Uma Abordagem Heurística para o Posicionamento e Encadeamento de Funções Virtuais de Rede em Ambientes *Online*

Samuel M. A. Araújo[1], **Fernanda S. H. de Souza**[2], **Geraldo R. Mateus**[1]

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG) - Belo Horizonte - MG

[2]Departamento de Ciência da Computação
Universidade Federal de São João del-Rei (UFSJ) - São João del-Rei - MG

smaa@dcc.ufmg.br, fsumika@ufsj.edu.br,mateus@dcc.ufmg.br

Resumo. *A Virtualização de Funções de Rede emerge com intuito de reduzir custos operacionais e prover flexibilidade no atendimento dos novos serviços de rede. Um dos principais desafios em tais ambientes é posicionar e encadear com eficiência as funções virtuais de rede sobre máquinas virtuais alocadas em servidores da rede física. Por se tratar de um ambiente online, as abordagens utilizadas devem gerar uma tomada de decisão rápida sobre o mapeamento, ou não, de uma requisição. Neste contexto, uma abordagem baseada nas metaheurísticas Greedy Randomized Adaptive Search Procedure e Variable Neighborhood Search foi proposta, e comparada a um método exato. Experimentos computacionais mostram que o método exato, apesar de gerar maiores lucros, possui um tempo de execução elevado. No entanto, a abordagem heurística apresentou um tempo de execução até* 810% *menor, reduziu o atraso fim a fim em até* 70%*, mas ao custo de uma queda no compartilhamento de servidores de até* 25%*, e no lucro de até* 11%*.*

Abstract. *Network Functions Virtualization has emerged to reduce operating costs and provide flexibility to serve the new network services. One of the main challenges in such environments is the placement and chain of the virtual network functions on virtual machines positioned over servers in the network. As an online environment is addressed, the approaches employed must generate a quick decision-making about the mapping, or not, of a network request. In this context, a solution based on Greedy Randomized Adaptive Search Procedure and Variable Neighborhood Search metaheuristics is proposed, and then compared with an exact method. The computational experiments show that the exact method, despite generating higher profits, has a high runtime. On the other hand, the heuristic approach showed a* 810% *improvement on runtime, reduce the end-to-end delay in* 70%*, but at the cost of a* 25% *decrement in the servers sharing, and* 11% *in profit.*

1. Introdução

Quando são tratados aspectos inerentes à Internet do Futuro, e consequente suporte às suas demandas, as principais preocupações são a flexibilidade e a capacidade de

possíveis alterações na infraestrutura física da rede. Neste sentido, as redes baseadas em virtualização de funções (*Network Functions Virtualization*, NFV) estão propiciando novas possibilidades computacionais, das quais muitas ainda são desconhecidas. Tais redes desvinculam as Funções de Rede (*Network Function*, NF) dos dispositivos de *hardware* legados, e as transfere para servidores virtualizados, que instanciam as NFs através de emulação sobre servidores físicos comerciais [Laghrissi and Taleb 2018]. As NFs podem ser virtualizadas em servidores distribuídos pelo substrato de rede (*Substrate Network*, SN). Cada função de rede virtualizada (*Virtual Network Function*, VNF) é responsável por um tratamento específico a um determinado fluxo de dados, *e.g.*, *Firewall* (FW), *Wide Area Network* (WAN), *Network Address Translation* (NAT), *Proxy*, *Intrusion Detection System* (IDS) [Yi et al. 2018, Laghrissi and Taleb 2018].

O problema de Posicionamento e Encadeamento de VNFs (VNF *Placement* and *Chaining*, VNF-PC) é intrínseco às arquiteturas de redes baseadas em NFV. O VNF-PC tem como desafio posicionar as VNFs demandadas por um cliente sobre servidores existentes no SN, e gerar um roteamento encadeado de tais VNFs de modo a prover um serviço de rede completo (*Network Service*, NS). Tal problema recebe como entrada uma sequência encadeada de VNFs (*Service Function Chaining*, SFC). Como exemplo, um NS que provê aspectos de segurança pode requerer que sejam mapeadas sequencialmente (encadeadas) as VNFs *proxy*, FW, IDS e antivírus [Jia et al. 2018].

As dificuldades de resolução do VNF-PC estão ligadas à sua natureza combinatória, sendo o problema pertencente à classe NP-difícil [Cohen et al. 2015]. Dados os requisitos dinâmicos das redes baseadas em NFV, existe uma necessidade de propostas de alocação de recursos que sejam capazes de encontrar soluções rápidas e *online* [Mijumbi et al. 2016]. Neste caso, cenários *online* são caracterizados pela chegada das SFCs de maneira desconhecida, onde não se sabe nenhum aspecto sobre a topologia, tempo de vida e recursos demandados. Devido à complexidade de resolução do VNF-PC, e sua função intrínseca no desenvolvimento de novas tecnologias com suporte a NFV, ele torna-se ideal para ser estudado e tratado com uso de heurísticas.

No mapeamento de SFCs em um contexto de computação em nuvem, um roteamento de VNFs com um atraso fim a fim maior, embora permita que o usuário execute um NS de maneira razoável, pode resultar em uma insatisfação em relação à qualidade de experiência percebida (*Quality of Experience*, QoE). Contextualizando tal afirmação, ao se efetuar uma alocação de recursos para determinado NS, o provedor de serviço recebe uma SFC para ser mapeada sobre o SN. Neste momento, percebe-se um *trade-off* entre bons roteamentos e o número de servidores compartilhados. Um posicionamento ótimo de VNFs para atender às demandas das SFCs com o menor número de instâncias de VNFs possível estimula o compartilhamento de servidores, e pode gerar custos mais baixos. De outra forma, pode-se encaminhar o tráfego pelo caminho mais curto possível, priorizando o roteamento e um baixo atraso fim a fim, e implantando as VNFs sempre que necessário. Esta opção induz roteamentos com menos saltos, mas certamente levará a maiores custos, pois potencialmente diminuir-se-ia o número de servidores compartilhados em detrimento a aspectos de QoE. Dadas as afirmações anteriores, depreendem-se as seguintes questões de pesquisa: *i*) Como os encadeamentos de VNFs, se realizados de modo exato (com garantia de otimalidade da solução) e heurístico, afetam o lucro dos servidores e o atraso fim a fim? *ii*) Em quais situações

o encadeamento de VNFs com menos saltos altera a taxa de aceitação? Para investigar tais questões, este artigo tem como principal contribuição uma abordagem heurística de rápida execução, baseada nas metaheurísticas *Greedy Randomized Adaptive Search Procedure* (GRASP) [Feo and Resende 1995] e Variable *Neighborhood Search* (VNS) [Mladenović and Hansen 1997] para a resolução do VNF-PC em um cenário *online*, e com restrições de atraso, precedência de VNFs, memória, CPU e largura de banda. A abordagem proposta realiza uma reserva de recursos do SN para garantir o atendimento de cada requisição, e emprega políticas de roteamento que estimulam o mapeamento das VNFs demandadas em regiões próximas aos pontos de origem e destino de tal SFC. Tais políticas são estabelecidas em estruturas de vizinhança, que possuem o intuito de gerar uma convergência mais rápida para a solução final. O princípio de tais estruturas é guiar a busca local a regiões do espaço de soluções que potencialmente possuem um atraso fim a fim dentro de um limite tolerado pela SFC, não investigando soluções infactíveis.

O restante do artigo está organizado como segue. Uma breve revisão dos trabalhos da literatura é apresentada na Seção 2. O VNF-PC é formalizado na Seção 3. A heurística é apresentada na Seção 4. As métricas e a análise de desempenho são mostradas nas Seções 5 e 6. Por fim, as conclusões e trabalhos futuros são discutidos na Seção 7.

2. Trabalhos Relacionados

A adoção das tecnologias baseadas em NFV começou em 2012 e ainda está em andamento, com pesquisas e desenvolvimento em ritmo acelerado [NFV White Paper 2012, Laghrissi and Taleb 2018, Yi et al. 2018, Li et al. 2020]. Segundo Yi *et al*., devido ao tempo de execução de uma heurística ser inferior ao de uma abordagem ótima, tal linha algorítmica é mais indicada para a resolução do VNF-PC. Este trabalho segue essa linha heurística, mas em vez de propor uma heurística gulosa, ou com estruturas de vizinhanças tradicionais, opta-se por empregar conceitos de redução do espaço de solução através de metaheurísticas com estruturas de vizinhanças bem definidas, explorando regiões promissoras do espaço de busca do problema.

De acordo com [Khoshkholghi et al. 2019], um dos principais desafios na resolução VNF-PC é reduzir os custos. Os autores definem os custos como os gastos com recursos de computação e comunicação necessários para prover um NS. Khoshkholghi *et al.* propõem dois algoritmos genéticos utilizando métodos de seleção por roleta e torneio. O objetivo das abordagens de Khoshkholghi *et al.* é reduzir tais custos. Similarmente, neste trabalho buscamos reduzir os custos dos provedores, mas em um cenário *online*, e analisando um *trade-off* entre a receita e o custo, de modo a maximizar o lucro final.

Os autores de [Bari et al. 2019] desenvolveram um orquestrador de SFCs, baseado em busca Tabu, com o objetivo de reduzir a emissão de carbono. Tal abordagem atua em um cenário *online*, e efetua o mapeamento de SFCs em locais que estimulem o compartilhamento de recursos físicos, permitindo que servidores não utilizados sejam colocados em espera. Bari *et al.* também propõem uma abordagem baseada em Programação Linear Inteira (*Integer Linear Programming*, ILP) para resolver instâncias menores, e gerar um limite para ser usado como um comparativo. Diferentemente da abordagem de Bari *et al.*, neste trabalho também são tratadas restrições de memória e atraso máximo fim a fim, sendo investigado um *trade-off* entre taxa de aceitação, lucro e o atraso gerado. Para tal comparação, utiliza-se um modelo exato, baseado em ILP,

semelhante ao utilizado em [Luizelli et al. 2017] e [Araujo et al. 2019]. O trabalho de Luizelli *et al.* também apresenta uma heurística para resolver o VNF-PC, mas basedo na metaheurística VNS e com o objetivo de minimizar o número de VNFs utilizadas.

3. Definição Formal do Problema

O VNF-PC pode ser modelado com um grafo direcionado ponderado $G = (N, L)$ representando o SN, onde N e L consistem nos conjuntos de nós e arcos. Cada nó $i \in N$ tem capacidades de CPU (*core*) e memória (MB), representadas por C_i e M_i, respectivamente. Assume-se que cada nó suporta instanciar qualquer VNF solicitada. Cada arco $(i, j) \in L$ tem uma capacidade de largura de banda BW_{ij} (Mbps) e um atraso (milissegundos). O atraso é representado pela função $delay(i, j)$. Seja F o conjunto de NFs, *i.e.*, $F = \{$NAT, FW, WAN...$\}$. Cada NF $f \in F$ pode ser requisitada por uma SFC, atribuída e virtualizada sobre uma instância de VNF já posicionada sobre um servidor do SN. Seja B_f o conjunto de todas as instâncias de NF de cada tipo $f \in F$. Cada instância $b \in B_f$ requer uma quantidade limitada de CPU C_{bf} (*core*) e memória M_{bf} (*MB*) que pode ser oferecida e partilhada entre diferentes SFCs. Quando uma instância de NF $b \in B_f$ é instanciada, ela oferece porções fracionadas de recursos para atender as SFCs, e consome uma parcela de recursos C_{bf} e M_{bf} dos servidores físicos. Assume-se que cada instância de NF mapeada $b \in B_f$ pode ser compartilhada por uma ou mais SFCs. Cada NF $f \in F$ tem um atraso de processamento, e pode gerar um aumento ou diminuição no fluxo de dados, causado pelo processamento associado à NF demandada. Como exemplo, um FW pode descartar alguns pacotes do fluxo de dados, e reduzir a largura de banda demandada.

Seja V o conjunto de SFCs. Cada requisição $v \in V$ possui um tempo de chegada (t^v_{in}), um tempo de duração (t^v_{dr}) e um atraso máximo suportado (t^v_{dl}). Cada SFC $v \in V$ é representada por um grafo direcionado acíclico $G^v = (N^v, L^v)$, onde N^v e L^v representam os conjuntos de nós e arcos virtuais. Seja F^v_f o conjunto das VNFs demandadas por uma SFC $v \in V$, tem-se que a união do ponto de origem s^v, destino d^v e F^v_f são representados pelo conjunto $N^v = \{s^v \cup d^v \cup F^v_f\}$. Assume-se que os nós s^v e d^v não demandam recursos, mas devem ser atribuídos em um nó físico específico do SN. Cada VNF $k \in F^v_f$ demanda capacidades de CPU (c^v_k) e memória (m^v_k) diferentes, e também gera um atraso de processamento $delay(k)$. Cada arco virtual possui uma demanda de largura de banda bw^v_{kl} necessária para o fluxo entre as VNFs, que pode ser alterada pelo ganho/perda de dados (τ), resultante da VNF previamente processada.

Seja t^{vpc}_{dl} o atraso efetivo de uma SFC $v \in V$ após cada nó virtual $v \in N^v$ ser posicionado e encadeado sobre o SN. Uma solução para o VNF-PC consiste em encontrar um mapeamento: $G^v \rightarrow G \mid t^{vpc}_{dl} \leq t^v_{dl}$. Tal ação pode ser dividida na atribuição de instâncias e posicionamento de VNFs[1], e no encadeamento dos arcos virtuais[2]. Se tal mapeamento for factível, a solicitação pode ser mapeada; caso contrário, é recusada. A Figura 1 ilustra 3 SFCs mapeadas sobre o mesmo servidor físico $i \in N$. Esta ação permite compartilhar os custos do respectivo servidor entre as SFCs envolvidas. A Figura 2 mostra uma alocação de recursos para 2 funções de rede $f \in F$ posicionadas sobre o servidor $i \in N$. No exemplo, 2 instâncias de NF são capazes de atender a 3 SFCs

[1]Para prover aspectos de privacidade e confiabilidade na rede, cada nó virtual de uma mesma SFC deve ser posicionado em um nó físico diferente, restrições chamadas de *anti-affinity rules* [Gao et al. 2018]

[2]Cada arco virtual pode ser mapeado em um único arco ou em um caminho com mais de um arco físico.

simultaneamente: a instância FW atende SFCs 1 e 2; e a instância NAT atende a SFC 3. Em cada VNF instanciada, ainda há uma parcela de recursos não utilizados, que podem ser designados caso alguma SFC aumente sua demanda.

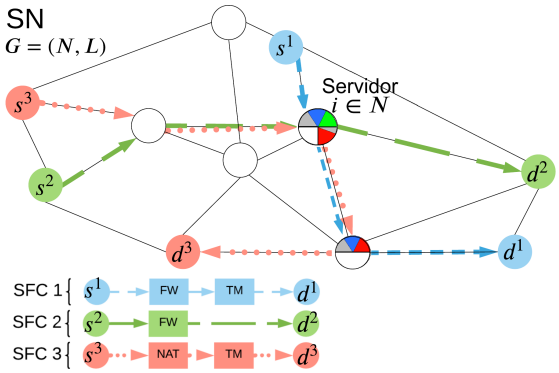


Figura 1. Mapeamento factível de 3 SFCs sobre um SN

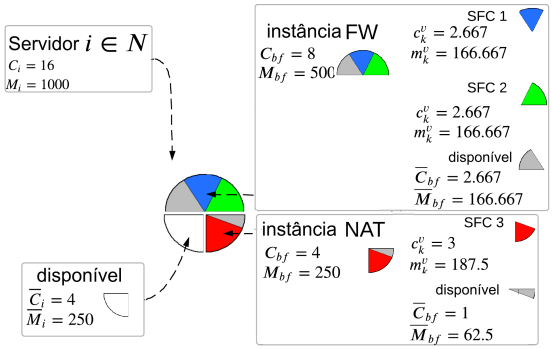


Figura 2. Compartilhamento de recursos em um servidor $i \in N$

4. Abordagem Proposta

A abordagem proposta é baseada nas metaheurísticas GRASP e VNS para solucionar o VNF-PC em um baixo tempo computacional e possui o objetivo de maximizar o lucro dos provedores, conforme a função objetivo a ser apresentada na Seção 5.

GRASP-RVNS (Algoritmo 1): O algoritmo GRASP é fundamentado em um processo iterativo, multipartida, que constrói uma solução factível de cada vez, e depois refina (otimiza) a referida solução através de uma busca local [Feo and Resende 1995].

Algoritmo 1 GRASP-RVNS (v, α)

1: $f^*, i \leftarrow 0, 0$
2: **enquanto** $i < MaxIter_1$ **faça**
3: $s' \leftarrow$ Construtivo (v, α) {fase de construção}
4: **se** $f(s') > f^*$ **então**
5: $s^*, f^* \leftarrow s', f(s')$
6: $i \leftarrow 0$
7: **senão**
8: $i++$
9: **fim se**
10: **fim enquanto** {$MaxIter_1$ iterações sem melhoras}
11: **se** $f^* > 0$ **então**
12: **devolve** RVNS (s^*, α) {busca local somente na melhor solução}
13: **senão**
14: **devolve** false {falha de mapeamento}
15: **fim se**

Algoritmo 2 Construtivo (v, α)

1: $s^v, d^v \leftarrow i \in N$ tal que $s^v = i, j \in N$ tal que $d^v = j$
2: **para cada** $k \in N_f^v$ **faça**
3: $i =$ elementoAleatorio(calculaLRC(k, α))
4: **se** $mapeia(k, i) =$ **falso então**
5: **devolve** false {falha de mapeamento de nós}
6: **fim se**
7: **fim para**
8: **para cada** $(k, l) \in N^v$ **faça**
9: $(i, j) \leftarrow noFisico(k, l)$
10: **se** $BFS(i, j) =$ **falso então**
11: **devolve** false {falha no encadeamento}
12: **fim se**
13: **fim para**
14: **se** $calculaAtraso(v) > t_{dl}^v$ **então**
15: **devolve** false {*Atraso* fim a fim inviável}
16: **fim se**
17: **devolve verdadeiro**

A abordagem apresentada neste artigo (Algoritmo 1) é uma variação do GRASP que aplica a busca local somente na melhor solução encontrada (linha 12), dentre um conjunto de soluções factíveis gerado (linhas 2 a 10). Caso nenhuma solução factível seja encontrada, a SFC é rejeitada (linha 14). Tal escolha de aplicar a busca local somente na melhor solução se dá pela necessidade de uma tomada de decisão rápida quanto ao mapeamento de uma SFC. O limite de iterações realizadas por tal abordagem é definido com um número de iterações sem melhora, através do parâmetro de $MaxIter_1$.

Heurística Construtiva (Algoritmo 2): Tal heurística é baseada na decomposição do VNF-PC em duas estruturas de repetição sequentes:

Laço I (linhas 2 a 7): Relativo ao posicionamento das instâncias de VNFs e à atribuição dos nós virtuais. Dada uma SFC $v \in V$, deve-se encontrar um conjunto de nós físicos que possuam uma capacidade de recursos residuais suficientes para a alocação das instâncias de VNFs solicitadas e para o mapeamento de cada VNF $k \in N_f^v$ demandada.

Laço II (linhas 8 a 13): Relativo ao encadeamento de VNFs. Dado um mapeamento válido de nós $k \in N^v$, para cada enlace $(k, l) \in L^v$ deve-se gerar um encadeamento no substrato que conecte os nós físicos cujos nós do enlace virtual foram mapeados.

Para cada VNF $k \in N_f^v$ existe uma lista de diferentes nós físicos $i \in N$ candidatos para hospedar a função de rede demandada (LC_k). 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 VNF $k \in N_f^v$ (LRC_k, linha 3 do Algoritmo 2) com elementos que atendam os critérios:

$$\{i \in LRC_k \mid h(i) \leq h_{max} - \alpha(h_{max} - h_{min})\}, \text{ onde } h(i) = hops(s^v, i) + hops(i, d^v)$$

Neste caso, h_{max} e h_{min} são respectivamente os maiores e menores valores de $h(i)$ encontrados, e $hops(i, j)$ uma função que retorna o número de arcos físicos intermediários presentes no menor caminho físico entre os nós i e j, calculado com o algoritmo de busca em largura (*Breadth-First Search*, BFS). Essa função é usada para controlar o mapeamento das VNFs demandadas em servidores próximos aos terminais s^v e d^v da SFC, e consequentemente controlar o atraso fim a fim resultante. Em tal implementação, o algoritmo BFS processa somente os enlaces que possuem uma banda residual maior que a banda demandada para o roteamento. Cada LRC_k gerada para cada VNF $k \in N_f^v$ é composta de nós que podem ser inseridos na solução sem gerar infactibilidades. 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 simultaneamente às adjacências dos nós terminais s^v e d^v. Por outro lado, um valor de α próximo a 0 tende a gerar uma LRC menos seletiva, e deixar a abordagem aleatorizada.

As Figuras 3 e 4 exemplificam a função $h(x)$ aplicada a alguns roteadores $i \in N$ de um SN qualquer. Na Figura 3, são inicialmente verificados os valores de $h(x)$ para cada nó físico do SN. Após estes cálculos, os valores $h_{max} = 10$ e $h_{min} = 5$ são encontrados. Aplicando tais valores e $\alpha = 0.9$ na Equação da LRC, gera-se um limite de 5.5, *i.e.*, se a função $h(x)$ de determinado nó obtiver um valor menor que 5.5, tal nó pode ser usado no processamento da SFC, do contrário ele é desprezado. No exemplo da Figura 4, aplicando $h_{max} = 10$, $h_{min} = 5$ e $\alpha = 0.5$ na Equação da LRC, gera-se um limite de 7.5, *i.e.*, se $h(x) < 7.5$ para algum nó $i \in N$ tal nó pode ser usado no processamento da SFC.

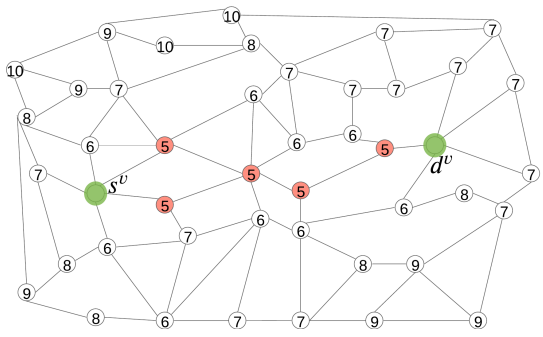


Figura 3. LRC seletiva, $\alpha = 0.9$

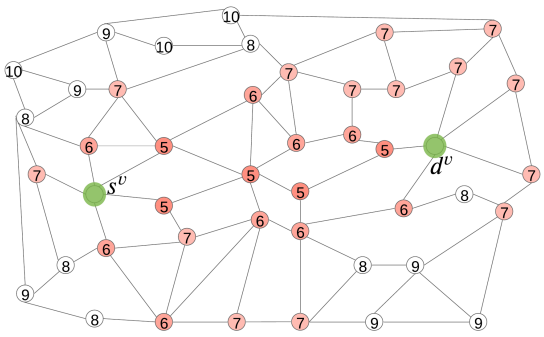


Figura 4. LRC moderada, $\alpha = 0.5$

RVNS: No VNP-PC o tamanho do espaço de soluções aumenta exponencialmente em relação ao número de componentes a serem processados. Tal característica eleva o número de vizinhanças a serem investigadas, e pode deixar a abordagem de refinamento computacionalmente lenta. Para se mitigar este problema, neste trabalho é utilizada uma abordagem modificada do VNS, chamada de *Reduced* VNS (RVNS). Neste caso, a busca local determinística é alterada para melhorar sua eficiência em termos de tempo computacional. Na implementação do RVNS, a cada iteração parte-se de uma solução s corrente (factível) para obter uma solução vizinha aleatória dentro de uma vizinhança $N(s)$. A cada iteração, a solução corrente s é submetida a uma busca local o que origina uma nova solução s'. Se a nova solução gerada s' for melhor que a solução anterior s, a busca continua a partir dela (s'), retornando-se à primeira estrutura de vizinhança. Caso contrário, a busca continua a partir da próxima estrutura de vizinhança. A condição de parada é definida pelo parâmetro $MaxIter_2$ iterações sem melhora.

Vizinhanças: São baseadas em movimentos de trocas factíveis do posicionamento de um nó virtual, e no reroteamento dos arcos virtuais incidentes sobre referido nó. Em tais estruturas, caso a factibilidade da nova solução não exista, a solução anterior é mantida.

i) **Vizinhança** $N_1(s)$**:** Baseada no reposicionamento de uma VNF $k \in N_f^v$ em uma região próxima aos nós físicos em que k 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. 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.

ii) **Vizinhança** $N_2^\alpha(s)$**:** Baseada no reposicionamento de uma VNF $k \in N_f^v$ 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 $(k, l) \in L^v$.

5. Métricas de Avaliação em Redes

As métricas utilizadas são baseadas nos artigos de [Laghrissi and Taleb 2018] e [Bari et al. 2019], e para serem entendidas deve-se descrever algumas variáveis e parâmetros apresentados respectivamente nas Tabelas 1 e 2.

Tabela 1. Variáveis $\in \{0, 1\}$

y^v	Se igual a 1 indica que a SFC $v \in V$ foi mapeada com sucesso
w_{bf}^i	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
z_{ki}^v	Se igual a 1 indica que um nó $k \in N^v$ da SFC $v \in V$ foi atribuído sobre o servidor físico $i \in N$
x_{ij}^{vkl}	Se igual a 1 indica que um arco virtual $(k, l) \in L^v$ da SFC $v \in V$ foi posicionado sobre o arco físico $(i, j) \in L$

Tabela 2. Parâmetros $\in \mathbb{R}_+$

α	Receita por $Mbps$ de banda utilizada
β	Receita por $core$ de CPU utilizado
γ	Receita por MB de memória utilizada
δ	Custo por $Mbps$ de banda alugada do InP
ε	Custo por core de CPU utilizado
ξ_{bf}	Custo para instanciar uma VNF b do tipo f
ζ	Custo por MB de memória utilizado
ϵ	Custo para manter ativo um servidor

Taxa de Aceitação: Métrica que consiste na razão entre o número de SFCs mapeadas e o número total de SFCs que chegaram até o momento t.

Atraso fim a fim: O QoE é um parâmetro subjetivo de ser medido, neste trabalho inferimos que um atraso fim a fim baixo tende a gerar uma qualidade de experiência maior ao usuário final. O atraso fim a fim de cada SFC é dado pela equação:

$$\sum_{(i,j)\in L}\sum_{(k,l)\in L^v} delay(i,j)x_{ij}^{vkl} + \sum_{i\in N}\sum_{k\in N_f^v} delay(k)z_{ki}^v$$

Tempo de processamento: Medido em segundos, através da biblioteca padrão *time.h*.

Lucro dos provedores: Função objetivo adotada por ser uma métrica importante do ponto de vista dos provedores. Neste caso, quanto maior a receita (Equação $\mathbb{R}$) e menores os custos de enlaces (Equação $\mathbb{CE}$) e servidores (Equação $\mathbb{CS}$), maior é o lucro. Equação definida como: $\mathbb{R} - \mathbb{CE} - \mathbb{CS}$.

$$\mathbb{R} = \sum_{v\in V} y^v \left(\sum_{(k,l)\in L^v} bw_{kl}^v\alpha + \sum_{k\in F_f^v}\left(c_k^v\beta + m_k^v\gamma\right)\right)$$

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

$$\mathbb{CS} = \sum_{i\in N}\left(S_i\epsilon + \sum_{f\in F}\sum_{b\in B_f} w_{bf}^i\xi_{bf} + \sum_{v\in V}\sum_{k\in F_f^v} z_{ki}^v\left(c_k^v\varepsilon + m_k^v\zeta\right)\right)$$

Espalhamento dos Nós Virtuais (ENV): Indica o quão conciso ou difuso, em termos de servidores usados, é o posicionamento médio dos nós virtuais $k \in N_f^v$ (VNFs) das diferentes SFCs ativas sobre a estrutura física do SN no momento t. Seja V^{at} o conjunto de SFCs ativas no momento t e S_i^t a variável que indica se o servidor físico $i \in N$ do SN está ativo no momento t, tem-se: $\dfrac{\sum_{i\in N} S_i^t}{\sum_{v\in V^{at}} |N_f^v|}$

6. Experimentos Computacionais

Os experimentos são realizados em um computador *Intel* Core i5-10210U 10*th* 4.20GHz, com 8 GB de RAM e o sistema operacional Ubuntu 20.04. O simulador é implementado em C++ e a abordagem exata é desenvolvida através da API CPLEX 12.6. Os parâmetros de iterações da heurística são definidos como $MaxIter_1 = 50$ e $MaxIter_2 = 300$. O objetivo de tais experimentos é mostrar, em um ambiente *online*, a eficiência da heurística e das diferentes estruturas de vizinhanças propostas. Para isso são feitas comparações com uma abordagem exata, similar a apresentada em [Luizelli et al. 2017, Araujo et al. 2019]. Os experimentos heurísticos são repetidos 15 vezes e mostrados com um intervalo de confiança de 95%.

6.1. Cenários de Simulação

Substrato Físico de Rede: Similarmente a [Jia et al. 2018], a topologia física utilizada para a geração dos SN foi retirada do repositório *http://topology-zoo.org*. Tal topologia, denominada *Cogent Network*, possui 186 nós e 214 links conectando países da América do Norte, Europa e Ásia. De modo similar a [Fischer et al. 2019], considera-se que cada arco e nó do SN possui uma configuração de recursos escolhida aleatoriamente entre $BW_{ij} \in \{20, 40, 60, 80, 100\}$, $C_i \in \{50, 60, 70, 80\}$ e $M_i \in \{1000, 2000, 3000, 4000\}$. Assim como [Jia et al. 2018], o cálculo do atraso de cada arco físico é ajustado proporcionalmente a seu comprimento geográfico, neste artigo calculado pela equação de Haversine[3], e perturbado por um número randômico definido entre $[0.008; 0.012]$.

[3]A fórmula de Haversine é uma equação de navegação, fundamentada na Lei dos Cossenos, e considera a curvatura da Terra para calcular a distância entre dois pontos.

Funções de Rede: As funções de rede $f \in F$ adotadas (Tabela 3) são comuns na literatura [Bari et al. 2019, Fischer et al. 2019]. Diversos fatores são relacionados aos custos de cada componente do SN, como *hardware*, licenças de *software*, energia, etc. Neste trabalho considera-se um modelo simplificado de custo, com os valores de $\delta = 0.025$, $\varepsilon = 0.125$, $\zeta = 0.25$ e $\epsilon = 30$. Assume-se 8 tipos de VNFs disponíveis para a atribuição sobre cada servidor $i \in N$. Para cada VNF $f \in F$, considera-se a disponibilidade de 6 instâncias com capacidades de memória, processamento e custos diferentes (Tabela 4). Os valores adotados para o cálculo da receita dos provedores são $\alpha = 0.05$, $\beta = 0.25$ e $\gamma = 0.5$. Na prática, esses valores não são reais, mas dados após uma avaliação preliminar da literatura e aproximados ao modelo. Caso esse modelo seja adotado em ambientes reais, tais parâmetros podem ser facilmente alterados.

Tabela 3. Tipos de VNFs

Tipo	cpu (core)	memória (MB)	fluxo (η^f)	atraso (ms)
FW small / large	2/ 4	200 / 400	0.9 / 0.9	0.8 / 0.5
NAT small / large	8 / 16	200 / 400	1.0 / 1.0	0.1 / 0.05
WAN optimizers	2	200	1.1	0.1
Encryption	4	400	1.2	0.8
Proxy	4	200	0.9	0.025
IDS	8	800	0.8	0.01

Tabela 4. Tipos de instâncias

instância b	1	2	3	4	5	6
C_{bf}	30	40	50	60	70	80
M_{bf}	250	500	1000	2000	3000	4000
ξ_{bf}	2	3	4	5	6	7

Cadeias de Funções de Serviço: De modo similar a [Sun et al. 2016], para a geração de cada SFC, dois servidores da rede são escolhidos aleatoriamente e definidos como pontos s^v e d^v. Entre tais pontos são associadas aleatoriamente entre $[2, 6]$ VNFs (Tabela 3). A banda demandada no nó de origem de cada SFC é escolhida aleatoriamente entre $[10, 20]Mbps$, e está sujeita a uma alteração na demanda de banda, implicada pela ação da VNF encadeada anteriormente (η^f). O t^v_{dl} de cada SFC é regido por uma distribuição uniforme entre $[500, 1000]ms$. Do mesmo modo que [Bari et al. 2019], o t^v_{dr} é baseado em uma distribuição exponencial com média $\mu = 1000$. O intervalo de chegada de cada SFC segue uma distribuição *Poisson* de média λ, e considerara 4 intervalos diferentes: *i*) $\lambda = 500$, onde 2 SFCs chegam a cada $1000t$; *ii*) $\lambda = 250$, onde 4 SFCs chegam a cada $1000t$; *iii*) $\lambda = 125$, onde 8 SFCs chegam a cada $1000t$; e *iv*) $\lambda = 62.5$, onde 16 SFCs chegam a cada $1000t$. Para cada valor de λ são geradas 1.000 SFCs.

6.2. Abordagens Avaliadas

As abordagens realizam o mapeamento de uma SFC com base nos recursos residuais do SN no momento t, e redimensionam, caso necessário, as instâncias de VNFs presentes nos servidores. O parâmetro α é variado em $\{0.6, 0.7, 0.8, 0.9, 1\}$. As abordagens são:

• ILP^i - Abordagem em ILP;

• $GRVNS^\alpha$ - Abordagem GRASP-RVNS com variação do parâmetro α. Como exemplo, a abordagem $GRVNS^{\alpha 5}$ utiliza o valor de $\alpha = 0.5$. Para uma comparação de valores apurada, são executados e mostrados os experimentos com e sem a busca local. Sendo a nomenclatura GR corresponde à abordagem GRASP e RVNS a busca local.

6.3. Resultados, Discussões e Análises

Lucro dos Provedores (Figura 5): Conforme definido na Seção 5, 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,

reduz os custos. Outro ponto que influi no lucro é relacionado a forma como os roteamentos dos arcos virtuais são distribuídos pelo SN. Neste caso, mapeamentos que usam menos arcos físicos, além de gerarem um atraso fim a fim mais baixo, podem prover lucros maiores, pois podem reduzir custos relativos ao encadeamento das VNFs.

Em todos os experimentos realizados, percebe-se que, ao aumentar o valor de α, o efeito da busca local no lucro tende a diminuir, como exemplo, na Figura 5 e $\lambda = 62.5$, na abordagem $GRVNS^{\alpha 6}$, a busca local é responsável por gerar $\approx 31\%$ do lucro, em contrapartida, na abordagem $GRVNS^{\alpha 9}$, essa parcela é de $\approx 18\%$. Essa diferença acontece por dois fatores inerentes à escolha do valor de α: *i*) quanto maior seu valor é acrescido, se aproximando de $\alpha = 0.9$, mais a LRC seleciona bons elementos para formar a solução inicial, gerando lucros maiores, independente da busca local; *ii*) com uma solução inicial de boa qualidade gerada pela abordagem GRASP, menor o espaço para melhoras na otimização realizada pela busca local RVNS. Observa-se também que um valor de $\alpha = 1$ deixa o algoritmo guloso, e este tende a não gerar boas soluções.

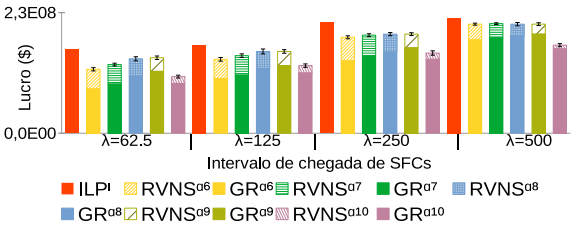


Figura 5. Lucro dos provedores

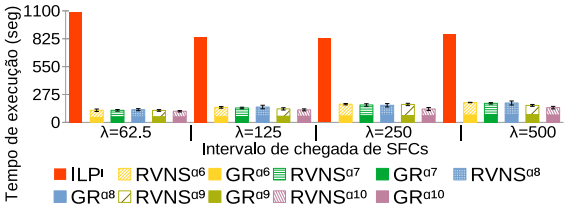


Figura 6. Tempo de execução

A abordagem ILP^i se destaca perante as outras abordagens por gerar um lucro maior em até $\approx 11\%$, justamente por prover uma alta taxa de aceitação (Figura 7) e uma quantidade de servidores compartilhados maior em até $\approx 25\%$ (Tabela 6). Dentre as variações do GRASP-RVNS, exceto para $\alpha = 1$, o maior lucro é obtido pelas variações que utilizam um α maior. O parâmetro $\alpha = 0,9$ gera uma exploração do espaço de soluções com uma maior seletividade de elementos presentes na LRC de cada nó, propiciando uma economia relativa a mapeamentos mais concisos em termos de arcos físicos utilizados. Em relação ao compartilhamento de servidores, a busca local também gera um efeito positivo. Como exemplo, nos experimentos com $\lambda = 62.5$, a abordagem $GRVNS^{\alpha 9}$ consegue aumentar os lucros, pois diminui o espalhamento dos nós virtuais em até 20.9% em comparação à variação sem busca local (Tabela 6). Outro ponto percebido é que, visto que o atraso fim a fim é uma restrição rígida para algumas SFCs (Figura 8), com α alto, a abordagem heurística consegue avaliar mais soluções factíveis em um tempo menor em relação a um α baixo. Um α baixo deixa a construção de soluções iniciais aleatorizada, onde várias soluções podem ser descartadas, pois $t_{dl}^{vpc} \not\leq t_{dl}^{v}$.

Análise do tempo de execução total (Figura 6): A abordagem ILP^i, apesar de gerar os maiores lucros (Figura 5), por resolver o problema de modo exato, gera um tempo de execução alto. Sendo de ≈ 18 minutos para $\lambda = 62.5$ e de ≈ 14 minutos para $\lambda = 500$, tempos impraticáveis para um cenário *online*. Constata-se que, em casos de muitas SFCs ativas simultaneamente (baixo λ), o SN tende a ficar fragmentado, e a resolução exata tende a ser mais custosa, pois as restrições tendem a ficar mais rígidas e difíceis de serem satisfeitas. Dificuldade percebida com base no aumento do tempo de execução em $\approx 25\%$ e na diminuição da taxa de aceitação em $\approx 35\%$, ambos verificados nos experimentos com

$\lambda = 62.5$ e $\lambda = 500$ junto à abordagem ILP^i(Figuras 6 e 7). Por outro lado, a abordagem GRVNS$^{\alpha}$ gera um tempo de execução baixo para todos os experimentos analisados.

Análise da taxa de aceitação (Figura 7): A abordagem $GRVNS^{\alpha}$ gera uma taxa de aceitação próxima e/ou ligeiramente superior à abordagem ILP^i em alguns casos, *e.g.* $\lambda = 62.5$. Enfatiza-se que as abordagens maximizam o lucro, e não a taxa de aceitação. Percebe-se que, à proporção que o valor de λ aumenta, a taxa de aceitação tende a crescer em todas as abordagens, aproximando-se de 100% para $\lambda = 500$. Ressalta-se que a quantidade de SFCs entrantes é a mesma nos quatro cenários, o fator que muda é o intervalo de chegadas e saídas. Observa-se que mesmo a busca local não tendo um efeito significativo na taxa de aceitação, ela é eficaz em aumentar o lucro (Figura 5) e melhorar o compartilhamento de servidores (Tabela 6). Em todos os experimentos realizados houve uma baixa dispersão do intervalo de confiança com uma alta taxa de aceitação e lucro para a abordagem GRVNS$^{\alpha 9}$. Tal comportamento demonstra que as diferentes estratégias de vizinhança propostas conseguem conduzir a heurística para bons ótimos locais até mesmo nos cenários mais difíceis, como no cenário com $\lambda = 62.5$.

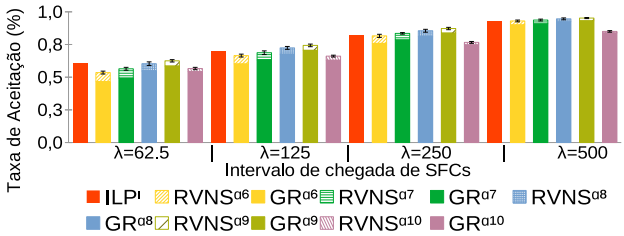
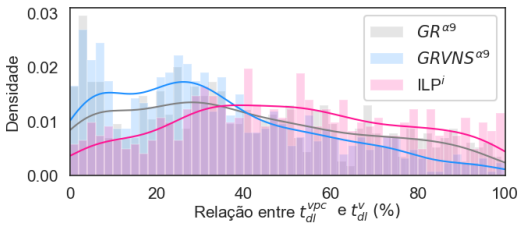


Figura 7. Taxa de aceitação

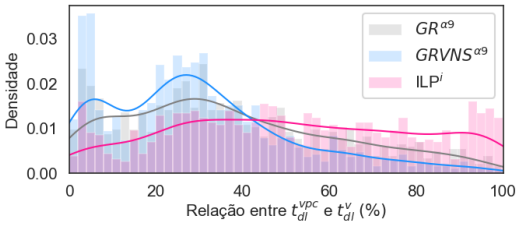
Analisando a variação heurística sem busca local percebe-se que a escolha do parâmetro α com um valor baixo falha mais ao gerar soluções factíveis em relação a um valor alto. Os resultados mostram-se superiores para valores entre $0.7 \sim 0.9$. Essa ação potencialmente reduz o número de soluções infactíveis para serem investigadas, devido ao alto atraso fim a fim; e implicitamente gera um balanceamento de carga sobre o SN. Com as SFCs mapeadas perto dos seus respectivos pontos de origem e destino, proporciona-se uma alocação de recursos distribuída igualitariamente em diferentes regiões do SN e reduzida em número de arcos físicos utilizados. Por outro lado, um valor de α alto gera um espaço de soluções mais restrito, ação que pode refletir em um alto espalhamento de VNFs, sendo até 25% maior se comparado à abordagem ILP^i para $\lambda = 62.5$ (Tabela 6).

Análise do atraso fim a fim (Figura 8): A variação $GRVNS^{\alpha 9}$ obteve resultados promissores em tempo de execução, taxa de aceitação e lucro. Deste modo, os gráficos das Figuras 8 a 10 são plotados sem os outros valores de α. A Figura 8 mostra um histograma da relação do atraso fim a fim máximo tolerado (t_{dl}^{v}) e o gerado após o mapeamento de cada SFC (t_{dl}^{vpc}). Como exemplo, uma SFC que possui um $t_{dl}^{v} = 600$ e $t_{dl}^{vpc} = 300$, indica que ela foi mapeada com um atraso fim a fim 50% menor que o máximo tolerado, ação que reflete em uma melhor QoE para o usuário. Do contrário, se t_{dl}^{v} for muito próximo de t_{dl}^{vpc}, tal valor se aproxima de 100%, e pode gerar insatisfações no cliente quanto a atrasos na conexão. Assim, mesmo que esse atraso fim a fim esteja abaixo ou limítrofe a t_{dl}^{v}, manter este aspecto baixo e controlado é benéfico para a qualidade da experiência do usuário final, e deve ser considerado. Em ambos experimentos (Figura 8), a abordagem heurística com busca local se destaca perante as outras abordagens 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.

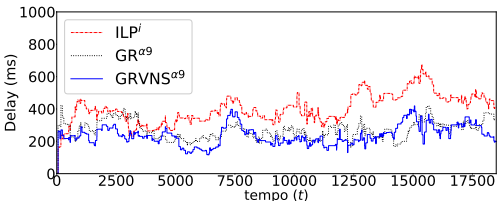


(a) $\lambda = 62.5$

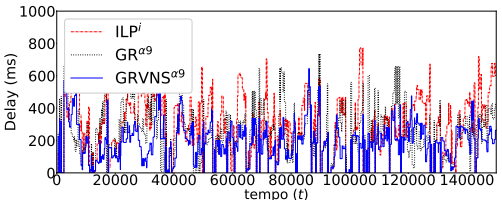


(b) $\lambda = 500$

Figura 8. Histograma da relação percentual entre t_{dl}^{v} e o t_{dl}^{vpc} de cada SFC



(a) $\lambda = 62.5$

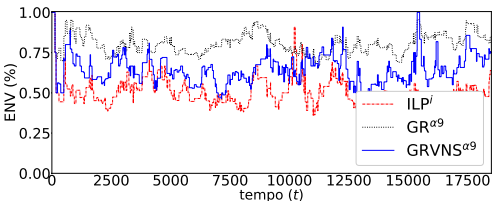


(b) $\lambda = 500$

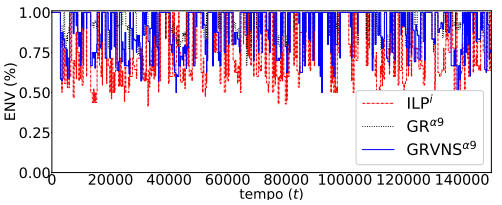
Figura 9. atraso fim a fim

Gráfico	9(a)	9(b)	10(a)	10(b)
ILP^{i}	$367ms$	$313ms$	47.3%	78.8%
$GRVNS^{\alpha 9}$	$240ms$	$183ms$	63.7%	90.1%
$GR^{\alpha 9}$	$299ms$	$235ms$	80.5%	94.5%

Tabela 5. Média $\overline{x}$



(a) $\lambda = 62.5$



(b) $\lambda = 500$

Figura 10. Espalhamento de Nós

Gráfico	9(a)	9(b)	10(a)	10(b)
ILP^{i}-$GRVNS^{\alpha 9}$	52.5%	70.1%	-25.7%	-12.5%
$GRVNS^{\alpha 9}$-$GR^{\alpha 9}$	-19.6%	-22.0%	-20.9%	-4.7%
ILP^{i}-$GR^{\alpha 9}$	22.6%	33.2%	-41.3%	-16.6%

Tabela 6. Variação $\frac{v_2 - v_1}{v_1} \times 100$

Nas Figuras 9 e 10 o eixo x apresenta o tempo de simulação em unidades de tempo (t) e o eixo y a métrica observada. Em ambos cenários (Figura 9), a abordagem 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 ser menor em relação ao custo de se manter um servidor ativo, assim, tal modelo prioriza o compartilhamento de recursos em detrimento a roteamentos mais curtos. Por outro lado, a abordagem 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 70% em comparação a abordagem exata (Figura 9 e Tabela 6). Inferindo uma correlação entre o atraso fim a fim e os encadeamentos entre VNFs, para a geração de um baixo atraso fim a fim, é necessário haver um bom roteamento dos arcos virtuais. Neste sentido, um roteamento que gera um menor atraso fim a fim, também é benéfico por se utilizar menos arcos do SN, e deixar mais recursos residuais livres para serem utilizados nos mapeamentos de outras SFCs.

Análise do espalhamento dos nós virtuais: Com um baixo intervalo na taxa de chegadas e saídas de novas SFCs (Figura 10(a)), a abordagem ILP^i consegue manter o espalhamento dos nós virtuais em 47.3%, *i.e.*, em média, cada servidor físico utilizado está sendo compartilhado por aproximadamente duas requisições diferentes (Tabela 6). Comparando a abordagem heurística com e sem busca local e $\alpha = 0.9$, constata-se que a aplicação de tal técnica aumenta o espalhamento de VNFs em até $\approx 20.9\%$ (Tabela 6), ação que implica em melhorias no lucro e na taxa de aceitação, mostradas nas Figuras 5 e 7. Dentre os cenários e métricas analisadas, a abordagem $GRVNS^{\alpha 9}$ é promissora em relação ao baixo tempo computacional (Figura 6), a constante e alta taxa de aceitação (Figura 7), e ao baixo atraso fim a fim gerado (Figura 9). Porém, a abordagem $GRVNS^{\alpha 9}$ tende a não gerar um compartilhamento de servidor tão promitente quanto a abordagem ILP^i (Tabela 6) e possui um lucro menor (Figura 5). Comportamento que em outros cenários pode ser prejudicial para o lucro, caso o custo de manter um servidor ativo seja ainda mais elevado em relação aos custos de utilização dos arcos.

7. Conclusões e Trabalhos Futuros

Neste trabalho foi apresentada uma abordagem heurística, chamada de GRASP-RVNS, para resolver o problema VNF-PC em um ambiente *online*, em um baixo tempo computacional e respeitando as demandas dos clientes, como, atraso fim a fim, memória, processamento e largura de banda. 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 $\alpha = 0.8$ e $\alpha = 0.9$. Por outro lado, percebeu-se que uma LRC com $\alpha = 1$ implica em resultados de baixa qualidade, devido ao mapeamento guloso.

Conforme esperado, um tratamento exato é custoso computacionalmente e impraticável. Por outro lado, o tratamento heurístico, apesar de abrir mão da garantia de otimalidade da solução, gera bons resultados, com uma alta taxa de aceitação, baixo atraso fim a fim, mas com um menor compartilhamento de servidores, ação que afetou negativamente o lucro. Como exemplo, no cenário com uma alta taxa de entrada e saída de SFCs ($\lambda = 62.5$) a abordagem heurística manteve a taxa de aceitação estável em relação à abordagem exata, gerou um lucro apenas $\approx 11\%$ menor; um tempo de processamento 810% mais baixo e um atraso fim a fim reduzido em 52.5%. Propõe-se em trabalhos futuros gerar uma abordagem híbrida, que combine as vantagens das técnicas de aprendizado de máquina com a abordagem heurística apresentada. Infere-se que, com a utilização de técnicas de aprendizado de máquina, pode-se reduzir o espaço de busca, e gerar uma menor probabilidade da heurística ficar retida em algum ponto de ótimo local.

Agradecimentos

Os autores agradecem ao CNPq, CAPES e FAPEMIG.

Referências

Araujo, S. A., de Souza, F. H., and Mateus, G. R. (2019). 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, Los Alamitos, CA, USA. IEEE Computer Society.

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.

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.

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.

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

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.

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.

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.

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.

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.

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.

NFV White Paper (2012). Network functions virtualisation: An introduction, benefits, enablers, challenges & call for action. *SDN and OpenFlow World Congress*.

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.

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.