UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UM MÉTODO DE OTIMIZAÇÃO COM PARÂMETROS DE DESEMPENHO PARA CLOUD NETWORK SLICES FOCADO NO LOCATÁRIO

LUCIAN BERALDO

ORIENTADOR: PROF. DR. FÁBIO LUCIANO VERDI

São Carlos – SP

Dezembro/2020

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UM MÉTODO DE OTIMIZAÇÃO COM PARÂMETROS DE DESEMPENHO PARA CLOUD NETWORK SLICES FOCADO NO LOCATÁRIO

LUCIAN BERALDO

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Sistemas Distribuídos e Redes de Computadores
Orientador: Prof. Dr. Fábio Luciano Verdi

São Carlos – SP

Dezembro/2020

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Defesa de Dissertação de Mestrado do candidato Lucian Beraldo, realizada em 29/01/2021.

Comissão Julgadora:

Prof. Dr. Fabio Luciano Verdi (UFSCar)

Prof. Dr. Paulo Matias (UFSCar)

Prof. Dr. Luiz Fernando Bittencourt (UNICAMP)

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

O Relatório de Defesa assinado pelos membros da Comissão Julgadora encontra-se arquivado junto ao Programa de Pós-Graduação em Ciência da Computação.

AGRADECIMENTOS

O presente trabalho de pesquisa foi realizado com o apoio parcial da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001 e da Comissão Europeia e Ministério da Ciência, Tecnologia, Inovação e Comunicação (MCTIC) através da RNP e CTIC no contexto da quarta chamada conjunta EU-BR, acordo #777067 (NECOS - *Novel Enablers for Cloud Slicing*). Agradeço ao meu orientador Prof. Dr. Fabio Luciano Verdi por conduzir o meu trabalho de pesquisa e ao Prof. Dr. Cesar Augusto Cavalheiro Marcondes pela co-orientação e contribuição com o trabalho de pesquisa. Também ao Prof. Dr. Panagiotis Papadimitriou e ao Angelos Pentelas pelas contribuições com a pesquisa e o artigo submetido, assim como à Universidade da Macedônia pela oportunidade de intercâmbio com tais pesquisadores.

RESUMO

O conceito de *Cloud Network Slicing* (CNS), emergente em conjunto com as redes móveis 5G, designa uma mudança de paradigma na forma com que redes computacionais são provisionadas, gerenciadas e operadas. Tal conceito impulsiona novas aplicações como realidade aumentada e virtual, transmissão de vídeo em resolução 4K e veículos autônomos interconectados. Aplicações essas que necessitam de tempo de latência mínimos, grande largura de banda de rede ou ambos. O funcionamento de tais serviços demanda da correta alocação de recursos de computação e rede, de forma isolada dos outros serviços da *Internet*. Normalmente, tais recursos estão espalhados por grandes áreas geográficas, como múltiplos países e continentes, o que implica na utilização de distintos provedores de infraestrutura. Característica que potencializa o desafio da máxima eficiência na alocação de recursos, que é tratada pela arquitetura funcional de CNS do projeto NECOS, a qual também é objeto de estudo do trabalho aqui apresentado. As CNS são consideradas aqui como um conjunto de *Slice Parts* de computação e rede, *i.e.* fragmentos que refletem a topologia da CNS com requisitos de recursos especificados pelo locatário e potencialmente múltiplas ofertas dos provedores por *Slice Part*. O problema da escolha otimizada de recursos foi modelado como um programa linear inteiro misto (*i.e. Mixed Integer Linear Program - MILP*), porém também foram elaborados dois algoritmos heurísticos para tratar cenários complexos com grande número de combinações. Os resultados experimentais, baseados em um ambiente simulado, em consonância com a arquitetura do projeto NECOS, indicam que a abordagem MILP teve melhor desempenho comparada com ambas as heurísticas, *i.e.* foram escolhidas ofertas de *Slice Parts* mais baratas, com uma boa quantidade de recursos, em um tempo de execução adequado. A contribuição principal está no desenvolvimento dos métodos de otimização baseados na abordagem de fragmentação e agregação, inseridos na nova arquitetura proposta pelo NECOS.

Palavras-chave: fatiamento de nuvem, fatiamento de rede, Programação Inteira Linear Mista, IBM CPLEX

ABSTRACT

Cloud Network Slicing (CNS), emerging alongside the 5G mobile network, comprises a paradigm shift in the way networks are provisioned, managed, and operated. Fundamentally, CNS fosters the deployment of a multitude of modern applications, *e.g.*, virtual and augmented reality, 4K video streaming, and autonomous vehicles, which require ultra-low latency, high bandwidth consumption, or both. Slicing promotes the realization of such services through the allocation of computing and network resource bundles, which, as CNS mandates, are isolated from the rest of the network. Typically, such resources are arranged into wide geographical areas (*e.g.*, into multiple countries or even continents), which implies that it is possible to pertain to distinct infrastructure providers. This exacerbates the already challenging problem of maximizing resource allocation efficiency, a feature commonly addressed by CNS architectures. In this respect, we study the optimal assignment of slices to multiple domains. Therefore, we account for slices as a collection of computing and network parts. Given specific resource requirements from slice tenants, and potentially multiple offers per slice part, we model the problem as a Mixed Integer Linear Program (MILP). We further design two heuristic algorithms, in order to mitigate the complexity intricacies that would be perceptible in large problem instances. Our evaluation results, based on a simulation environment aligned with the NECOS architecture, indicate that the MILP approach had a better performance compared to both the heuristics in choosing the cheapest offers with a fair amount of performance parameters in an adequate execution time. Our main contribution stands on the optimization methods based on the split and combine approach inserted in the novel NECOS' CNS architecture.

Keywords: Cloud Network Slicing, slicing, mixed integer linear programming, MILP, heuristic, optimization, IBM CPLEX.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

Capítulo 1

INTRODUÇÃO

O presente capítulo apresenta o contexto das slices, focando em suas aplicações e apresenta a motivação para a realização deste trabalho, sendo a introdução do texto de dissertação.

A capacidade de abstrair fatias (*i.e. slices*) representa uma mudança de paradigma significativa e necessária na relação entre clientes e provedores de serviços computacionais. Aplicações de entretenimento como transmissões em tempo real (*streaming*) de vídeos com resolução em 4K e realidade virtual demandam cada vez mais largura de banda de transmissão (HU et al., 2017; GE et al., 2017; KENNY; BROUGHTON, 2013) à medida que cresce a preferência dos consumidores por imagem, som e sensação de imersão cada vez realistas. Esta forma de consumir conteúdo tem ganhado muitos adeptos com a popularização de televisores inteligentes (*smart TVs*), *smartphones*, *tablets* e óculos de realidade virtual capazes de exibir conteúdo personalizado, no momento em que o espectador está disponível, *i.e. Video on Demand* (VoD). A utilização das ferramentas mencionadas anteriormente, de forma fluida e sem interrupções por problemas de conexão instável, denominada qualidade de experiência, do inglês *Quality of Experience* (QoE), é diretamente afetada pela estabilidade de velocidade das conexões com a Internet. Também a promessa de adoção em massa de veículos autônomos interconectados, desde carros de passeio até veículos autônomos não tripulados (*drones*) de entrega, tem como requisito vital que tempos de latência mínimos sejam realidade.

Outras aplicações que podem se beneficiar das inovações das *slices* se encontram na área de turismo, em que museus e parques com grande movimentação, poderão disponibilizar conteúdo em alta definição para turistas sem a necessidade de sobrecarregar a rede com vários acessos à *core cloud*, disponibilizando o conteúdo necessário para seu funcionamento localmente, na chamada *edge cloud*; a promessa de utilização em massa de carros autônomos interconectados tem como requisito para seu funcionamento a utilização tempos de resposta ultra rápidos, por volta

de 1 milissegundo, para que os veículos possam tomar decisões em tempo real; aplicações de cidades inteligentes com a utilização de Internet das Coisas, *i.e. Internet of Things* (IoT), também dependerão das inovações oriundas da implementação da quinta geração de redes móveis (5G) e das *slices*, uma vez que contribuem para a diminuição do consumo de bateria para os dispositivos se comunicarem com as antenas de rádio base, uma característica crucial para estes equipamentos, que possuem pouca ou nenhuma alimentação de energia elétrica constante e dependem de bateria interna para seu funcionamento.

Tais aplicações só serão possíveis em consequência de funcionalidades que as *slices* com as redes 5G proporcionarão em comparação às redes 4G. Destas funcionalidades, se destacam o aumento da largura de transmissão (*bandwidth*) em aproximadamente 100 vezes, o tempo de resposta (*response time*) 20 vezes menor e a capacidade de criar *slices* para isolar o tráfego e garantir o desempenho constante de serviços interconectados (DING et al., 2017).

Cloud Network Slice é um conceito que emerge em conjunto com as redes móveis 5G, propondo isolar o tráfego de rede, recursos de infraestrutura (*i.e.* rede e computação). Seu objetivo principal é customizar o uso de recursos fim a fim para atingir demandas de certas aplicações, apresentando-se como promessa para atingir os avanços supracitados. Algumas arquiteturas propõem o uso da combinação entre SDN e NFV, que provêm formas de adequar redes programáveis ao contexto das *slices*. Neste sentido, o projeto *Novel Enablers for Cloud Slicing* (NECOS) propõe uma arquitetura para a implementação das *Cloud Network Slices* na infraestrutura atual da rede mundial de computadores (*Internet*), elucidando características deste tema recente e contribuindo para que novos padrões sejam estabelecidos para a definição de *slices*. Por se tratar de um conceito emergente, existem ainda diferentes definições para as *Cloud Network Slices*, porém no presente trabalho será considerado no escopo da arquitetura definida pelo projeto NECOS. As *Cloud Network Slices* tratadas no contexto do projeto NECOS caracterizam-se como um subconjunto das *slices* e que engloba recursos de computação e armazenamento de forma separada dos recursos de rede. Isto possibilita que diferentes tipos de provedores de infraestrutura possam participar das propostas de criação das *slices*.

A definição e alocação das *slices* requer que sejam elencados os recursos de infraestrutura que farão parte dessa abstração, posteriormente eles precisam ser selecionados de forma eficiente e planejada, para que sejam cumpridos os requisitos de desempenho e de custo financeiro dos locatários que a utilizarão. Com a escolha eficiente de recursos computacionais, é possível também qualificar as *slices* com a eficiência energética, aspecto cada vez mais relevante para os locatários, com foco em abordagens sustentáveis (BOCKEN et al., 2014) e na computação verde (*green computing*) (HACHICHA; YONGSIRIWIT; GAALOUL, 2016). Tais recursos de-

vem atravessar diferentes domínios para que sejam disponibilizados próximos ao usuário final e, ocasionalmente, atravessar diferentes unidades federativas, as quais possuem características diversas como: legislação, infraestrutura, preço, estabilidade e demanda disponíveis. A escolha correta dos recursos pode propiciar o balanceamento de carga entre servidores com sobrecarga e melhorar tempos de processamento.

A escolha correta dos recursos de infraestrutura é característica crucial para que a *slice* possa cumprir seu funcionamento esperado e, devido à necessidade de alocar recursos próximos ao locatário, deve possibilitar a participação de vários provedores de infraestrutura que atuam em regiões diversas. Com a participação de diferentes provedores na oferta de recursos computacionais, de armazenamento e de rede, existe a necessidade de o método de escolha permitir que o provedor esconda informações sensíveis de sua infraestrutura, as quais não devem ser divulgadas para seus concorrentes. Além disso, é necessário que haja a capacidade de comparar propostas com níveis variados de desempenho, *i.e.* com diferentes tecnologias utilizadas que influenciam diretamente nos custos das propostas.

No contexto apresentado, é proposto um modelo baseado em fragmentação e agregação (*i.e. split and combine*), que foca no desempenho ótimo de uma *Cloud Network Slice*. Particularmente, tal modelo considera ofertas (*i.e. Resource Options*) de rede e nuvem oriundas de múltiplos provedores, alinhadas com um cenário que tenta representar aplicações reais. Tal característica contribui para que o método encontre soluções com baixo custo financeiro, considerando também restrições de desempenho das partes constituintes da *Cloud Network Slice*, *i.e.* os nós e as arestas da mesma. O método inicialmente divide a *slice* em vários segmentos de *datacenter* e rede de acordo com a aplicação que será utilizada; posteriormente utiliza de Programação Inteira Linear Mista, *i.e. Mixed Integer Linear Programming* (MILP) para a escolha dos recursos de infraestrutura *Cloud Network Slices* no contexto do projeto NECOS; finalmente, com os recursos de infraestrutura já escolhidos, é alocada uma *slice* combinando tais recursos.

Almeja-se com os resultados do presente trabalho, contribuir para o avanço do estado da arte na área de otimização para *slices*. Para atingir tal objetivo, foi utilizada a arquitetura de *Cloud Network Slice* do projeto NECOS que tem como objetivo criar e orquestrar *slices*, a qual possui um módulo de loja virtual no formato de *marketplace*, *e.g.* Bondan et al. (2019), onde os recursos de infraestrutura estão disponíveis para o locatário selecionar. Pelo ponto de vista da implementação, foi desenvolvido um método baseado em MILP e duas heurísticas para resolver o problema apresentado. O primeiro foca nas restrições necessárias de acordo com a especificação do locatário, junto com uma função objetivo que almeja minimizar o custo da

Cloud Network Slice. As duas heurísticas foram desenvolvidas para lidar com a complexidade do problema, o qual geralmente é perceptível em grandes escalas, *e.g.* muitos fragmentos de *slice* (*i.e. Slice Parts*) e muitas *Resource Options*.

Os resultados apresentados aqui demonstram o MILP como o melhor método para selecionar *Resource Options* com o menor preço financeiro e também provendo bons recursos de infraestrutura para o locatário. Apesar dos bons resultados, o MILP consome mais recursos da estação de trabalho utilizada para os experimentos, comparado a ambas as heurísticas. Os experimentos mostram resultados promissores na alocação ótima dos recursos de infraestrutura mantendo carga financeira mínima para o locatário. Por fim, a contribuição principal deste trabalho é o novo método MILP para a escolha de *Resource Options* advindos de provedores de infraestrutura no contexto das *Cloud Network Slices* com base na arquitetural funcional do projeto NECOS.

1.1 Organização da dissertação

É apresentado a seguir o *outline* do texto de dissertação:

1. Capítulo 1: Contém a introdução do tema apresentado, com o contexto, a motivação e um resumo da proposta;
2. Capítulo 2: Apresenta os conceitos que são utilizados neste trabalho, fornecendo a base teórica para a implementação apresentada posteriormente;
3. Capítulo 3: Introduz o projeto NECOS e a arquitetura utilizada para a implementação do trabalho de dissertação;
4. Capítulo 4: Contextualiza o cenário do projeto aqui apresentado e o compara a outros trabalhos similares;
5. Capítulo 5: Explicita a proposta do projeto de dissertação e o que se pretende conseguir com ele;
6. O Capítulo 6 mostra os resultados atingidos.

Capítulo 2

CONCEITOS UTILIZADOS

O capítulo a seguir introduz os conceitos que serão utilizados ao longo do presente texto de dissertação.

O fatiamento (*slicing*) de recursos é uma abstração que permite a alocação de recursos físicos ou virtuais, disponibilizados por provedores de rede, nuvem ou terceiros, de forma isolada para um locatário sem a exposição da infraestrutura física que a compõe. Com o isolamento da *slice*, locatários podem executar seus serviços em múltiplos domínios, *i.e. multi-domain*, uma característica de organizações que atuam em várias localizações geográficas, tendo de tratar devidamente o tráfego quando seus usuários se movimentam entre vários domínios.

As *slices* podem agregar um contrato no formato de Acordo de Nível de Serviço, *i.e. Service Level Agreement* (SLA), que contém parâmetros mínimos de serviço e restrição de custos especificados pelo locatário, que devem ser seguidos pelo provedor da *slice*. Tais restrições podem ser exemplificadas como tempo de resposta (*response time*), capacidade de processamento, armazenamento, largura de banda de rede, nível de confiabilidade, localização geográfica desejável, janela temporal de alocação, etc. As restrições contidas no SLA vigorarão desde o momento da instanciação até a finalização das *slices* (ALLIANCE, 2016). Apesar da possibilidade de inserção do conceito de SLA nas *slices*, neste trabalho e também no projeto NECOS, não foram tratadas as negociações de SLA, pois foge do escopo da proposta dos mesmos.

A abstração de *slices* encontra grande aplicabilidade em mercados de nicho, *i.e. vertical markets*, voltado para públicos específicos com atendimento personalizado, podendo alocar *slices* em regiões onde os públicos se localizam. Para o gerenciamento, a orquestração dos recursos é comumente realizada de forma distribuída para amenizar a carga de complexidade e também para possibilitar a utilização recursiva de virtualização (CLAYMAN, 2018).

Para atingir o requisito de tempo de resposta ultra rápido esperado pelas redes 5G, a abstração

de *slices* precisa permitir a replicação de conteúdo entre os nós (*i.e. hosts*) da topologia em locais diversos. Dessa forma o serviço de *cache* deve ser disponibilizado próximo ao locatário que o utilizará, reduzindo a distância entre servidor e cliente e, consequentemente o tempo de resposta do serviço. Em consequência disso, para a garantia de privacidade na replicação de conteúdo em locais diversos, é necessário que haja o isolamento do fluxo de rede, assim como os recursos de computação devem possuir níveis de desempenho garantidos e não transparecer informações com outros serviços compartilhados nos *datacenters* (DCs) (SUN et al., 2018).

Após a descoberta dos recursos que farão parte da *Cloud Network Slice*, ela é dividida em *Slice Parts*, os quais são fragmentos que refletem a topologia da *slice* em nós e arestas, *i.e. DC Slice Parts* e *Net Slice Parts* respectivamente. O escopo desta dissertação de mestrado e do projeto NECOS não contemplam a forma com que serão fragmentados os *Slice Parts*, nem a quantidade mínima ou máxima de fragmentos que serão criados, o que demanda de investigação sobre a quantidade ótima de *Slice Parts* em uma *slice*. Porém, uma abordagem que se encaixa neste contexto é criar um *DC Slice Parts* por localização geográfica e por funcionalidade necessária, *e.g.* servidor *Web*, *firewall*, etc. Os *Slice Parts* contém a descrição dos componentes de cada fragmento necessário para o funcionamento da *slice*, sendo que os do tipo DC contém informações específicas de computação e armazenamento, enquanto que os do tipo Net possuem informações de comunicação e tráfego da *Cloud Network Slice*. Os *Slice Parts* possuem restrições inerentes ao todo da *Cloud Network Slice*, como localização geográfica, preço financeiro, parâmetros de desempenho, etc.

Apesar do inovador conceito de *slices* permitir que o dimensionamento de recursos seja feito sem o conhecimento de que tipo de aplicação será executada nele, sua relação custo-benefício será mais atrativa quanto maior forem conhecidos os requisitos de sua utilização. O dimensionamento e escolha das *slices* pode ser feito de forma manual, por profissionais que possuem experiência na aplicação que será executada, podendo aumentar ou diminuir pontualmente a quantidade de recursos de acordo com o comportamento da aplicação através do monitoramento constante de sua taxa de utilização. Em contrapartida, métodos de otimização são utilizados para realizar a escolha dos recursos de infraestrutura, mas neste, de forma automatizada. No método de otimização linear é elaborada uma função linear, chamada de função objetivo, que representa matematicamente o problema da escolha dos recursos de infraestrutura, atribuindo variáveis a cada parâmetro de desempenho e custo financeiro que deve ser considerado na escolha dos recursos. Inclusive, para a delimitação dos intervalos nos quais estão inseridas as variáveis da otimização linear, usam-se restrições às quais a função objetivo está sujeita.

Slices conceitualmente podem ser divididas em três categorias, fatiamento de rede (*i.e. Network Slicing*, fatiamento de nuvem (*i.e. Cloud Slicing*) e por fim o fatiamento de rede e nuvem (*i.e. Cloud Network Slicing*) (CLAYMAN, 2018).

- As *Network Slices* podem ser definidas como a capacidade de abstrair e isolar os recursos de rede através da virtualização dos mesmos, compondo uma camada separada da infraestrutura física. O isolamento nesta camada se dá pela atribuição de Funções de Rede Virtualizadas, *i.e. Virtualized Network Functions* (VNFs), específicas para cada locatário. O isolamento do tráfego gerado pode ser garantido com a utilização conjunta de Redes Virtuais Locais, *i.e. Virtual Local Area Network* (VLAN), Redes Privadas Virtuais, *i.e. Virtual Private Networks* (VPN) e com a criptografia garantindo a privacidade do tráfego de rede gerado;

- As *Cloud Slices* representam a abstração de isolamento dos recursos computacionais que são efetivamente usados para a execução das aplicações. Tal abstração pode ser exemplificada pela criação de máquinas virtuais, *i.e. Virtual Machines*, máquinas físicas, *i.e. bare metal* e *containers* caracterizando-se como os *hosts* destas aplicações;

- Por fim, as *Cloud Network Slices* agregam as *Network Slices* e as *Cloud Slices* para fornecer a abstração de uma infraestrutura completa de *hosts* e as interfaces de comunicação que estes *hosts* utilizam para a comunicação com outros. As *Cloud Network Slices* serão o foco principal do trabalho apresentado neste texto.

Os atores envolvidos nas *Cloud Network Slices* podem ser descritos como o provedor de infraestrutura, *i.e. Infrastructure Provider* (InP), o locatário (*i.e. tenant*) e o usuário final. O provedor é geralmente representado por organizações que possuem a infraestrutura física e também as redes de comunicação interconectando a infraestrutura. O locatário é o indivíduo ou organização, que utiliza os recursos computacionais de um ou vários provedores, gerenciando e criando serviços que serão distribuídos aos usuários finais. O provedor disponibiliza os recursos físicos ou virtualizados para que os locatários possam executar suas aplicações nestes recursos, por uma taxa cobrada em certo período de tempo pelo provedor. No contexto do trabalho de mestrado aqui apresentado, a taxa (preço) cobrada pelos recursos é estipulada por dia, *i.e.* quanto o locatário irá pagar para utilizar o recurso durante 24 horas. Tal interação entre o provedor e o locatário pode ser recursiva, em que o locatário repassa seus recursos virtualizados para outros provedores ou locatários, desta forma, se tornando um provedor, porém este tipo de interação não será tratado aqui. O usuário final é o ator que utiliza os serviços provisionados pelo locatário, não repassando para outros usuários (ORDONEZ-LUCENA et al., 2017).

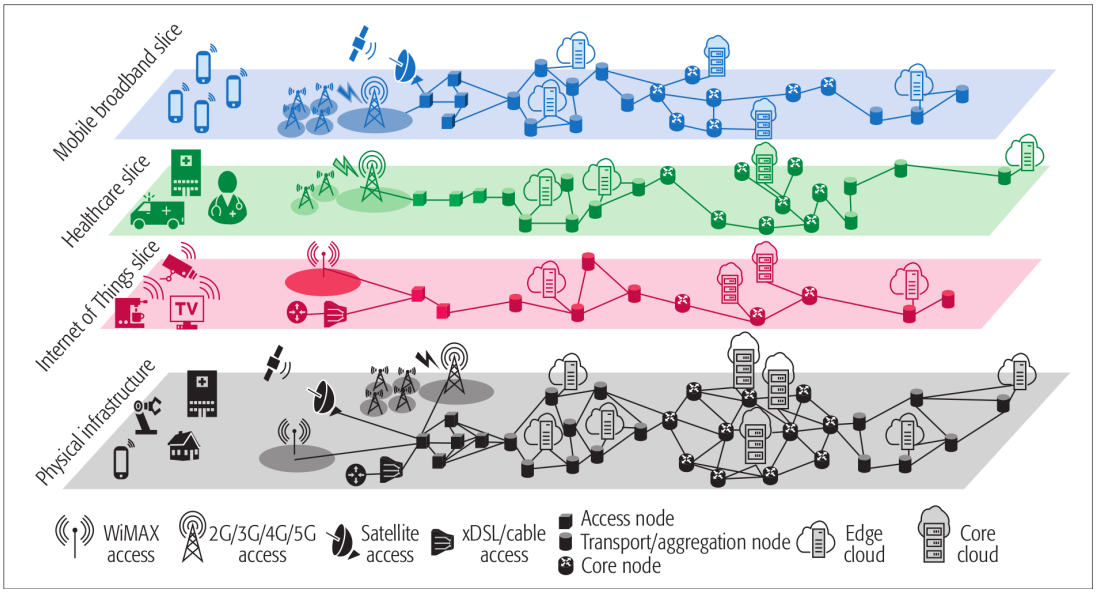


Figura 2.1: Exemplo de *slices* para propósitos diversos.
Fonte: (ORDONEZ-LUCENA et al., 2017)

A Figura 2.1 ilustra o conceito das *Cloud Network Slices* no contexto das redes 5G. Nela é possível observar três *slices* em cores distintas (vermelha, verde e azul), sendo que as três possuem propósitos diversos, abstraídas da mesma infraestrutura física localizada na camada inferior. A camada inferior, na cor cinza, representa a topologia da infraestrutura física, contendo antenas de acesso de redes móveis como o 5G e *Worldwide Interoperability for Microwave Access* (WiMAX); antenas de conexão via satélite; links cabeados de conexão com alta taxa de transmissão como Ethernet, cabo coaxial e fibra ótica; equipamentos de interconexão como *switches*, roteadores, *hubs* e equipamentos contidos na nuvem central (*core cloud*) e de borda (*edge cloud*). Logo acima , nas cores vermelha, verde e azul estão representadas camadas de isolamento abstrato representando as *Cloud Network Slices*, onde são selecionados recursos que comumente são usados para um serviço de Internet das Coisas, *i.e. Internet of Things* (IoT), serviços emergenciais de saúde e de internet móvel respectivamente.

Para a utilização do recente conceito de *Cloud Network Slices*, o trabalho aqui apresentado utiliza algumas definições básicas para sua concepção, elas estão apresentadas a seguir. Recurso computacional é um componente de um sistema de computação, *e.g.* processamento, armazenamento, memória, rede, etc, podendo ele ser físico (*i.e. bare metal*) ou virtualizado. Recursos de rede são elementos de comutação de pacotes utilizados para interconectar os *hosts* que possuem os recursos computacionais, *i.e.* roteadores, *switches*, *hubs*, etc, tais recursos de rede também podem ser físicos ou virtualizados. No contexto deste trabalho e também no projeto NECOS, são considerados apenas recursos físicos. Tais recursos computacionais e de rede

possuem métricas de desempenho (*e.g. consumo de memória, armazenamento em disco, uso do processador, etc*), consumo médio energético, portas de conexão para comunicação e capacidade de uso pré estabelecidos, informações importantes para que eles sejam utilizados em aplicações (GUERZONI et al., 2017).

O termo virtualização utilizado para as *Cloud Network Slices* refere-se à abstração de recursos computacionais, de rede e funções lógicas oferecidas aos usuários ou aplicações, tal abstração possibilita o gerenciamento inteligente do ciclo de vida permitindo que se faça um uso mais eficiente dos recursos virtualizados. Com isso é possível aplicar a automatização nas etapas do gerenciamento, economizando recursos financeiros em comparação aos métodos tradicionais com intervenção humana. Tal economia financeira se deve principalmente ao uso mais eficiente de consumo energético, que representa boa parte dos custos atrelados (RAHMAN; DESPINS; AFFES, 2013). A virtualização dos ativos de rede e nuvem torna possível a criação de redes virtuais completamente independentes do hardware utilizado, facilitando a orquestração dos ativos, compreendendo a migração, o escalonamento (*i.e. upgrade/downgrade*) e a desalocação de sistemas. Tal possibilidade ocorre devido à facilidade para se adicionar novas funcionalidades sem a preocupação com camadas inferiores à camada de aplicação.

As Redes Definidas por Software, *i.e. Software Defined Networks* (SDN), remetem à utilização de virtualização para a implementação de redes computacionais, enquanto que as Funções de Rede Virtuais, *i.e. Network Function Virtualization* (NFV) são as implementações de forma sistematizada das funções de redes. Ou seja, NFV são aplicações SDN desacopladas das especificidades dos equipamentos de hardware, cujo procedimento é definido pelo Instituto Europeu de Padrões para Telecomunicações, *i.e. European Telecommunications Standards Institute* (ETSI) [1]. Já as VNF, por sua vez, são instâncias de software que provêm funcionalidades de rede computacional, como *firewall*, comutação e análise de pacotes, Sistemas de Detecção de Intrusão *i.e. Intrusion Detection Systems* (IDS), *Network Address Translation* (NAT), *Domain Name Server* (DNS), etc. As VNFs são executadas nos recursos de infraestrutura e podem ser abstraídas como blocos de montar espalhados pelas *Cloud Network Slices* (ORDONEZ-LUCENA et al., 2017).

Uma das maiores vantagens da utilização de NFVs é a capacidade substituir sistemas de propósito especial por sistemas com hardware genérico, mas com software específico para o problema a ser resolvido. Os sistemas de propósito especial, que comumente são vendidos na forma de caixas pretas (*i.e. black box*), possuem a desvantagem de custo de capital (CAPEX) relativamente elevado pelas poucas empresas que oferecem tais equipamentos. Os custos

[1] ⟨https://www.etsi.org/technologies/nfv⟩.

de operação (OPEX) também costumam ser altos na abordagem de caixa preta, pela mão de obra especializada necessária para a manutenção do *hardware* e o suporte aos sistemas de gerenciamento de tais equipamentos. Outra vantagem na utilização de NFVs é a capacidade de utilização de redes virtuais centralizadas em servidores do tipo *commodity*, amplamente difundidos em *Datacenters* (ABUJODA; PAPADIMITRIOU, 2016).

O encadeamento de serviço (*service chain*) é uma implementação que representa a sequência ordenada de NFVs, recorrente em *Cloud Network Slices*, as quais possuem um ou vários fluxos de rede atravessando-as. As *service chains* podem ser representadas através de um grafo bidirecional, onde cada vértice representa uma NFV com restrições especificadas e cada aresta representa um link que interliga duas NFVs. Neste grafo, os vértices possuem restrições como quantidade de núcleos da Central de Processamento Unitário (CPU), quantidade de Memória de Acesso Randômico (RAM), armazenamento e localização geográfica, já as arestas possuem restrição de largura de banda. Conjuntamente, o *Network Service Embedding* (NSE) trata da criação de um grafo interligando *datacenters* de localizações específicas, a fim de suprir uma ordem de execução das *service chains*, seguindo certos requisitos. Tais requisitos estão geralmente contidos em um SLA, estabelecendo os parâmetros de desempenho que os recursos contidos na NSE deverão atender (DIETRICH et al., 2017; DIETRICH; ABUJODA; PAPADIMITRIOU, 2015).

O termo orquestração aplicado no contexto das *slices* caracteriza a ação de provisionar serviços e infraestruturas virtualizados, utilizando o monitoramento dos recursos computacionais a fim de gerenciar sua utilização. O monitoramento dos recursos permite gerenciar a elasticidade de uma fatia de rede, *i.e.* que se aumente ou diminua o número de nós (*scale in/out*); também é possível que se expanda ou diminua a quantidade de recursos em um mesmo nó (*scale up/down*). Já a orquestração de NFVs pode ser definida como o gerenciamento e coordenação automatizados do ciclo de vida das VNFs. Dentre o conjunto de funções para gerenciar o ciclo de vida das VNFs está a etapa de instanciação, manutenção e finalização das funções de forma otimizada. As otimizações que um orquestrador pode fazer incluem a escolha de localização geográfica das VNFs, local do ponto final de acesso da rede, tempo de resposta esperado, carga de utilização desejada, custo monetário, a escolha de rotas para evitar *blacklists* de endereços IP, etc (GUERZONI et al., 2017; SOUSA et al., 2019).

Métodos de otimização são comumente utilizados para a escolha de recursos de infraestrutura no contexto de NSE e *service chain* (DIETRICH et al., 2017; ABUJODA; PAPADIMITRIOU, 2016; ARAÚJO; SOUZA; MATEUS, 2018, 2018). Estes métodos comumente utilizam MILP e métodos heurísticos para representar o problema e tentar encontrar uma solução

ótima, *i.e.* evitando pontos máximos ou mínimos locais para encontrar o máximo ou mínimo global em uma função, cuja representação gráfica possui vários picos e vales. Nos métodos de otimização por MILP são elaboradas funções lineares, chamadas de funções objetivo, que representam matematicamente o problema da escolha dos recursos de computação, armazenamento e rede, atribuindo variáveis a cada parâmetro de desempenho e custo financeiro que devem ser considerados na escolha dos recursos. A função objetivo está sujeita às chamadas restrições, que são equações lineares que representam intervalos para os valores das variáveis da função objetivo, *e.g.* a necessidade da quantidade de um recurso ser positiva ou maior que zero.

Nos trabalhos encontrados através da revisão sistemática apresentada no Capítulo 4 é bastante comum a utilização da biblioteca de otimização CPLEX da IBM. A biblioteca CPLEX implementa, entre outros, o algoritmo Simplex, que representa a função objetivo e as restrições através de uma matriz com as variáveis sendo a quantidade de colunas, e a quantidade de equações sendo o número de linhas. Nele é utilizada a técnica de matriz canônica para manipular a matriz de forma a encontrar o valor máximo ou mínimo para a função objetivo. A biblioteca também utiliza métodos heurísticos e outras técnicas que auxiliam no processo de otimização em diferentes situações.

Capítulo 3

NECOS

A seguir será apresentado o projeto NECOS, que foca em definir o conceito de Cloud Network Slices, especificando um escopo delimitado para provar o conceito de slices com uma visão focada em inovação e também no mercado.

Para avançar o estado da arte em *Cloud Network Slices*, o projeto *Novel Enablers for Cloud Slicing* (NECOS)[1], uma parceria entre universidades e empresas europeias e brasileiras, propõe o estudo de técnicas e provas de conceito para tornar o *Cloud Network Slicing* uma realidade. Os principais objetivos do NECOS são o de prover uma plataforma para a utilização dos conceitos de *Slice as a Service* (SaaS), realizar a descoberta dos recursos computacionais disponibilizados pelos provedores de infraestrutura, utilizar a abstração de *Cloud Network Slices* para compor abstrações leves, *i.e. lightweight slice* com a utilização de orquestração para controlar o ciclo de vida das *Cloud Network Slices*.

O conceito de *lightweight slice* se torna muito conveniente no contexto das redes 5G, pois os equipamentos que se encontram na *edge cloud*, ou seja, próximos ao usuário final, geralmente possuem pouco poder computacional se comparados com os de *datacenters* na *core cloud*. Outra característica endereçada pelo *lightweight slice* é a dinamicidade, onde *Slice Parts* ou o todo das *slices* necessitam ser alocados e terminados de forma dinâmica. Equipamentos como servidores de estações de radio base, *smartphones*, servidores empresariais com capacidade ociosa, entre outros elementos da *edge cloud*, podem tornar realidade a utilização de tempos de resposta ultra rápidos com a replicação dos serviços próximos aos usuários finais (LIU et al., 2018).

É então proposta a arquitetura apresentada na Figura 3.1 para a definição da *Cloud Network Slice* no projeto NECOS. Nesta Figura 3.1 é possível observar três divisões principais, o domínio

[1] ⟨http://www.h2020-necos.eu/⟩.

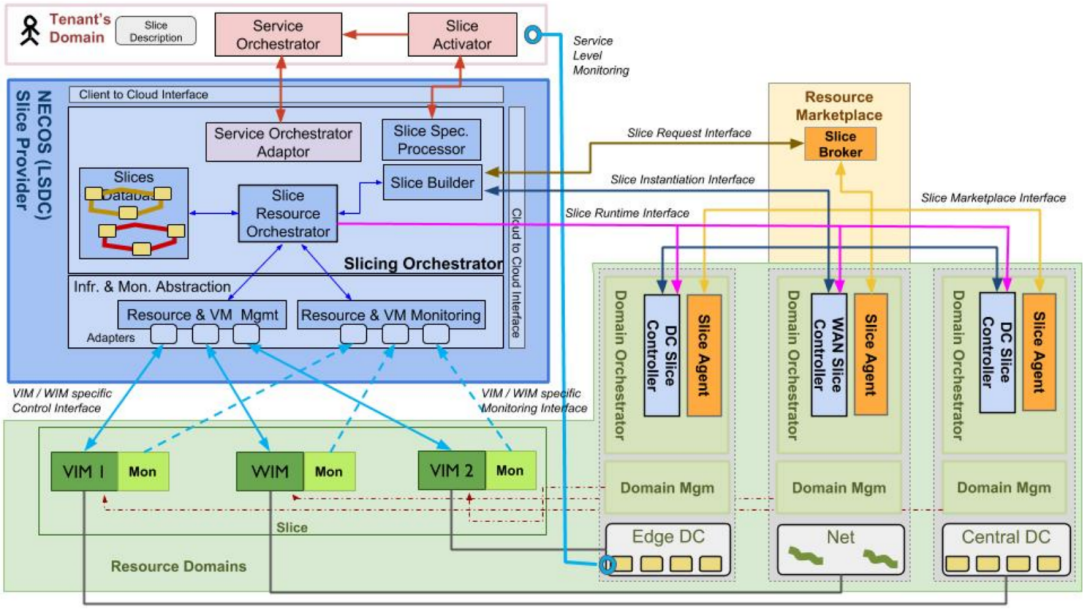


Figura 3.1: Arquitetura funcional do NECOS.
Fonte: (CLAYMAN, 2018)

do locatário, o NECOS *Lightweight Slice Defined Cloud* (LSDC) e o domínio do orquestrador. No domínio do locatário (*tenant*), o ator *tenant* é uma organização ou indivíduo que solicita a *slice* ao módulo *Slice Provider*; o locatário também possui o módulo *Slice Activator* que é responsável por fragmentar a *slice* em *Slice Parts*, o *Slice Activator* envia a fragmentação gerada para o *Slice Broker* e também tem o propósito de receber as especificações da *slice* após a etapa combinação e instanciação ao final do processo; o *Service Orchestrator* permite que o locatário controle o ciclo de vida, da instanciação à finalização dos serviços que irá utilizar; é possível também que o locatário realize o monitoramento dos serviços hospedados através da interface *Service Level Monitoring* ou simplesmente delegar tal tarefa ao NECOS (CLAYMAN, 2018).

Ainda na Figura 3.1, o NECOS LSDC permite a criação da *slice* como um todo, sendo que o *Slice Builder* se comunica com o *Resource Marketplace*, onde se encontram os *Slice Parts* disponibilizadas pelos provedores do Domínio de Recursos (*Resource Domain*); o *Slice Builder* é encarregado de combinar os (*Slice Parts*) ao final do processo; o *Slice Resource Orchestrator* orquestra a *slice* e o serviço como um todo; a abstração *Infrastructure and Monitoring Abstraction* (IMA) permite a instanciação dos serviços e gerenciar informações de monitoramento dos gerenciadores de infraestrutura virtual, *i.e. Virtual Infrastructure Managers* (VIMs) como OpenStack Tacker[2] e VMWare[3]; a IMA também permite a instanciação dos gerenciadores de infraestrutura de área ampla, *i.e. Wide-area network Infrastructure Managers* (WIMs) que ge-

[2] ⟨https://docs.openstack.org/tacker/latest/⟩.
[3] ⟨https://docs.vmware.com/en/VMware-vCloud-NFV/⟩.

renciam os recursos de rede da *slice*, como o Sonata[4]. O módulo *Slice Broker* é responsável por comunicar-se com os *Slice Agents* a fim de realizar a descoberta de *Slice Parts*, cada *Slice Agent* representa unicamente um provedor de infraestrutura, sendo prevista a utilização de vários *Slice Brokers* por unidade federativa (CLAYMAN, 2018). O escopo do trabalho aqui apresentado, na otimização da escolha de recursos se encaixa dentro do módulo *Slice Builder*, o qual é apresentado com mais detalhes na Figura 3.2.

Para estabelecer a criação de uma *Cloud Network Slice* na arquitetura apresentada anteriormente, são necessárias quatro etapas principais: primeiramente deve-se realizar a fragmentação da requisição de *slice* em *Slice Parts*; posteriormente se faz a descoberta dos recursos de infraestrutura que serão integrados em tal abstração; em seguida há a escolha dos recursos aptos a serem agregados; por fim se dá a agregação dos recursos na abstração para a instanciação das aplicações necessárias. A descoberta dos recursos deve ser feita em uma infraestrutura conhecida parcialmente ou em sua totalidade, neste processo, cada provedor oferece os seus recursos disponíveis e os respectivos custos para que os mesmos sejam selecionados pelo locatário e a *Cloud Network Slice* seja formada.

Para alcançar os objetivos do projeto NECOS, duas abordagens distintas foram identificadas para a descoberta dos recursos computacionais e de rede, são elas *Push mode* e *Pull mode* (CLAYMAN, 2018). Na abordagem *push mode*, o provedor de recursos é requisitado toda vez que um locatário solicita a criação de uma *Cloud Network Slice*. Em contrapartida, na abordagem *Pull mode*, os provedores divulgam os recursos disponíveis e seus respectivos custos financeiros, com tais informações é formado um catálogo com os recursos divulgados pelos provedores, que posteriormente será usado para a escolha dos recursos. No trabalho aqui apresentado não será tratada a descoberta de recursos, pois estamos assumindo que os mesmos já estão disponíveis no *marketplace*.

Após a descoberta dos recursos, a *Cloud Network Slice* é fragmentada em vários pedaços divulgados em *Slice Parts*, que irão hospedar funções de rede virtuais e aplicações leves. Os *Slice Parts* são distinguidos entre *DC Slice Parts* e *Net Slice Parts*. A primeira categoria é composta de *Virtual Deployment Units* (VDU), que são os elementos de computação básicos de uma *slice*, neste trabalho é assumido de forma simplificada que cada *DC Slice Part* tem apenas uma VDU. Já os *Net Slice Parts* representam os *links* que conectam os *DC Slice Parts*. Com os fragmentos já formados em passos anteriores, o *Slice Broker* faz uma requisição aos provedores cadastrados, que então verificam se possuem recursos suficientes para atender ao pedido, podendo atender um *Slice Part* isolado ou a *Cloud Network Slice* inteira. Tal abordagem

[4] ⟨https://github.com/sonata-nfv/tng-sp-ia/wiki/WAN-Infrastructure-Manager⟩.

possui uma problemática devido à necessidade de fragmentação da *Cloud Network Slice*, tarefa que requer um conhecimento amplo do propósito a que servirá a *Cloud Network Slice*, assim podendo fragmentá-la de forma eficiente. Com o conhecimento da infraestrutura através dos *Slice Parts*, o otimizador do *Slice Builder* pode então escolher os recursos que desejar e formar a *Cloud Network Slice* de forma eficiente. Tal abordagem de se fragmentar a *slice* em *Slice Parts* e realizar a requisição de fragmentos separados trata com atenção o requisito de que, alguns provedores, podem não concordar em divulgar toda sua infraestrutura, pois estariam em situação de desvantagem comercial perante seus concorrentes (ABUJODA; PAPADIMITRIOU, 2015). Para tal, os provedores precisam divulgar apenas informações necessárias para a atual requisição do locatário.

O fluxo simplificado de mensagens, utilizado para a implementação dos módulos *Slice Builder*, *Slice Agent* e *Slice Broker* da arquitetura apresentada anteriormente (Figura 3.1), é detalhado a seguir na Figura 3.2, sendo que o conjunto de interações se dá na seguinte forma:

1. Inicialmente, o locatário interage com o módulo de *Marketplace*, que funciona como uma loja virtual com os recursos computacionais e de rede disponíveis para serem contratados. O locatário indica os recursos que necessita para suas atividades, assim o *Marketplace* encaminha esta interação na forma de um arquivo descritivo YAML para o módulo *Slice Builder*, com os recursos computacionais e de rede necessários para a aplicação do locatário. Nesta interação com o locatário, assume-se que é especificado exatamente os recursos que o locatário precisa, podendo ser expandido ou diminuído posteriormente em uma nova requisição;

2. O *Slice Builder* processa o arquivo YAML, transformando-o em um *Partially Defined Template* (PDT) com os recursos mínimos que a aplicação do locatário irá utilizar, assim como a localização geográfica em que tais recursos precisam estar. Um exemplo do arquivo PDT pode ser verificado na Figura 3.3a e Figura 3.4a;

3. Em seguida, o *Slice Builder* faz uma requisição ao *Slice Broker*, que atua como um agente intermediador que vende e compra propostas, recebendo a requisição do locatário;

4. O *Slice Broker* encaminha a requisição do locatário na forma de PDTs para todos os *Slice Agents* analisarem a proposta;

5. Posteriormente, cada *Slice Agent* que pode suprir a demanda do pedido do locatário, envia sua proposta, com os recursos computacionais e custo financeiro, ao *Slice Broker*. A resposta do provedor é identificada como *Resource Option*, sendo que cada *Resource Option* representa a proposta de apenas um provedor para um *Slice Part*. As respostas

dos provedores, através dos *Slice Agents*, é representada por um arquivo YAML definido como *Partially Defined Template with Resources Alternatives* (SRA). Um exemplo de SRA pode ser verificado nas Figuras 3.3b e 3.4b;

6. No próximo passo, o *Slice Broker* irá agregar todas as propostas dos *Slice Agents* em um único arquivo, encaminhando-o posteriormente para o *Slice Builder*;

7. Finalmente o *Slice Builder* utiliza o arquivo YAML com as SRAs para escolher a *Resource Option* mais adequada, através de métodos de otimização, levando em consideração o custo e atributos de desempenho de cada *Slice Part*. Este passo é o qual a presente dissertação de mestrado está focada.

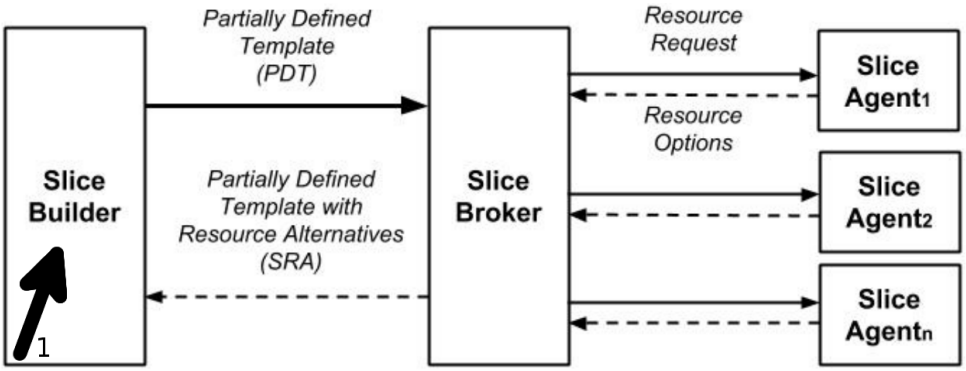


Figura 3.2: Fluxo de mensagens arquivos YAML.

Fonte: (PAPADIMITRIOU, 2018)

O conjunto de etapas apresentado anteriormente na Figura 3.2 representa a criação inicial da *Cloud Network Slice*. Quando o módulo de monitoramento IMA identifica que é necessário realizar a elasticidade para aumentar ou diminuir a quantidade de recursos de infraestrutura da *slice* criada, deve-se fazer uma nova solicitação ao *Slice Broker* e consequentemente um novo ciclo é iniciado. Para a realização da etapa de escolha das *Resource Options* (indicado pela seta número 1 na Figura 3.2), passo no qual foca este trabalho de pesquisa, o módulo *Slice Builder* é utilizado.

```yaml
dc-slice-part:
   name: dc-slice1
   slice-constraints:
      geographic: brazil.saopaulo.saocarlos
   resource-priority:
      memory-mb: 0.2
      storage-mb: 0.2
      cpu-number: 0.2
      power-kwh-day: 0.4 #More priority
   cost:
      dc-model:
         model: COST_PER_PHYSICAL_MACHINE_PER_DAY
         value-euros: {lower_than_equal: 10}
   slice-timeframe:
      service-start-time: {100918: 10 pm} # MMDDYY HH [AM|PM] UTC
      service-stop-time: {101018: 10 pm} # MMDDYY HH [AM|PM] UTC
   vdus: # defining load balancer VDU
      - dc-vdu:
           id: load_balancer
           epa-attributes:
              host-epa:
                 cpu-model: PREFER_COREi9
                 cpu-architecture: PREFER_X86_64
                 cpu-vendor: PREFER_INTEL
                 cpu-number: 3 #Minimum number of cores
                 storage-mb: {greater_or_equal: 5000}
                 storage-type: SSD #Optional value
                 memory-mb: {greater_or_equal: 8000}
```

(a) *DC Slice Part* requisição do locatário.

```yaml
dc-slice-part:
   name: dc-slice1 # Same string of the PDT YAML
   slice-constraints:
      geographic: brazil.saopaulo.saocarlos
   cost:
      dc-model:
         model: COST_PER_PHYSICAL_MACHINE_PER_DAY
         value-euros: 9 # Cost of only this slice part
   slice-timeframe:
      service-start-time: {100918: 10 pm} # MMDDYY HH [AM|PM] UTC
      service-stop-time: {101018: 10 pm} # MMDDYY HH [AM|PM] UTC
   dc-slice-controller:
      dc-slice-provider: IBM
   vdus: # defining load balancer VDU
      - dc-vdu:
           id: load_balancer
           epa-attributes:
              host-epa:
                 cpu-model: COREi9
                 cpu-architecture: X86_64
                 cpu-vendor: INTEL
                 cpu-number: 4
                 storage-mb: 6000
                 storage-type: SSD # Optional value
                 memory-mb: 8192
                 power-kwh-day: 30
```

(b) *DC Slice Part* proposta do provedor.

Figura 3.3: Exemplo arquivos YAML *DC Slice Parts*.
Fonte: Adaptado de (CLAYMAN, 2018).

```yaml
net-slice-part:
   name: net-slice1
   slice-constraints:
      geographic: brazil.saopaulo.saocarlos
   cost:
      net-model:
         model: COST_PER_LINK_PER_DAY
         value-euros: {lower_than_equal: 45}
   slice-timeframe:
      service-start-time: {100918: 10 pm} # MMDDYY HH [AM|PM] UTC
      service-stop-time: {101018: 10 pm} # MMDDYY HH [AM|PM] UTC
   links:
      - dc-part1:
           name: dc-slice1
      - dc-part2:
           name: dc-slice2
   constraints:
      bandwidth-mb: 10 #Minimum
```

(a) *Net Slice Part* requisição do locatário.

```yaml
net-slice-part:
   name: net-slice1
   slice-constraints:
      geographic: brazil.saopaulo.saocarlos
   cost:
      net-model:
         model: COST_PER_LINK_PER_DAY
         value-euros: 45 # Cost of only this slice part
   slice-timeframe:
      service-start-time: {100918: 10 pm} # MMDDYY HH [AM|PM] UTC
      service-stop-time: {101018: 10 pm} # MMDDYY HH [AM|PM] UTC
   wan-slice-controller:
      wan-slice-provider: telefonica
   links:
      - dc-part1:
           name: dc-slice1
           dc-slice-provider: IBM
      - dc-part2:
           name: dc-slice2
           dc-slice-provider: Amazon
   constraints:
      bandwidth-mb: 10
```

(b) *Net Slice Part* proposta do provedor.

Figura 3.4: Exemplo arquivos YAML *Net Slice Part*.
Fonte: Adaptado de (CLAYMAN, 2018).

Capítulo 4

TRABALHOS RELACIONADOS

Antes de entrar nos detalhes do modelo de otimização desenvolvido, o capítulo a seguir apresenta os trabalhos relacionados com o tema do projeto proposto, onde foi utilizada uma revisão sistemática através da ferramenta StArt para o auxílio do método de busca e organização dos artigos científicos.

Foi realizada uma revisão sistemática através da ferramenta StArt (FABBRI et al., 2016), para o auxílio do levantamento e organização do conteúdo sobre o estado da arte em métodos de otimização para seleção de recursos de infraestrutura de nuvem, mas não se delimitando apenas aos resultados desta revisão. A revisão disponibilizada contém todos os artigos recuperados pelos motores de busca utilizando a *string* de busca definida, assim como alguns artigos encontrados em referências durante a revisão que se mostraram relevantes para o tema. São também explicitados os artigos efetivamente analisados pela seleção dos mais importantes através do resumo de cada artigo. Os arquivos necessários para a utilização da revisão sistemática realizada estão disponíveis publicamente para consulta[1]. Na época em que foi realizada a revisão sistemática, foram encontrados poucos trabalhos técnicos na literatura sobre métodos de otimização específicos para *Cloud Network Slices*. Foram separados alguns artigos com grande relevância para o tema desta dissertação, os quais são apresentados a seguir.

O DistNSE (ABUJODA; PAPADIMITRIOU, 2016) é um método heurístico de otimização, implementado de forma distribuída, que encoraja a colaboração entre provedores de NFVs, enquanto provém privacidade para as informações sensíveis da infraestrutura dos mesmos. Os serviços de rede virtuais utilizam tais funções de rede para sua execução, e a escolha das opções mais eficientes de acordo com parâmetros previamente definidos é uma tarefa crucial para tal. O método apresentado utiliza o conceito de *Network Service Embedding* (NSE), em que NFVs são integradas em uma cadeia de serviço, havendo uma ordem pré definida de execução das NFVs.

[1] ⟨https://zenodo.org/record/4582772#.YEFK8_tKiV4⟩.

A plataforma permite que provedores distintos possam competir de forma descentralizada para oferecer o melhor custo/benefício para o locatário. Dessa forma provendo autonomia para os provedores que oferecem seus recursos e criando chances igualitárias de competição para todos os provedores. Outra característica do DistNSE é a garantia de privacidade do provedor ao não divulgar a topologia e o nível de utilização de toda sua infraestrutura para os provedores concorrentes. Os autores validaram o modelo através de um simulador em Python, onde o mesmo se mostrou pouco custoso para os clientes ao escolher a proposta com menor valor monetário, e também fomentando competição entre os diferentes provedores para todas as NFVs em uma *service chain*. Foi concluído que o método utiliza pouca banda de transmissão nos *datacenters* devido à escolha de rota mais curta entre as NFVs e também que a quantidade de mensagens de controle trocadas é proporcional ao número de provedores participantes na disputa da cadeia de serviço. Todavia, os autores não mencionam a forma como é descrita a topologia e os recursos oferecidos pelos provedores, deixando dúvidas como é possível garantir a privacidade do mesmo.

O trabalho apresentado por Dietrich et al. (2017) propõe um método de otimização utilizando MILP e heurísticas, intitulado NESTOR, que funciona de forma similar a um orquestrador de NFVs, utilizando recursos de diferentes provedores. O método usa uma camada de composição de serviços de rede entre os locatários e os provedores, buscando tratar da privacidade sobre informações sensíveis de infraestrutura e competição justa entre os provedores. A camada criada tenta resolver a alocação de recursos nos casos em que um único provedor não é capaz de fornecer todas as NFVs necessárias para a cadeia de serviço devido a restrições como localização e banda de rede disponível. Este método pode ser usado em aplicações que dependem de armazenamento de *cache* para acesso rápido de conteúdos frequentemente procurados. Também aplica o conceito de ranqueamento para as VNFs ofertadas, elencando-as de acordo com a quantidade de recursos disponíveis e possibilitando o mapeamento em diferentes *datacenters* controlados por diferentes provedores. É a única solução encontrada pelos autores que possibilita a modificação dos NFVs de acordo com a necessidade de largura de banda de rede. Os autores tentam resolver o alto tempo de execução da otimização, pelo *solver*, através de uma técnica de arrendondamento, diminuindo a complexidade do problema inicial para utilizar programação linear. A solução proposta cumpre o objetivo de reduzir custos para aumentar o acesso às soluções pelos clientes, porém gera soluções de rede que não são ótimas devido à técnica de arredondamento, podendo restringir o público a clientes que não são muito exigentes.

Addis et al. (2015) utilizam o padrão do ETSI para gerenciamento e orquestração de SDN, onde consta a necessidade de se considerarem as características específicas de cada aplicação para a especificação das VNFs, como atributos de localização, armazenamento e processa-

mento. Os autores formulam um método de otimização utilizando MILP, porém também elabora uma abordagem meta heurística para tratar a complexidade do problema. O objetivo definido é maximizar a utilização da banda de rede, considerando o posicionamento geográfico ótimo e também o custo mínimo para as VNFs utilizadas. É levado em consideração o tempo de resposta necessário e o correto dimensionamento da quantidade de CPUs e memória RAM para aplicações em larga escala e com o tempo de processamento adequado ao seu funcionamento. Os testes realizados pelos autores, usando como caso de uso a implementação de um *Firewall* em VNF, indicaram que o dimensionamento de tráfego, em conjunto com a minimização do custo em infraestrutura, tende a utilizar quantidades menores de instâncias de NFV em comparação com o dimensionamento de tráfego de forma isolada. Foi observado também que é necessário cerca de 20% mais recursos de infraestrutura comparando à abordagem padrão de direcionamento de pacotes e a *fastpath* também analisada, o que mostra um ganho considerável na redução de custos do serviço para os locatários. Os autores não analisaram a privacidade dos provedores ao divulgar seus recursos de infraestrutura, o que mostra a necessidade de uma olhar mais realista para a aplicação.

Outra abordagem que foca na alocação eficiente de recursos dentro de *datacenters* é Pentelas et al. (2020). O trabalho também usa o conceito de uma rede de serviços para organizar VNFs entre os servidores de acordo com sua capacidade ociosa. Os autores dão atenção particular à forma de alocação de múltiplos tipos de recursos entre VNFs e servidores através de um mecanismo de seleção. Para tal, foi desenvolvida uma heurística que considera duas dimensões de recursos (*i.e.* CPU e RAM), a mesma foi comparada com um método MILP, que também exercita tais dimensões, sendo também utilizada na comparação uma heurística que não exercita nenhum aspecto de multidimensionalidade. A partir dos resultados, foi demonstrado que houveram ganhos significativos ao incorporar noções de múltiplas dimensões no conceito de NSE, que preza pela utilização balanceada entre os servidores. A heurística desenvolvida teve melhor desempenho que os outros métodos, contribuindo para maximizar os lucros de provedores de infraestrutura com a maximização do uso de seus recursos. Apesar dos bons resultados, este trabalho não leva em consideração aspectos focados no locatário.

Um artigo também baseado no padrão para orquestração e gerenciamento de SDNs, desenvolvido pelo ETSI, incorpora capacidade de processamento dos componentes de computação no contexto de *Network Slicing* (ORDONEZ-LUCENA et al., 2017). Neste trabalho, os autores utilizam um método baseado em MILP e também um algoritmo heurístico guloso para maximizar a largura de banda de rede considerando localização geográfica e custo financeiro. Os resultados mostram uma melhoria de 20% do MILP na alocação de recursos de infraestrutura se comparado ao algoritmo guloso. Porém este trabalho não considera aspectos de privacidade

do provedor de infraestrutura. Por fim, outro trabalho (HA; LE, 2017) foca na combinação de aspectos do conceito de *Cloud Slicing* e *Network Slicing* da infraestrutura das redes 5G. O trabalho gera resultados otimizados para equipamentos de redes sem fio baseando-se em restrições que combinam antenas de rádio base e recursos de *cloud*.

Os trabalhos anteriormente apresentados mostram métodos de otimização em contextos similares, porém não abordam o conceito inovador de *Cloud Network Slices*, identificado como avanço necessário no estado da arte deste assunto. Neste trabalho, pretende-se avançar o estado da arte juntando preferências de parâmetros de desempenho do locatário com custos financeiros anunciados por vários provedores de infraestrutura. A abordagem tomada consiste em fragmentar a *slice* em *Slice Parts*, selecionar através de otimização o conjunto de recursos para essas partes, posteriormente, combinar as partes. Devido ao fato de fragmentar a *slice*, o problema torna-se mais simples para ser tratado por métodos de otimização.

Capítulo 5

MÉTODOS UTILIZADOS

O capítulo a seguir apresenta os métodos utilizados neste trabalho de pesquisa, descrevendo quais foram utilizados e como pretende-se avançar o estado da arte no tema tratado.

Embora as técnicas de otimização sejam bem conhecidas para o contexto de escolha de *service chains* e *Network Service Embedding*, como mostrado no Capítulo 4, este projeto de mestrado propõe a aplicação da otimização usando o método de MILP no contexto de *Cloud Network Slices* da arquitetura funcional idealizada no projeto NECOS. Tal abordagem caracteriza uma novidade na área, pois até o presente momento não foi encontrado nenhum trabalho similar na revisão sistemática descrita no Capítulo 4.

O trabalho aqui apresentado se situa no passo após a descoberta de recursos oferecidos pelos provedores através dos *Slice Agents* da Figura 3.2. Para tal são usados como parâmetros o preço financeiro que o provedor de infraestrutura divulga para cada *Resource Option*, os requisitos mínimos de recursos de infraestrutura e o consumo energético dos *DC Slice Parts* e dos *Net Slice Parts*, especificados em um arquivo YAML (SRA). Desta forma, o custo financeiro dos recursos divulgados através do *Marketplace* é estabelecido pelo provedor de infraestrutura, pois ele possui o conhecimento de sua infraestrutura como um todo. É possível para o provedor avaliar os locais em que seus recursos serão hospedados, procurando por menores custos de manutenção dos equipamentos, devido à falta de mão de obra por exemplo; avaliar os locais onde o custo agregado de energia elétrica é mais elevado, devido a políticas governamentais e equipamentos com maior consumo; estabelecer regiões que possuem maior procura por recursos, devido à estarem em regiões muito populosas ou de alto índice de desenvolvimento tecnológico; é possível também verificar regiões onde o valor do terreno utilizado pode encarecer a hospedagem dos recursos.

Apesar de os requisitos mínimos para a *Cloud Network Slice* estarem intrinsecamente rela-

cionados ao serviço que será executado nela, necessitando de um conhecimento profundo sobre sua aplicação que apenas o locatário possui, o conceito de *Cloud Network Slice* é agnóstico ao serviço que é executado nele. Como exemplo podem ser citados serviços de *Web Server*, que geralmente necessitam de quantidades razoáveis de memória RAM e de CPU; serviços de *streaming* de vídeo precisam de grandes quantidades de armazenamento, assim como tempos de resposta que não debilitem a experiência ao assistir os vídeos e talvez haja necessidade de armazenamento de alto desempenho com tecnologias como *Solid State Drive* (SSD) e *Serial Attached SCSI* (SAS). A informação sobre o consumo energético dos *DC Slice Parts*, componentes da *Cloud Network Slice*, se faz necessária em casos onde o locatário prefira utilizar equipamentos que possam oferecer nível de desempenho computacional similar, mas que agreguem baixo consumo de energia. Tal característica pode propiciar selos de eficiência energética e de empresa amiga do meio ambiente para os locatários.

Para gerenciar o requisito de *lightweight slice* e facilitar a participação de vários provedores de infraestrutura, a *Cloud Network Slice* é fragmentada em vários *Slice Parts* interconectados, porém a forma como é feita tal fragmentação não entra no escopo deste trabalho. Este conjunto de *Slice Parts* é composto de fragmentos de *slice* de *datacenter* representados por *DC Slice Parts* e de fragmentos de rede representados por *Net Slice Parts*. O *DC Slice Part* contém os recursos de computação e de armazenamento, já o *Net Slice Part* interconecta os *DC Slice Parts* e estabelece os requisitos de rede. A etapa de fragmentação da *Cloud Network Slice* é realizada previamente e não entra no escopo do trabalho aqui apresentado.

Após a fragmentação da *Cloud Network Slice* em *Slice Parts*, estes fragmentos com as especificações de recursos de infraestrutura em forma de *Partially Defined Template*, são enviados para os provedores através do *Slice Broker* e dos *Slice Agents*. Os provedores que atendem aos requisitos especificados nos PDTs, como localização e consumo energético enviam suas propostas no formato de de SRAs para o *Slice Broker* que encaminha ao *Slice Builder* escolher quais SRAs farão parte da *Cloud Network Slice*.

5.1 Formulação MILP

Nesta seção será discutido o modelo matemático, apresentando a função objetivo e as restrições do método MILP. Esté método foi desenvolvido para o processo de escolha dos *Slice Parts*, que formarão a *Cloud Network Slice* requisitada pelo locatário, em cenários com muita complexidade gerada pelas muitas combinações ao se escalar o problema. Para a formulação, assume-se a existência de duas topologias complementares: a topologia abstrata, criada a partir

da solicitação do locatário e a topologia física, criada pelas ofertas dos provedores para a topologia abstrata. A topologia abstrata, especificada pelo locatário através do NECOS *marketplace*, representa os recursos desejados pelo locatário em uma única solicitação. Já a topologia física, é gerada a partir do arquivo YAML (SRA) com os *Slice Parts* ofertados pelos provedores (*Resource Options*), *i.e.* representa o que os provedores possuem para suprir a demanda desejada pelo locatário.

Como exemplo, uma topologia abstrata pode ser gerada pela requisição do locatário nos arquivos YAML da Figura 3.3a e Figura 3.4a. Esta topologia abstrata pode ser exemplificada com a Figura 5.1, em que o locatário solicita a criação de uma *Cloud Network Slice* entre as cidades de São Carlos, no Brasil e Tessalônica, na Grécia. A *slice* possui dois *DC Slice Parts*, *dc-slice1_t* e *dc-slice2_t* em São Carlos, representando dois *datacenters* regionais no Brasil e um terceiro, *dc-slice3_t*, na Grécia. Por fim, os *Net Slice Parts*, *net-slice1_t* e *net-slice2_t* interconectam os *DC Slice Parts* citados anteriormente.

A topologia abstrata contém os requisitos do locatário para a especificação da CNS, *i.e.* os *DC Slice Parts* interconectados pelos *Net Slice Parts* que farão partes da CNS, com seus respectivos requisitos de recursos. Com foco na solicitação da topologia abstrata (PDT), os provedores de infraestrutura farão ofertas de *Resource Options* contendo suas propostas, através dos *Slice Agents*. Por conseguinte, um exemplo de topologia física, para suprir a requisição do locatário citada anteriormente, pode ser observado nos arquivos YAML da Figura 3.3b e Figura 3.4b. Tais *Resource Options* representados no arquivo YAML geram a topologia física (SRA) apresentada na Figura 5.2.

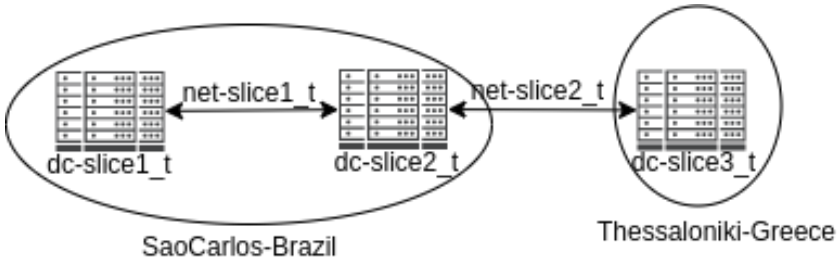


Figura 5.1: Exemplo topologia abstrata (PDT) da *Slice*.

Para a representação matemática deste cenário, é necessária a formalização dos conjuntos utilizados, a qual é apresentada a seguir:

Definição 1. *Seja D_V o conjunto de todos os DC Slice Parts (datacenter) da solicitação do locatário (PDT), E_V o conjunto de todos os Net Slice Parts (rede) que interconectam os DC Slice Parts. Juntos, os conjuntos D_V e E_V geram a topologia abstrata.*

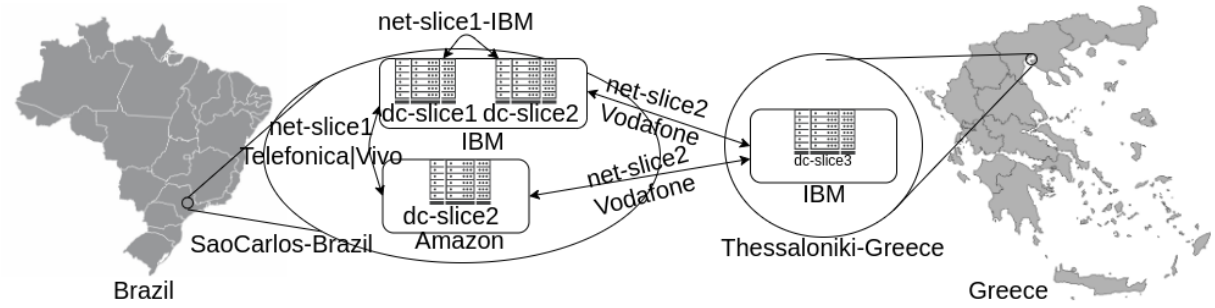


Figura 5.2: Examplo topologia física (SRA) da *Slice*.

Definição 2. *Seja D o conjunto de todos os DC Slice Parts oferecidos pelos provedores (SRA), E o conjunto de todos os Net Slice Parts ofertados que interconectam os DC Slice Parts. Juntos, os conjuntos D e E geram a topologia física.*

Definição 3. *Seja v a quantidade de memória RAM em megabytes (MB), s a quantidade de armazenamento em disco (MB), t a quantidade de núcleos da CPU e e o consumo energético, em kilowatt-hour (kWh) por dia, então o conjunto $R = [v,s,t,e]$ inclui as diferentes possibilidades de recursos de infraestrutura ofertados pelos provedores através dos DC Slice Parts físicos (SRA).*

Definição 4. *Seja E_V o conjunto de todos os Net Slice Parts que interconectam os DC Slice Parts abstratos, E_V^{-1} o conjunto simétrico dos mesmos Net Slice Parts, então a concatenação $E_V \frown E_V^{-1}$ possibilita mostrar ao método de otimização que devem ser considerados links de rede com comunicação em ambas as direções.*

Definição 5. *Seja E o conjunto de todos os Net Slice Parts que interconectam os DC Slice Parts físicos, E^{-1} o conjunto simétrico dos mesmos Net Slice Parts, então a concatenação $E \frown E^{-1}$ também compreende direções opostas na comunicação dos links ofertados.*

A Equação 5.1 a seguir mostra a concatenação dos conjuntos apresentados anteriormente na Definição 4 e Definição 5 de *Net Slice Parts* físicos e abstratos com seus conjuntos simétricos. Tal formulação possibilita mostrar ao método de otimização que devem ser consideradas todas as possibilidades fornecidas a partir da requisição do locatário na topologia abstrata (PDT) em conjunto com a topologia física dos provedores (SRA), *e.g. E = (dc-slice1_IBM, dc-slice2_IBM)* $\frown$ *(dc-slice2_IBM, dc-slice1_IBM).*

$$\begin{aligned} E_V = E_V \frown E_V^{-1}, E_V^{-1} = [(v,u) : (u,v) \in E_V] \\ E = E \frown E^{-1}, E^{-1} = [(j,i) : (i,j) \in E] \end{aligned} \tag{5.1}$$

Definição 6. *Seja m^u_{1k} o recurso m do tipo k do Resource Option 1 ofertado por um provedor de infraestrutura para o DC Slice Part u, então m^u_{ik} é o conjunto de quantidades para cada recurso m do tipo k de cada Resource Option i dos provedores de infraestrutura para o DC Slice Part u conforme Equação 5.2.*

$$m^u_{ik} = [m^u_{1k}, m^u_{2k}, ..., m^u_{nk}], \forall u \in D_V, k \in R \tag{5.2}$$

Cada quantidade de recurso m do tipo k contido no *DC Slice Part* físico i ofertado para suprir a requisição de *DC Slice Part* abstrato u deve ser no mínimo o valor especificado pelo locatário (t) para o mesmo recurso k do *DC Slice Part u*. Tal informação está representada na Equação 5.3 e é recuperada do arquivo YAML gerado pela solicitação do locatário.

$$m^u_{ik} >= m^t_{uk}, \forall i \in D, u \in D_V, k \in R \tag{5.3}$$

A Equação 5.4 representa a normalização da quantidade de recursos oferecidos por cada provedor para a solicitação do locatário. Essa informação pode ser interpretada como sendo a oferta (*Resource Option*), de um provedor de infraestrutura para o recurso k contida no *DC Slice Part* físico i em resposta à solicitação u do locatário. O valor resultante dessa normalização varia entre zero e um para cada tipo de recurso. Tal ação tem o intuito de manter a proporcionalidade entre valores de diferentes tipos de recursos, evitando priorizar parâmetros que possuem ordens de grandeza maiores. *E.g.* a quantidade de memória RAM pode variar entre um *gigabyte* (GB) e 1000 GB, *i.e.* ordens de magnitude maior do que a quantidade de núcleos de CPU, variando normalmente entre um e dez.

$$m'^u_{ik} = \frac{m^u_{ik} - min(m^u_k)}{max(m^u_k) - min(m^u_k)}, \forall i \in D, k \in R, u \in D_V \tag{5.4}$$

Com o intuito de representar o quão bom é cada *DC Slice Part i* ofertado por determinado provedor, a Equação 5.5 especifica um fator de risco atrelado a cada oferta (*Resource Option*) e comparado com todas as ofertas dos outros provedores. A equação pode ser interpretada como sendo o fator de risco da *Resource Option i* ofertada para o *DC Slice Part* abstrato u. O valor do fator de risco varia entre zero e um, sendo que quanto mais próximo o fator de risco chega de um, maior é o risco para o locatário. Em contrapartida, quanto mais próximo de zero, menos risco existe para o locatário nesta oferta, pois ele terá mais recursos do que o necessário. A ideia relacionada ao fator de risco é que, com um risco alto, o locatário terá estritamente o que solicitou, o que pode se tornar problemático se a quantidade de recursos foi dimensionada de

forma errônea. Ne Equação 5.5, o parâmetro d representa a prioridade que o locatário escolhe para cada indicador de desempenho, *i.e.* para expressar qual indicador é mais importante para o locatário.

$$r_i^u = \sum_{k \in \{v,s,t\}} (d_k(1 - m_{ik}^{,u})) + d_e m_{ie}^{,u}, \forall i \in D, k \in R, u \in D_V \tag{5.5}$$

É importante mencionar que o consumo energético pode ser considerado como informação sensitiva para alguns provedores de infraestrutura, então alguns atores podem preferir não compartilhar tal informação. Nestes casos, o parâmetro de consumo energético deve ser removido do fator de risco na Equação 5.5. A Equação 5.6 representa que a somatória da prioridade de todos os parâmetros de desempenho de um *DC Slice Part* resulta em um, *i.e.* quando a prioridade de um elemento é aumentada, a prioridade de outros elementos deve diminuir para manter a restrição. *E.g.* na Figura 3.3a, s somatória da prioridade da quantidade de memória RAM (0.2) para o locatário, armazenamento (0.2), CPU (0.2) e consumo energético (0.4) resulta em um, sendo que qualquer mudança em um deles reflete no outro.

$$\sum_{k \in R} d_k = 1 \tag{5.6}$$

A Equação 5.7 formaliza que o o custo final c_i^u para o *DC Slice Part* físico i é dependente do preço financeiro c estipulado pelo provedor e o fator risco r para o locatário. O preço p_i^u deve ser no máximo o especificado pelo locatário durante a requisição (PDT), como na Equação 5.8.

$$c_i^u = p_i^u(1 + r_i^u), \forall i \in D, u \in D_V \tag{5.7}$$

$$p_i^u <= p_u^t, \forall i \in D, u \in D_V \tag{5.8}$$

É representado na Equação 5.9 que, o custo final c_i^u para o *DC Slice Part* físico i, assim como o custo final c_{ij}^{uv} para o *Net Slice Part* físico estão contidos no conjunto dos números reais, positivos e diferentes de zero. Foi assumido que o preço financeiro p_i^u é informado pelo provedor de infraestrutura no arquivo YAML com as *Resource Options* (SRA). Todavia, a forma com que o preço financeiro foi calculado pelo provedor, não está no escopo deste trabalho. Apesar deste fato, o preço financeiro pode ser calculado através de outra função de otimização que foca nos aspectos do provedor, *e.g.* (DIETRICH et al., 2017) e (PENTELAS et al., 2020), tentando encontrar a máxima eficiência no uso dos recursos para oferecer um preço competitivo.

$$(c_i^u, c_{ij}^{,uv}) \in \mathbb{R}_+^* \tag{5.9}$$

De forma similar ao fator de risco dos *DC Slice Parts* físicos (*Resource Options*), da Equação 5.5, também é calculado o custo final c_{ij}^{uv}, sendo a largura de banda de rede, um fator de risco para o *Net Slice Part*. Tal equação representa um *link* de rede conectando dois *DC Slice Parts* físicos, i e j, que são *Resource Options* (SRA) para os *DC Slice Parts* abstratos u conectado a v, da requisição (PDT) do locatário. Com tal combinação, o provedor tem a chance de oferecer *Slice Parts* interconectados dentro de sua própria infraestrutura, evitando que fluxos de rede façam um caminho custoso ao acessar a *Internet* pública, posteriormente tendo de voltar para o *datacenter* do provedor. Junto ao preço financeiro anunciado pelo provedor, é usada a largura de banda b_{ij}^{uv} para definir um fator de risco que privilegia *links* que suportam mais fluxos de rede, representado na Equação 5.10.

$$c_{ij}^{,uv} = p_{ij}^{uv}(1 + \frac{1}{b_{ij}^{uv}}), \forall (i,j) \in E, (u,v) \in E_V \tag{5.10}$$

Em casos onde existe mais de um *Resource Option* para *Net Slice Parts*, será escolhida a opção com menor custo final c, como especifica a Equação 5.11. Tal abordagem foca na redução da complexidade da função objetivo e também na melhor escolha de *Net Slice Parts* para o locatário. De modo similar ao *DC Slice Part*, o preço financeiro c do *Net Slice Part* deve ser no máximo aquele especificado pelo locatário na requisição (PDT) da *slice*, como especificado na Equação 5.12. Em contrapartida, a largura de banda de rede deve ser no mínimo aquela especificada pelo locatário, como na Equação 5.13, *i.e.* para que se atinja a desempenho desejada na aplicação.

$$c_{ij}^{uv} = min[c_{1ij}^{,uv}, c_{2ij}^{,uv}, ..., c_{nij}^{,uv}], \forall (i,j) \in E, (u,v) \in E_V \tag{5.11}$$

$$c_{ij}^{uv} <= p_{uv}^t, \forall (i,j) \in E, (u,v) \in E_V \tag{5.12}$$

$$b_{ij}^{uv} >= b_{uv}^t, \forall (i,j) \in E, (u,v) \in E_V \tag{5.13}$$

Por fim, a formulação da função objetivo do método de MILP é apresentada na Equação 5.14, a mesma pode ser dividida em duas partes: a parte à esquerda da soma refere-se aos *DC Slice Parts* enquanto que a parte à direita refere-se aos *Net Slice Parts*. Na parte à esquerda (DC) a variável de seleção x é multiplicada pelo parâmetro de custo final c_i^u. Na parte à direita (*Net*) tem-se a variável de seleção y multiplicada pelo parâmetro de custo final c_{ij}^{uv}, do *Net Slice Part* que interconecta o par de *DC Slice Parts* físicos (i, j). O par de *DC Slice Parts* físicos (i, j)

representa as *Resource Options* de um ou dois provedores, para a requisição dos *DC Slice Parts* abstratos (u, v). É assumido que o provedor de infraestrutura irá oferecer *Resource Options* de *Net Slice Parts*, para interconectar o par de *DC Slice Parts* físicos *(i,j)*, apenas se o mesmo é capaz de interconectá-los, usando sua própria infraestrutura, ou terceirizando tal tarefa. A função objetivo possui sua normalização e outras restrições nas Equações 5.15, 5.16, 5.17, 5.18 e 5.19.

Minimize:

$$\sum_{u \in D_V} \sum_{i \in D} x_i^u c_i^u + \sum_{(u,v) \in E_V} \sum_{(i,j) \in E} y_{ij}^{uv} c_{ij}^{uv} \tag{5.14}$$

Subject to:

$$2y_{ij}^{uv} <= x_i^u + x_j^v, \forall (i,j) \in E, (u,v) \in E_V \tag{5.15}$$

$$\sum_{i \in D}^{n} x_i^u = 1, \forall u \in D_V \tag{5.16}$$

$$x_i^u \in \{1|0\}, \forall i \in D, \forall u \in D_V \tag{5.17}$$

$$\sum_{(i,j) \in D}^{n} y_{ij}^{uv} = 1, \forall (u,v) \in D_V \tag{5.18}$$

$$y_{ij}^{uv} \in \{1|0\}, \forall (i,j) \in D, \forall (u,v) \in D_V \tag{5.19}$$

A representação matemática de que um *Net Slice Part* y_{ij}^{uv} conecta dois *DC Slice Parts*, x_i^u e x_j^v, encontra-se na Restrição 5.15 da função objetivo, *i.e.* se este *link* é selecionado, então os dois *DC Slice Parts* que o link conecta devem ser selecionados. Apenas uma oferta (*Resource Option*) para o *DC Slice Part* abstrato *u* deve ser selecionada, tal oferta deve ser aquela representada por *i*. Tal sentença é representada pela Restrição 5.16 e Restrição 5.17. A Restrição 5.17, também representa que a variável de seleção *x*, vai ter obrigatoriamente os valores 0 ou 1, *i.e.* a escolha ou não, respectivamente, do *DC Slice Part* físico. Cada provedor possui apenas um *Resource Option* (SRA) por *DC Slice Part* abstrato. De forma similar, a Restrição 5.18 e Restrição 5.19 representam que apenas um *Net Slice Part* da topologia física (SRA) será selecionado para conectar dois *Slice Parts* de *datacenter* i e j.

Toda a formulação apresentada anteriormente foi implementada na linguagem de programação *Python*, com a utilização da biblioteca de otimização *IBM ILOG CPLEX Optimization Studio v12.8 - Student*[1] para a implementação do método por MILP. Devido ao fato do IBM CPLEX necessitar de licença para utilização do *software*, o ambiente de execução não pôde ser disponibilizado publicamente como um todo, porém o código em Python implementado está disponível

[1] https://www.ibm.com/analytics/cplex-optimizer.

publicamente[2], os resultados obtidos serão apresentados a seguir.

[2] ⟨https://zenodo.org/record/4582772#.YEFK8_tKiV4⟩.

Capítulo 6

RESULTADOS

Neste capítulo serão apresentados detalhes sobre a implementação e também detalhes dos cenários utilizados para os experimentos, assim como os resultados obtidos com tais experimentos.

A implementação do trabalho da dissertação aqui apresentado segue o fluxo simplificado de mensagens no formato YAML (PDT e SRA) definido pelo projeto NECOS na Figura 3.2 do Capítulo 3. Durante a etapa de descoberta de recursos, a solicitação da *Cloud Network Slice* já fragmentada em *DC Slice Parts* e *Net Slice Parts* abstratos, é feita aos provedores disponíveis por intermédio do *Slice Broker* e dos *Slice Agents*. Nesta solicitação são especificados os requisitos para a *slice* em sua integralidade, a qual pode ser representada por um grafo bidirecional, sendo a topologia abstrata, em que cada nó é um *DC Slice Part* e cada aresta é um *Net Slice Part* que interconecta dois *DC Slice Parts*.

Após a solicitação do locatário, é feita uma verificação de interconexão entre os *DC Slice Parts*. Nesta verificação são analisados todos os *DC Slice Parts* abstratos contidos no conjunto dos *Net Slice Parts* (E_V), assim são selecionados para a próxima etapa apenas os *DC Slice Parts* que estão interconectados. Posteriormente à verificação de interconexão dos *DC Slice Parts*, são analisadas as respostas dos provedores de infraestrutura através das SRAs. Na Figura 3.3b e Figura 3.4b é possível observar um exemplo desta resposta, onde o provedor IBM oferece uma proposta (*Resource Option*) para o *dc-slice1* da Figura 3.3b, enquanto que o provedor Telefônica oferece uma proposta para o *net-slice1* da Figura 3.4b que interconecta o *dc-slice1* e o *dc-slice2*.

Na Figura 3.3b, os campos descritos no arquivo YAML da resposta do provedor são interpretados da seguinte forma:

- *name* é o identificador único do *DC Slice Part* para esta requisição do locatário;

- *geographic* restringe o local em que o *DC Slice Part* deve ser hospedado, sendo usado para delimitar uma região geográfica próxima ao locatário, almejando tempos de resposta baixos;
- *cost* define o modelo de preço financeiro do *DC Slice Part*, que geralmente é cobrado por dia a cada *DC Slice Part*, é o valor efetivamente cobrado pelo provedor;
- *slice-timeframe* delimita a janela de tempo do *DC Slice Part*, com o momento de início, onde será realizada a alocação do *DC Slice Part* e fim, em que o *DC Slice Part* é desalocado e portanto deixa de existir;
- *vdus* são os *Virtual Deployment Units*, máquinas físicas que o provedor oferece para o locatário, os componentes internos dos VDUs são usados pelo provedor para compor o preço divulgado para o *DC Slice Part*. Nesta implementação está sendo assumido apenas um VDU por *DC Slice Part*;
- *id* identifica cada VDU, geralmente pelo nome do serviço que será executado nele;
- *instance-count* delimita a quantidade de instâncias da máquina no *datacenter* do provedor;
- *hosting* indica se o VDU será dedicado ou compartilhado;
- *epa-attributes* lista os componentes e seus atributos que estarão disponíveis neste VDU.

Para o *Net Slice Part* físico da Figura 3.4b, os campos com o mesmo nome dos *DC Slice Parts* são interpretados da mesma forma, a única diferença está no campo *links*, que representa as duas partes de DC que o mesmo irá conectar. Os passos citados anteriormente são realizados previamente ao escopo do trabalho aqui apresentado, a implementação feita na linguagem de programação Python, para validar os conceitos apresentados, é elucidada a seguir. As SRAs representando as propostas dos provedores exemplificadas anteriormente são validadas verificando-se a sintaxe e a semântica dos valores dos campos apresentados anteriormente na Figura 3.3b e também na Figura 3.4b.

Posteriormente à verificação dos campos do arquivo YAML, são comparados os atributos de desempenho das SRAs (*Resource Options*) ofertadas pelos provedores com base nos requisitos contidos nos PDTs da Figura 3.3a e Figura 3.4a. Estes requisitos são enviados pelo *Slice Broker*, para os *Slice Agents* analisarem se possuem os recursos de infraestrutura necessários para atender a atual requisição de *Slice Part* advinda do locatário. As SRAs que foram aceitas, *i.e* que ofertaram *Slice Parts* condizentes com a requisição são então encaminhados para a função de otimização.

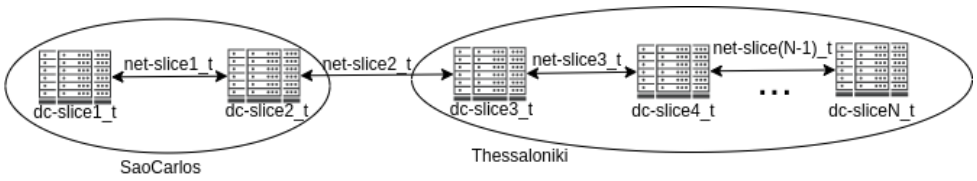
O método de otimização utilizando MILP foi implementado com o auxílio da biblioteca de otimização CPLEX da IBM para otimização. Esta biblioteca implementa várias técnicas de otimização, incluindo o algoritmo Simplex, utilizando MILP e heurística para encontrar o ponto máximo global de uma função dada como entrada. A versão utilizada foi *IBM ILOG CPLEX Optimization Studio v12.8 - Student*, por ser uma ferramenta gratuita para uso acadêmico e encontrada com certa frequência em aplicações similares para otimização de escolha de NFVs (LUIZELLI et al., 2015; LI; Qian, 2015; MAROTTA et al., 2017). A escolha se deve também pela facilidade de integração com a linguagem de programação Python, mas um plano futuro seria o de comparar o desempenho com otimizadores lineares de outras ferramentas, como Matlab[1], R[2], Python SciPy[3], Python PuLP[4], Microsoft Excel Solver[5], entre outros.

Para validar o método proposto utilização a formulação MILP apresentada na Seção 5.1, foram criados quatro cenários diferentes baseados no exemplo das Figuras 5.1 e 5.2. Estes cenários têm como propósito comparar o método MILP proposto com dois outros métodos heurísticos desenvolvidos: **Heurístico 1** seleciona os *DC Slice Parts* físicos ofertados pelos provedores baseando-se no custo final de cada *DC Slice Part*, *i.e.* o fator de risco associado com o preço financeiro da Equação 5.7; o método **Heurístico 2** utiliza apenas o preço financeiro apresentado pelo provedor, em Euros, para a seleção. Todos os dados e resultados dos experimentos apresentados aqui, assim como as análises de consumo de recursos da estação de trabalho utilizada para executar os experimentos estão disponíveis publicamente[6].

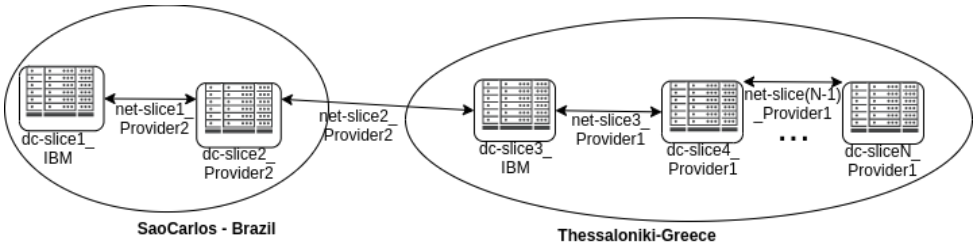
6.1 Cenários para os experimentos

O Cenário 1, representado na Figura 6.1, é composto por uma quantidade variável de *DC Slice Parts* abstratos e apenas uma oferta (*Resource Option*) para cada, *i.e.* uma *Cloud Network Slice* grande com nenhuma competição entre os provedores de infraestrutura. Este cenário é utilizado como uma base de comparação, para verificar o impacto da etapa de *parsing* dos arquivos YAML e do processamento das informações nos três métodos de otimização. Neste cenário, a Figura 6.1a é a topologia da requisição do locatário (PDT), enquanto que a Figura 6.1b contém as *Resource Options* (SRA) dos provedores, onde existem 3, 10, 50, 100, 150 e 200 *DC Slice Parts* em ambas as topologias.

[1] ⟨https://www.mathworks.com/products/optimization.html⟩.
[2] ⟨http://lpsolve.r-forge.r-project.org/⟩.
[3] ⟨https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html⟩.
[4] ⟨https://pythonhosted.org/PuLP/#⟩.
[5] ⟨https://www.solver.com/⟩.
[6] ⟨https://zenodo.org/record/4582772#.YEFK8_tKiV4⟩.



(a) Cenário 1 - topologia abstrata (PDT).



(b) Cenário 1 - topologia física (SRA).

Figura 6.1: Cenário 1 - Múltiplos *DC Slice Parts* e uma oferta dos provedores de infraestrutura.

De forma similar, o Cenário 2, apresentado na Figura 6.2, é composto por uma quantidade fixa de três *DC Slice Parts*, ambos na topologia abstrata e física, mas com uma quantidade variável de ofertas (*Resource Options*). As ofertas são apenas para o segundo *DC Slice Part* abstrato, *i.e.* existem muitos provedores competindo pelo mesmo *DC Slice Part*. Este cenário tem como intuito comparar como cada método de otimização escolhe as ofertas dos provedores. Também tem o intuito de verificar o impacto da otimização na utilização dos recursos da estação de trabalho utilizada para a execução, variando-se o número de *Resource Options* em 3, 10, 50, 100, 150, e 200 *DC Slice Parts*. Para tal cenário, é utilizada a mesma topologia abstrata (PDT) da Figura 5.1, porém com apenas três *DC Slice Parts*, a qual contém a solicitação do locatário.

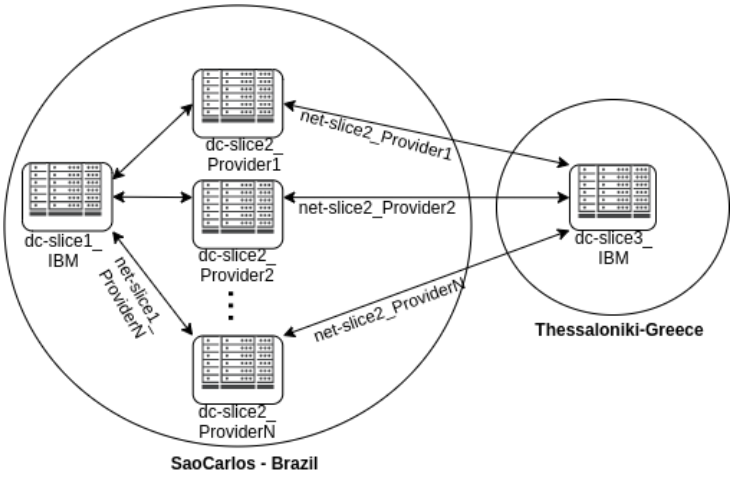


Figura 6.2: Cenário 2 - Três *DC Slice Parts* e múltiplas ofertas dos provedores em um nó.

Para avaliar o impacto de uma abordagem mais realista para o contexto de *Cloud Network*

Slices, o Cenário 3 apresenta um número variável de *DC Slice Parts* na topologia inteira da *slice*, em ambas as topologias abstrata e física. Sendo que as ofertas (*Resource Options*) são feitas apenas para o segundo *DC Slice Part* da *slice*. Todavia, este cenário permite que se observe os efeitos de ambos os Cenários 1 e Cenário 2 juntos. Também é utilizada a mesma topologia abstrata da Figura 6.1a como solicitação do locatário.

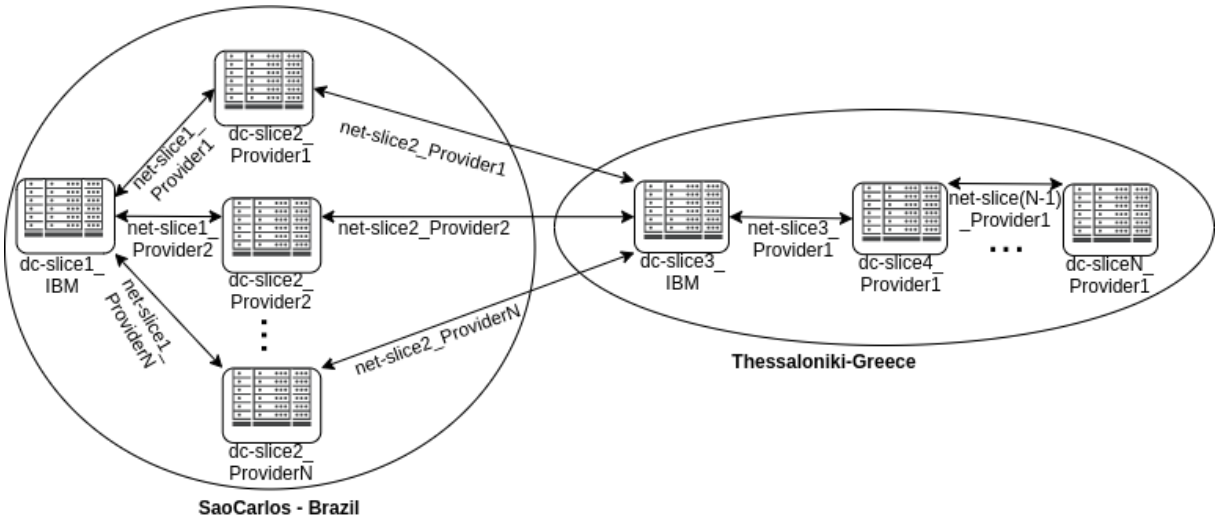


Figura 6.3: Cenário 3 - Múltiplos *DC Slice Parts* e três ofertas de provedores em um único nó.

Por fim, o Cenário 4 da Figura 6.4 tenta ilustrar aspectos de uma situação real, onde provedores ofertam *Resource Options* para diferentes *DC Slice Parts* abstratos, *i.e.* são ofertados para mais de um *DC Slice Part* abstrato. Este cenário possui um número fixo de apenas cinco *DC Slice Parts* abstratos na solicitação do locatário. O foco deste cenário é validar o comportamento da solução com um número exponencial de possibilidades, entre as ofertas de *DC Slice Part* e *Net Slice Part*. Devido à necessidade de gerar arquivos YAML com todos os *Resource Options*, para servir como entrada do programa, com o intuito de simplificar este teste, a quantidade de ofertas varia entre 10 e 25. Tais ofertas são distribuídas randomicamente entre os *DC Slice Parts* e em seguida os *Net Slice Parts* são gerados manualmente para interconectar todos eles na topologia. Como trabalho futuro, é planejado implementar um quinto cenário variando-se também a quantidade de *DC Slice Parts*, ao invés de mantê-los fixos, assim como pretende-se gerar mais *Resource Options* de forma a conseguir grandes quantidades de combinações.

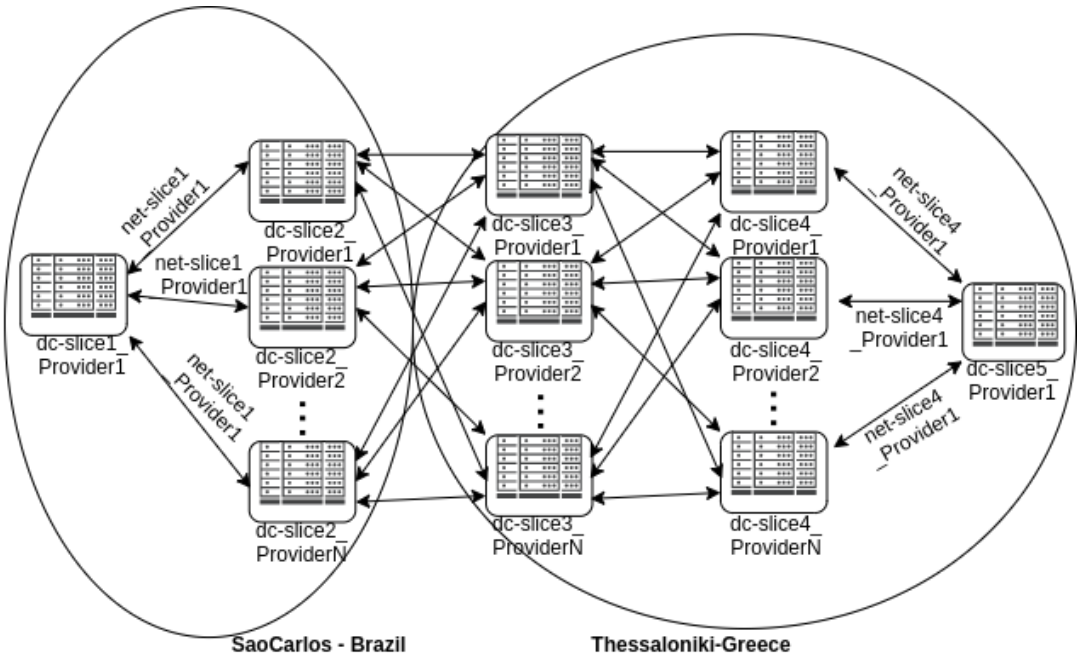


Figura 6.4: Cenário 4 - Cinco *DC Slice Parts* e múltiplas ofertas de provedores em vários nós.

6.2 Experimento 1 - A influência do fator de risco na escolha dos *Slice Parts*

Nesta seção serão apresentados os resultados do Experimento 1, utilizando-se os quatro cenários explicados na seção anterior. O Experimento 1, apresentado na Figura 6.5, foi elaborado com o intuito de entender como o fator de risco (Equação 5.5), baseado na quantidade de recursos de desempenho (CPU, RAM, armazenamento, energia e preço), influencia nos três métodos de otimização comparados (MILP, Heurístico 1, Heurístico 2). Tais recursos estão contidos nos *DC Slice Parts* físicos oferecidos pelos provedores de infraestrutura. Para verificar este comportamento, foi implementado o Cenário 3 com um número fixo de *DC Slice Parts*, ambos físicos e abstratos. Portanto, o tamanho da *Cloud Network Slice* não cresce e o número de *Resource Options* dos provedores é sempre o mesmo.

Este cenário apresentado no parágrafo anterior, foi executado múltiplas vezes (100) para evitar um enviesamento da amostra. Cada execução foi feita com um diferente conjunto de parâmetros, armazenando a quantidade de recursos das *Resource Options* selecionadas. Esta alternativa para evitar o enviesamento permite verificar se alguns recursos ou o preço financeiro influenciam na seleção. Para gerar randomicamente o conjunto de parâmetros para os *DC Slice*

Parts e *Net Slice Parts*, foi utilizado o gerador de números pseudoaleatórios do Linux[7] como entrada, modificando os valores dentro do intervalo especificado na Tabela 6.1.

Quantidade de núcleos de CPU	1 - 100
Tamanho da memória RAM (GB)	4 - 400
Tamanho do armazenamento em disco (GB)	2 - 2000
Consumo energético (kWh)	10 - 100
Preço financeiro (Euro)	1 - 100

Tabela 6.1: Intervalos dos números pseudoaleatórios utilizados.

O conjunto de parâmetros são gerados e especificados no arquivo YAML (SRA) com as ofertas (*Resource Options*) dos provedores em cada execução. É importante mencionar que os preços financeiros, tanto dos *DC Slice Parts* quanto dos *Net Slice Parts* oferecidos pelos provedores, não estão relacionados com a quantidade de recursos que estes possuem, *i.e.* um *Slice Part* com grande capacidade de recursos não significa que o mesmo possui um preço elevado. Este mecanismo simula a competição entre os provedores, que podem fazer promoções e ofertarem preços diferentes em datas específicas, por exemplo.

Os resultados são apresentados na Figura 6.5d, por meio do gráfico tipo *candlestick*, que representa os pontos mínimo e máximo, assim como o quartil inferior e o quartil superior da distribuição de métricas. O gráfico mostra uma leve tendência para a seleção de baixos valores para o consumo energético no método de otimização por MILP. *I.e.* o terceiro quartil do método MILP (69) é menor que dos métodos Heurístico 1 (73) e Heurístico 2 (75). Este resultado pode ser explicado pelo parâmetro de prioridade do consumo energético ser maior (0.4) que dos outros parâmetros de desempenho (0.2), *i.e.* o locatário optou por dar maior prioridade para *DC Slice Parts* com menor consumo energético. Em contrapartida, na Figura 6.5a, Figura 6.5b e Figura 6.5c, o mínimo, máximo e os quartis são praticamente os mesmos, o que representa uma distribuição aleatória na escolha de tais parâmetros. Tal comportamento pode ser explicado pela prioridade especificada pelo locatário nesta requisição ser a mesma para estes parâmetros.

Finalmente, a Figura 6.5e mostra claramente a escolha de baixos preços em todos os três métodos de otimização, evitando ofertas com mais de dez Euros, sendo o mínimo, máximos e os quartis bastante parecidos. Este resultado aponta que o preço financeiro é o parâmetro principal na escolha dos *DC Slice Parts*, *i.e.* o preço influencia mais que os outros parâmetros na escolha, até mesmo com o peso dos parâmetros de desempenho sendo o mesmo. É importante mencionar que em uma situação diferente, como um possível Cenário 4, o fator de risco dos *Net Slice Parts*, que interconectam os *DC Slice Parts*, pode ter um impacto maior na competição.

[7] https://wiki.archlinux.org/index.php/Random_number_generation.

6.3 Experimento 2 - Desempenho do método MILP

Com a ideia de testar o quão bom é o método MILP desenvolvido, comparado com o Heurístico 1 e Heurístico 2, foi implementado o Cenário 4 da Figura 6.4 com uma quantidade fixa de 5 *DC Slice Parts* e 10 ofertas (*Resource Options*) espalhadas pelos diferentes nós. O experimento foi executado 100 vezes para cada método, cada vez com diferentes preços para os *DC Slice Parts* e os *Net Slice Parts*. Estas execuções representam 100 solicitações diferentes de locatários, *e.g.* na primeira execução tem-se, para o segundo *DC Slice Parts* a oferta mais barata, do provedor X, custando quatro Euros; para o terceiro *DC Slice Part*, a mais barata custa dois Euros; para o segundo *Net Slice Part*, que conecta os *DC Slice Parts* mencionados anteriormente, a oferta mais barata custa 5 Euros. Na segunda execução, estes valores mudam aleatoriamente novamente, e assim por diante.

Na Figura 6.6 são apresentados dois gráficos do tipo *candlestick*, sendo um com a distribuição da somatória dos custos finais e outro dos preços, respectivamente, de todos os *Slice Parts* escolhidos por cada método. A Figura 6.6a mostra a somatória dos custos finais dos *Slice Parts* escolhidos. É importante salientar que o método MILP possui um desempenho melhor que os outros, *i.e.* na maioria dos casos o locatário vai pagar o menor preço e terá mais recursos disponíveis para sua *slice*. Da mesma forma, a Figura 6.6b mostra que o locatário vai pagar menos Euros na maioria dos casos, utilizando o método, *i.e.* o locatário vai pagar este valor pela *slice* inteira.

6.4 Experimento 3 - Carga de trabalho na estação de trabalho

Por fim, o Experimento 3 foca na investigação da utilização de recursos computacionais (*workload*), durante a execução dos quatro cenários apresentados, combinando com os três métodos de otimização. O cenário foi executado em uma estação de trabalho (*workstation*), utilizada para os experimentos, podendo ser definido como o *Slice Builder* da Figura 3.2. O computador utilizado para este experimento é uma estação de trabalho dedicada (*bare-metal*) que possui quatro núcleos físicos na CPU, sendo que a mesma possui 8 *threads*, é fabricada com tecnologia de 14 nanômetros (nm) e possui frequência de operação de 2.2 gigahertz (GHz); o computador também possui 64GB de memória RAM; disco rígido (*i.e. Hard Drive - HD*) do tipo SATA, versão 3.1 e que opera a 7200 Rotações Por Minuto (RPM).

Neste experimento, cada cenário e cada método de otimização são executados 30 vezes,

para evitar enviesamento dos dados. Em cada execução, as seguintes métricas de carga de trabalho são medidas para o programa inteiro, assim como para o método de otimização separadamente: porcentagem do uso de CPU, tempo de execução em segundos e o consumo de memória RAM (MB). As métricas referentes ao programa inteiro incluem o *parsing* dos arquivos YAML e o pré-processamento dos metadados. Em contrapartida, as métricas da função de otimização implementada em Python, foram medidas separadamente das etapas anteriores. Os tamanhos dos arquivos YAML da requisição do locatário (PDT) e também das respostas dos provedores de infraestrutura com suas ofertas (SRAs) foram medidos na Figura 6.7.

O gráfico apresentado na Figura 6.8a mostra o cálculo da mediana da porcentagem de uso de CPU durante a execução da função de otimização em Python, nas 30 execuções. Foi utilizada a função *cpu_percent*[8] da biblioteca Python *system and process utilities* (*psutil*), que mede a diferença do tempo de uso da CPU entre o começo e o fim da função de otimização. Ainda na Figura 6.8a, as funções de otimização dos métodos Heurístico 1 e Heurístico 2 utilizaram zero porcento de CPU durante sua execução, nos casos onde há menos de 150 ofertas de *DC Slice Parts* físicos. Possivelmente, a razão para isto ter acontecido é que o tempo de execução da função de otimização, separada do resto do programa em Python, é muito rápido (menos de 0.005 segundos), *i.e.* a execução da função foi muito rápida para ser medido corretamente o uso de CPU. É também possível observar que o uso de CPU da função de otimização no Cenário 1, Cenário 2 e Cenário 3 é sempre menor de 100%, *i.e.* utiliza praticamente um núcleo inteiro da CPU. Em contrapartida, no Cenário 4, utiliza mais de um núcleo nos casos com 10 e 25 ofertas (*Resource Option*). Isto acontece, pois o IBM CPLEX, utilizado para a resolver a otimização com MILP, consegue paralelizar melhor o Cenário 4, pois o número de combinações é maior que nos outros cenários, assim o *solver* da IBM consegue tratar o problema mais eficientemente.

Em suma, a quantidade de recursos consumidos durante a execução dos experimentos não apresentou-se muito alta, para os cenários propostos. Possivelmente isto ocorreu devido à baixa quantidade de combinações gerada nestes cenários. Apesar disto, é previsto que o consumo de recursos cresça bastante, adicionando-se mais ofertas ao Cenário 4 e também em um novo possível quinto cenário, com mais *DC Slice Parts* abstratos, em uma *slice* com comprimento maior. Como demonstrado pelos experimentos aqui apresentados, ao se aumentar a quantidade de *Slice Parts* e de ofertas, aumenta-se também a quantidade de combinações e consequentemente de recursos consumidos.

Após a realização dos experimentos e análise dos resultados, foi observado um problema na Restrição 5.15, da função objetivo do método MILP. Ao se exercitar as possibilidades de

[8] https://psutil.readthedocs.io/en/latest/#psutil.cpu_percent

valores para as variáveis da restrição da Equação 5.15, temos os casos a seguir:

$$(x_i^u = 0) \wedge (x_j^v = 0) \implies (y_{ij}^{uv} = 0) \tag{6.1a}$$

$$(x_i^u = 0) \wedge (x_j^v = 1) \implies (y_{ij}^{uv} = 0) \tag{6.1b}$$

$$(x_i^u = 1) \wedge (x_j^v = 0) \implies (y_{ij}^{uv} = 0) \tag{6.1c}$$

$$(x_i^u = 1) \wedge (x_j^v = 1) \implies (y_{ij}^{uv} = 0) \vee (y_{ij}^{uv} = 1) \tag{6.1d}$$

$$(y_{ij}^{uv} = 1) \implies (x_i^u = 1) \wedge (x_j^v = 1) \tag{6.1e}$$

$$(y_{ij}^{uv} = 0) \implies [(x_i^u = 0) \wedge (x_j^v = 0)] \vee [(x_i^u = 1) \wedge (x_j^v = 1)] \tag{6.1f}$$

1

- Os valores das variáveis da Equação 6.1a e Equação 6.1e funcionam corretamente pela restrição apresentada anteriormente na Equação 5.15;
- Os valores da Equação 6.1b e Equação 6.1c também funcionam corretamente, pois se apenas um *DC Slice Part* é escolhido, o link que interconecta ele não pode ser escolhido, pois ficará com a outra extremidade sem conexão;
- Já os valores da Equação 6.1d são problemáticos, mas funcionam para ambos os valores de y_{ij}^{uv} devido às restrições da Equação 5.16 e Equação 5.18 apresentados anteriormente, que forçam o valor de $y_{ij}^{uv} = 1$, pois os dois *DC Slice Parts* que o *Net Slice Part* interconecta foram escolhidos;
- Por fim, os valores a Equação 6.1f também são problemáticos, pois se um *Net Slice Part* não é escolhido, não possível concluir se os *DC Slice Parts* foram escolhidos ou não.

Para corrigir os problemas citados anteriormente, a nova forma correta para substituir a restrição da função objetivo da Equação 5.15, seria:

$$x_i^u + x_j^v \leq 1 + y_{ij}^{uv} \tag{6.2a}$$

$$y_{ij}^{uv} \leq x_i^u \tag{6.2b}$$

$$y_{ij}^{uv} \leq x_j^v, \forall (i,j) \in E, (u,v) \in E_V \tag{6.2c}$$

Exercitando-se os valores para a nova equação:

$$(x_i^u = 0) \wedge (x_j^v = 0) \implies (y_{ij}^{uv} = 0) \tag{6.3a}$$

$$(x_i^u = 0) \wedge (x_j^v = 1) \implies (y_{ij}^{uv} = 0) \tag{6.3b}$$

$$(x_i^u = 1) \wedge (x_j^v = 0) \implies (y_{ij}^{uv} = 0) \tag{6.3c}$$

$$(x_i^u = 1) \wedge (x_j^v = 1) \implies (y_{ij}^{uv} = 1) \tag{6.3d}$$

$$(y_{ij}^{uv} = 1) \implies (x_i^u = 1) \wedge (x_j^v = 1) \tag{6.3e}$$

$$(y_{ij}^{uv} = 0) \implies (x_i^u = 0) \wedge (x_j^v = 0) \tag{6.3f}$$

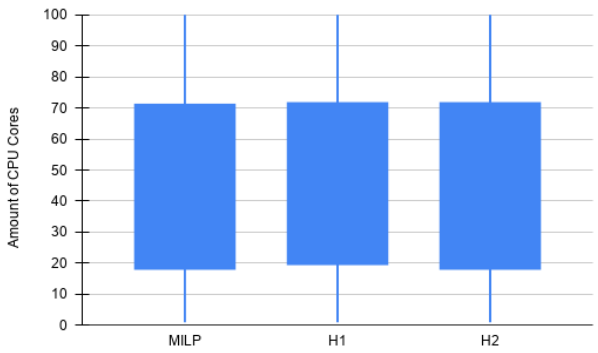
- Os valores da Equação 6.3a e Equação 6.3e funcionam corretamente pela restrição da Equação 6.2b e Equação 6.2c;
- Por fim, os valores da Equação 6.3b, Equação 6.3c, Equação 6.3d, Equação 6.3f também funcionam corretamente pela desigualdade da Equação 6.2a;

Não houve tempo hábil para testar esta modificação no código implementado em Python, porém espera-se que os resultados de consumo de recursos, na estação de trabalho, mudem sutilmente, mas os resultados da comparação da eficiência entre os métodos de otimização continuem o mesmo.

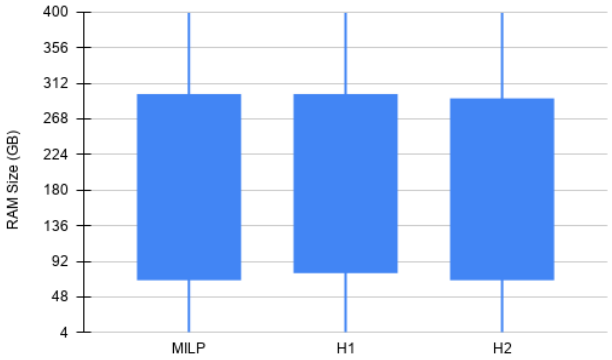
Dois aspectos importantes do provisionamento de uma *Cloud Network Slice* são a dinamicidade e a elasticidade (VERDI, 2019), *i.e.* a habilidade da *slice* ser provisionada e terminada em um período específico de tempo. Não foram exercitados tais aspectos neste trabalho. Porém, de acordo com outros estudos do projeto NECOS, uma *Cloud Network Slice* com dois *DC Slice Parts* e quatro VDUs, *i.e.* 8 *DC Slice Parts* na simplificação que foi assumida aqui, levaria cerca de 2000 segundos para ser provisionada (PINHEIRO; ABELéM, 2019). Este tempo inclui a alocação de máquinas físicas e a instalação do serviço que será utilizado nelas. Um cenário deste tipo pode ser comparado ao Cenário 4 do experimento apresentado aqui, com cinco *DC Slice Parts* na topologia abstrata e 25 ofertas (*Resource Options*) dos provedores de infraestrutura. De acordo com a Figura 6.9b, a mediana do tempo para tal situação seria 6 segundos, para todo o código Python implementado, *i.e.* representa uma pequena fração do tempo de provisionamento (0.3%). Desta forma, o tempo do programa de otimização é negligenciável se comparado com os próximos passos, ao se provisionar a *slice*. A partir de tal análise, em um caso de uso com grande dinamicidade para a *Cloud Network Slice*, onde há provisionamento e terminação das *Slice Parts* de forma frequente, *e.g.* um carro autônomo e interconectado com outros carros, se movendo por uma rodovia, o tempo utilizado para o processo de otimização não será um problema.

6.5 Trabalhos futuros

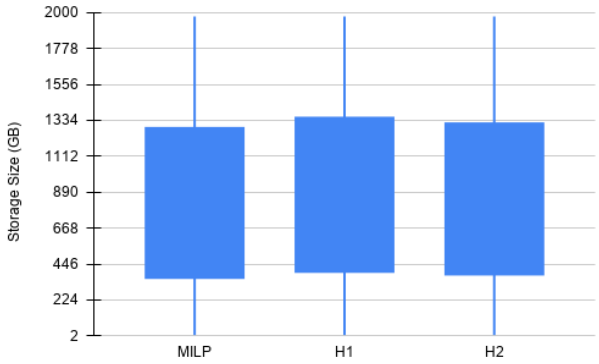
Até o presente momento, a solução apresentada não possui o conceito de localização geográfica dos *Slice Parts*, na formulação do problema, o que pode ser considerado em planos futuros. Ao tratar de georreferência, a função objetivo do método MILP pode selecionar *Slice Parts* próximos ao locatário, de forma a permitir tempos de resposta bastante baixos. Também é planejada a adição de mais ofertas no Cenário 4, onde se espera que o número de combinações cresça exponencialmente e, em seguida, observar o comportamento do método MILP e das heurísticas desenvolvidas. Outro aspecto seria propor um Cenário 5, onde ambas as ofertas dos provedores e o tamanho da topologia irão crescer. Adicionalmente, seria também interessante comparar o método MILP desenvolvido aqui, com outros métodos estado-da-arte, como o NESTOR (DIETRICH; ABUJODA; PAPADIMITRIOU, 2015). Outro experimento interessante que poderia ser realizado, é comparar o tempo do processo de otimização, integrado com a inicialização dos serviços que serão executados dentro da *Cloud Network Slice*, até a fase de utilização do serviços, assim podendo-se analisar o nível de dinamismo suportado pela implementação, *i.e.* quantas vezes os *Slice Parts* podem ser inicializadas e terminadas em certo período de tempo.



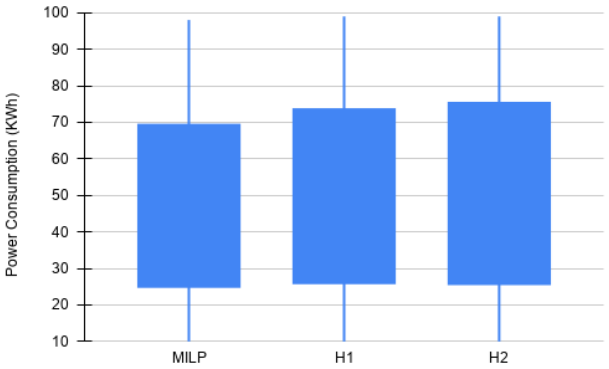
(a) Preferência: quantidade núcleos de CPU.



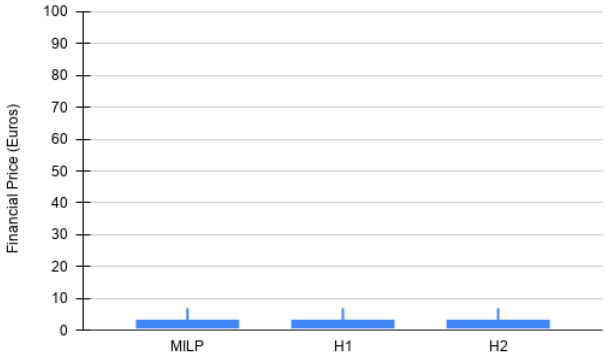
(b) Preferência: memória RAM.



(c) Preferência: armazenamento em disco.

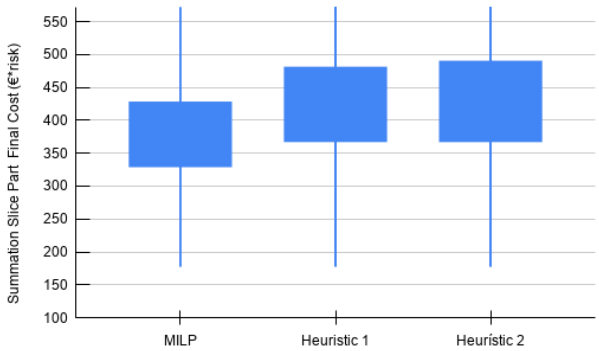


(d) Preferência: consumo energético.

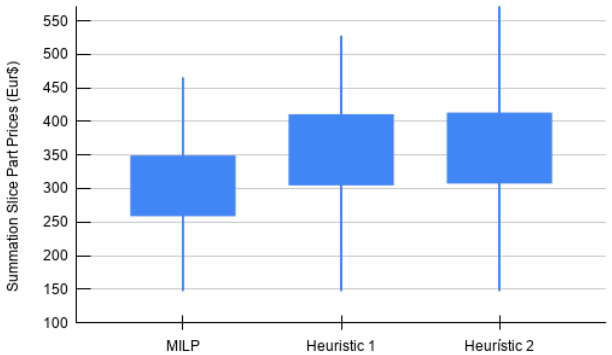


(e) Preferência: preço financeiro.

Figura 6.5: Distribuição da quantidade de recursos escolhidos no Cenário 3.



(a) Somatória do custo final (Euro*risco)



(b) Somatória dos preços (Euro)

Figura 6.6: Preço e custo final totais comparados entre os métodos no Cenário 4.

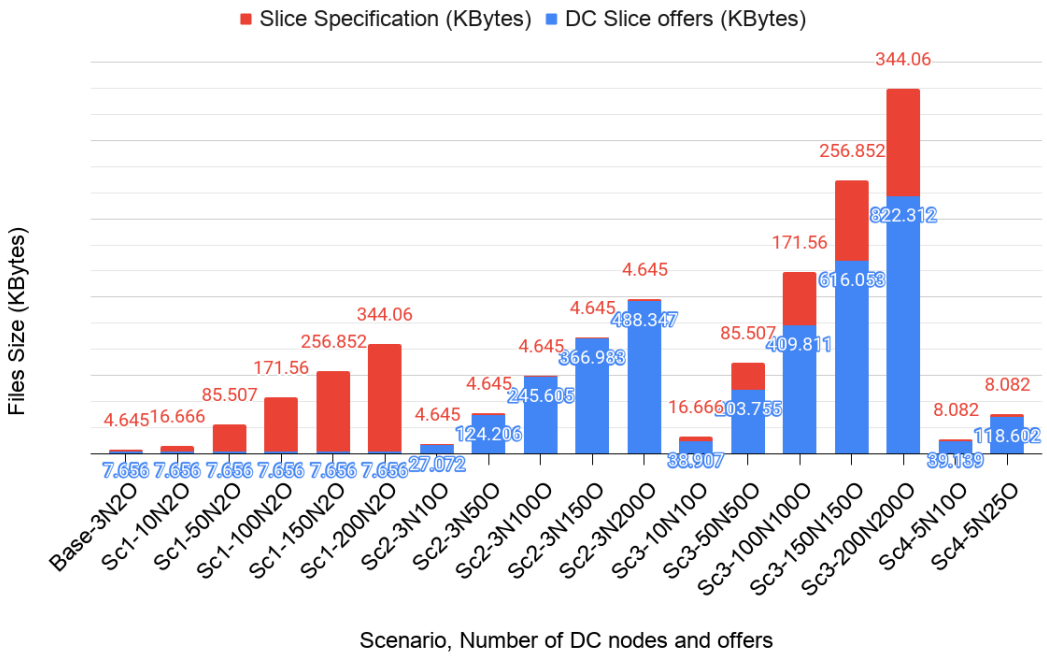
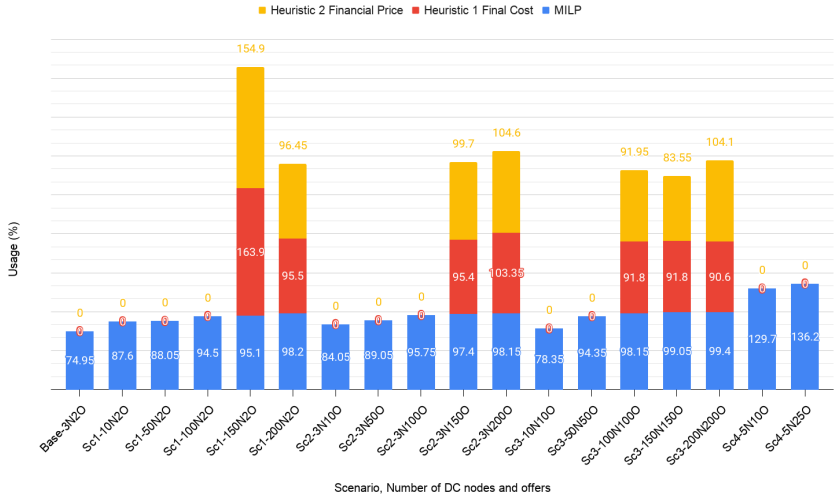
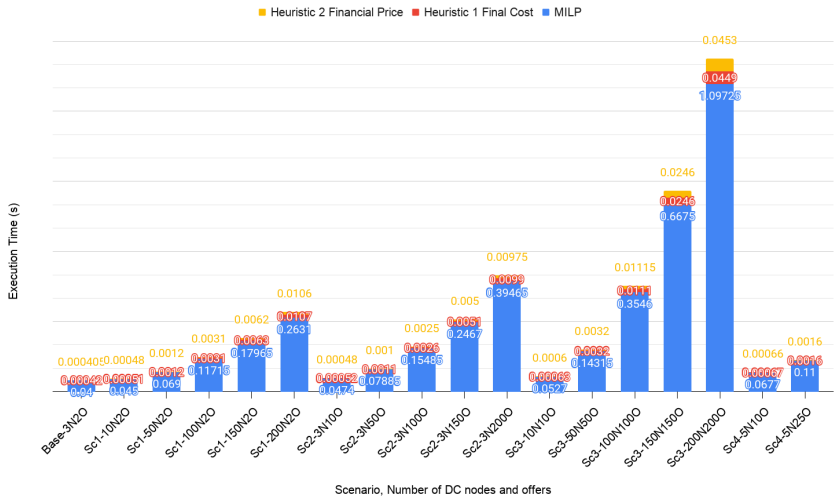


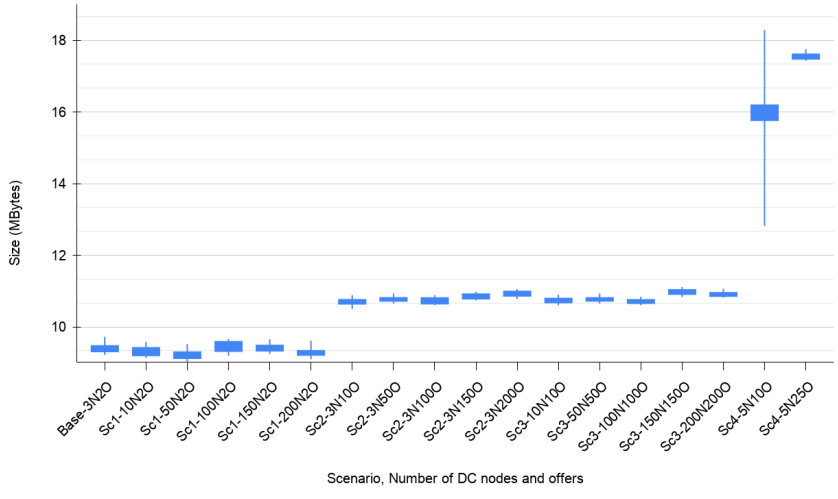
Figura 6.7: Tamanho arquivos YAML (KB) nos Diferentes Cenários.



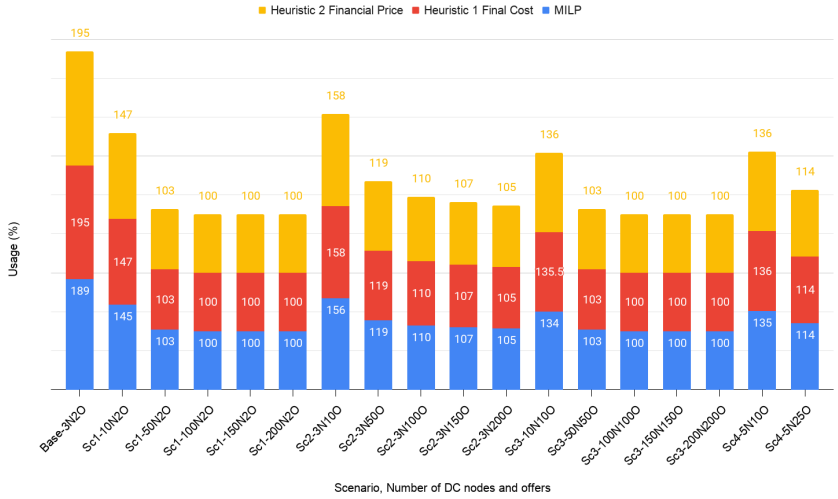
(a) Mediana do uso de CPU (%) - Função de Otimização.



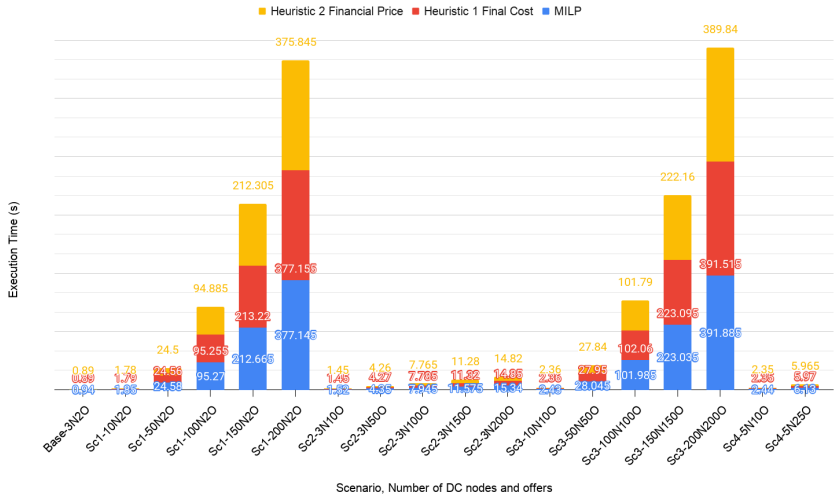
(b) Mediana do tempo de execução (s) - Função de Otimização.



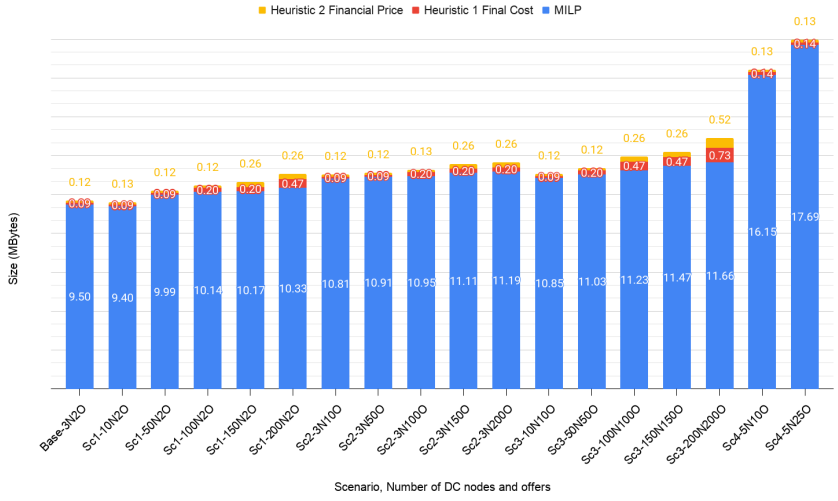
(c) Mediana do consumo RAM (MB) - Função de Otimização MILP.



(a) Mediana do uso de CPU (%) - Programa Inteiro.



(b) Mediana do tempo de execução (s) - Programa Inteiro.



(c) Mediana do uso de RAM (MB) - Programa Inteiro.

Capítulo 7

CONCLUSÕES

A presente dissertação de mestrado foca, principalmente, em preencher uma lacuna no campo de estudo de novos métodos de otimização, para escolha de recursos, no contexto das *Cloud Network Slices*. Foi criado um modelo matemático baseado na proposta de fatiamento, emergente com as redes 5G, apresentada pelo projeto NECOS, porém o modelo pode ser implementado também em outros contextos. Para investigar o desempenho do modelo criado, foram desenvolvidos métodos de MILP, Heurístico 1 e Heurístico 2. Com os experimentos realizados, foi possível mostrar que o método MILP possui uma melhor desempenho na escolha das ofertas com os menores preços financeiros, assim como a escolha de bons recursos de infraestrutura. Apesar do método MILP gerar mais carga de trabalho na máquina utilizada, se comparado com os métodos heurísticos, ainda assim foi mostrado também que esta carga não é tão grande nos cenários com poucas combinações propostos aqui, ao ponto de causar lentidão na estação de trabalho. *I.e.* ainda existe espaço para que a utilização dos recursos cresça durante experimentos.

Também é importante apontar as limitações do trabalho aqui apresentado. Considerando que ainda é necessário escalar o número de combinações, para estressar os métodos de otimização e verificar seu comportamento. Também existe a possibilidade de que o parâmetro de consumo energético não seja divulgado pelos provedores de infraestrutura. Desta forma, ainda é necessária mais investigação sobre a possibilidade de removê-lo da formulação, porém ficou entendido que os objetivos do mestrado foram alcançados.

REFERÊNCIAS

ABUJODA, A.; PAPADIMITRIOU, P. Midas: Middlebox discovery and selection for on-path flow processing. In: *2015 7th International Conference on Communication Systems and Networks (COMSNETS)*. [S.l.: s.n.], 2015. p. 1–8. ISSN 2155-2487.

ABUJODA, A.; PAPADIMITRIOU, P. Distnse: Distributed network service embedding across multiple providers. In: *2016 8th International Conference on Communication Systems and Networks (COMSNETS)*. [S.l.: s.n.], 2016. p. 1–8. ISSN 2155-2509.

ADDIS, B. et al. Virtual network functions placement and routing optimization. In: *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*. [S.l.: s.n.], 2015. p. 171–177.

ALLIANCE, N. *Description of network slicing concept*. 2016. Accessed 2019-10-07. Disponível em: ⟨https://www.ngmn.org/fileadmin/user_upload/161010_NGMN_Network_Slicing_framework_v1.0.8.pdf⟩.

ARAÚJO, S. M. A.; SOUZA, F. S. H. de; MATEUS, G. R. Abordagens descentralizadas para o mapeamento de redes virtuais em ambientes multidomínio. In: *Anais do XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. Porto Alegre, RS, Brasil: SBC, 2018. ISSN 2177-9384. Disponível em: ⟨https://ojs.sbc.org.br/index.php/sbrc/article/view/2441⟩.

BOCKEN, N. et al. A literature and practice review to develop sustainable business model archetypes. *Journal of Cleaner Production*, v. 65, p. 42 – 56, 2014. ISSN 0959-6526.

BONDAN, L. et al. Fende: Marketplace-based distribution, execution, and life cycle management of vnfs. *IEEE Communications Magazine*, v. 57, n. 1, p. 13–19, January 2019. ISSN 1558-1896.

CLAYMAN, S. *D3.1: NECOS System Architecture and Platform Specification.* 2018. Accessed 2020-10-07. Disponível em: ⟨http://www.maps.upc.edu/public/NECOS\\%20D3.1\\%20final.pdf⟩.

DIETRICH, D.; ABUJODA, A.; PAPADIMITRIOU, P. Network service embedding across multiple providers with nestor. In: *2015 IFIP Networking Conference (IFIP Networking)*. [S.l.: s.n.], 2015. p. 1–9.

DIETRICH, D. et al. Multi-provider service chain embedding with nestor. *IEEE Transactions on Network and Service Management*, v. 14, n. 1, p. 91–105, March 2017. ISSN 1932-4537.

DING, Z. et al. A survey on non-orthogonal multiple access for 5g networks: Research challenges and future trends. *IEEE Journal on Selected Areas in Communications*, v. 35, n. 10, p. 2181–2195, Oct 2017. ISSN 0733-8716.

FABBRI, S. et al. Improvements in the start tool to better support the systematic review process. In: *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2016. (EASE '16). ISBN 9781450336918. Disponível em: ⟨https://doi.org/10.1145/2915970.2916013⟩.

GE, C. et al. Toward qoe-assured 4k video-on-demand delivery through mobile edge virtualization with adaptive prefetching. *IEEE Transactions on Multimedia*, v. 19, n. 10, p. 2222–2237, Oct 2017. ISSN 1520-9210.

GUERZONI, R. et al. Analysis of end-to-end multi-domain management and orchestration frameworks for software defined infrastructures: an architectural survey. *Transactions on Emerging Telecommunications Technologies*, v. 28, n. 4, p. e3103, 2017. E3103 ett.3103. Disponível em: ⟨https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3103⟩.

HA, V. N.; LE, L. B. End-to-end network slicing in virtualized ofdma-based cloud radio access networks. *IEEE Access*, v. 5, p. 18675–18691, 2017.

HACHICHA, E.; YONGSIRIWIT, K.; GAALOUL, W. Energy efficient configurable resource allocation in cloud-based business processes (short paper). In: *On the Move to Meaningful Internet Systems: OTM 2016 Conferences*. Cham: Springer International Publishing, 2016. p. 437–444. ISBN 978-3-319-48472-3.

HU, Y. et al. Dynamic vr live streaming over mmt. In: *2017 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. [S.l.: s.n.], 2017. p. 1–4. ISSN 2155-5052.

KENNY, R.; BROUGHTON, T. *Domestic demand for bandwidth: An approach to forecasting requirements for the period 2013-2023*. [S.l.], 2013. Disponível em: ⟨http://www.broadbanduk.org/wp-content/uploads/2013/11/BSG-Domestic-demand-for-bandwidth.pdf⟩.

LI, X.; Qian, C. The virtual network function placement problem. In: *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. [S.l.: s.n.], 2015. p. 69–70.

LIU, H. et al. Mobile edge cloud system: Architectures, challenges, and approaches. *IEEE Systems Journal*, v. 12, n. 3, p. 2495–2508, Sep. 2018. ISSN 1932-8184.

LUIZELLI, M. C. et al. 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)*. [S.l.: s.n.], 2015. p. 98–106. ISSN 1573-0077.

MAROTTA, A. et al. A fast robust optimization-based heuristic for the deployment of green virtual network functions. *Journal of Network and Computer Applications*, v. 95, p. 42 – 53, 2017. ISSN 1084-8045. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S1084804517302461⟩.

ORDONEZ-LUCENA, J. et al. Network slicing for 5g with sdn/nfv: Concepts, architectures, and challenges. *IEEE Communications Magazine*, v. 55, n. 5, p. 80–87, May 2017. ISSN 0163-6804.

PAPADIMITRIOU, P. *D4.1: Provisional API and Information Model Specification.* 2018. Accessed 2020-10-07. Disponível em: ⟨http://www.maps.upc.edu/public/D4.1_final-1.pdf⟩.

PENTELAS, A. et al. Network service embedding with multiple resource dimensions. In: *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*. [S.l.: s.n.], 2020. p. 1–9. ISSN 2374-9709.

PINHEIRO, B.; ABELéM, A. *D6.2: Complete Report on Validation and Demonstration of the Integrated Platform.* 2019. Accessed 2020-10-07. Disponível em: ⟨http://www.maps.upc.edu/public/D6.2\\%20final.pdf⟩.

RAHMAN, M. M.; DESPINS, C.; AFFES, S. Analysis of capex and opex benefits of wireless access virtualization. In: *2013 IEEE International Conference on Communications Workshops (ICC)*. [S.l.: s.n.], 2013. p. 436–440. ISSN 2164-7038.

SOUSA, N. F. S. de et al. Network service orchestration: A survey. *Computer Communications*, v. 142-143, p. 69 – 94, 2019. ISSN 0140-3664. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0140366418309502⟩.

SUN, G. et al. Low-latency orchestration for workflow-oriented service function chain in edge computing. *Future Generation Computer Systems*, v. 85, p. 116 – 128, 2018. ISSN 0167-739X. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0167739X18300153⟩.

VERDI, F. L. *D5.2: Intelligent Management and Orchestration.* 2019. Accessed 2020-10-07. Disponível em: ⟨http://www.maps.upc.edu/public/D5.2\\%20final.pdf⟩.

GLOSSÁRIO

5G – *Redes Móveis de Quinta Geração*

CAPEX – *Capital Expenditure*

CNS – *Cloud Network Slice*

CPU – *Central de Processamento Unitário*

DC – *Datacenters*

DNS – *Domain Name Server*

ETSI – *European Telecommunications Standards Institute*

GB – *gigabyte*

GHz – *gigahertz*

HD – *Hard Drive*

IDS – *Intrusion Detection Systems*

InP – *Infrastructure Provider*

IoT – *Internet of Things*

MB – *megabyte*

MILP – *Mixed Integer Linear Program*

NAT – *Network Address Translation*

NECOS – *Novel Enablers for Cloud Slicing*

NFP – *Network Function Provider*

NFV – *Network Function Virtualization*

NSE – *Network Service Embedding*

OPEX – *Operational Expenditure*

PDT – *Partially Defined Template*

QoE – *Quality of Experience*

RAM – *Random Access Memory*

RPM – *Rotações Por Minuto*

SAS – *Serial Attached SCSI*

SDN – *Software Defined Networks*

SLA – *Service Level Agreement*

SRA – *Partially Defined Template with Resources Alternatives*

SSD – *Solid State Drive*

SaaS – *Slice as a Service*

VDU – *Virtual Deployment Units*

VIM – *Virtual Infrastructure Manager*

VLAN – *Virtual Local Area Network*

VNF – *Virtualized Network Functions*

VPN – *Virtual Private Networks*

VoD – *Video on Demand*

WIM – *Wide-area network Infrastructure Manager*

WiMAX – *Worldwide Interoperability for Microwave Access*

YAML – *YAML Ain't Markup Language*

nm – *nanômetros*

Apendice A

VALIDAÇÃO DA FORMULAÇÃO

A seguir é apresentado um exemplo baseado nas topologias da Figura 5.1 e Figura 5.2 que foi utilizado para exercitar manualmente o comportamento da formulação MILP desenvolvida, verificando possíveis falhas e lacunas a serem preenchidas. Primeiramente são apresentados nas Tabelas A.1 e A.2 os valores para todos as *Resource Options* dos provedores para os *Slice Parts* requeridas pelo locatário.

	DC Slice Part abstract topology			**Resource priority**	**DC Slice Part physical topology**			
Name	dc-slice1	dc-slice2	dc-slice3	N/A	dc-slice1	dc-slice2	dc-slice2	dc-slice3
Location	SaoCarlos	SaoCarlos	Thessaloniki	N/A	SaoCarlos	SaoCarlos	SaoCarlos	Thessaloniki
Price (Euro/day)	<=10	<=10	<=10	N/A	9	7	8	5
Provider		N/A		N/A	IBM	IBM	Amazon	IBM
RAM (Mb)	>=8000	>=4000	>=4000	1/5	8192	10000	4096	4096
CPU	>=3	>=1	>=6	1/5	4	3	2	8
Storage (Mb)	>=5000	>=2000	>=10000	1/5	6000	40000	3000	20000
Power(kWh/day)		N/A		2/5	30	25	20	50

Tabela A.1: Valores para os recursos e preços dos *DC Slice Parts*.

	Net Slice Part abstract topology		**Net Slice Part physical topology**			
Name	net-slice1	net-slice2	net-slice1	net-slice1	net-slice1	net-slice2
Price (Euro/day)	<=45	<=50	45	40	5	40
Provider	N/A		Telefonica	Vivo	IBM	Vodafone
Bandwidth (Mbps)	>=10	>=80	10	10	1000	80
PoP1	dc-slice1	dc-slice2	dc-slice1	dc-slice1	dc-slice1	dc-slice2
PoP2	dc-slice2	dc-slice3	dc-slice2	dc-slice2	dc-slice2	dc-slice3

Tabela A.2: Valores para largura de banda e preços dos *Net Slice Parts*.

Utilizando tais valores apresentados anteriormente, na Tabela A.3 e Tabela A.4 são cal-

culados os parâmetros fator de risco, preço final e variáveis utilizados no método MILP e na Heurística 2.

$$E_V^{-1} = [\text{(dc-slice2\_t,dc-slice1\_t),(dc-slice3\_t,dc-slice2\_t]}$$

$$E_V = [\text{(dc-slice1\_t,dc-slice2\_t),(dc-slice2\_t,dc-slice3\_t)}] \frown [\text{(dc-slice2\_t,dc-slice1\_t),(dc-slice3\_t,dc-slice2\_t)}]$$

$$E^{-1} = [\text{(dc-slice2\_IBM,dc-slice1\_IBM),(dc-slice2\_Amazon,dc-slice1\_IBM),(dc-slice2\_Amazon,dc-slice1\_IBM),(dc-slice3\_IBM,dc-slice2\_IBM),(dc-slice3\_IBM,dc-slice2\_Amazon)}]$$

$$E = [\text{(dc-slice1\_IBM,dc-slice2\_IBM),(dc-slice1\_IBM,dc-slice2\_Amazon),(dc-slice1\_IBM,dc-slice2\_Amazon),(dc-slice2\_IBM,dc-slice3\_IBM),(dc-slice2\_Amazon,dc-slice3\_IBM)}] \frown [\text{(dc-slice2\_IBM,dc-slice1\_IBM),(dc-slice2\_Amazon,dc-slice1\_IBM),(dc-slice2\_Amazon,dc-slice1\_IBM),(dc-slice3\_IBM,dc-slice2\_IBM),(dc-slice3\_IBM,dc-slice2\_Amazon)}]$$

$m_v^{\text{dc-slice1\_t}} = [8192]$	$m_s^{\text{dc-slice1\_t}} = [6000]$	$m_t^{\text{dc-slice1\_t}} = [4]$	$m_e^{\text{dc-slice1\_t}} = [30]$
$m_v^{\text{dc-slice2\_t}} = [10000, 4096]$	$m_s^{\text{dc-slice2\_t}} = [40000, 3000]$	$m_t^{\text{dc-slice2\_t}} = [3, 2]$	$m_e^{\text{dc-slice2\_t}} = [25, 20]$
$m_v^{\text{dc-slice3\_t}} = [4096]$	$m_s^{\text{dc-slice3\_t}} = [20000]$	$m_t^{\text{dc-slice3\_t}} = [8]$	$m_e^{\text{dc-slice3\_t}} = [50]$

$$m'^{\text{dc-slice1\_t}}_{\text{dc-slice1\_IBM}v} = \frac{8192}{8192} = 1 \qquad m'^{\text{dc-slice1\_t}}_{\text{dc-slice1\_IBM}s} = \frac{6000}{6000} = 1$$
$$m'^{\text{dc-slice1\_t}}_{\text{dc-slice1\_IBM}t} = \frac{4}{4} = 1 \qquad m'^{\text{dc-slice1\_t}}_{\text{dc-slice1\_IBM}e} = \frac{30}{30} = 1$$

$$m'^{\text{dc-slice2\_t}}_{\text{dc-slice2\_IBM}v} = \frac{10000-4096}{10000-4096} = 1 \qquad m'^{\text{dc-slice2\_t}}_{\text{dc-slice2\_IBM}s} = \frac{40000-3000}{40000-3000} = 1$$
$$m'^{\text{dc-slice2\_t}}_{\text{dc-slice2\_IBM}t} = \frac{3-2}{3-2} = 1 \qquad m'^{\text{dc-slice2\_t}}_{\text{dc-slice2\_IBM}e} = \frac{25-20}{25-20} = 1$$

$$m'^{\text{dc-slice2\_t}}_{\text{dc-slice2\_Amazon}v} = \frac{4096-4096}{10000-4096} = 0 \qquad m'^{\text{dc-slice2\_t}}_{\text{dc-slice2\_Amazon}s} = \frac{3000-3000}{40000-3000} = 0$$
$$m'^{\text{dc-slice2\_t}}_{\text{dc-slice2\_Amazon}t} = \frac{2-2}{3-2} = 0 \qquad m'^{\text{dc-slice2\_t}}_{\text{dc-slice2\_Amazon}e} = \frac{20-20}{25-20} = 0$$

$$m'^{\text{dc-slice3\_t}}_{\text{dc-slice3\_IBM}v} = \frac{4096}{4096} = 1 \qquad m'^{\text{dc-slice3\_t}}_{\text{dc-slice3\_IBM}s} = \frac{20000}{20000} = 1$$
$$m'^{\text{dc-slice3\_t}}_{\text{dc-slice3\_IBM}t} = \frac{8}{8} = 1 \qquad m'^{\text{dc-slice3\_t}}_{\text{dc-slice3\_IBM}e} = \frac{50}{50} = 1$$

$d_v = 1/5$	$d_s = 1/5$	$d_t = 1/5$	$d_e = 2/5$

$$r_i^u = \sum_{k\in\{v,s,t\}} (d_k(1-m_{ik}^{\prime u})) + d_e m_{ie}^{\prime u}$$

$$r^{\text{dc-slice1\_t}}_{\text{dc-slice1\_IBM}} = \frac{1}{5}(1-1)+\frac{1}{5}(1-1)+\frac{1}{5}(1-1)+\frac{2}{5}1 = \frac{2}{5}$$
$$r^{\text{dc-slice2\_t}}_{\text{dc-slice2\_IBM}} = \frac{1}{5}(1-1)+\frac{1}{5}(1-1)+\frac{1}{5}(1-1)+\frac{2}{5}1 = \frac{2}{5}$$
$$r^{\text{dc-slice2\_t}}_{\text{dc-slice2\_Amazon}} = \frac{1}{5}(1-0)+\frac{1}{5}(1-0)+\frac{1}{5}(1-0)+\frac{2}{5}0 = \frac{3}{5}$$
$$r^{\text{dc-slice3\_t}}_{\text{dc-slice3\_IBM}} = \frac{1}{5}(1-1)+\frac{1}{5}(1-1)+\frac{1}{5}(1-1)+\frac{2}{5}1 = \frac{2}{5}$$

$$c^{\text{dc-slice1\_t}}_{\text{dc-slice1\_IBM}} = 9(1+\frac{2}{5}) = 9\frac{7}{5} = \frac{63}{5} \qquad c^{\text{dc-slice2\_t}}_{\text{dc-slice2\_IBM}} = 7(1+\frac{2}{5}) = 7\frac{7}{5} = \frac{49}{5}$$
$$c^{\text{dc-slice2\_t}}_{\text{dc-slice2\_Amazon}} = 8(1+\frac{3}{5}) = 8\frac{8}{5} = \frac{64}{5} \qquad c^{\text{dc-slice3\_t}}_{\text{dc-slice3\_IBM}} = 5(1+\frac{2}{5}) = 5\frac{7}{5} = \frac{35}{5}$$

$$c'^{\text{(dc-slice1\_t,dc-slice2\_t)}}_{\text{Telefonica(dc-slice1\_IBM,dc-slice2\_Amazon)}} = 45(1+\frac{1}{10}) = 45\frac{11}{10} = \frac{495}{10}$$
$$c'^{\text{(dc-slice1\_t,dc-slice2\_t)}}_{\text{Vivo(dc-slice1\_IBM,dc-slice2\_Amazon)}} = 40(1+\frac{1}{10}) = 40\frac{11}{10} = \frac{445}{10}$$
$$c'^{\text{(dc-slice1\_t,dc-slice2\_t)}}_{\text{(dc-slice1\_IBM,dc-slice2\_IBM)}} = 5(1+\frac{1}{1000}) = 5\frac{1001}{1000} = \frac{5005}{1000}$$

Tabela A.3: Demonstração dos cálculos de parâmetros e variáveis - Parte 1.

Ao final dos cálculos da Tabela A.3 as letras que multiplicam cada uma das frações representam variáveis que devem ser calculadas pelo método de otimização, *i.e.* existe mais de uma oferta (*Resource Option*) para determinado *Slice Part*. Também é possível interpretar que as frações que não possuem tais letras são as que também não possuem *Resource Options*.

$$c^{(\text{dc-slice2\_t,dc-slice3\_t})}_{(\text{dc-slice2\_IBM,dc-slice3\_IBM})} = 40(1+\frac{1}{80}) = 40\frac{81}{80} = \frac{3240}{80}$$

$$c^{(\text{dc-slice2\_t,dc-slice3\_t})}_{(\text{dc-slice2\_Amazon,dc-slice3\_IBM})} = 40(1+\frac{1}{80}) = \frac{3240}{80}$$

$c^{(\text{dc-slice1\_t,dc-slice2\_t})}_{(\text{dc-slice1\_IBM,dc-slice2\_Amazon})} = min[\frac{495}{10}, \frac{445}{10}] = \frac{445}{10}$ $\quad$ $c^{(\text{dc-slice1\_t,dc-slice2\_t})}_{(\text{dc-slice1\_IBM,dc-slice2\_IBM})} = \frac{5005}{1000}$

$c^{(\text{dc-slice2\_t,dc-slice3\_t})}_{(\text{dc-slice2\_IBM,dc-slice3\_IBM})} = \frac{3240}{80}$ $\quad$ $c^{(\text{dc-slice2\_t,dc-slice3\_t})}_{(\text{dc-slice2\_Amazon,dc-slice3\_IBM})} = \frac{3240}{80}$

Minimize:

$$x^{\text{dc-slice1\_t}}_{\text{dc-slice1\_IBM}} c^{\text{dc-slice1\_t}}_{\text{dc-slice1\_IBM}} + x^{\text{dc-slice2\_t}}_{\text{dc-slice2\_IBM}} c^{\text{dc-slice2\_t}}_{\text{dc-slice2\_IBM}} +$$
$$x^{\text{dc-slice2\_t}}_{\text{dc-slice2\_Amazon}} c^{\text{dc-slice2\_t}}_{\text{dc-slice2\_Amazon}} + x^{\text{dc-slice3\_t}}_{\text{dc-slice3\_IBM}} c^{\text{dc-slice3\_t}}_{\text{dc-slice3\_IBM}} +$$
$$y^{(\text{dc-slice1\_t,dc-slice2\_t})}_{(\text{dc-slice1\_IBM,dc-slice2\_Amazon})} c^{(\text{dc-slice1\_t,dc-slice2\_t})}_{(\text{dc-slice1\_IBM,dc-slice2\_Amazon})} +$$
$$y^{(\text{dc-slice1\_t,dc-slice2\_t})}_{(\text{dc-slice1\_IBM,dc-slice2\_IBM})} c^{(\text{dc-slice1\_t,dc-slice2\_t})}_{(\text{dc-slice1\_IBM,dc-slice2\_IBM})} +$$
$$y^{(\text{dc-slice2\_t,dc-slice3\_t})}_{(\text{dc-slice2\_IBM,dc-slice3\_IBM})} c^{(\text{dc-slice2\_t,dc-slice3\_t})}_{(\text{dc-slice2\_IBM,dc-slice3\_IBM})} +$$
$$y^{(\text{dc-slice2\_t,dc-slice3\_t})}_{(\text{dc-slice2\_Amazon,dc-slice3\_IBM})} c^{(\text{dc-slice2\_t,dc-slice3\_t})}_{(\text{dc-slice2\_Amazon,dc-slice3\_IBM})} =$$

$$1 * \frac{63}{5} + x^{\text{dc-slice2\_t}}_{\text{dc-slice2\_IBM}} * \frac{49}{5} + x^{\text{dc-slice2\_t}}_{\text{dc-slice2\_Amazon}} * \frac{64}{5} + 1 * \frac{35}{5} +$$
$$y^{(\text{dc-slice1\_t,dc-slice2\_t})}_{(\text{dc-slice1\_IBM,dc-slice2\_Amazon})} \frac{445}{10} +$$
$$y^{(\text{dc-slice1\_t,dc-slice2\_t})}_{(\text{dc-slice1\_IBM,dc-slice2\_IBM})} \frac{5005}{1000} + y^{(\text{dc-slice2\_t,dc-slice3\_t})}_{(\text{dc-slice2\_IBM,dc-slice3\_IBM})} \frac{3240}{80} +$$
$$y^{(\text{dc-slice2\_t,dc-slice3\_t})}_{(\text{dc-slice2\_Amazon,dc-slice3\_IBM})} \frac{3240}{80} =$$

$$\frac{63}{5} + a * \frac{49}{5} + b * \frac{64}{5} + \frac{35}{5} + c * \frac{445}{10} + d * \frac{5005}{1000} + e * \frac{3240}{80} + f * \frac{3240}{80} =$$

Tabela A.4: Demonstração dos cálculos de parâmetros e variáveis - Parte 2.

A Tabela A.5 apresenta as diferentes possibilidades para os valores das variáveis da função objetivo, sendo que é assumido o valor binário para a escolha ou não de determinada oferta.

DC2		Net1		Net2		
a	**b**	**c**	**d**	**e**	**f**	**TotalCost**
1	0	1	0	1	0	114,4
1	0	1	0	0	1	114,4
0	1	1	0	1	0	117,4
0	1	1	0	0	1	117,4
1	0	0	1	1	0	74,905
1	0	0	1	0	1	74,905
0	1	0	1	1	0	77,905
0	1	0	1	0	1	77,905

Tabela A.5: Diferentes possibilidades para variáveis calculadas pela função objetivo.