possuem uma sequência lógica de etapas e cada etapa é caracterizada por um conjunto de escolhas a fim de otimizar um objetivo. Em problemas de maximização, uma solução s' é chamada ótimo local se a função objetivo de s' for maior que a função objetivo de uma solução s vizinha. Uma solução s' é dita como ótima global se seu valor for o maior valor possível dentre todas as soluções existentes (Figura D.1).

Figura D.1: Exemplos de ótimo local e global



Fonte: Elaborado pelo autor.

Os algoritmos para solucionar problemas de otimização podem ser definidos como:

Algoritmo guloso: é um algoritmo distinguido por sempre fazer suas escolhas com base na melhor solução naquele instante, *i.e.*, faz uma escolha baseada em um ponto de ótimo local, com o intuito de que essa escolha leve a uma solução de boa qualidade ou ótima global (Cormen et al., 2001). Simplificadamente, um algoritmo guloso, a partir de um princípio de resolução, constrói uma solução, um elemento de cada vez. O Algoritmo 8 representa o funcionamento de tal abordagem.

Algoritmo 8 Guloso

1: Define os elementos candidatos a fazerem parte da solução s
2: Aplica uma função gulosa em cada elemento candidato a fazer parte da solução s
3: Classifica os elementos conforme o valor da função gulosa
4: Adiciona o elemento com melhor classificação à solução s
5: **devolve** s

Um algoritmo guloso funciona bem para uma ampla faixa de problemas, como para o problema de caminhos mais curtos (Cormen et al., 2001). Mas, por metodicamente sempre fazer suas escolhas com base na melhor solução no instante em questão, os algoritmos gulosos são limitados no sentido de sempre fornecer a mesma solução, quando iniciados de um mesmo ponto de partida. Tal estratégia se torna ineficiente para problemas pertencentes à classe NP-difícil, pois não se conhece um princípio guloso que conduza a uma geração de solução com garantias de otimalidade. Uma boa escolha de elemento para fazer parte da solução, apesar de ser ótima localmente, nem sempre conduz a uma solução ótima global.

Heurística Semigulosa: para tentar solucionar tais problemas presentes nos algoritmos gulosos, Hart and Shogan (1987) propuseram as chamadas heurísticas semigulosas. Uma heurística semigulosa tenta contornar a convergência da geração da solução para pontos de ótimos locais. Para tal ação, ela escolhe o elemento que irá fazer parte da solução com certo grau de aleatoriedade entre os elementos mais promissores, de acordo com uma função gulosa. Em tal algoritmo, existe uma Lista Restrita de Candidatos (LRC) que pode ser entendida como um conjunto de elementos candidatos a solução do problema, em que tais elementos são os mais bem classificados de acordo com uma função de avaliação. Uma heurística semigulosa constrói uma solução, um elemento de cada vez, com uma avaliação iterativa feita na LRC.

Algoritmo 9 Construtivo Semiguloso

1: **enquanto** solução s não construída **faça**
2: **para cada** elemento candidato **faça**
3: Aplique uma função gulosa de avaliação ao elemento
4: **fim para**
5: Classifique os elementos de acordo com seus valores de função gulosa
6: Coloque os elementos em uma LRC
7: Selecione um elemento da LRC aleatoriamente
8: Adiciona o elemento à solução s
9: **fim enquanto**
10: **devolve** s

O algoritmo 9 apresenta o funcionamento da heurística semigulosa. Após a avaliação de cada elemento com uma função gulosa (linha 3), pode-se escolher qual elemento irá fazer parte da LRC. No momento de escolher os elementos que irão compor a LRC

(linha 6), Hart and Shogan (1987) propuseram dois mecanismos distintos, o primeiro baseado na quantidade de elementos existentes, neste caso são inseridos os k melhores candidatos; e o segundo baseado em um valor $\alpha \in [0, 1]$, no caso são inseridos os elementos candidatos com valores maiores do que $\alpha \times best$. No caso, *best* é uma variável que representa o valor da função de avaliação do melhor elemento.

Hart and Shogan (1987) também apresentam uma estrutura de repetição de aplicação da heurística construtiva semigulosa, em que sempre é armazenada a melhor solução encontrada. Tal conceito serviu de base para a característica multipartida adotada posteriormente na metaheurística *Greedy Randomized Adaptive Search Procedure* (GRASP), proposta por Feo and Resende (1995). A propriedade multipartida apresentada aumenta a região explorada do espaço de soluções, permitindo a investigação de diferentes áreas do espaço de busca.

O Algoritmo 10 representa o funcionamento da heurística multipartida semigulosa apresentada por Hart and Shogan (1987). Em tal algoritmo, são construídas n soluções s' (linha 4). Caso a solução gerada seja melhor que a solução encontrada anteriormente s, *i.e.*, $f(s') > f(s)$, tal solução é então armazenada (linha 5). Por fim, o algoritmo retorna à melhor solução encontrada após um número de iterações (linha 8). Tal número de iterações é definido como um critério de parada (linha 2).

Algoritmo 10 Multipartida (Hart and Shogan, 1987)

```
1: s ← NULL
2: enquanto critério de parada não satisfeito faça
3:    s' ← Construtivo Semiguloso
4:    se f(s') > f(s) então
5:       s ← s'
6:    fim se
7: fim enquanto
8: devolve s
```

De maneira complementar aos estudos apresentados por Hart and Shogan (1987), Feo and Resende (1995) propõem que a solução de um problema consiste em um conjunto de elementos que viabiliza suas respectivas restrições. Nesse sentido, cada elemento diferente que pode pertencer a tal solução, pode ser concebido como pertencente a um conjunto, denominado Lista de Candidatos (LC). Uma LRC é uma estrutura de dados que contém um subconjunto reduzido de elementos candidatos a pertencer à solução, *i.e.* $LRC \subseteq LC$. Neste caso, um parâmetro α pode ser usado na construção da solução semigulosa inicial para controlar a quantidade de elementos presentes nesta LRC, *i.e.*, controla a seletividade/aleatoriedade de elementos que irão ser selecionados. Em problemas de minimização, caso existam vários elementos candidatos a pertencer a tal solução, quanto maior for o parâmetro α (mais próximo a 1) mais elementos irão compor a LRC, gerando uma maior variabilidade na solução e deixando o algoritmo aleatorizado; por outro lado,

quanto menor for o parâmetro α (mais próximo a 0), a LRC irá tender a possuir menos elementos, tornando o algoritmo mais seletivo (guloso).

Metaheurística VNS: metaheurísticas são procedimentos de alto nível, que coordenam heurísticas simples, para geração de soluções de boa qualidade em problemas de otimização combinatória computacionalmente difíceis (Resende and Ribeiro, 2016). Em detrimento a um tratamento exato, indicam-se as abordagens metaheurísticas em função de seu baixo custo computacional, mas ao preço da não garantia de otimalidade. A metaheurística VNS, proposta por Mladenović and Hansen (1997), é um algoritmo simples, amplamente aplicado em problemas de otimização.

Diferentemente de uma heurística semigulosa, a metaheurística VNS não segue uma linha multipartida para explorar diferentes regiões do espaço de solução. A metaheurística VNS tem como princípio criar soluções e combiná-las para melhorar a solução corrente através da utilização de diferentes vizinhanças para o problema. Ela tenta explorar com trocas sistemáticas de estruturas de vizinhança, diferentes locais do espaço de solução, sendo sempre a melhor solução usada como solução corrente (Mladenović and Hansen, 1997). Tal ação consiste em mecanismos para modificar a solução corrente e escapar de ótimos locais. O Algoritmo 11 representa o funcionamento da metaheurística VNS.

Algoritmo 11 VNS

```
1:  s ← s_0        {Solução inicial}
2:  enquanto critério de parada não satisfeito faça
3:      k ← 1      {Tipo de estrutura de vizinhança}
4:      enquanto k ≤ r faça
5:          Gere um vizinho qualquer s' ∈ N^k(s);
6:          s'' ← Busca Local(s');
7:          se f(s'') ¡ f(s) então
8:              s, k ← s'', 1
9:          senão
10:             k ← k + 1
11:         fim se
12:     fim enquanto
13: fim enquanto
14: devolve s
```

No processo iterativo, a melhor solução s recebe a solução corrente s_0 (linha 1). O parâmetro k indica o valor da vizinhança a ser explorada e r o limite de vizinhanças (linha 5). Em cada iteração é gerada uma solução vizinha s' na vizinhança k, sendo tal solução submetida a uma busca local (linha 6). Caso a solução gerada pela busca local (s'') seja melhor que a corrente, ela é armazenada como a melhor solução (linha 7), e o processo reinicia-se da estrutura de vizinhança inicial ($k = 1$), caso contrário o valor de k é incrementado e a próxima vizinhança examinada. O critério de parada pode ser definido como uma quantidade fixa de iterações, tempo limite de processamento, ou ainda uma quantidade de iterações sem melhora (estrutura de repetição entre as linhas 2 e 12 do algoritmo 11). Por fim, o algoritmo retorna à melhor solução encontrada (linha 14).

Apêndice E

Análise Complementar dos Experimentos do Capítulo 5

E.1 Lucro Total

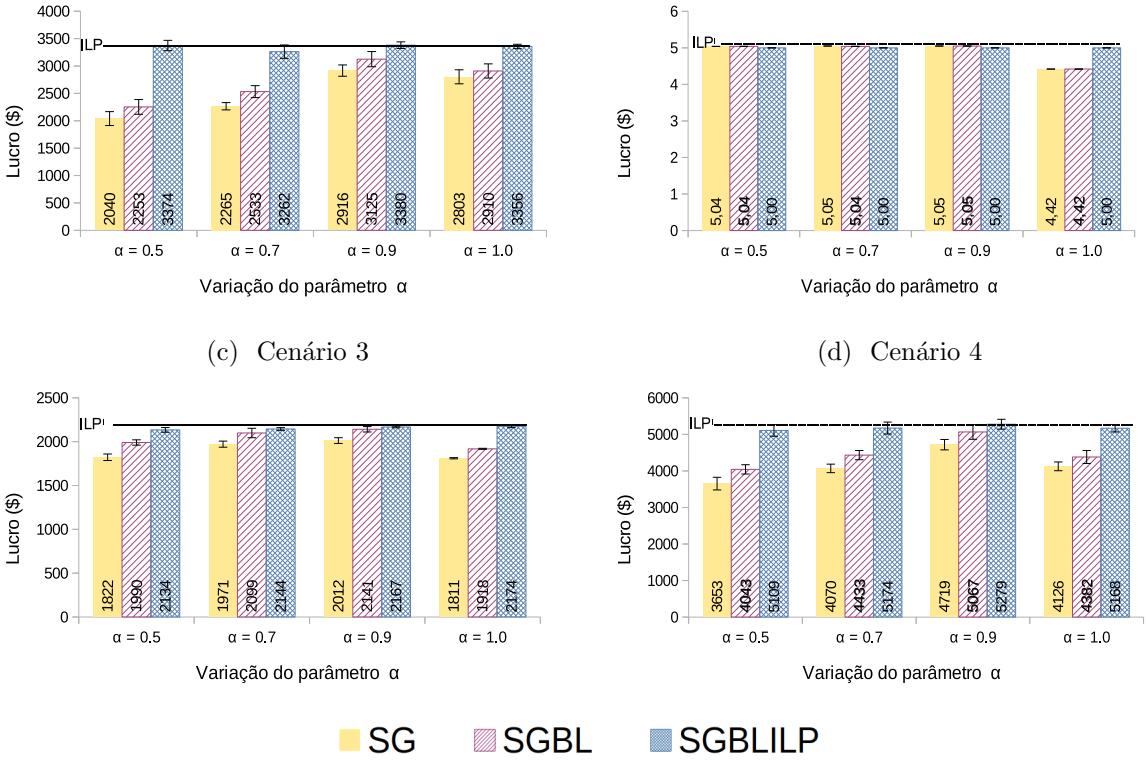
Conforme a função objetivo (Equação 4.16), o lucro está relacionado com a taxa de aceitação das SFCs (Figuras E.3 e 5.7). Quanto maior o número de SFCs mapeadas, maior a receita acumulada, e maior também a quantidade de servidores físicos ativos aptos para serem compartilhados, ação que, consequentemente, mitiga os custos operacionais. Outro ponto que influi no lucro dos provedores é relacionado ao quão os roteamentos dos arcos virtuais são concisos ou distribuídos pelo SN (ver em anexo, Figura 5.9). Como premissa, os mapeamentos que usam menos arcos físicos podem gerar lucros maiores, pois se reduzem os custos relativos ao roteamento das VNFs; e deixam mais arcos não usados para outras SFCs utilizarem, o que potencialmente melhora a taxa de aceitação.

Nos experimentos realizados no cenário 4, percebe-se que, ao se intensificar a chegada de SFCs, o efeito da busca local tende a aumentar. Como exemplo, se compararmos as Figuras E.1d e 5.5d, na heurística com busca local e $\alpha = 0.9$, o efeito da busca local aumentou o lucro em $\approx 7.3\%$ no cenário esparso, e $\approx 11.7\%$ no cenário denso. Indicando que a busca local é mais efetiva em cenários mais densos, em que existe uma maior disputa por recursos do substrato de rede.

Diferentemente dos experimentos com o cenário mais simples (Figuras E.1b e 5.5b), no qual a busca local não gera melhorias no lucro, devido ao algoritmo heurístico semiguloso por si só, produzir soluções de boa qualidade. Em um cenário com mais SFCs ativas simultaneamente a geração de boas soluções torna-se mais complicada. Tal aumento de dificuldade é decorrente de fatores como a fragmentação do SN, e o compartilhamento de servidores já ativos no SN. Neste caso, a busca local gera melhorias significativas estatisticamente no lucro, mais perceptíveis em cenários com uma alta entrada e saída de SFCs, aumentando em até $\approx 12\%$ o lucro (Cenário 4 denso, $\lambda = 0.9$, Figura 5.5d).

Figura E.1: Lucro total dos provedores no cenário esparso

(a) Cenário 1 (b) Cenário 2



(c) Cenário 3 (d) Cenário 4

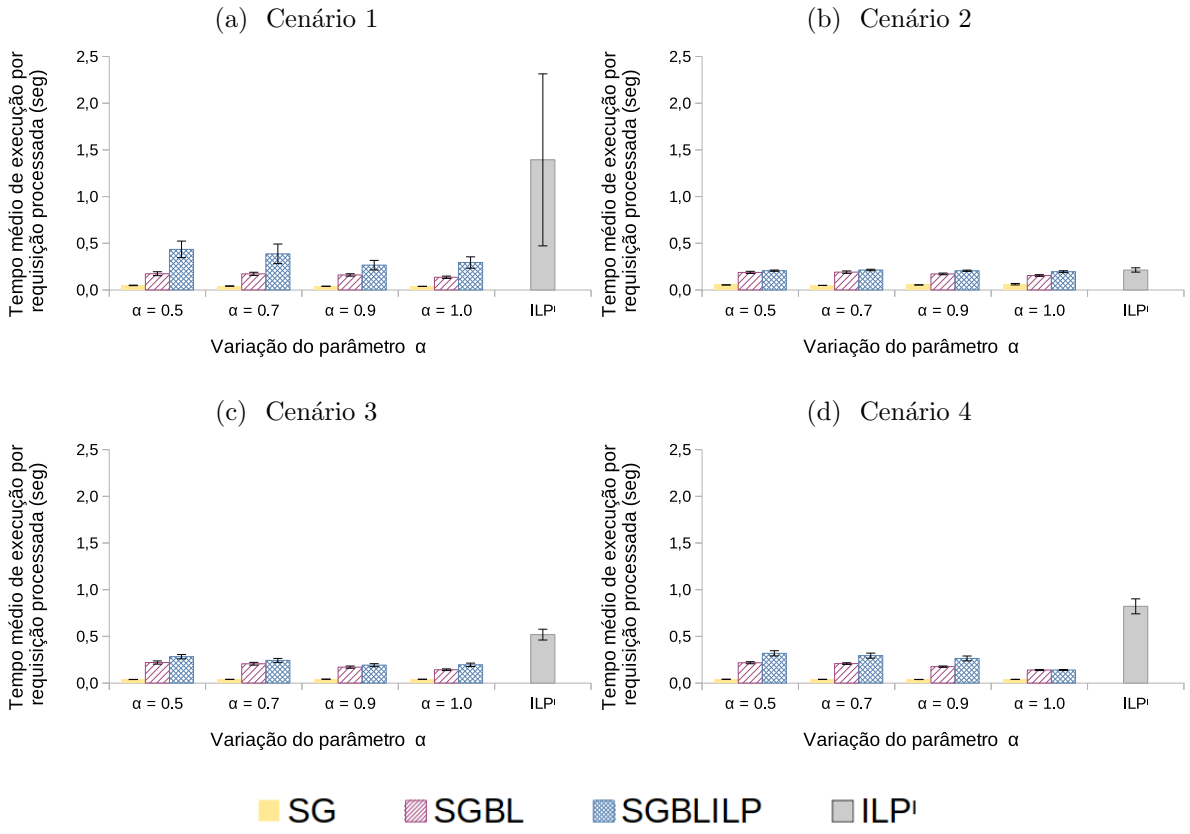
SG SGBL SGBLILP

Fonte: Elaborado pelo autor.

E.2 Tempo de Execução

Similarmente aos experimentos do Capítulo 4, percebe-se que o algoritmo exato em cenários com menos SFCs ativas simultaneamente, produz um menor tempo de processamento por SFC. Fato decorrente de referidos cenários possuírem menos componentes físicos para serem processados, gerando assim um número menor de variáveis e restrições no modelo matemático a ser resolvido. Como exemplo, o algoritmo ILP^i possui um tempo médio de processamento de $\approx 0.8seg$ no cenário 4 esparso, e de $\approx 1.7seg$ no cenário 4 denso. O algoritmo ILP^i, apesar de ser o algoritmo que regularmente gera o maior lucro (Figuras E.1 e 5.5), por processar de modo exato todas as restrições de todas as SFCs entrantes e ativas, possui um tempo de execução alto. Como exceção tem-se o Cenário 2, o mais simples por não haver SFCs ativas simultaneamente. Neste caso, o *solver* CPLEX, de reconhecida eficiência na literatura, consegue um desempenho satisfatório em tempo de execução, justamente por haverem poucas restrições e variáveis para serem processadas.

Figura E.2: Tempo de execução médio por SFC processada no cenário esparso



Fonte: Elaborado pelo autor.

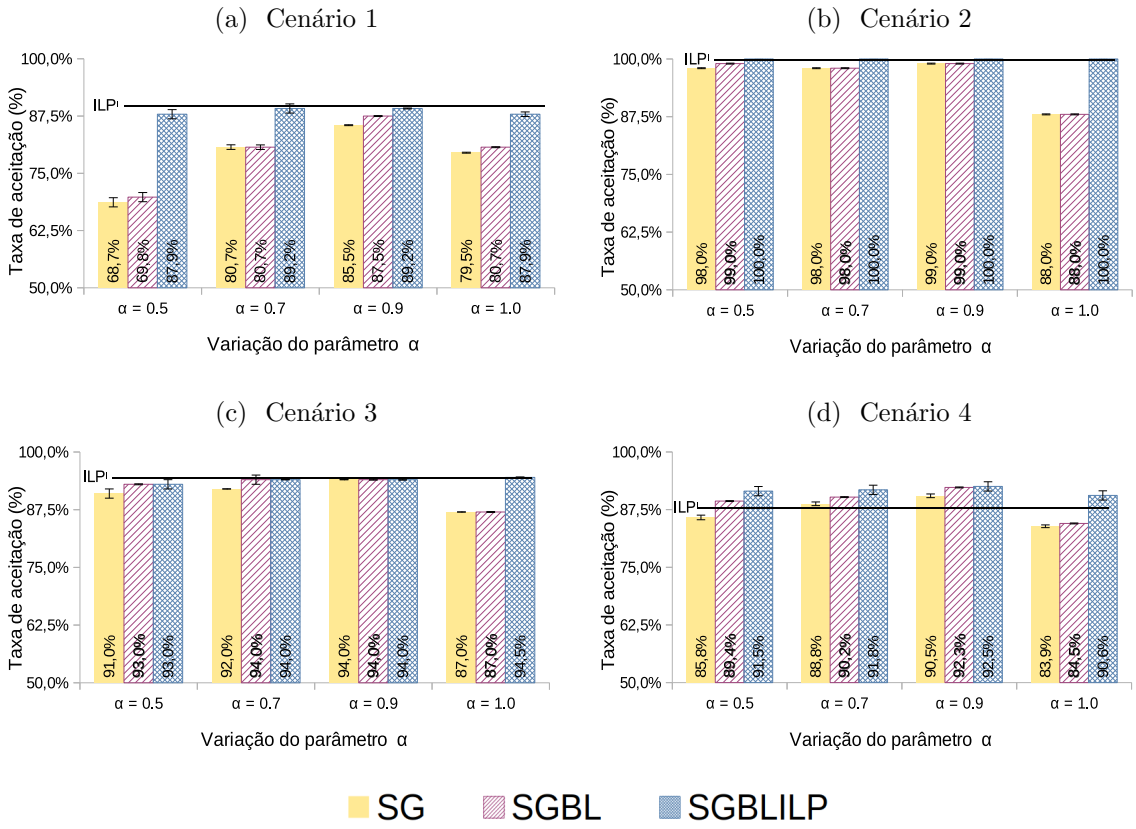
E.3 Taxa de Aceitação

Conforme o número de SFCs ativas aumenta simultaneamente, a taxa de aceitação tende a diminuir. A taxa de aceitação é de 100% nos experimentos do cenário 2, justamente por serem mapeamentos mais simples, e não existir uma disputa por recursos do substrato de rede para se realizar o mapeamento. Neste sentido, a taxa de aceitação é maior nos experimentos esparsos em relação aos densos. Nos experimentos com cenários densos, mais SFCs estão ativas em simultâneo, o consumo de recursos físicos é maior, e consequentemente a taxa de aceitação tende a ser afetada negativamente. Como exemplo, a taxa e aceitação do algoritmo SGBL e $\alpha = 0.9$ é de $\approx 87,5\%$ no cenário 1 esparso (Figura E.3a), e de $\approx 81,8\%$ no cenário 1 denso (Figura 5.7a).

A Tabela E.1 mostra o percentual de mapeamento segmentado das diferentes classes no cenário 4 de SFCs. Tal cenário, por abarcar as SFCs das diferentes classes, se mostra relevante de ser analisado. Neste caso, o algoritmo SGBLILP, além de aumentar o tempo de execução (Figuras E.2 e 5.6), não se mostrou promissora em melhorar a taxa

de aceitação, se for considerada uma comparação com o algoritmo SGBL com $\alpha = 0.9$ (Tabela E.1). Assim, apesar da heurística construtiva não possuir uma garantia de mapeamento com sucesso, na prática, como percebido na Tabela E.1, o algoritmo SGBLILP não melhorou o desempenho em relação ao algoritmo SGBL. Ou seja, nos casos em que uma SFC é rejeitada pela heurística, na prática, realmente tratava-se de uma infactibilidade, visto que o algoritmo exato não conseguiu reverter tal rejeição.

Figura E.3: Taxa de aceitação final no cenário esparso



Fonte: Elaborado pelo autor.

Observando os percentuais de mapeamentos de cada algoritmo no cenário 4 (Tabela E.1), percebe-se que os algoritmos analisados geram um comportamento igualitário não priorizando o mapeamento das SFCs de uma classe em detrimento a outra, independentemente do cenário analisado. Este fato reflete uma política de mapeamento neutra. Contudo, o valor de α pode influenciar diretamente na taxa de aceitação das SFCs de cada classe. Como mostrado na Tabela E.1, no cenário 4 e denso, a variação do α entre 0.5 e 0.9 aumenta a taxa de aceitação da classe 1 em até 22.5%, enquanto na classe 2 essa variação foi de $\approx 1\%$. Isso acontece devido à classe de serviços de rede 2 já possuir uma taxa de aceitação alta mesmo com um α menor. Por outro lado, a classe 1 só gera uma taxa de aceitação alta com um $\alpha = 0.9$. Isso acontece devido ao atraso fim a fim ser mais rígido na classe 1 ($150ms$), assim, muitas soluções iniciais podem ser geradas e descartadas devido às restrições de atraso fim a fim terem sido extrapoladas (Equação

4.11). Na classe 2 este comportamento é menos significativo, visto que lá o atraso fim a fim tolerado é bem maior ($300ms$) e tais restrições se tornam menos rígidas. Nesta análise, as mesmas observações realizadas para a classe 2 podem ser extrapoladas para a classe 3, no qual o atraso fim a fim tolerado é ainda maior ($30000ms$).

Tabela E.1: Relação percentual de SFCs mapeadas no Cenário 4

	Cenário esparso			Cenário denso		
	Classe 1	Classe 2	Classe 3	Classe 1	Classe 2	Classe 3
ILP^i	69/83	115/125	89/100	61/83	108/125	79/100
SG α0.5	53.6/83	119.6/125	90.9/100	48.6/83	115.5/125	85/100
SG α0.7	61.6/83	119.0/125	92.8/100	56.8/83	113.8/125	86.7/100
SG α0.9	68.2/83	119.3/125	91.1/100	65.0/83	115.2/125	87.6/100
SG α1.0	64.2/83	108.0/125	86.2/100	61.4/83	109.6/125	79.6/100
SGBL α0.5	58.4/83	120.9/125	95.8/100	50.0/83	120.2/125	88.4/100
SGBL α0.7	63.8/83	120.2/125	93.8/100	57.6/83	119.2/125	87.4/100
SGBL α0.9	70.8/83	120.2/125	93.3/100	68.0/83	120.3/125	89.6/100
SGBL α1.0	65.5/83	107.9/125	86.8/100	63.0/83	110.9/125	80.4/100
SGBLILP α0.5	69.7/83	119.1/125	93.0/100	53.9/83	118.0/125	83.3/100
SGBLILP α0.7	70.6/83	119.0/125	93.0/100	56.9/83	118.8/125	88.8/100
SGBLILP α0.9	72.2/83	119.3/125	93.4/100	65.6/83	117.4/125	88.6/100
SGBLILP α1.0	70.2/83	116.4/125	94.0/100	60.9/83	110.4/125	81/100

Fonte: Elaborado pelo autor.

E.4 Espalhamento dos Nós Virtuais

No cenário 4 e esparso, o algoritmo ILP^i consegue gerar um bom nível de compartilhamento de recursos, mantendo o espalhamento dos nós virtuais em $\approx 54.8\%$, *i.e.*, em média cada servidor físico ativo está sendo compartilhado por duas SFCs diferentes (Tabela E.2). Por outro lado, a heurística SGBL, apesar de renunciar a um tratamento exato em prol de ter um bom tempo de execução, gera um espalhamento de VNFs também elevado, com valor próximo a $\approx 67.8\%$ (Tabela E.2). No que lhe concerne, a heurística com busca local consegue melhorar significativamente o espalhamento em relação ao algoritmo que não possui a etapa de busca local, sendo esta diferença de $\approx 30.5\%$. Com base em tais dados, percebe-se que a busca local além de melhorar o lucro (Figuras E.1 e 5.5), aumenta o compartilhamento de recursos físicos, aspecto que é um dos princípios das redes baseadas em tecnologias NFV. Um resumo numérico das Figuras 5.8a e 5.8b pode ser visto nas Tabelas E.2 a E.3.

Tabela E.2: Resumo do Gráfico 5.8a

Média ($\overline{x}$)		
ILP^i	SG α0.9	SGBL α0.9
54.82%	88.56%	67.83%
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP^i-SG α0.9	ILP^i-SGBL α0.9	SG α0.9-SGBL α0.9
−38.10%	−19.18%	30.56%

Tabela E.3: Resumo do Gráfico 5.8b

Média ($\overline{x}$)		
ILP^i	SG α0.9	SGBL α0.9
50.02%	71.01%	49.99%
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP^i-SG α0.9	ILP^i-SGBL α0.9	SG α0.9-SGBL α0.9
−29.55%	0.05%	42.01%

Fonte: Elaborado pelo autor.

E.5 Espalhamento dos Arcos Virtuais

Neste experimento, quanto mais baixo for o valor aferido, menos arcos físicos estão sendo utilizados para o mapeamento de determinada SFC. Um resumo numérico das Figuras 5.9a e 5.9b pode ser visto nas Tabelas E.4 a E.5. Analisando o algoritmo SGBL, constata-se que a busca local consegue melhorar o espalhamento dos arcos virtuais sobre os arcos físicos em até $\approx 61.7\%$ (Tabela E.5). Este fato demonstra que as estruturas de vizinhanças desenvolvidas, apesar de restringirem a região de mapeamento de uma VNF (Figuras 5.1a a 5.1c), ação que afeta o compartilhamento de servidores físicos (Figura 5.8), geram um roteamento mais conciso entre os nós virtuais de uma SFC. Outro ponto levantado é que as abordagens que possuem um espalhamento de arcos menor, como as heurísticas propostas (Tabelas E.4 e E.5), além de fragmentarem menos o SN, geram menos impactos em possíveis vulnerabilidades, ou falhas em componentes da rede física.

Tabela E.4: Resumo do Gráfico 5.9a

Média ($\overline{x}$)		
ILP^i	SG α0.9	SGBL α0.9
5.22%	5.54%	4.36%
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP^i-SG α0.9	ILP^i-SGBL α0.9	SG α0.9-SGBL α0.9
−5.76%	19.90%	27.22%

Tabela E.5: Resumo do Gráfico 5.9b

Média ($\overline{x}$)		
ILP^i	SG α0.9	SGBL α0.9
5.51%	6.12%	3.80%
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP^i-SG α0.9	ILP^i-SGBL α0.9	SG α0.9-SGBL α0.9
−10.02%	45.01%	61.17%

Fonte: Elaborado pelo autor.

E.6 Atraso Médio Fim a Fim

Um elevado atraso, superior ao um limite t^v_{dl}, inviabiliza a solução corrente, fator que eleva o número de rejeições. Assim, mesmo que esse atraso fim a fim esteja abaixo ou limítrofe a t^v_{dl}, manter este aspecto baixo e controlado é benéfico para a qualidade

da experiência do usuário final, e deve ser considerado. Em ambos os experimentos, a heurística com busca local se destaca perante a outros algoritmos por mapear mais SFCs com um baixo atraso fim a fim, justamente por controlar tal aspecto com o parâmetro α e por SFCs com menos saltos gerarem um custo mais baixo, fator que aumenta o lucro. Um resumo numérico das Figuras 5.10a e 5.10b pode ser visto nas Tabelas E.6 a E.7.

Tabela E.6: Resumo do Gráfico 5.10a

Média ($\overline{x}$)		
ILP^i	SG α0.9	SGBL α0.9
103.87ms	109.16ms	85.39ms
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP^i-SG α0.9	ILP^i-SGBL α0.9	SG α0.9-SGBL α0.9
-4.84%	21.64%	27.82%

Tabela E.7: Resumo do Gráfico 5.10b

Média ($\overline{x}$)		
ILP^i	SG α0.9	SGBL α0.9
97.11ms	113.45ms	69.37ms
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP^i-SG α0.9	ILP^i-SGBL α0.9	SG α0.9-SGBL α0.9
-14.36%	40.05%	63.53%

Fonte: Elaborado pelo autor.

Apêndice F

Clusterização: Introdução a Algoritmos e Métricas

F.1 K-*Means*: Introdução

Trata-se de um algoritmo de clusterização, baseado em centroides, que utiliza de medidas de distância para identificar aspectos de similaridades entre diferentes registros (dados), e formar os diferentes *clusters* (agrupamentos) (MacQueen, 1967). Cada registro a ser clusterizado possui um conjunto de características (*features*) a serem analisadas. No problema VNF-PC, podem ser consideradas como características o número de VNFs, latitude e longitude dos servidores, e demanda por recursos, como banda, processamento, memória, etc. A intuição do K-*Means* é formar *clusters* de dados, em que os dados no mesmo *cluster* sejam minimamente diferentes entre si, e com baixa similaridade em relação aos elementos dos outros *clusters*. O algoritmo K-*Means* possui uma complexidade de tempo da ordem de $\mathcal{O}(nki)$ e de espaço de $\mathcal{O}(n(d+k))$, na qual n é o número de elementos, k o número de *clusters*, e i o número de iterações (Jin and Han, 2010).

Existem diferentes métodos para o cálculo da distância entre dois registros diferentes, como a distância de *Manhattan*, *Mahalanobis*, e Euclidiana (Jain et al., 1999). Por ser uma das mais populares, inicialmente, neste trabalho é adotada a Distância Euclidiana. A Distância Euclidiana é empregada para refletir a dissimilaridade entre dois padrões. Outras medidas de distância podem ser usadas para caracterizar a similaridade conceitual entre referidos padrões. Sejam q e p dois registros distintos e i o índice da i-ésima característica de cada registro, tal distância pode ser definida como:

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{F.1}$$

No algoritmo K-*Means* existem k *clusters*, sendo que cada *cluster* possui um centroide (MacQueen, 1967). O Algoritmo 12 demonstra o funcionamento do K-*Means*. Inicialmente, cada centroide de cada grupo k é inicializado aleatoriamente (linha 1). Durante

o treinamento do modelo, calcula-se a Distância Euclidiana das características de cada dado até os centroides dos k *clusters* existentes, atribuindo cada dado existente ao *cluster* mais próximo, *i.e.*, minimizando a Distância Euclidiana (linha 3). O reposicionamento de cada centroide é calculado através da distância média entre todos os dados atribuídos à cada *cluster* na iteração corrente (linha 4). Assim, uma iteração completa é definida com a atribuição de cada dado do conjunto de observação até o centroide mais próximo, e do cálculo do reposicionamento de cada centroide (estrutura de repetição entre as linhas 2 e 5). Em nosso caso, por padrão da biblioteca *Sklearn* (Pedregosa et al., 2011), o limite de iterações do K-*Means* (critério de parada) é definido em 300 iterações sem alterações no posicionamento dos centroides (linha 4).

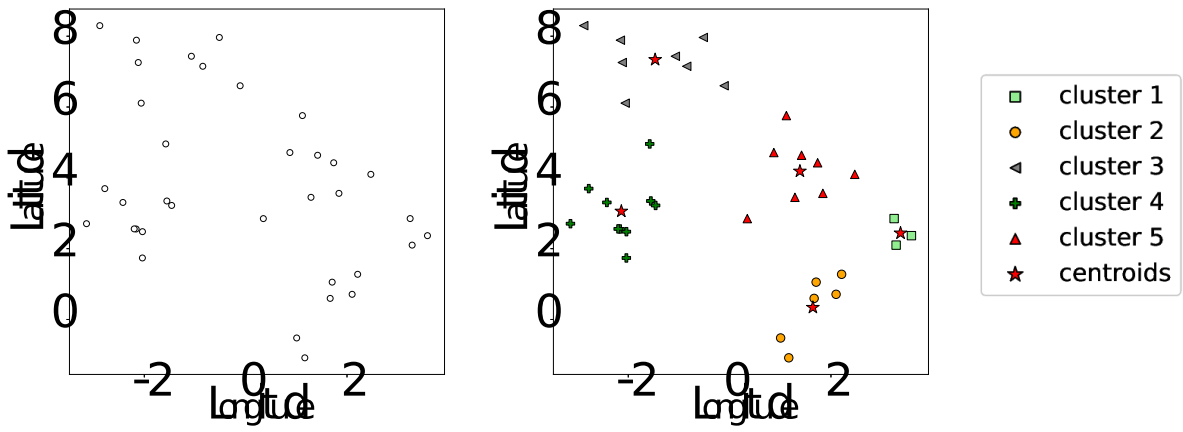
Algoritmo 12 K-*Means* (k)

1: Atribuição inicial do posicionamento dos k centroides
2: **enquanto** critério de parada não satisfeito **faça**
3: Atribui cada dado ao seu centroide mais próximo, minimizando a equação F.1.
4: Calcula o novo posicionamento de cada centroide
5: **fim enquanto**
6: **devolve** k *clusters*

Como exemplo, considere que se deseja clusterizar um conjunto de servidores genéricos distribuídos geograficamente por coordenadas de latitude e longitude, a Figura F.1a exemplifica a distribuição física desses servidores no plano. Clusterizando os servidores da Figura F.1a com o algoritmo K-*Means*, sendo $k = 5$, geram-se 5 *clusters* e seus respectivos centroides, como mostrado na Figura F.1b. Para exemplificar, na Figura F.1, as características consideradas de cada registro do conjunto de dados são suas respectivas latitudes e longitudes, mas outras características podem ser utilizadas, como: capacidade de processamento, memória residual, consumo de energia, etc.

Figura F.1: Exemplo de um conjunto de servidores clusterizados pelo K-*Means*

(a) Servidores do SN (b) SN clusterizado

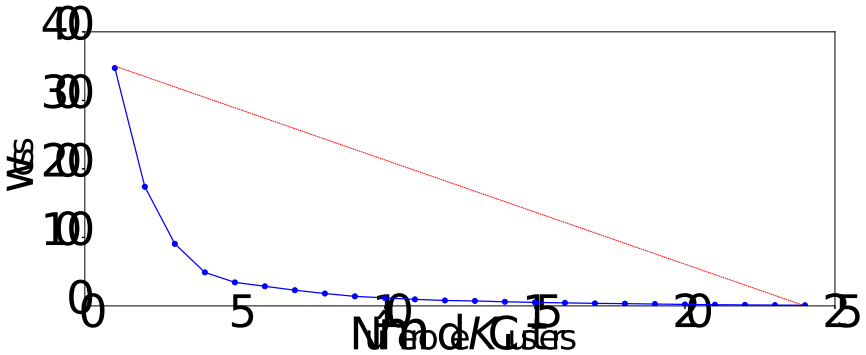


Fonte: Elaborado pelo autor.

Dentre outros, o algoritmo K-*Means* (Algoritmo 12) possui dois fatores principais a serem ajustados, o posicionamento inicial dos centroides de cada *cluster* (linha 1), e o número de k *clusters* existentes (parâmetro de entrada). Neste trabalho, o posicionamento inicial dos centroides é realizado através do algoritmo chamado na literatura de *k-means++* (Arthur and Vassilvitskii, 2007). No *k-means++* a inicialização dos centroides é efetuada de modo aleatorizado, mas com uma estratégia que busca uma ponderação entre soma dos quadrados intra-clusters (*Within-Clusters Sum-of-Squares*, WCSS, também chamada inércia) em relação ao centroide mais próximo. O algoritmo *k-means++* garante uma inicialização mais inteligente dos centroides em relação ao modelo tradicional, tendendo a uma convergência mais rápida. Devido ao chamado momento de inércia e a Distância Euclidiana, o *k-means* tende a gerar *clusters* esféricos.

Em alguns trabalhos da literatura, a determinação do número ideal de k *clusters* é definida com uso de técnicas estatísticas, como o *Elbow Method* (Figura F.2) ou o *Silhouette Coefficient* (Figura F.3) (Thorndike, 1953; Rousseeuw, 1987; Pearson et al., 2004; Song et al., 2018). Segundo Thorndike (1953), no *Elbow Method* é procurada uma quantidade de agrupamentos em que o WCSS seja a menor possível, gerando um ponto de equilíbrio entre a maior homogeneidade dentro do *cluster* e a maior diferença entre *clusters*. Heuristicamente, tal ponto pode ser achado calculando o ponto da curva mais distante de uma reta (vermelha) traçada entre os pontos iniciais e finais calculados (Figura F.2). Repare que, aplicando o *Elbow Method* (Figura F.2) na clusterização da Figura F.1a, percebe-se que o ponto de equilíbrio, chamado cotovelo, é justamente para o valor $k = 5$, indicando que um bom número de *clusters* é 5 para este caso.

Figura F.2: Redução da inércia com aumento do número de *clusters* (*Elbow method*)



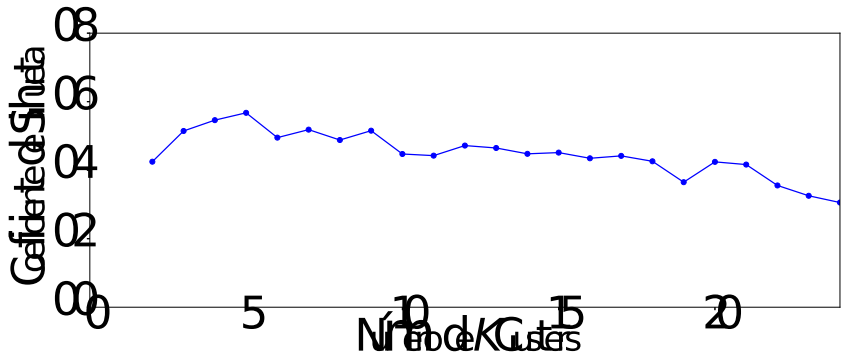
Fonte: Elaborado pelo autor.

Para o cálculo do *Silhouette Coefficient* deve-se conhecer as seguintes variáveis: $s(o)$, que representa o coeficiente de silhueta do ponto de dados o; $a(o)$, que consiste na distância média entre o e todos os outros pontos de dados pertencentes ao *cluster* em questão; e $b(o)$, que simboliza a distância média mínima de o para todos os outros *clusters* aos quais o não pertence. Assim:

$$s(o) = \frac{b(o) - a(o)}{max\{a(o), b(o)\}} \tag{F.2}$$

Segundo Rousseeuw (1987), observando o *Silhouette Coefficient*, existem três situações para serem ponderadas: *i*) se o valor for próximo a 1, os agrupamentos estão bem separados uns dos outros, e o ponto de dados *o* é bem demarcado dentro do *cluster* ao qual pertence; *ii*) se o valor for próximo a 0, os *clusters* são similares, e a distância entre eles não são significativas (denotam *clusters* sobrepostos); e *iii)* se o valor for próximo a -1, a clusterização não ocorreu com sucesso. O *Silhouette Coefficient* gerado para a clusterização da Figura F.1a é mostrado na Figura F.3. Neste caso, procura-se o maior valor para o *Silhouette Coefficient*, que no caso é de 5, gerando o mesmo valor encontrado pelo *Elbow Method*.

Figura F.3: *Silhouette Coefficient* gerado pela clusterização dos dados da Figura F.1 através do algoritmo K-*Means*.



Fonte: Elaborado pelo autor.

Apesar de muito usadas, ressalta-se que o *Elbow method* e *Silhouette Coefficient*, são técnicas estatísticas heurísticas, que, eventualmente, podem gerar resultados ruins. Referidas técnicas consideram especificamente os dados de entrada do problema, desconsiderando implicações específicas na aplicação em questão, no caso, a redução da dimensionalidade de componentes do SN. Neste trabalho, a qualidade de cada *cluster* está diretamente ligada à redução em número de componentes, e ao quão promitente em capacidade de mapeamento essa redução pode ser. Proeminência que deve ser ponderada tanto por aspectos inerentes ao provedor, como a taxa de aceitação e o lucro; quanto para o cliente, como o atraso fim a fim de cada SFC mapeada.

F.2 Algoritmo *Hierarchical Clustering*: Introdução

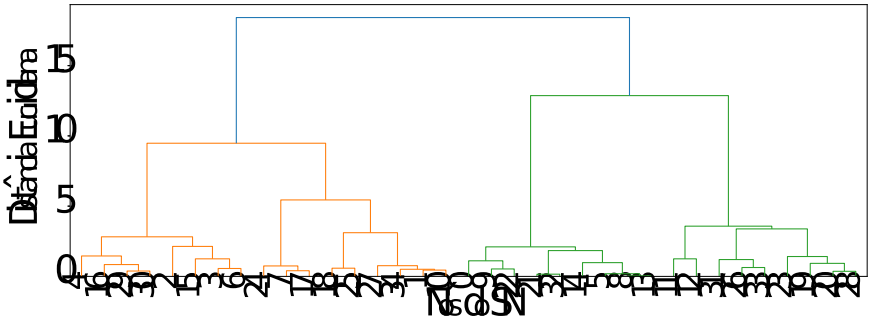
O *Hierarchical Clustering* (Algorítimo 13) é um algoritmo *bottom-up* que inicia com $k = N$ *clusters*, e prossegue aninhando os *clusters* dois a dois, até se obter $k = 1$ cluster. Neste sentido, cada dado é inicialmente considerado um *cluster* de um único elemento (linha 1), na aplicação em questão, os roteadores do SN. Em seguida, são calculadas as distâncias entre todos os *clusters* existentes (linha 4), e os dois *clusters* mais próximos são aninhados em um mesmo *cluster* (linha 6). Assim, são realizados sucessivos aninhamentos, e o algoritmo termina quando há apenas um único *cluster* (Jain et al., 1999) (linha 2).

Algoritmo 13 *Hierarchical Clustering*

1: Transforme cada elemento da entrada em um *cluster* com 1 único elemento
2: **enquanto** número de *clusters* maior que 1 **faça**
3: **para** cada par de *clusters* c_n e c_m **faça**
4: calcula a distância $d(c_n, c_m)$
5: **fim para**
6: Aninha o par de *clusters* c_n, c_m que possuem a menor distância (*merge*)
7: **fim enquanto**
8: **devolve** dendrograma

O *Hierarchical Clustering* possui uma complexidade de tempo da ordem de $\mathcal{O}(n^3)$ e de espaço de $\Omega(n^2)$, na qual n é o número de dados a ser clusterizado. Complexidades elevadas em relação ao K-*Means*, se for considerado que, na prática, o número de caraterísticas e valor de K é inferior a n. Similarmente ao K-*Means*, podem ser usadas diferentes medidas de similaridade, mas, em conformidade com o K-*Means*, optou-se por manter o uso da Distância Euclidiana (linha 4). Ao final, o *Hierarchical Clustering* retorna um diagrama (linha 8), também chamado dendrograma, que representa os agrupamentos aninhados com seus respectivos níveis de similaridade (Figura F.4).

Figura F.4: Dendrograma gerado pela clusterização dos nós da Figura F.1



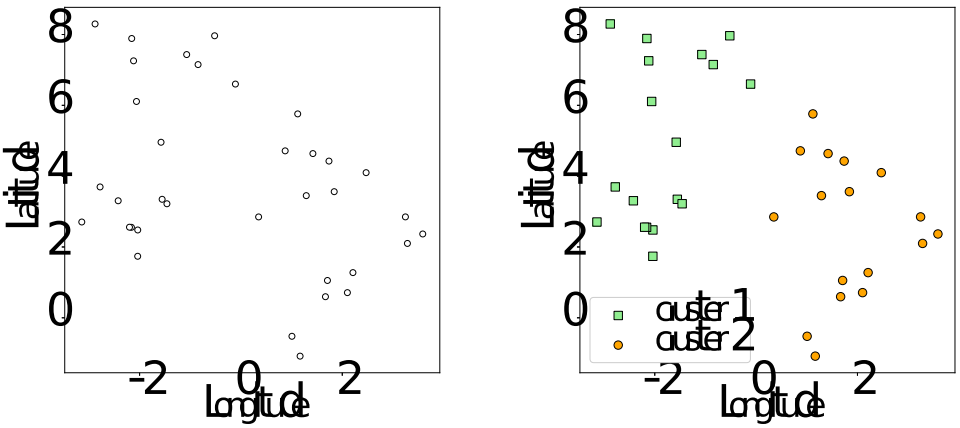
Fonte: Elaborado pelo autor.

Como exemplo, considere que se deseja clusterizar o mesmo conjunto de servidores clusterizados anteriormente com o algoritmo K-*Means* (Figura F.1), mas agora com o algoritmo *Hierarchical Clustering*. Aplicando o Algoritmo 13, obtêm-se o dendrograma (Figura F.4). Na parte inferior do dendrograma, o algoritmo começa com 35 nós do SN de exemplo, em que, cada um é atribuído a um *cluster* individual. Após, par a par, os *clusters* mais próximos são então aninhados, até que exista apenas um *cluster* no topo do dendrograma. A altura no dendrograma em que cada par de *clusters* são aninhados representa a distância entre dois *clusters* no espaço. A decisão do número de *clusters* que melhor separa os nós do servidor pode ser escolhida observando-se os aninhamentos do dendrograma. Heuristicamente, a melhor escolha do número de *clusters* é dada pelo número de linhas verticais no dendrograma cortadas por uma linha horizontal que pode atravessar verticalmente a distância máxima sem interceptar um *cluster* diferente. Pelo algoritmo a tendência é o número de *clusters* ser igual a 2, devido à maior distância vertical existente.

Clusterizando os servidores da Figura F.5a com o algoritmo *Hierarchical Clustering*, geram-se 2 *clusters*, como mostrado na Figura F.5b. Repare que, por possuir um modelo de clusterização baseado em conectividade e não em centroides, o *Hierarchical Clustering* gera uma clusterização diferente da gerada pelo K-*Means* (Figura F.1).

Figura F.5: Conjunto de servidores clusterizados pelo algoritmo *Hierarchical Clustering*

(a) Servidores do SN (b) SN clusterizado



Fonte: Elaborado pelo autor.

F.3 Algoritmo DBSCAN: Introdução

O DBSCAN é um algoritmo, baseado em conceitos de densidade, amplamente utilizado para a formação de *clusters* de formas arbitrárias (esféricas ou não) (Singh and Meshram, 2017). Tal algoritmo descobre formas aleatórias de *clusters*, e consegue lidar com ruídos (*outliers*) eficientemente. Contudo, tal técnica pode gerar resultados ruins para conjuntos de dados de densidade variada e/ou desbalanceada. Neste algoritmo, a Distância Euclidiana (Equação F.1) é usada como uma medida de similaridade para os dados. O *DBSCAN* possui uma complexidade de tempo da ordem de $\mathcal{O}(n^2)$ e de espaço de $\mathcal{O}(n)$, onde n é o número de dados a ser clusterizado (Shinde and Sankhe, 2017). Para o entendimento do DBSCAN, deve se conhecer alguns parâmetros e conceitos, são eles:

I Parâmetros: EPS, define um raio máximo em que dois dados (pontos) podem estar para serem considerados do mesmo cluster; e $MPts$, define o número mínimo de pontos necessários em uma região para garantir uma densidade desejada;

II Conceitos: núcleo, consiste em um ponto que possui um mínimo de pontos (MinPts) dentro de um raio de distância (EPS); borda, é um ponto que não possui nenhum outro ponto dentro de um raio EPS de distância, porém está inserido dentro de um raio EPS de outro ponto; e ruído, é um ponto que não está presente no raio EPS de nenhum núcleo, e também não possui nenhum ponto dentro do seu raio EPS.

O processo iterativo do algoritmo DBSCAN é exemplificado no Algoritmo 14. Este recebe como entrada o conjunto N de pontos a serem clusterizados, e os parâmetros EPS e MPts. Inicialmente, tal algoritmo escolhe um ponto $p \in N$ arbitrário para ser classificado (linha 2). Após, tal ponto pode ser classificado como núcleo, borda ou ruído (linha 3). Posteriormente, se o ponto arbitrário p é um núcleo, todos os pontos a sua volta com um raio de distância EPS formam um *cluster* (linha 6). Repare que caso necessário, reclassificações podem ocorrer. Após classificar todos os pontos da região do núcleo p, a estrutura de repetição retorna para a etapa de escolha de um nó arbitrário (linha 2). O algoritmo termina quando todos os pontos tiverem sido classificados (linha 1). Em referido algoritmo, os pontos classificados como ruído são aqueles em regiões de baixa densidade (linha 6). Assim, quando o valor de EPS é baixo, alguns pontos podem não ser agrupados, sendo classificados como ruído. Por outro lado, quando o EPS é alto, *clusters* que sejam próximos podem ser mesclados como um só.

Diferentemente do algoritmo K-*Means*, o algoritmo DBSCAN não requer como entrada o número de *clusters* procurado. Como exemplo, considere que se deseja clusterizar o mesmo conjunto de servidores genéricos clusterizados anteriormente com o algoritmo

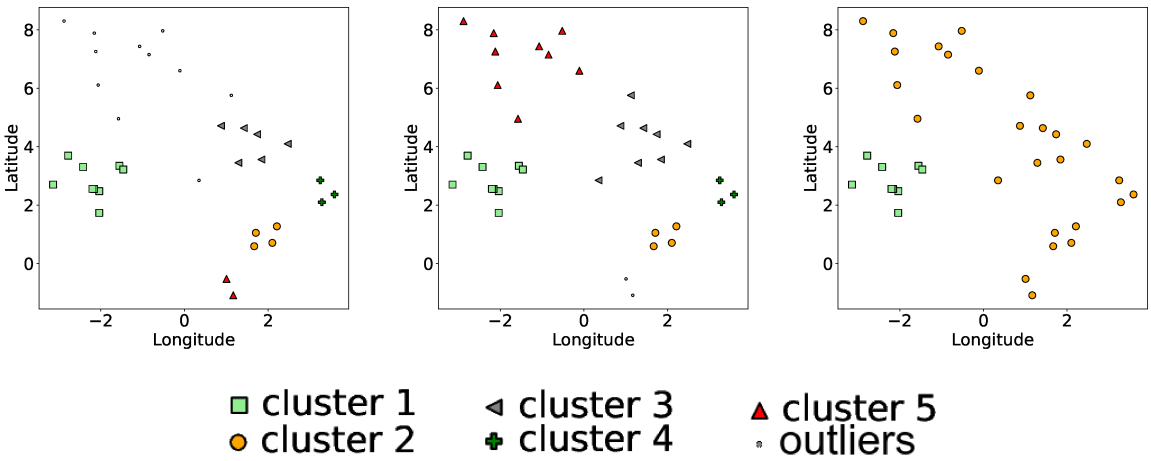
Algoritmo 14 DBSCAN (n,eps,MPts)

```
1: enquanto exitir ponto p ∈ N não classificado faça
2:    p ← aleatório(N)
3:    p ←classifica(p,MPts)
4:    se p = nucleo então
5:       para cada q ∈ N faça
6:          se d(p, q) ≤ EPS então
7:             q ← cluster(p) {q é classificado como na borda de p}
8:          fim se
9:       fim para
10:   fim se
11: fim enquanto
```

K-*Means* (Figura F.1b) e com o algoritmo *Hierarchical Clustering* (Figura F.5b). Empregando o algoritmo DBSCAN, gera-se a clusterização mostrada na Figura F.6. No exemplo, mostra-se que o valor de EPS é fundamental neste algoritmo, e para ser definido deve-se realizar diferentes experimentações, isolando e analisando o efeito dos diferentes fatores estados. Conforme os algoritmos K-*Means* e *Hierarchical Clustering*, o número ideal de *clusters* encontrado foi 5. Mas, percebe-se que, com o DBSCAN, alguns nós físicos foram rotulados como ruídos (*outliers*), fator que pode não ser bom para a aplicação de fato. De maneira resumida, a Tabela F.1, mostra as principais características de cada algoritmo de clusterização apresentado.

Figura F.6: Servidores clusterizados pelo algoritmo DBSCAN, variando o valor de EPS

(a) $EPS = 1.0, MPts = 3$ (b) $EPS = 1.25, MPts = 3$ (c) $EPS = 1.50, MPts = 3$



Fonte: Elaborado pelo autor.

Tabela F.1: Comparação entre os diferentes Algoritmos de clusterização

Técnica de Agrupamento	K-*Means*	Herárquico	DBSCAN
Categoria	Baseado em centroides e partições	Aglomerativo, *bottom-up*	Baseado em densidade
Método para encontrar o número de *clusters*	*Elbow method* e *Silhouette Coefficient*,	Dendrograma	Automático

Fonte: Elaborado pelo autor.

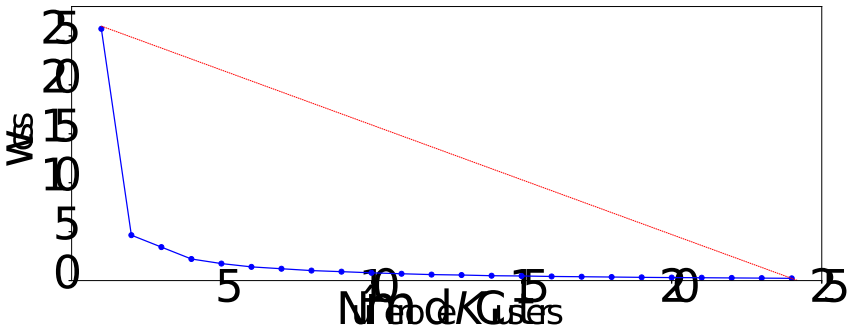
Apêndice G

Clusterização: Análise Complementar do Capítulo 6

G.1 Clusterização com o K-*Means*

Interpretando o *Elbow Method*, apresentado na Figura G.1, o ponto de equilíbrio entre a maior homogeneidade no *cluster* e a maior diferença entre *clusters*, é o ponto da curva mais distante da reta vermelha traçada entre os pontos iniciais e finais, sendo justamente $k = 3$ (cotovelo). Analisando matematicamente tais distâncias, tem-se que k_3 está a 11.3 unidades de distância da reta vermelha, k_2 está a 11.2. Conforme a Figura 6.2b, constata-se que $k = 3$ gera uma boa clusterização do SN, justamente por a rede *Cogent* ser visualmente particionada em duas grandes regiões, e a região mais densa, também particionada em duas, gerando um número equilibrado de roteadores em cada cluster.

Figura G.1: WCSS (*Elbow Method*) da clusterização com o algoritmo *K-Means*

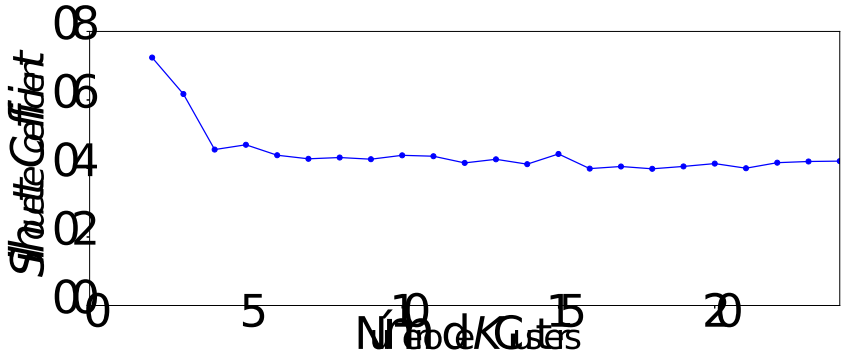


Fonte: Elaborado pelo autor.

Uma das caraterísticas da métrica WCSS, é que ela deve ser a menor possível. Assim, percebe-se ainda na Figura G.1, que ao se aumentar o valor de K, tal métrica continua em declínio, se aproximando do valor 0, mas de maneira suavizada. Tal comportamento indica que os *clusters* gerados podem ser bons, mas não implicam em uma melhora considerável em relação à soma dos quadrados intra-clusters.

No *Silhouette Coefficient*, coeficientes próximos a 1 indicam que a amostra está longe dos *clusters* vizinhos, o que é bom. Por outro lado, um valor próximo de 0 indica que a amostra está no limite, ou muito perto do limite de decisão entre dois *clusters* diferentes. Observando a Figura G.2, um bom valor encontrado para tal métrica é $k = 2$, o que condiz com a Figura 6.2a, pois a topologia pode ser separável em dois grandes *clusters*. Contudo, percebe-se que o valor de $k = 3$ também gera um alto *Silhouette Coefficient*, corroborando o valor encontrado com o *Elbow Method*.

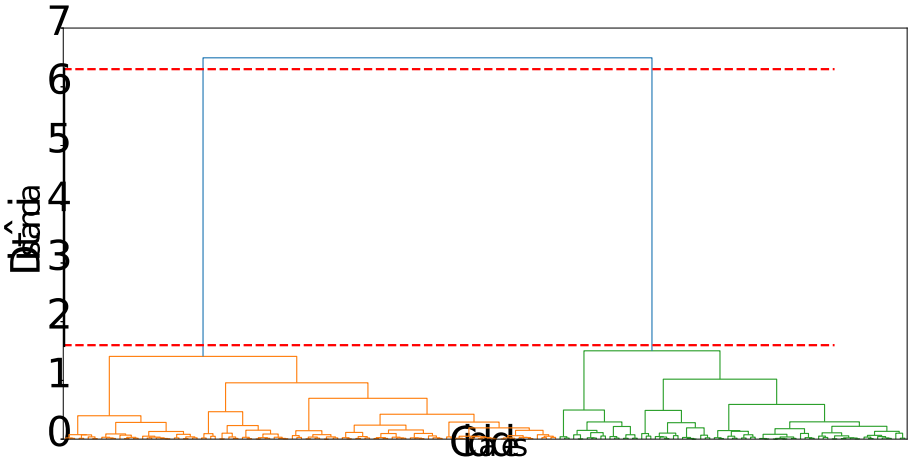
Figura G.2: *Silhouette Coefficient* da clusterização com o algoritmo *K-Means*



Fonte: Elaborado pelo autor.

G.2 Clusterização com *Hierarchical Clustering*

Figura G.3: Dendrograma da clusterização com o algoritmo *Hierarchical Clustering*



Fonte: Elaborado pelo autor.

Uma regra adotada para identificar o número de *clusters* deste algoritmo é selecionar a maior distância no eixo y, em que não existem linhas verticais interseccionadas por linhas horizontais (mostradas como vermelhas na Figura G.3). Adotando esta regra, o número de *clusters* indicado de ser adotado é 2, justamente pela topologia ser visivelmente separável em duas regiões geográficas (Figura 6.4a). Diferentemente do algoritmo K-*Means*, o *Hierarchical Clustering* minimiza a variância dos *clusters* aninhados, e não a soma dos quadrados intra-clusters (Pedregosa et al., 2011), assim, as técnicas *Elbow Method* e *Silhouette Coefficient* não são adequadas para a análise deste algoritmo. Depreende-se que a clusterização gerada pelo *Hierarchical Clustering* pode ser tendenciosa em tal cenário, justamente pela ampla separação geográfica existente na topologia, em qual os *endpoints* que estão distribuídos fisicamente entre dois continentes, fator que induz a análise do dendrograma a gerar o valor $K = 2$.

G.3 Clusterização com DBSCAN

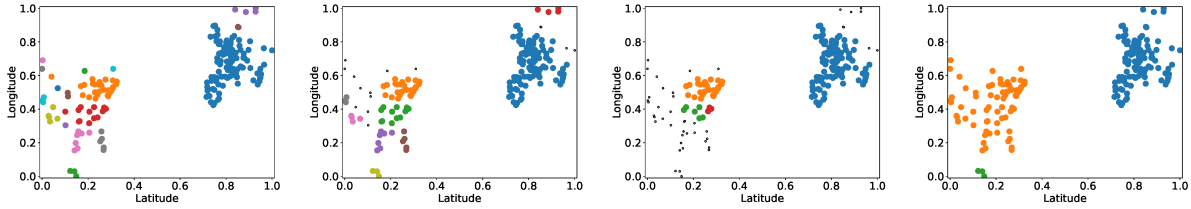
Similarmente ao K-*Means* e ao *Hierarchical Clustering*, o DBSCAN é um algoritmo alternativo para identificar *clusters* em um conjunto de dados. Diferentemente das técnicas citadas anteriormente, o DBSCAN gera uma definição automática do número de *clusters*, e possui dois hiperparâmetros principais para serem ajustados, o EPS e o MPts. Por padrão, a biblioteca *Sklearn* (Pedregosa et al., 2011) define o EPS como 0.5. Neste trabalho tal valor de 0.5 foi usado como referência de centralidade, sendo experimentados alguns valores adjacentes intervalados em 0.05, sendo EPS $\in \{0.05, 0.10, ... , 1\}$. Para o parâmetro MPts, a biblioteca *Sklearn* (Pedregosa et al., 2011) define como padrão o valor $MPts = 5$, neste caso os valores explorados foram os adjacentes a 5 com um intervalo de 1, sendo MPts $\in \{1, 2, ..., 10\}$. Com base em uma análise visual prévia, alguns *clusters* gerados são reportados na Figura G.4.

Diferentemente do K-*Means*, o DBSCAN possui a característica de realizar a clusterização com base em conceitos de densidade, e trabalha com princípios de detecção de ruídos. Tais ruídos são nós não clusterizados, representados pelos pontos pretos na Figura G.4. Devido a estes conceitos de densidade e ruído, a métrica WCSS não pode ser aplicada neste algoritmo, nem o dendrograma, pois eles avaliam aspectos diferentes. Em uma análise inicial, em conformidade com o comportamento descrito por Singh and Meshram (2017), o DBSCAN gerou resultados ruins para conjuntos de dados, devido à densidade de pontos variada em algumas regiões da topologia analisada. Percebe-se que a geração dos melhores *clusters* no cenário avaliado foi com o parâmetro EPS mantido abaixo de 0.1 (Figura G.5a). Mesmo assim, os *clusters* gerados são muito disformes, e houve uma

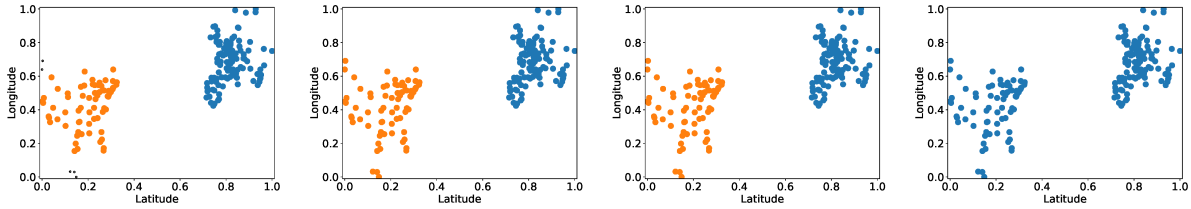
presença grande de pontos classificados erroneamente como ruídos, mas que, na verdade, são servidores físicos da rede, e não podem ser descartados.

Figura G.4: Visualização da clusterização realizadas com o algoritmo DBSCAN

(a) $EPS = 0.05$ $MinPts = 1$ (b) $EPS = 0.05$ $MinPts = 3$ (c) $EPS = 0.05$ $MinPts = 5$ (d) $EPS = 0.1$ $MinPts = 3$

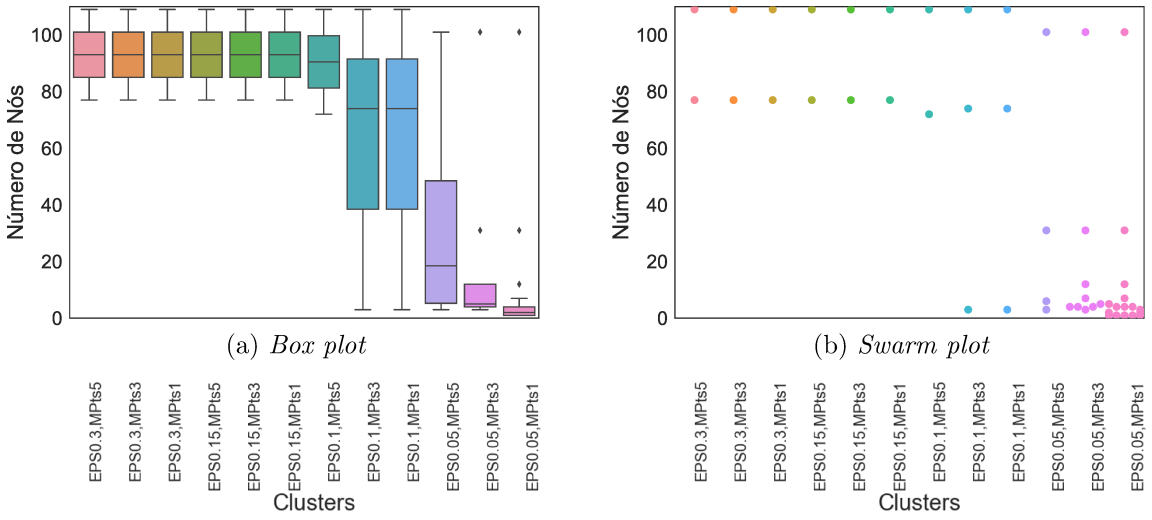


(e) $EPS = 0.1$ $MinPts = 5$ (f) $EPS = 0.3$ $MinPts = 3$ (g) $EPS = 0.3$ $MinPts = 10$ (h) $EPS = 0.4$ $MinPts = 10$



Fonte: Elaborado pelo autor.

Figura G.5: Variação no número de nós dos *clusters*, gerado com o DBSCAN



(a) *Box plot* (b) *Swarm plot*

Fonte: Elaborado pelo autor.

Em todos os experimentos feitos com as variações dos parâmetros EPS e MPts, sempre houve uma alta dispersão nos pontos, como mostrado nos diagramas *Box plot* (Figura G.5a). Tal algoritmo gerou *clusters* disformes, com uma densidade média muito díspar, além de em alguns momentos gerar erros de clusterização, atribuindo como ruído

um servidor físico do SN. Devido a este comportamento ruim, os experimentos com este algoritmo de clusterização não serão continuados, e será dado foco no algoritmo K-*Means*.

Apêndice H

Análise Complementar dos Experimentos do Capítulo 6

H.1 Lucro Total

Similarmente aos experimentos realizados na Subseção 4.2.2, o lucro gerado pelos algoritmos de mapeamento é influenciado diretamente pela taxa de aceitação (Figura 6.8). O algoritmo ILP, por realizar o processamento de cada SFC considerando o SN residual completo, dados os baixos custos de arcos, em alguns casos realiza o mapeamento de uma SFC de modo a priorizar o compartilhamento de servidores e reduzir potenciais aumentos nos custos de mapeamento. Verifica-se que essa redução no uso de diferentes servidores é de até $\approx 10.7\%$ do algoritmo ILP em relação ao algoritmo ILPS12HC (Tabela H.3). Esse aumento no compartilhamento no número de servidores implica em um uso maior de arcos, em que o algoritmo ILP utiliza $\approx 13.8\%$ mais arcos físicos que o algoritmo ILPS12HC (Tabela H.4). Pondera-se que esse comportamento de centralização de consumo de recursos em nós, de modo a aumentar o compartilhamento de servidores, pode potencialmente fragmentar o SN, fator que pode afetar negativamente a taxa de aceitação em cenários com uma chegada e saída mais acentuada de SFCs.

H.2 Taxa de Aceitação

Tabela H.1: Figura 6.8a estratificada em classes de serviços, mostrando as médias e o intervalo de confiança ($IC = 95\%$)

Número de	classe 1		classe 2		classe 3	
clusters	Média	IC	Media	IC	Média	IC
1	73,61%	4,95%	84,24%	7,26%	78,60%	6,43%
2	77,23%	4,35%	89,84%	3,40%	83,80%	2,65%
4	71,69%	7,12%	82,08%	9,46%	77,00%	8,83%
6	69,88%	6,65%	81,68%	8,42%	75,70%	8,18%
8	69,52%	8,34%	82,00%	8,85%	76,70%	8,65%
10	73,13%	3,45%	84,96%	5,00%	80,30%	5,03%
12	74,34%	4,08%	86,48%	3,68%	81,20%	4,13%
14	68,19%	2,27%	88,72%	4,17%	81,90%	5,56%
16	72,89%	7,74%	86,08%	3,13%	79,30%	4,50%

Fonte: Elaborado pelo autor.

Tabela H.2: Figura 6.8b estratificada em classes de serviços, mostrando as médias e o intervalo de confiança ($IC = 95\%$)

Número de	classe 1		classe 2		classe 3	
clusters	Média	IC	Media	IC	Média	IC
1	73,61%	4,95%	84,24%	7,26%	78,60%	6,43%
2	77,23%	4,35%	89,84%	3,40%	83,80%	2,65%
4	76,75%	4,34%	89,68%	3,59%	84,00%	3,16%
6	74,58%	3,69%	87,84%	4,35%	82,20%	5,41%
8	74,82%	5,68%	88,00%	4,43%	81,40%	4,58%
10	71,08%	6,29%	85,28%	7,17%	79,20%	7,37%
12	73,13%	4,84%	86,16%	4,83%	81,00%	4,46%
14	74,22%	6,81%	86,96%	6,45%	80,80%	6,23%
16	63,98%	3,31%	82,24%	3,59%	77,80%	4,99%

Fonte: Elaborado pelo autor.

H.3 Espalhamento dos Nós Virtuais

Um resumo numérico da Figura 6.9 pode ser visto na Tabela H.3. O algoritmo ILP obteve um espalhamento de nós virtuais ligeiramente melhor em comparação com os outros algoritmos, principalmente no começo dos mapeamentos. Como exemplo, o espalhamento de nós do algoritmo ILP é $\approx 6.23\%$ maior em relação ao algoritmo ILPs12KM, e $\approx 10.71\%$ maior em relação ao algoritmo ILPs12HC (Tabela H.3). Neste caso, por o algoritmo ILP processar todos os nós físicos para gerar um mapeamento, é realizado um mapeamento com um maior compartilhamento de servidores em detrimento a roteamentos mais alongados sobre o SN. Por outro lado, o algoritmo que realiza os mapeamentos nos SN induzidos, pode em algum momento não compartilhar um servidor já ativo, que está sendo usado por outra SFC, justamente por ele não pertencer ao $\overline{SN}$ induzido processado.

Tabela H.3: Resumo do Gráfico 6.9

Média ($\overline{x}$)		
ILP	ILPs12KM	ILPs12HC
53.58%	57.14%	60.01%
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP-ILPs12KM	ILP-ILPs12HC	ILPs12KM-ILPs12HC
−6.23%	−10.71%	−4.78%

Fonte: Elaborado pelo autor.

H.4 Espalhamento dos Arcos Virtuais

Estes experimentos mostram o quão conciso ou difuso é o roteamento médio de arcos virtuais sobre SN. Infere-se que um algoritmo que gere um alto espalhamento de arcos em um cenário com uma frequência alta de chegada e saída de SFCs, potencialmente fragmentará mais o SN, pois consumirá recursos de um maior número de arcos físicos. Um resumo numérico da Figura 6.10 pode ser visto na Tabela H.4.

Tabela H.4: Resumo do Gráfico 6.10

Média ($\overline{x}$)		
ILP	ILPs12KM	ILPs12HC
5.31%	4.33%	4.66%
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP-ILPs12KM	ILP-ILPs12HC	ILPs12KM-ILPs12HC
22.57%	13.81%	−7.15%

Fonte: Elaborado pelo autor.

H.5 Atraso Médio Fim a Fim

Um resumo numérico da Figura 6.11 pode ser visto na Tabela H.5. Percebe-se que o algoritmo tradicional, com uma visão global do SN, centraliza os mapeamentos sobre os nós físicos já ativos (Figura 6.11 e Tabela H.4), e pode eventualmente mapear uma SFC de forma espalhada sobre os arcos do SN, gerando um alto atraso fim a fim (Tabela H.5), o que pode afetar a qualidade da experiência do usuário final. Como exemplo, o atraso fim a fim gerado pelo algoritmo ILPs12KM é $\approx 14.8\%$ menor se comparado ao algoritmo ILP, e 6.92% mais baixo se comparado ao algoritmo ILPs12HC.

Tabela H.5: Resumo do Gráfico 6.11

Média ($\overline{x}$)		
ILP	ILPs12KM	ILPs12HC
$93.52ms$	$81.462ms$	$87.517ms$
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP-ILPs12KM	ILP-ILPs12HC	ILPs12KM-ILPs12HC
14.80%	6.85%	-6.92%

Fonte: Elaborado pelo autor.

Apêndice I

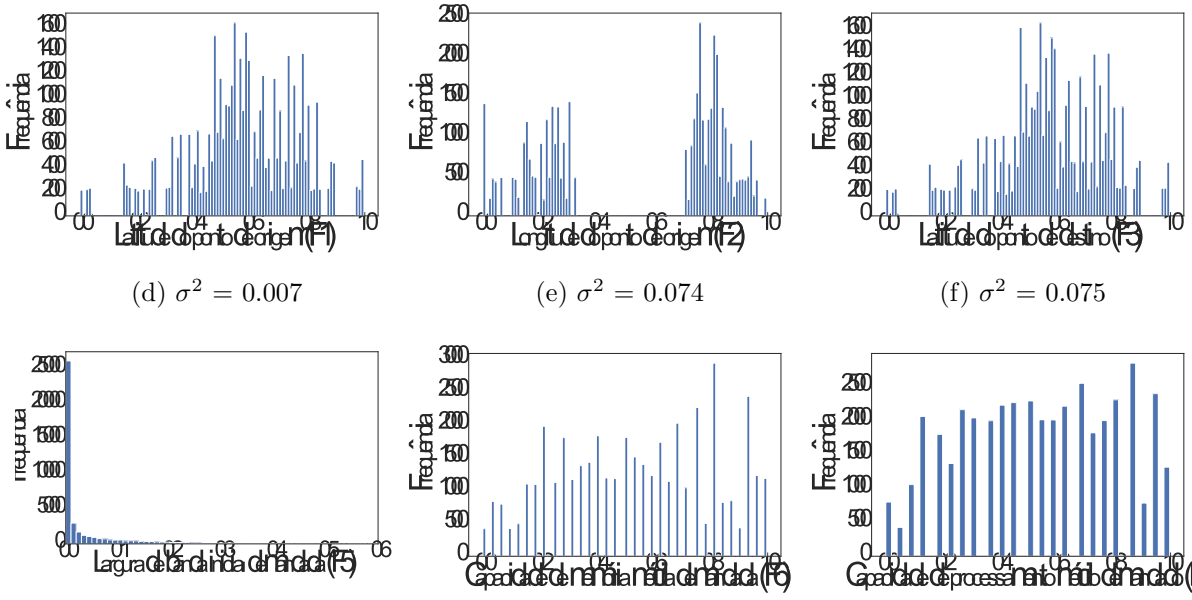
Análise Complementar dos Experimentos do Capítulo 7

I.1 *Dataset*: Análise de Variância

Neste caso definimos um limite de variação (*Variance Threshold*) para determinar quais recursos devem ser removidos. A definição deste *Threshold* é feita com base em vários experimentos preliminares, onde é analisado o impacto que a retirada de referida característica gera na aplicação proposta. Estipulou-se um limite de 0.02, onde características abaixo deste limite foram excluídas por implicarem em uma baixa variação de dados. As Figuras I.1 e I.2 mostram o cálculo de cada respectiva variância.

Figura I.1: Variação das caraterísticas do *dataset* (parte I)

(a) $\sigma^2 = 0.037$ (b) $\sigma^2 = 0.103$ (c) $\sigma^2 = 0.036$

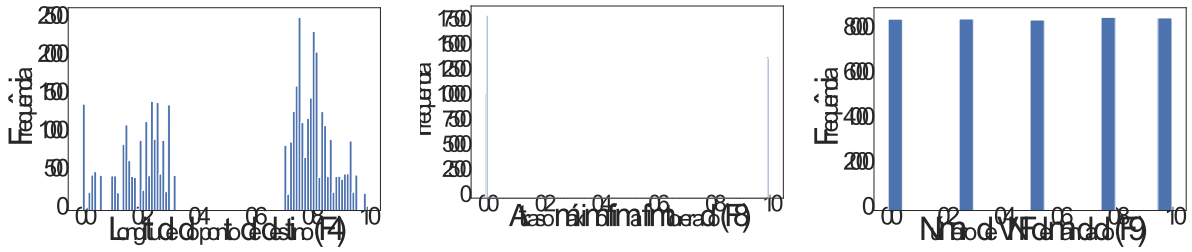


(d) $\sigma^2 = 0.007$ (e) $\sigma^2 = 0.074$ (f) $\sigma^2 = 0.075$

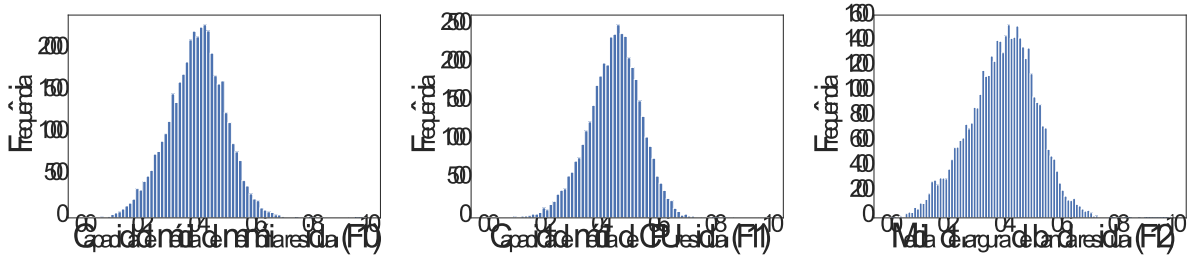
Fonte: Elaborado pelo autor.

Figura I.2: Variação das caraterísticas do *dataset* (parte II)

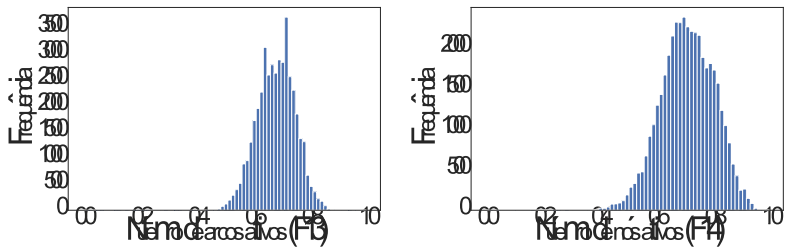
(a) $\sigma^2 = 0.102$ (b) $\sigma^2 = 0.219$ (c) $\sigma^2 = 0.125$



(d) $\sigma^2 = 0.01$ (e) $\sigma^2 = 0.009$ (f) $\sigma^2 = 0.014$



(g) $\sigma^2 = 0.005$ (h) $\sigma^2 = 0.01$



Fonte: Elaborado pelo autor.

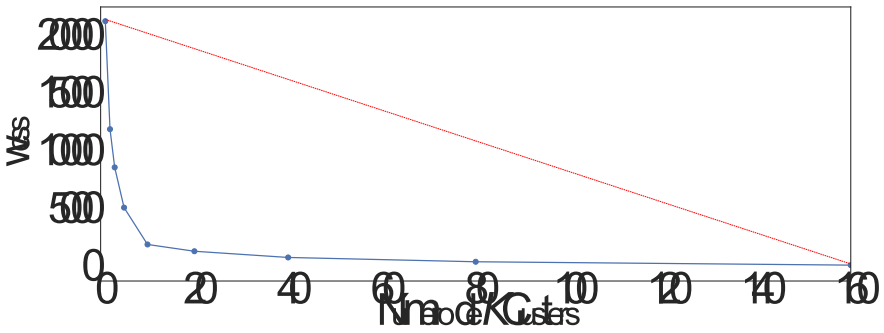
I.2 Clusterização: Análise Complementar

Inicialmente, é aplicado o *Elbow Method* de modo a detectar em qual valor de k a métrica WCSS se torna mais atrativa (Figura I.3). Tal variação é mostrada com o número de *clusters* até $k = 160$, pois valores maiores geraram alterações significativas. Pode-se observar na Figura I.3 que a queda na soma da distância ao quadrado, começa a reduzir menos significantemente após $k = 5$ (ponto mais distante da reta vermelha). Portanto, o ponto de equilíbrio entre a maior homogeneidade no *cluster* e a maior diferença entre *clusters*, chamado cotovelo, é justamente para o valor de $k = 5$.

Outra métrica que pode ser aplicada para a análise do valor de K, é o *Silhouette Coefficient*. Semelhantemente à Figura I.3, para melhorar a visualização, a Figura I.4 foi

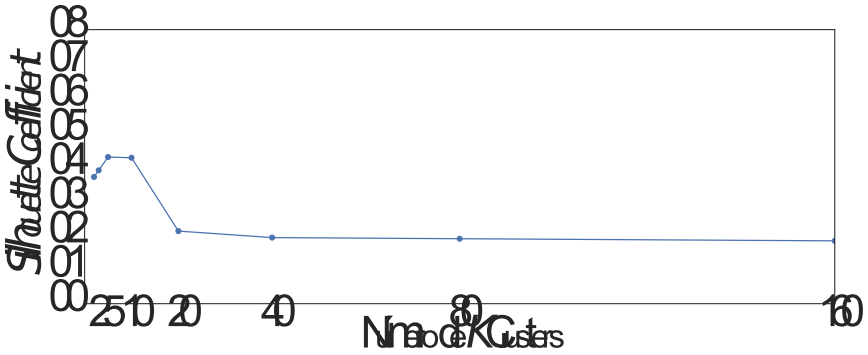
mostrada com a variação do número de *clusters* até $k = 160$, pois valores maiores não geraram alterações expressivas. Observando a Figura I.4, infere-se que à medida que o valor de k aumenta, a tendência é que o *Silhouette Coefficient* reduza. Ou seja, quanto maior o k, menor a inércia, e proporcionalmente menor o *Silhouette Coefficient*.

Figura I.3: WCSS (*Elbow Method*) da clusterização com o algoritmo *K-Means*



Fonte: Elaborado pelo autor.

Figura I.4: *Silhouette Coefficient* da clusterização com o algoritmo *K-Means*



Fonte: Elaborado pelo autor.

Sob um olhar estatístico, as métricas WCSS e *Silhouette Coefficient* indicam que um bom valor a ser usado na clusterização é $k = 5$. Contudo, deve-se considerar qual impacto que essas escolhas do valor de k causam na redução de componentes a serem processados no mapeamento de cada SFC.

I.3 Ponderação de Valores: Análise Complementar

A Tabela I.1 mostra uma versão completa da Tabela 7.2, e mostra também o impacto da variação deste parâmetro no mapeamento das SFCs entrantes. Do mesmo modo

que ao se omitir as características $F8$ e $F12$, a taxa de rejeição aumenta ao se aplicar pouco peso nas referidas ponderações. No caso de um valor de Υ muito baixo, as caraterísticas $F8$ e $F12$ assumem um valor mais significativo, ficando na mesma ordem de grandeza que as caraterísticas $F1$ a $F4$. Assim, o cálculo das distâncias entre as caraterísticas que irão gerar os SN induzidos é gerado de forma igualitária, o que não implicou em resultados promissores. Este aspecto acontece pelas caraterísticas relacionadas à localização geográfica serem fundamentais na viabilidade do problema. Ao se dar o mesmo peso às características $F1$ a $F4$, e $F8$ e $F12$, gera-se SNs induzidos com rotas que potencialmente são benéficas para otimizar as restrições de largura de banda e atraso fim a fim, mas que em muitos casos não são factíveis em relação à localização geográfica dos componentes.

Tabela I.1: Impacto da variação do parâmetro Υ com k fixado em 2000 (IC= 95%)

Variação de parâmetro	Taxa de aceitação (%)	IC	Tempo de processamento por SFC	IC	Lucro ($)	IC
ILPr2000 sem $f8$ e $F12$	79,99	0,98	1,024	0,091	2758,072	253,040
ILPr2000 e Υ=100	84,45	1,18	1,043	0,087	2773,229	209,683
ILPr2000 e Υ=90	84,29	1,21	1,063	0,073	2839,628	218,421
ILPr2000 e Υ=80	84,06	0,54	1,116	0,109	2833,876	219,013
ILPr2000 e Υ=70	84,19	0,83	1,083	0,077	2820,203	178,202
ILPr2000 e Υ=60	84,29	1,07	1,100	0,063	2873,234	174,249
ILPr2000 e Υ=**50**	**84,58**	**0,73**	**1,062**	**0,103**	**2885,648**	**196,765**
ILPr2000 e Υ=40	83,81	1,01	1,122	0,132	2712,138	202,157
ILPr2000 e Υ=30	83,38	1,25	1,115	0,162	2792,027	187,742
ILPr2000 e Υ=20	82,89	0,88	1,100	0,082	2831,078	193,356
ILPr2000 e Υ=10	80,23	0,89	1,391	0,764	2726,598	153,586
ILPr2000 e Υ=1	6,46	1,80	0,092	0,020	171,452	77,488
Abordagem ILP^i	81,40	5,87	1,763	0,167	2828,250	155,592

Fonte: Elaborado pelo autor.

Ponderando as caraterísticas $F8$ e $F12$ com valores intermediários, como $\Upsilon = 50$, o algoritmo gera uma maior taxa de aceitação em relação aos outros experimentos, e mantêm os lucros altos. Nota-se que o algoritmo ILPr2000 com $\Upsilon = 50$ possui uma taxa de aceitação média de $84,58\%$ e um IC de apenas $0,73\%$. Em contrapartida, ao algoritmo tradicional ILP^i possui uma taxa de aceitação média de $81,4\%$ e um IC alto, de $5,87\%$. Contudo, como os ICs envolvem a mesma faixa de valor, não se pode afirmar que o algoritmo ILPr2000 e $\Upsilon = 50$ tende a se comportar melhor que o algoritmo ILP^i em termos de taxa de aceitação e lucro. Mas, por outro lado, em relação ao tempo de processamento, o algoritmo ILPr2000 foi $\approx 66\%$ mais rápido, mostrando que os mapeamentos de SFCs em SNs induzidos, não afetam negativamente os lucros, nem a taxa de aceitação, mas pode reduzir significativamente o tempo de processamento em relação a um algoritmo tradicional.

I.4 Taxa de Aceitação

Analisando a taxa de aceitação da abordagem de Clusterização por Consumo de Recursos por extratos de classes de serviços (Tabela I.2), percebe-se que a variação observada não é estatisticamente significante para os valores de k analisados. Tal percepção reflete políticas neutras de mapeamento das SFCs, onde o mapeamento das SFCs tem a mesma prioridade, *i.e.*, nenhum NS é privilegiado em relação a outro. Tal percepção é importante, pois mostra que a abordagem não prioriza o mapeamento dos serviços da classe 2, sendo teoricamente mais simples, em prol de obter uma taxa de aceitação mais alta. Percebe-se que as abordagens propostas, apesar de reduzirem o espaço de otimização em relação ao modelo de otimização com visão completa do SN, induzem à geração de políticas promissoras para o mapeamento de uma nova SFC, generalizando bem para os diferentes cenários avaliados, não diminuindo a taxa de aceitação.

Tabela I.2: Taxa de aceitação mostrada na Figura 7.7a, estratificada em classes de serviços, mostrados as médias e o IC de 95% (IC)

Número de	classe 1		classe 2		classe 3	
clusters	Média (%)	IC(%)	Media (%)	IC (%)	Média (%)	IC (%)
ILP	73,61	4,95	84,24	7,26	78,60	6,43
500	79,76	3,01	92,72	1,39	85,00	1,55
1000	78,43	2,05	92,56	1,30	85,70	1,58
1500	78,19	1,84	92,00	1,62	84,30	1,51
2000	77,47	1,63	89,68	1,51	84,20	1,99
2500	77,11	2,96	89,12	1,65	83,30	1,88
3000	75,42	1,48	88,72	1,09	82,20	1,00
3500	74,10	3,10	88,80	1,14	80,40	1,82
4000	71,93	2,57	87,60	2,37	78,90	1,52

Fonte: Elaborado pelo autor.

I.5 Espalhamento dos Nós Virtuais

Um resumo numérico da Figura 7.8 pode ser visto na Tabela I.3.

Tabela I.3: Resumo do Gráfico 7.8

Média ($\overline{x}$)		
ILP	ILPs12	ILPr3000
55.60%	57.34%	51.39%
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP-ILPs12	ILP-ILPr3000	ILPs12-ILPr3000
−3.03%	8.19%	11.57%

Fonte: Elaborado pelo autor.

I.6 Espalhamento dos Arcos Virtuais

Um resumo numérico da Figura 7.9 pode ser visto na Tabela I.4.

Tabela I.4: Resumo do Gráfico 7.9

Média ($\overline{x}$)		
ILP	ILPs12	ILPr3000
5.29%	4.50%	5.02%
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP-ILPs12	ILP-ILPr3000	ILPs12-ILPr3000
17.57%	5.44%	−10.32%

Fonte: Elaborado pelo autor.

I.7 Atraso Médio Fim a Fim

Um resumo numérico da Figura 7.10 pode ser visto na Tabela I.5.

Tabela I.5: Resumo do Gráfico 7.10

Média ($\overline{x}$)		
ILP	ILPs12	ILPr3000
100.80ms	88.36ms	100.32ms
Variação percentual ($\frac{v_2-v_1}{v_1} \times 100$)		
ILP-ILPs12	ILP-ILPr3000	ILPs12-ILPr3000
14.07%	0.47%	−11.92%

Fonte: Elaborado pelo autor.

I.8 Divergência de KL: Introdução

Antes de introduzir a divergência de KL, deve-se fundamentar o conceito de entropia. A entropia é uma métrica que fornece uma medida da incerteza, também chamada desordem, contida em uma distribuição de probabilidade (Nalewajski, 2006). A divergência de KL é uma métrica de entropia, baseada na divergência de uma distribuição de dados atual, e outra verificada em outro momento (Kullback and Leibler, 1951). Neste trabalho, a divergência de KL é usada como uma ferramenta para medir a diferença entre duas distribuições de probabilidade, *i.e.*, se ao processar uma nova SFC, os requisitos envolvidos no mapeamento desta SFC alteram as distribuições de probabilidades envolvidas no

treinamento do modelo no passado.

A divergência de KL é uma técnica baseada em uma função logarítmica que generaliza a entropia informacional clássica, e está diretamente relacionada à métrica de informação de Fisher (Nalewajski, 2006). A métrica de informação de Fisher é uma forma de medir a quantidade de informação que uma variável aleatória possui. A divergência de KL tornou-se um marco na teoria da probabilidade e na teoria da informação, com aplicações nas mais diferentes áreas, como: Gong et al. (2021); Karampasi et al. (2020); Sakamoto et al. (2015). De forma simplificada, a divergência de KL, quando aplicada para medir a diferença entre duas distribuições, informa uma quantidade de entropia perdida ou adicionada quando são alterados os dados.

Adotando $P(x)$ e $Q(x)$ respectivamente como as distribuições de probabilidade anterior e posterior à chegada de uma SFC, de uma característica x ser demandada. A divergência de KL pode ser representada por:

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) log \frac{P(x)}{Q(x)} \tag{I.1}$$

(para o caso de uma distribuição discreta de probabilidade)

I.9 Análise dos Momentos de Retreinamento

I.9.1 Momento 1

Pela função objetivo adotada, quanto mais SFC forem mapeadas, maior será a receita gerada, junto a uma maior probabilidade de compartilhamento de servidores e redução de custos. Por certo, quanto mais SFCs estão ativas simultaneamente, e como os servidores possuem limites de memória e processamento, maior a probabilidade de novos servidores serem ativados para atender os novos pedidos. Complementando o raciocínio, observando a Tabela I.6 e a Figura I.5, percebe-se que ao final das $50000t$, o atendimento das SFCs com o algoritmo ILP até $20000t$, seguido do algoritmo ILPkr entre $2000t$ e $50000t$, geraram um lucro final $\approx 7.15\%$ maior que o processamento realizado somente com o algoritmo ILP. Baseado nos resultados positivos, percebe-se que o instante $20000t$, indicado pela divergência de KL, define assertivamente o momento de se treinar o modelo.

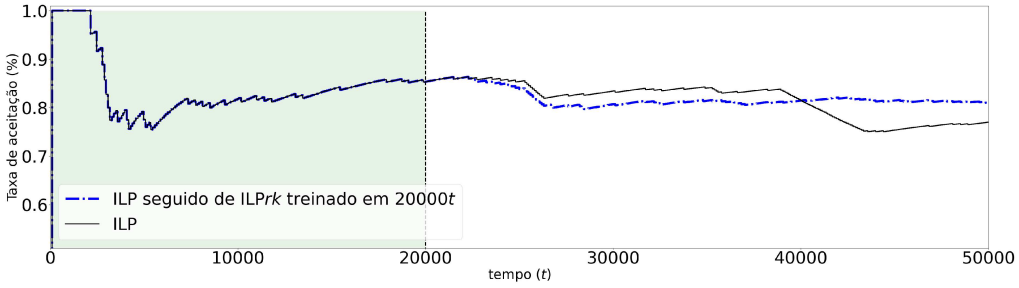
Tabela I.6: Dados dos mapeamentos das SFCs de $0t$ a $50000t$

			SFCs mapeadas (%)		
Algoritmo	Tempo de execução (seg)	Lucro ($)	classe 1	classe 2	classe 3
ILP	424	7160	0	0	77.26
ILP seguido de ILPrk treinado em 20000	369	7672	0	0	81.28

Fonte: Elaborado pelo autor.

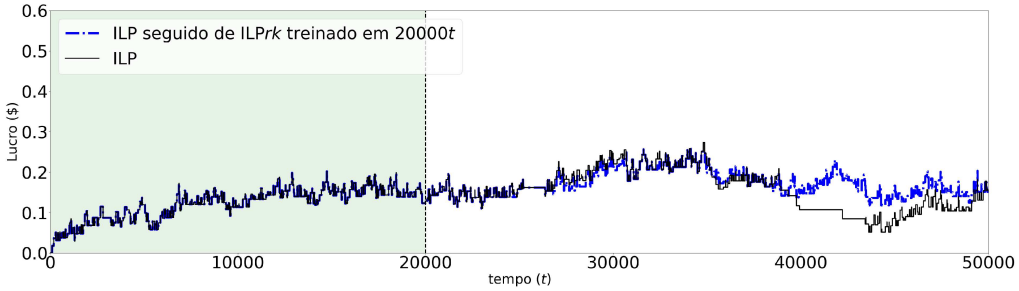
Análise do lucro: comparando as Figuras I.5 e I.6, percebe-se uma relação forte entre a taxa de aceitação e o lucro. Pela função objetivo adotada, quanto mais requisição forem mapeadas, maior será a receita gerada, e maior também a probabilidade de compartilhamento de servidores, potencialmente reduzindo os custos. Por certo, quanto mais SFCs estão ativas simultaneamente, e como os servidores possuem limites de memória e processamento, maior a probabilidade de novos servidores serem ativados para atender os novos pedidos. Complementando o raciocínio, observando a tabela I.6, percebe-se que ao final das $50000t$, o atendimento das SFCs com o algoritmo ILP até $20000t$, seguido do algoritmo ILPkr entre $2000t$ e $50000t$, geraram um lucro final $\approx 7.15\%$ maior que o processamento realizado somente com o algoritmo ILP. Baseado nos resultados positivos, percebe-se que o instante $20000t$, indicado pela divergência de KL, define assertivamente o momento de se treinar o modelo.

Figura I.5: Impacto do retreinamento na taxa de aceitação



Fonte: Elaborado pelo autor.

Figura I.6: Impacto do retreinamento no lucro, computado até $50000t$



Fonte: Elaborado pelo autor.

I.9.2 Momento 2

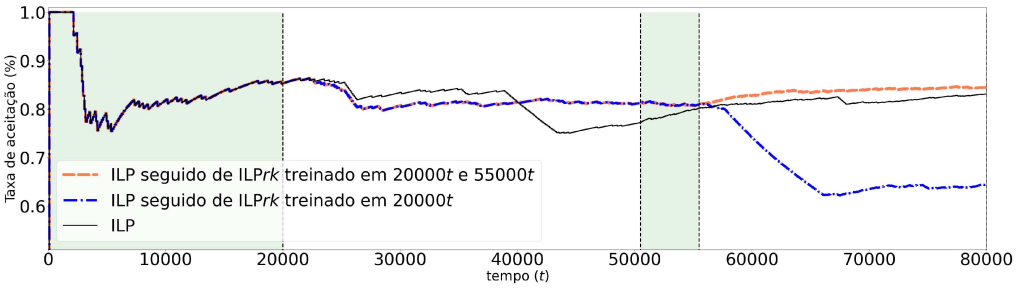
Análise do lucro: pela função objetivo adotada, quanto mais SFC forem mapeadas, maior será a receita gerada, e maior também a probabilidade do compartilhamento de servidores, reduzindo os custos. Com base em tal formulação, o algoritmo que realizou o treinamento somente em $20000t$, por ter uma redução na taxa de aceitação, teve seu lucro reduzido. Perfazendo o raciocínio, a tabela I.7 mostra que ao final das $80000t$, o atendimento das SFCs com o algoritmo ILP até $20000t$, seguido do algoritmo ILPkr entre $2000t$ e $80000t$, geraram um lucro final acumulado ≈ 10.97 superior ao algoritmo treinado somente no instante $20000t$, e ≈ 6.37 superior ao algoritmo exato. Baseado nos resultados promissores, percebe-se que o momento $50000t$, apontado pela divergência de KL, define corretamente o momento de se gerar um novo modelo.

Tabela I.7: Dados dos mapeamentos das SFCs de $0t$ a $80000t$

			SFCs mapeadas (%)		
Algoritmo	Tempo de execução (seg)	Lucro ($)	classe 1	classe 2	classe 3
ILP	1288	13353	0	89.51	80.49
ILP seguido de ILPrk treinado em 20000	871	12799	0	52.68	70.43
ILP seguido de ILPrk treinado em 20000 e 55.000	902	14203	0	90.05	82.36

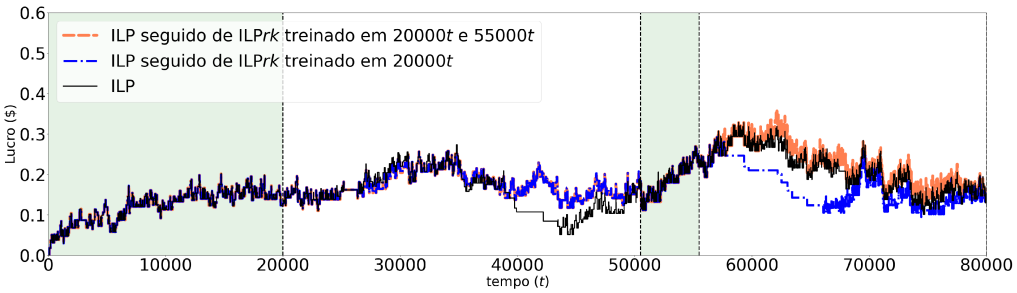
Fonte: Elaborado pelo autor.

Figura I.7: Impacto do retreinamento na taxa de aceitação



Fonte: Elaborado pelo autor.

Figura I.8: Impacto do retreinamento no lucro



Fonte: Elaborado pelo autor.

I.9.3 Momento 3

O algoritmo treinado somente no momento 20000t teve uma taxa de mapeamento regular para o mapeamento das SFCs da classe 1 (Figura I.9). Isso se deve às demandas das SFCs desta classe se aproximarem às da classe 3. Assim, em alguns casos o algoritmo ILPrk, mesmo sem conhecer as SFCs da classe 1, conseguiu realizar os mapeamentos. Por outro lado, o algoritmo retreinado em 20000t e 55000t, apesar de também conhecer as SFCs da classe 3, não teve um desempenho tão satisfatório (39.91%). Neste caso, o algoritmo priorizou os mapeamentos das SFCs da classe 2 em detrimento aos da classe 1.

As SFCs da classe 1 são mais desafiadoras de serem mapeadas, e mesmo com o modelo retreinado, a taxa de aceitação desta classe ficou em $\approx$ 64.51% (Tabela I.8). Deve-se ressaltar que no cenário avaliado, esta classe de serviço possui uma taxa de aceitação reduzida se comparado ao lucro (Figura I.10). Isto acontece por neste momento estarem chegando SFCs das 3 classes no mesmo intervalo de tempo, e seguindo a distribuição apresentada na Subseção 4.1. Logo, existe uma maior disputa por recursos do SN, e, similarmente aos experimentos do Capítulo 3, a taxa de aceitação decresce.

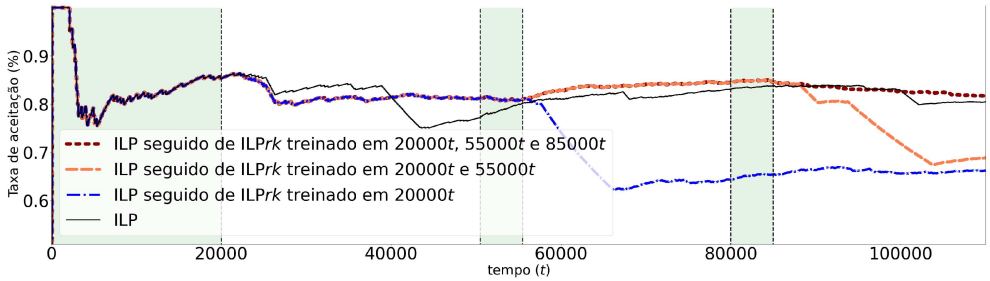
Análise do lucro: similarmente ao reportado nas Figuras I.6 e I.8, o algoritmo que possui a maior taxa de aceitação, gera os maiores lucros (tabela I.8). Tangendo o raciocínio anterior, ao final das 110000t, o atendimento das SFCs com o algoritmo ILP que sempre efetuou o retreinamento gerou um lucro final acumulado $\approx$ 7.23% superior ao algoritmo treinado nos instantes 20000t e 55000t, e $\approx$ 1.2% superior ao algoritmo ILP tradicional (Tabela I.8).

Tabela I.8: Dados dos mapeamentos das SFCs de $0t$ a $110000t$

Algoritmo	Tempo de execução (seg)	Lucro ($)	SFCs mapeadas (%) classe 1	classe 2	classe 3
ILP	3393	23949	64.91	85.18	79.72
ILP seguido de ILPrk retreinado em 20000	2098	22996	58.87	62.21	70.63
ILP seguido de ILPrk retreinado em 20000 e 50000	2191	22601	39.91	70.89	73.63
ILP seguido de ILPrk retreinado em 20000, 50000 e 80000	2164	24236	64.51	86.51	81.90

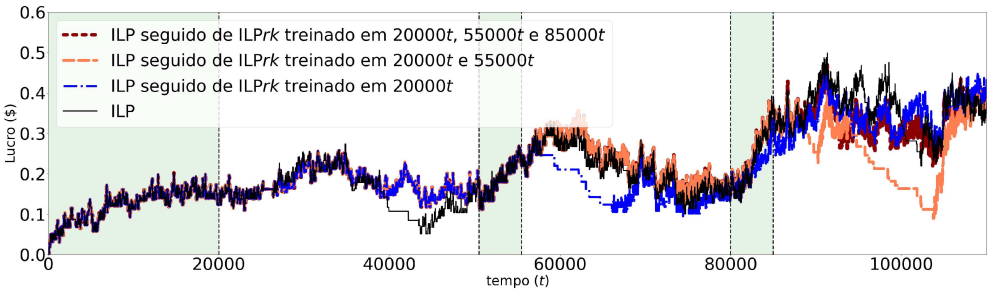
Fonte: Elaborado pelo autor.

Figura I.9: Impacto do retreinamento na taxa de aceitação



Fonte: Elaborado pelo autor.

Figura I.10: Impacto do retreinamento no lucro, computado até $110000t$



Fonte: Elaborado pelo autor.