Lucio Agostinho Rocha

Abordagem Híbrida para Alocação de Máquinas Virtuais em Nuvens Computacionais

Campinas
2013

Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação

Lucio Agostinho Rocha

Abordagem Híbrida para Alocação de Máquinas Virtuais em Nuvens Computacionais

Tese de doutorado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos exigidos para a obtenção do título de Doutor em Engenharia Elétrica. Área de concentração: Engenharia de Computação.

Orientador: Prof. Dr. Eleri Cardozo

Este exemplar corresponde à versão final da tese defendida por Lucio Agostinho Rocha, e orientada pelo Prof. Dr. Eleri Cardozo

---

Campinas
2013

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

Rocha, Lucio Agostinho, 1982-
R582a Abordagem híbrida para alocação de máquinas virtuais em nuvens computacionais / Lucio Agostinho Rocha. – Campinas, SP : [s.n.], 2013.

Orientador: Eleri Cardozo.
Tese (doutorado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Computação em nuvem. 2. Redes de computadores. 3. Algoritmos genéticos. 4. Pesquisa operacional. I. Cardozo, Eleri,1954-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Hybrid approach for virtual machines allocation in computational clouds
Palavras-chave em inglês:
Cloud computing
Computer networks
Genetic algorithms
Operational research
Área de concentração: Engenharia de Computação
Titulação: Doutor em Engenharia Elétrica
Banca examinadora:
Eleri Cardozo [Orientador]
Romis Ribeiro de Faissol Attux
Maurício Ferreira Magalhães
Bruno Richard Schulze
Cesar Augusto Cavalheiro Marcondes
Data de defesa: 18-10-2013
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE DOUTORADO

Candidato: Lucio Agostinho Rocha

Data da Defesa: 18 de outubro de 2013

Título da Tese: "Abordagem Híbrida para Alocação de Máquinas Virtuais em Nuvens Computacionais"

Prof. Dr. Eleri Cardozo (Presidente): _____

Prof. Dr. Bruno Richard Schulze: _____

Prof. Dr. Cesar Augusto Cavalheiro Marcondes: _____

Prof. Dr. Romis Ribeiro de Faissol Attux: _____

Prof. Dr. Maurício Ferreira Magalhães: _____

Resumo

A Computação em Nuvem é um modelo para a oferta de serviços sob demanda na Internet. Neste modelo, um provedor de serviço de nuvem oferece serviços de processamento e armazenamento de informação por meio da virtualização de uma infraestrutura computacional, composta de servidores, equipamentos de comunicação, sistemas de armazenamento de dados, aplicativos, dentre outros. Na Computação em Nuvem as máquinas virtuais (VMs) são entidades que permitem o compartilhamento seguro de recursos na nuvem. Visando obter economia de escala, os provedores de serviço operam dezenas de *data centers* cada qual abrigando milhares de servidores e interconectados por redes de comunicação de alta capacidade. O consumo de energia nesta ampla infraestrutura física é responsável por uma parcela cada vez mais significativa dos custos operacionais das nuvens computacionais, além de contribuirem com a emissão de gases responsáveis pelo efeito estufa. Neste sentido, o processamento de informação em larga escala com baixas emissões de carbono, a chamada Computação Verde, assume um papel importante na Computação em Nuvem. O interesse pela Computação Verde tem motivado o surgimento de várias estratégias de alocação de recursos computacionais (VMs) em *data centers* com o objetivo de otimizar o consumo de energia na nuvem. Esta tese propõe uma estratégia capaz de otimizar o consumo de energia na infraestrutura de nuvem mantendo níveis de qualidade de serviço (QoS) na rede de comunicação dentro de limites definidos pelo provedor de serviço. Essa estratégia de otimização híbrida combina algoritmos genéticos, programação linear inteira mista e simulação de redes.

Palavras-chave: Computação em Nuvem. Computação Bio-inspirada. Alocação de Recursos em Nuvem.

Abstract

Cloud Computing is a model for the offering of on demand services through the Internet. In this model, a cloud service provider offers processing and data storage services by virtualizing a computing infrastructure, composed of servers, communication equipments, storage systems, and applications, among others. In Cloud Computing virtual machines (VMs) are the entities that allow the secure sharing of resources in the cloud. In order to achieve economies of scale, service providers operate tens of data centers each one holding thousands of servers and interconnected by a high capacity communication network. The energy consumption in this wide physical infrastructure responds by an increasing and significant portion of the operational costs of the cloud, and contributes to the emission of greenhouse gases. In this scenario, the large scale information processing with low emissions of carbon dioxide, the Green Computing, is becoming a major concern in Cloud Computing. The interest in Green Computing has motivated the emergence of strategies for allocating computing resources (VMs) in data centers with the objective of optimizing the energy consumption in the cloud. This thesis proposes a strategy able to optimize the energy consumption in cloud infrastructure keeping levels of quality of service (QoS) in the network communication within thresholds defined by service provider. This strategy of hybrid optimization combines genetic algorithms, mixed integer linear programming, and network simulations.

Keywords: Cloud Computing. Bio-inspired Computing. Resource Allocation in Cloud.

Sumário

Agradecimentos

Agradeço primeiramente a Deus, que é o princípio, o meio e o fim de todas as coisas.

Aos meus pais, Jorge e Maria Clarete, pelas orientações, alegrias, apoio e compreensão nesse meu caminho dedicado aos estudos. Aos meus irmãos, Fernanda e Wagner, pelo companheirismo e alegria de sempre. Agradeço aos meus familiares pelo incentivo, e pelas orientações do vô Anselmo (*in memorian*).

Agradeço ao meu orientador, Prof. Dr. Eleri Cardozo e à Prof. Dr. Eliane G. Guimarães, pela amizade verdadeira e orientações que tanto auxiliaram os meus estudos.

Agradeço ao Prof. Dr. Luis Fernando Faina, da Universidade Federal de Uberlândia, pelo incentivo para a conclusão dessa pesquisa.

Ao Prof. Dr. Rodrigo Calheiros, da Universidade de Melbourne, Austrália, pelo incentivo à pesquisa.

Agradeço também aos verdadeiros amigos do Laboratório de Computação e Automação (LCA) que contribuíram direta e/ou indiretamente para a conclusão desse trabalho, e que merecem menção: Ricardo Souza, Fernando Paolieri, Leonardo Olivi, Diego Rodrigues, Fábio Teixeira, Fernando Pinho, Yi Ling Liu, Paulo Gurgel, Rossano Pablo, Luisa Fernanda, Eric Rohmer, Guilherme Feliciano. Amigos sem os quais esse trabalho não poderia ter sido concluído.

À CAPES, pelo apoio financeiro.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

*aaS Everything as a Service

AG Algoritmo Genético

AMI Amazon Machine Images

API Application Programming Interface

BGP Border Gateway Protocol

BSP Backward Speculative Placement

CBR Constant Bit Rate

CHP Combined Heat and Power

Conama ConamaConselho Nacional do Meio Ambiente

CRAC Computer Room Air Conditioning

CSA Cloud Security Aliance

DCIM Data Center Infrastructure Management

DMTF Distributed Management Task Force

EC2 Elastic Compute Cloud

FPGA Field Programmable Gate Array

IaaS Infrastructure as a Service

ISP Internet Service Provider

JSON JavaScript Object Notation

KVM Kernel-based Virtual Machine

LSP Label Switch Path

MAPI Management API

MIB Management Information Base

MILP Mixed Integer Linear Programming

MPLS MultiProtocol Label Switching

MTU Maximum Transmition Unit

NAM Network Animator

NF Next Fit

NFS Network File System

NIST National Institute of Standards and Technology

OCC Open Cloud Consortium

OGF Open Grid Forum

OMG Object Management Group

OSPF Open Shortest Path First

PaaS Platform as a Service

PDU Power Distribution Unit

PL Programação Linear

PUE Power Usage Effectiveness

RaaS Robotic as a Service

RAM Random Access Memory

RIP Routing Information Protocol

RISC Reduced Instruction Set Computer

SaaS Software as a Service

SLA Service Level Agreement

SNIA Storage Networking Industry Association

SNMP Simple Network Management Protocol

TCO Total Cost Ownership

UDP User Datagram Protocol

UPS Uninterruptible Power Supply

VLAN Virtual Local Area Network

VM Virtual Machine

WSM Weighted Sum Method

Capítulo 1

Introdução

Este capítulo descreve inicialmente a Computação em Nuvem, o tema central desta tese. O capítulo contempla também as motivações, a proposta e as contribuições desta tese, bem como a forma como o texto está organizado.

1.1 Visão Geral da Computação em Nuvem

O conceito da Computação em Nuvem remonta a meados da década de 60 com a afirmação do cientista da computação John McCarthy, reconhecido por seus estudos em inteligência artificial e criador da linguagem de programação Lisp (FRANGULYAN, 2009):

> *"Computação pode algum dia ser organizada como uma utilidade pública."* (Tradução do autor)

No entanto, a Computação em Nuvem começou a se destacar em meados de 2007 como um modelo empresarial, apesar de sua base teórica já ser estabelecida em modelos de Computação em Grades (WLADAWSKY-BERGER, 2003).

Atualmente é comum que muitas aplicações sejam acessadas pela Internet e mantidas em *data centers*. Essas aplicações são oferecidas de forma transparente para seus usuários, ou seja, com pouca (ou nenhuma) referência à infraestrutura física que as mantêm. Por conta dessa abstração é comum empregar o termo "nuvem" para indicar a abstração da conectividade que existe entre os *endpoints*.

A Fig. 1.1 apresenta uma visão geral dos principais componentes encontrados em ambientes de nuvem. A descrição desses componentes é detalhada referência (AHRONOVITZ et al., 2010). Nessa figura destacamos os relacionamentos entre os componentes Provedor de Serviços, Consumidor de Serviços e Desenvolvedor de Serviços, os quais são interligados por tecnologias de comunicação baseadas em padrões abertos, e descritos como segue:

- Provedor de Serviços: é observado um nível crescente de abstração em cada uma das camadas de serviços. Por exemplo, infraestruturas de nuvem são altamente dependentes do tipo de *hardware* escolhido, além do *Hypervisor*. O suporte à virtualização é oferecido pelo componente *Hypervisor* sobre o Sistema Operacional ou diretamente sobre o *hardware*. A virtualização de recursos de *hardware* e *software* oferece a base para que um amplo conjunto de Aplicações de Nuvem sejam oferecidas de várias formas diferentes por meio de serviços. O componente de Gerência deve incluir funcionalidades para contabilidade de

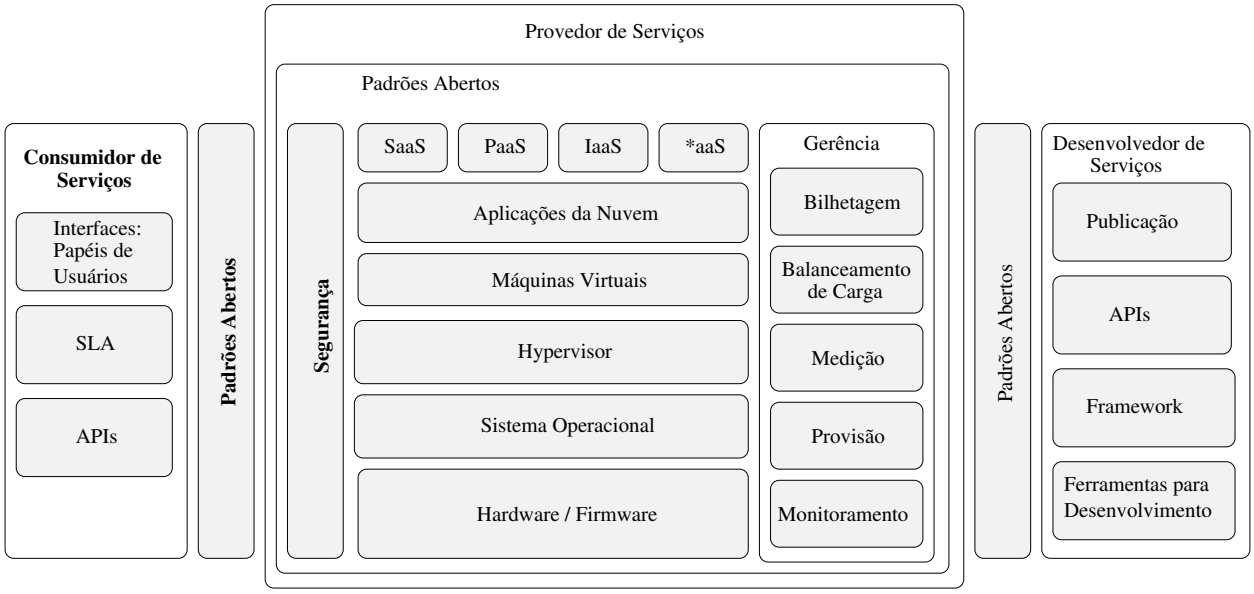

Figura 1.1: Visão Geral dos Principais Componentes para Ambientes em Nuvem.

uso, balanceamento de carga, medição de tráfego e uso de recursos, provisionamento de máquinas virtuais, e demais funções para monitoramento. O componente Segurança deve incluir funcionalidades tanto para a segurança do tráfego de dados quanto para a segurança relacionada ao acesso ao ambiente. Tanto os componentes para gerência quanto os componentes de segurança devem abordar desde as camadas inferiores do *hardware/firmware* até as camadas superiores no nível das aplicações e infraestrutura de suporte;

- Consumidor de Serviços: tem acesso aos recursos por meio de interfaces baseadas em papéis de usuários no ambiente e regras de acesso compatíveis com os mecanismos de segurança da nuvem. Além disso, SLAs *(Service Level Agreement)* são necessários para garantir a qualidade da oferta de serviços e disponibilidade de acesso. Além disso, são utilizadas APIs (*Application Programming Interface*) para estabelecer a comunicação com os serviços oferecidos;

- Desenvolvedor de Serviços: utiliza uma gama de serviços para desenvolver novas aplicações para a nuvem. Esses serviços utilizam padrões abertos para se comunicar e estender as funcionalidades dos serviços já existentes. O desenvolvimento de novos serviços envolve a publicação deles na Web, o uso de APIs, *frameworks* próprios para o desenvolvimento, e demais ferramentas de suporte.

Mais especificamente, o termo "nuvem" foi emprestado da telefonia (FRANGULYAN, 2009), mas também é comum encontrar esse termo em referências de redes quando se deseja abstrair a ampla gama de recursos entre os *endpoints*. São utilizados serviços de forma transparente, sem a necessidade de se conhecer o local onde os serviços são mantidos fisicamente.

Existe uma grande diversidade de aplicações que podem executar nessa nuvem, cada uma delas com os mais diferentes requisitos. Em função disso, existe o desafio por parte dos provedores de serviços em atender aos requisitos desse conjunto diverso de aplicações. Nesse modelo, fatias do poder computacional dos nós da rede são oferecidos, reduzindo os custos para fornecer uma infraestrutura interna para prover os serviços. Esse modelo também baseia-se na alocação

de recursos apenas pelo período de utilização dos serviços contratados. Isso permite que as empresas façam cotações dinâmicas sobre os períodos de melhor custo × benefício para uso de tais serviços. Como exemplo, pode ser economicamente mais vantajosa a transferência de uma grande quantidade dados quando menos usuários utilizarem a rede do provedor de serviços.

A Fig. 1.2 é baseada no artigo de Pinal (PINAL, 2010), e mostra que o surgimento da Computação em Nuvem foi decorrente de diversos outros modelos de computação distribuída. Na base dessa figura está a Computação em Grades, que herdou características da Computação em Clusters e da Computação de Alto Desempenho (ou Super Computação). As Grades forneceram a base para a descrição de um modelo genérico de Computação Utilitária, onde os usuários utilizam os recursos e serviços da rede, sem se preocuparem com a forma como são disponibilizados ou como são oferecidos remotamente. A Computação em Nuvem expandiu as possibilidades de oferta de serviços, seja de plataforma (PaaS - *Platform as a Service*), de *software* (SaaS - *Software as a Service*), de infraestrutura (IaaS - *Infrastructure as a Service*), de robótica (RaaS - *Robotic as a Service*), e muitos outros genericamente denominados *aaS (*Everything as a Service*).

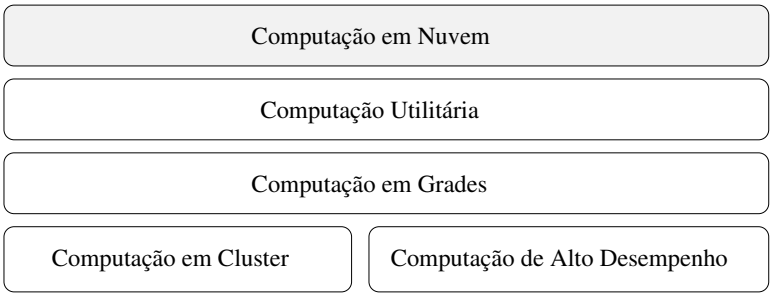

Figura 1.2: Tecnologias que Contribuíram para o Surgimento da Computação em Nuvem.

1.2 Motivações

O avanço das tecnologias de informação e comunicação (TICs) tem sido acompanhado pelo crescimento vertiginoso das aplicações voltadas para a Web, gerando novas necessidades para a indústria de computação. Uma questão que tem sido apontada como primordial é o uso sustentável das infraestruturas de suporte a aplicações para a Web.

A motivação para essa tese parte do princípio de buscar o uso sustentável de recursos computacionais. A natureza dinâmica de uma nuvem exige a otimização dos processos para a alocação de recursos com o objetivo de reduzir o consumo de energia e realizar o balanceamento de carga em *data centers*. A alocação de recursos na granularidade de máquinas virtuais (VMs - *Virtual Machines*) deve considerar não apenas os seus requisitos convencionais de processamento, como CPU, memória RAM (*Random Access Memory*) e espaço de armazenamento em disco, mas também a interconexão de rede entre os servidores da nuvem.

A alocação de VMs pode ser incluída na categoria de problemas nos quais métodos bio-inspirados são factíveis. Métodos bio-inspirados são úteis em muitos cenários, principalmente quando métodos clássicos não são aplicáveis, quando o tempo computacional para se alcançar uma solução for alto, ou quando um problema lida com dados incertos ou imprecisos. Especificamente em relação à alocação de VMs, a razão é a natureza discreta desse problema combinada com o fato de que as restrições de rede são difíceis de incorporar em métodos formais de alocação.

Nesse contexto, propõe-se a utilização de uma abordagem híbrida que combine programação matemática, simulação e AGs (Algoritmos Genéticos) para a alocação de VMs em nuvens, levando-se em consideração métricas de rede. A intenção é contribuir para reduzir o consumo de energia quando da execução de múltiplas instâncias de usuários concorrentes.

Essa motivação é baseada em pesquisa bibliográfica para avaliar as novas tendências de aplicações distribuídas em ambientes sustentáveis. Estudos recentes (ARMBRUST et al., 2009) na área de computação distribuída apontam tendências para desvincular a infraestrutura de *hardware* da infraestrutura de desenvolvimento de *software*, ou seja, o usuário de aplicações distribuídas utiliza os serviços de ambientes pré-configurados. Essa declaração está de acordo com as expectativas do mercado quanto ao aumento da produção, oferta e redução dos custos para a manutenção de uma complexa infraestrutura de *software* e *hardware* (CANONICAL, 2012). A evolução da computação distribuída tende a buscar mecanismos de computação interconectados, independentes da infraestrutura física na qual executam, com redução da complexidade de gerência e desenvolvimento.

Segundo a OMG (*Object Management Group*) (CLOUD-STANDARDS.ORG., 2010) e parceiros como a DMTF (*Distributed Management Task Force*), OGF (*Open Grid Forum*), SNIA (*Storage Networking Industry Association*), OCC (*Open Cloud Consortium*) e CSA (*Cloud Security Aliance*), é esperado um crescimento empresarial na utilização de Computação em Nuvem para os próximos anos. A Dra. Kate Keahey, pesquisadora que frequentemente escreve sobre Computação em Nuvem e em Grades, afirmou em 2010 que (SCHIFF, 2010):

> *"(...) Grades irão ser reorganizadas ou agregadas na Computação em Nuvem... Eu penso que em 5 anos algo como 80 a 90% da computação poderia ser baseada em nuvem."* (Tradução do autor)

A Fig. 1.3 mostra a previsão de crescimento dos mercados de Computação em Nuvem. De acordo com a pesquisa reportada por Ried et al. (RIED et al., 2010), o SaaS representará um mercado que crescerá para $92.8 bilhões em 2016, acompanhado pelo menor crescimento dos demais tipos de serviços.

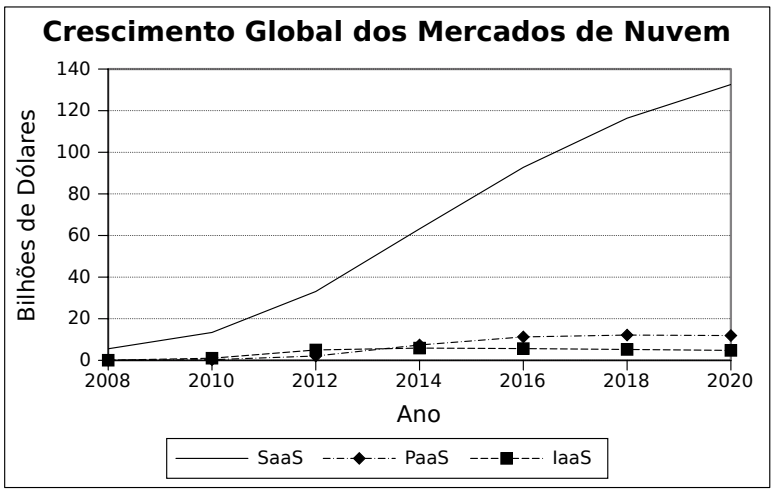

Figura 1.3: Crescimento dos Mercados de Computação em Nuvem (RIED et al., 2010).

Diversos autores (MARTINS, 2010) afirmam que, em um futuro próximo, a tecnologia de Grades reunirá recursos e pessoas de diversas localidades, com atividades distintas, em uma

infraestrutura de inter-relacionamentos comuns. De acordo com Schiff (SCHIFF, 2010), infraestruturas de Computação em Grades falham em prover facilidade no desenvolvimento de serviços distribuídos, mas são soluções bem consolidadas para a integração confiável e segura de aplicações distribuídas. Isso contribui para que aplicações em Grade sejam voltadas mais para a pesquisa acadêmica e demais instituições de pesquisa, ainda que o seu uso empresarial seja possível (WLADAWSKY-BERGER, 2003). Por outro lado, a Computação em Nuvem é apoiada fortemente na virtualização como tecnologia chave para o suporte a uma grande quantidade de sistemas operacionais e aplicações oferecidas como serviços. No entanto, as aplicações na nuvem são confinadas ao escopo do ambiente virtualizado, sendo necessário o desenvolvimento de novos serviços de rede para a integração segura e confiável dos serviços oferecidos nesses ambientes.

1.3 Proposta

A Computação em Nuvem traz uma mudança de paradigma do modelo de computação tradicional para um modelo voltado para o consumo de serviços. Na literatura, alguns autores (BREITMAN; VITERBO, 2010) afirmam que essa mudança na computação tem um paralelo com o cenário do final do século XIX, no período da Revolução Industrial, quando as fábricas eram responsáveis pela produção de sua própria energia motriz. Atualmente, é natural que as fábricas utilizem a energia de que precisam e paguem pelo seu consumo. Essa mesma visão é projetada para as TICs: espera-se que algumas poucas empresas especializadas sejam responsáveis pela gestão e comercialização dos serviços de computação, ao invés de cada empresa manter esses serviços em seus próprios domínios.

Propõe-se nessa tese uma nova estratégia para melhorar a eficiência do consumo de energia de infraestruturas de Computação em Nuvem entre provedores de *data centers*. A tese apresenta uma estratégia de otimização para minimizar o consumo de energia em infraestruturas de Computação em Nuvem compostas de agrupamentos de servidores e redes de comunicação, as quais interconectam os servidores a *backbones* na Internet. A estratégia de otimização proposta é capaz de alocar máquinas virtuais de tal forma que a energia consumida nos servidores e na rede seja minimizada, preservando níveis aceitáveis de qualidade de serviço (QoS) na rede que interconecta os *data centers*. Essa estratégia segue uma abordagem híbrida para combinar o processamento de alocações com Algoritmos Genéticos e Programação Linear Inteira Mista (MILP - *Mixed Integer Linear Programming*) (GOLDBARG; PACCA; LUNA, 2001a) (CHINNECKE, 2012a). As soluções produzidas pelo Algoritmo Genético são refinadas por um *solver* de programação inteira, que processa a alocação ótima de VMs de acordo com um modelo linear da infraestrutura da nuvem. O tráfego dessa alocação é posteriormente simulado por um simulador de rede com a finalidade de avaliar a QoS relativa a vazão, atraso fim-a-fim e perda de pacotes. No processo de busca, VMs são alocadas de tal forma que servidores menos eficientes são evitados, e o tráfego na rede é encaminhado para roteadores mais eficientes. Como resultado, servidores e roteadores podem entrar em estado de baixo consumo de energia sem comprometer a qualidade do serviço oferecido aos usuários da nuvem. Essa estratégia proposta pode auxiliar provedores de serviços de nuvem a avaliar os cenários operacionais de acordo com a computação de seus domínios e os recursos de rede oferecidos para seus clientes. Essa estratégia possui alto potencial para paralelismo, tonando-a factível para a otimização energética de grandes infraestruturas de nuvem. Os resultados obtidos com a ferramenta REALcloudSim, desenvolvida na tese, mostram que a otimização simultânea da energia necessária para a computação e provisão de recursos de rede produz melhores resultados do que aqueles que consideram cada uma dessas

facilidades individualmente.

1.4 Contribuições do Trabalho Proposto

As principais contribuições do trabalho estão focadas no tema Computação Verde, mais precisamente em estratégias para otimizar o consumo de energia em nuvens computacionais. Esta tese propõe:

- Uma modelagem baseada em Programação Linear Inteira Mista (MILP) para a alocação de VMs em servidores de ambientes de nuvem, considerando restrições de capacidade da rede de comunicação interconectando os *data centers*;
- Um estratégia híbrida de alocação de VMs que considera métricas de qualidade de serviço na rede de comunicação interconectando os *data centers*. Esta estratégia combina a modelagem baseada MILP, algoritmos genéticos e simulação de redes de comunicação;
- Uma ferramenta de otimização para simular as alocações de VMs segundo a estratégia híbrida proposta com o objetivo de simplificar o estudo e análise da viabilidade das alocações em redes de referência.

1.5 Organização desse Trabalho

A estrutura da tese está organizada como segue:

- Capítulo 2: descreve os trabalhos relacionados à linha de pesquisa, referente à alocação e escalonamento de recursos em ambientes de nuvem;
- Capítulo 3: apresenta o modelo de alocação de VMs em nuvem baseado em MILP;
- Capítulo 4: apresenta a estratégia híbrida para otimizar o consumo de energia mantendo os níveis de qualidade de serviço na rede dentro de limites estabelecidos pelo provedor de serviços;
- Capítulo 5: descreve a avaliação e os resultados obtidos na pesquisa;
- Capítulo 6: apresenta as considerações finais da tese e propõe sugestões de trabalhos futuros.

Capítulo 2

Trabalhos Relacionados

Este capítulo define os principais termos relacionados à Computação em Nuvem e detalha os principais trabalhos que abordam técnicas de alocação de VMs em infraestruturas de nuvem.

2.1 Introdução

Antes de discutir as estratégias para a alocação de recursos em nuvens é interessante notar que se trata de um problema em aberto. De acordo com Gonçalves et al. (GONÇALVES et al., 2011), provedores de nuvem utilizam *data centers* com uma grande quantidade de servidores, os quais possuem enorme espaço físico, e exigem o controle adequado do consumo de energia. Limitações quanto à escalabilidade dessa infraestrutura, congestionamento do tráfego de dados na rede e subutilização de servidores são problemas que merecem atenção.

Um *data center* agrupa de centenas a milhares de componentes, desde *switches* e roteadores a poderosos servidores, balanceadores de carga e dispositivos de armazenagem. Essas características exigem que a infraestrutura tenha suporte adequado para o funcionamento, o que inclui a provisão adequada de energia, ar-condicionado, segurança, monitoramento e redundância de dados. Equipamentos especializados requerem configuração e treinamento do distribuidor, o que pode encarecer os custos para a provisão de uma infraestrutura particular (VERDI et al., 2010).

Neste capítulo, a ênfase está na descrição dos trabalhos relacionados, os quais são agrupados de acordo com uma taxonomia proposta pelo autor, de forma a simplificar a classificação das principais características das soluções encontradas na literatura.

2.2 Computação em Nuvem

Atualmente, é comum que muitas aplicações sejam acessadas pela Internet e mantidas em *data centers*. Essas aplicações são oferecidas de forma transparente para seus usuários, ou seja, com pouca (ou nenhuma) referência à infraestrutura física que as mantêm. A Computação em Nuvem (*Cloud Computing*) surge como um modelo de computação para acessar serviços oferecidos sob demanda, com transparência de localidade e provisão dinâmica de recursos (servidores, armazenamento, aplicações e quaisquer outros serviços) (VERDI et al., 2010) (ENDO et al., 2010). Vaquero et al. (VAQUERO et al., 2009) definem a Computação em Nuvem como:

> *"Um conjunto de recursos virtuais facilmente usáveis e acessíveis tais como hardware, plataformas de desenvolvimento e serviços. Estes recursos podem ser dinamicamente reconfigurados para se ajustarem a uma carga variável, permitindo a otimização do uso dos recursos. Este conjunto de recursos é tipicamente explorado através de um modelo pay-per-use com garantias oferecidas pelo provedor através de SLAs."*(Tradução do autor)

O NIST (*National Institute of Standards and Technology*) é uma agência governamental do Departamento de Comércio dos Estados Unidos que tem o objetivo de promover padrões de TIC para ampliar a segurança da informação. De acordo com a especificação dessa agência (MELL; GRACE, 2011), as seguintes características podem ser enumeradas:

1. Oferta de serviços sob demanda: alocação dinâmica dos serviços requisitados (*resource pooling*), sem interação humana com o provedor dos serviços;

2. Amplo acesso aos recursos computacionais: por meio de diversos protocolos padronizados, para uma grande variedade de dispositivos como computadores pessoais, dispositivos móveis, *tablets*, entre outros;

3. *Resource Pooling*: por meio de um modelo multi-locatário, recursos computacionais são oferecidos para múltiplos clientes. Existe uma abstração da localidade onde os serviços são oferecidos com relação aos servidores, *data centers*, ou mesmo região geográfica. Os recursos oferecidos incluem processamento de alto desempenho, armazenagem, largura de banda, memória, entre outros.

4. Elasticidade: os serviços devem ser alocados e desalocados rapidamente apenas no decorrer da requisição do usuário. Para este, tem-se a ilusão de uma gama infinita de recursos disponíveis;

5. Medição: a infraestrutura deve oferecer mecanismos para dinamicamente otimizar a disponibilidade de recursos com o uso de mecanismos de medição, gerência e controle do uso dos recursos computacionais oferecidos.

Outras características comumente encontradas em ambientes que utilizam Computação em Nuvem (ENDO et al., 2010) são:

- Escalabilidade: o aumento do número de usuários não implica em redução do desempenho da nuvem, ou seja, mais recursos são alocados internamente, ou de domínios parceiros para suprir a demanda e manter o desempenho. A oferta de recursos sob demanda viabiliza a oferta de serviços para um número maior de usuários. Os recursos serão alocados apenas pelo período contratado, reduzindo a subutilização da rede de serviços. Essa característica implica na elasticidade da oferta de recursos para muitos usuários concorrentes;

- Modelo *pay-per-use*: cobrança proporcional ao uso dos recursos. A Computação em Nuvem é um exemplo de *utility computing* (computação vista como uma utilidade) porque a oferta desses serviços é similar à de outros tradicionais, onde o usuário paga pelo fornecimento de eletricidade, água, gás natural ou redes de telefonia;

- Virtualização: O usuário tem a ilusão de que interage com os recursos de um *host* real quando, na verdade, utiliza um ambiente abstrato que simula o acesso físico do *host* no qual estariam hospedados.

Na Computação em Nuvem um provedor de serviços pode participar de um ou mais modelos para a oferta de serviços em nuvem. Ainda assim, a administração do domínio é responsável por controlar a infraestrutura, sistema operacional, servidores, operações de persistência, e demais requisitos para a oferta de serviços para uma grande quantidade de usuários concorrentes. Os modelos para prestação de serviços em nuvem são os seguintes (CUNNINGHAM, 2009) (MELL; GRACE, 2011):

- IaaS (*Infrastructure as a Service*): destaca a importância da infraestrutura na provisão de serviços. O provedor é capaz de oferecer uma infraestrutura de armazenagem, processamento e demais recursos de *hardware* e *software* de maneira transparente para o usuário. O aluguel de máquinas virtuais de acordo com os requisitos do usuário quanto à CPU, memória, espaço em disco e largura de banda, é um exemplo de serviço do tipo IaaS. Exemplo de provedores: Amazon EC2 (AMAZON WEB SERVICES, 2012) e FlexiScale (FLEXIANT, 2010);

- PaaS (*Platform as a Service*): destaca a importância do *framework* para o desenvolvimento de aplicações na nuvem. O usuário é capaz de desenvolver suas próprias aplicações, respeitando o modelo de desenvolvimento, comunicação, armazenagem, linguagens de programação e demais serviços oferecidos pelo *framework* da nuvem. Exemplo de provedores: Ning (NING, 2010), BungeeLabs (BUNGEE LABS, 2010) e Microsoft Windows Azure Platform (MICROSOFT, 2012b);

- SaaS (*Software as a Service*): O provedor de serviços habilita a execução de aplicações para uso exclusivo do cliente, ou aplicações para uso do cliente fornecidas pelo próprio provedor ou terceiros, tais como aplicativos de *e-mail* empresariais, grupos de discussão, ferramentas para edição de *sites* e demais aplicações que são compartilhadas por um grande número de usuários, tais como formulários Web, ferramentas de suporte (*Help Desk*), *workflows* empresariais, entre outros. Exemplo de provedores: Zoho (ZOHO, 2010), Salesforce (SALESFORCE, 2010), Google Apps (GOOGLE APPS, 2010) e Facebook (FACEBOOK, 2012).

A infraestrutura da nuvem é oferecida em diversos níveis de abrangência, ou modelos de disponibilidade, de acordo com as finalidades da organização e de seus usuários (CUNNINGHAM, 2009) (MELL; GRACE, 2011):

- Nuvem Privada: centrada no domínio de uma organização. A motivação é não expor publicamente interfaces para interação com serviços privados do domínio, mas sim fornecer acesso apenas para usuários e administradores que façam parte do domínio em questão;

- Nuvem Pública: disponível para uso público em geral, com interfaces públicas acessíveis pela Internet;

- Nuvem Comunitária: ambientes onde os participantes de uma comunidade específica compartilham de uma infraestrutura com aspectos comuns (por exemplo, relativo à segurança, padrões de comunicação, interfaces, etc.). Essa infraestrutura possui uma escala menor do que a nuvem pública, porém maior do que uma nuvem privada;

- Nuvem Híbrida: uma combinação de infraestruturas com modelos distintos (pública, privada, comunitária), onde cada infraestrutura se mantém independente das demais, porém

compartilham serviços entre si, e têm a preocupação quanto à portabilidade das aplicações que executam nos diferentes domínios.

A infraestrutura de rede é fundamental para aplicações que executam na nuvem. Na Computação em Nuvem cria-se um ambiente para suportar o desenvolvimento e a disponibilização de aplicações na Internet. Do ponto de visto do usuário, prioriza-se o esforço para a criação e uso de aplicações, mais do que a configuração e a atualização da infraestrutura física ou do conjunto de *frameworks* de suporte ao desenvolvimento.

Os principais tipos de técnicas de virtualização utilizadas em ambientes de nuvem são a virtualização completa, paravirtualização, virtualização assistida por *hardware* e virtualização com *containers*. Os detalhes de cada uma delas são descritas pelo autor desta tese em capítulo de livro (ROCHA et al., 2012) e em (ROCHA, 2013).

2.3 Classificação de Métodos de Alocação de VMs

Alocação de VMs (*VM placement*) é o processo de atribuição de máquinas virtuais a *hosts* físicos (HYSER, 2008) (MICROSOFT, 2012a) (MISHRA; SAHOO, 2011). No âmbito dessa tese, esse processo é formulado com base nos requisitos que essas VMs possuem e que precisam ser inteiramente satisfeitos por servidores físicos em ambientes de nuvem. Por se tratar de um processo sob demanda, e que envolve centenas (até milhares) de VMs com atividades concorrentes, a automação desse processo é de grande valia. Por outro lado, é notório que grande parte dos provedores de serviços de nuvem ocultam como esse processo é feito, por razões comerciais. De acordo com Shankar e Bellur (SHANKAR; BELLUR, 2010), esse processo pode ser classificado em duas grandes categorias:

1. Consolidação de servidores com o intuito de otimizar o consumo de energia: algoritmos de alocação de VMs com o objetivo de minimizar a quantidade de *hosts* físicos necessários para instanciar um determinado conjunto de VMs e suprir suas demandas. Ou seja, busca-se reduzir a quantidade de *hosts* físicos que satisfazem uma demanda específica;

2. Maximizar os recursos oferecidos para as aplicações: algoritmos de alocação de VMs que têm o objetivo de otimizar os recursos oferecidos pela infraestrutura à medida que ocorre variação da demanda das VMs alocadas. Ou seja, busca-se otimizar a quantidade mínima de *hosts* físicos suficientes para acomodar variações da demanda.

Muitos trabalhos sobre alocação de VMs são descritos na literatura, porém métodos híbridos são relativamente novos. Observa-se também que, geralmente, os algoritmos propostos são utilizados com situações e métricas diferentes para pequenos conjuntos de VMs e máquinas físicas (MILLS; FILLIBEN; DABROWSKI, 2011), o que torna a comparação entre métodos difícil de ser reproduzida.

Shang et al. (SHANG et al., 2011) identificam três características com relação às decisões de alocação de VMs em nuvens:

1. Alocação orientada à reserva: um tipo de instância de VM é requisitada mediante pagamento prévio por um determinado período;

2. Alocação sob demanda: o acesso à instâcia é imediato, e a cobrança é baseada em taxas definidas pelo provedor;

3. Alocação em *spot markets*: em um *pool* de recursos onde VMs podem migrar entre *data centers*, as instâncias requisitadas por usuários serão iniciadas quando os preços de uso estiverem abaixo de um limiar previamente estipulado. O esquema de cobrança varia ao longo do tempo.

Ainda de acordo com o NIST (MILLS; FILLIBEN; DABROWSKI, 2011), dois níveis de alocações devem ser considerados: alocação em *cluster* na nuvem, e alocação em nós desses *clusters*. Técnicas de alocação de VMs utilizadas no nível de *cluster* fazem sentido quando é exigida comunicação entre as VMs, e o uso de *switches* locais potencialmente melhora o desempenho dessa comunicação. Ainda assim, diversos trabalhos da literatura utilizam técnicas de alocação de VMs em *pools* de máquinas físicas. Os critérios usados para escolher um *cluster* podem ser combinados com as heurísticas de alocação de VMs nesse *cluster*, de forma a gerar algoritmos básicos de alocação que consideram a escolha do *cluster* e a escolha do nó que satisfaz os critérios de alocação dentro desse *cluster*:

- Critério para escolha do *cluster*: a) *Least-Full First*: ordena o conjunto de *clusters* do menos ocupado para o mais ocupado; b) *Percent Allocated*: ordena os *clusters* de acordo com a proporção de VMs que foram alocadas; c) *Random*: mantém a mesma probabilidade de seleção de um *cluster*, independentemente da quantidade de VMs que nele foram alocadas.

- Heurísticas para escolha dos nós dentro do *cluster*: a) *First Fit*: o primeiro nó que satisfaz a requisição é alocado; b) *Least-Full First*: ordena os nós do menos ocupado para o mais ocupado; c) *Most-Full First*: ordena os nós do mais ocupado para o menos ocupado; d) *Next Fit*: o próximo nó do *cluster* que possui capacidade para satisfazer a requisição; e) *Random*: mantém a mesma probabilidade de seleção de um nó, independente das quantidade de VMs que nele foram alocadas; f) *Tag & Pack*: realiza uma marcação em determinados nós, de forma que apenas determinados tipos de VMs possam ser alocadas neles.

Por outro lado, existem diversas soluções comerciais para alocação de recursos em nuvens: Lanamark Suite (LANAMARK, 2012), VMware Capacity Planner (VMWARE, 2012), IBM WebSphere CloudBurst (IBM, 2012), PlateSpin Recon (NETIQ, 2012). Todas essas ferramentas fazem o balanceamento de carga na nuvem de acordo com os requisitos de VMs, porém nenhuma delas considera as características da rede que suporta a nuvem.

É fato que existe uma grande quantidade de trabalhos relativos a técnicas de alocação de VMs. Diante da dificuldade de se detalhar as muitas variações de métodos da literatura, propomos a taxonomia a seguir como alternativa para agrupar diferentes linhas de pesquisa de acordo com as suas principais características, como mostra a Fig. 2.1.

A camada 1 identifica se o método de alocação de VMs possui suporte para migração ou não. O suporte à migração é contemplado em diversos trabalhos (HYSER, 2008) (ROCHA et al., 2011) (MILLS; FILLIBEN; DABROWSKI, 2011) (BOBROFF; KOCHUT; BEATY, 2007), porém adiciona complexidade ao processo de alocação porque deve considerar a conectividade de rede quando VMs migram entre *data centers* ou domínios diferentes, e é necessário prover mecanismos adicionais de segurança para garantir a integridade dos dados durante a migração. A garantia de conectividade após o processo de migração depende da forma como o *software* de virtualização atua. Como exemplo, VMware vMotion (MURPHY, 2011) define que é necessário manter as mesmas configurações de rede no *data center* de destino, ou seja, endereço IP, máscara

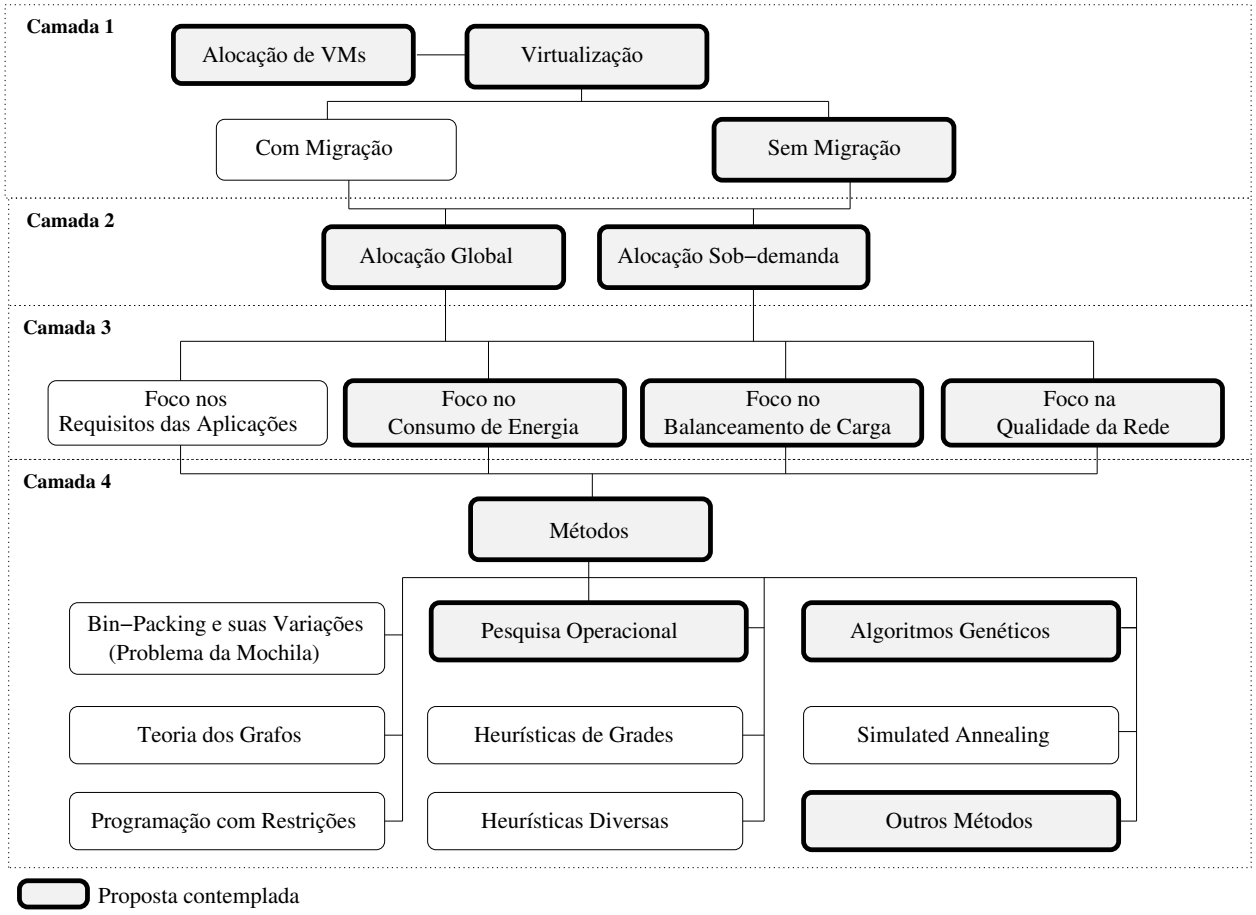

Figura 2.1: Taxonomia para Classificação de Métodos de Alocação de VMs.

de sub-rede, *gateway* e configurações de VLAN (*Virtual Local Area Network*). Dessa maneira, a VM deve migrar para um domínio de destino com as mesmas características da sua rede de origem, caso se queira manter a conectividade.

Sem o suporte à migração, uma vez que a VM é instanciada ela permanece no *host* físico até o término de sua execução. Dessa forma, cada requisição implica em uma nova imagem virtual instanciada no respectivo *host* físico da infraestrutura. Trabalhos nessa categoria são descritos nas referências (MENG; PAPPAS; ZHANG, 2010) (CALCAVECCHIA et al., 2012) (MEHDI et al., 2011) (ALICHERRY; LAKSHMAN, 2012) (MISHRA; SAHOO, 2011) (ZHANG et al., 2010) (YAZIR et al., 2010) (TSAKALOZOS; ROUSSOPOULOS; DELIS, 2011).

A camada 2 define em que momento as VMs são instanciadas. A alocação global faz a alocação de todas as VMs requisitadas de uma única vez. Essa abordagem simplifica o processo de balanceamento de carga porque permite prever como será feita a distribuição com base nos requisitos de todas as VMs. A alocação global é utilizada nas referências (MENG; PAPPAS; ZHANG, 2010) (TSAKALOZOS; ROUSSOPOULOS; DELIS, 2011) (ALICHERRY; LAKSHMAN, 2012) (MISHRA; SAHOO, 2011) (ZHANG et al., 2010) (WU et al., 2012). Por outro lado, a estratégia sob demanda assume que, uma vez que a requisição para instanciar uma VM chegue, esta deva ser instanciada tão logo seja possível. Essa estratégia tem a vantagem de, teoricamente, não limitar a quantidade de alocações, desde que haja recursos disponíveis nos *hosts* da infraestrutura. Porém, torna mais difícil o balanceamento de carga, uma vez que não se

conhecem, a priori, os requisitos das VMs. Essa estratégia é abordada nas referências (HYSER, 2008) (BOBROFF; KOCHUT; BEATY, 2007) (MILLS; FILLIBEN; DABROWSKI, 2011) (CALCAVECCHIA et al., 2012) (MEHDI et al., 2011) (ROCHA et al., 2011) (YAZIR et al., 2010).

A camada 3 define o foco que a estratégia adota, e essa estratégia pode estar contida em uma ou mais categorias. O foco nos requisitos das aplicações diz respeito a QoS e SLA entre os requisitos das aplicações e provedor de serviços. Podem ser citados aspectos referentes ao tempo reservado para execução de tarefas, garantia de banda, custos de utilização, quantidade de recursos reservados nos *hosts*, entre outros. Trabalhos descritos em (BOBROFF; KOCHUT; BEATY, 2007) (MILLS; FILLIBEN; DABROWSKI, 2011) (TSAKALOZOS; ROUSSOPOULOS; DELIS, 2011) (ROCHA et al., 2011) (YAZIR et al., 2010) pertencem a esta categoria.

O foco no consumo de energia diz respeito a otimizar a eficiência energética resultante das alocações referentes à quantidade de *hosts* necessários para prover as alocações, ou otimizar as alocações nos *hosts* mais eficientes. A motivação é reduzir a quantidade de recursos da nuvem que são necessários para satisfazer a demanda. Exemplo de trabalhos nessa categoria são reportados em (CALCAVECCHIA et al., 2012) (ZHANG et al., 2010) (ROCHA et al., 2011) (WU et al., 2012).

O foco no balanceamento de carga tem o objetivo de otimizar a quantidade de alocações por *host*, reduzir a latência entre VMs comunicantes, prover alocações de VMs comunicantes em *hosts* físicos mais próximos em relação ao número de *hops* da rede de comunicação da infraestrutura, previsão de demanda de ingresso de VMs com base em matrizes de tráfego. Exemplo de trabalhos nessa categoria são descritos em (HYSER, 2008) (MILLS; FILLIBEN; DABROWSKI, 2011) (TSAKALOZOS; ROUSSOPOULOS; DELIS, 2011) (MISHRA; SAHOO, 2011) (ZHANG et al., 2010).

O foco na qualidade da rede diz respeito a avaliar as alocações quanto ao desempenho do tráfego de rede das VMs. Trabalhos reportados em (MENG; PAPPAS; ZHANG, 2010) (ALICHERRY; LAKSHMAN, 2012) (WU et al., 2012) (BIRAN et al., 2012) (ROCHA et al., 2011) são exemplos nessa categoria. Os trabalhos que mais se aproximam da proposta desta tese se encontram nessa categoria, e serão detalhados na próxima seção.

A camada 4 agrupa os muitos métodos de alocação de VMs. Grande parte das heurísticas em nuvens são baseadas em heurísticas de grades. Mecanismos de alocação mais simples têm a vantagem de prover a solução mais rápida, mas têm a desvantagem de serem menos susceptíveis ao balanceamento de carga, como apresentado em Mills et al. (MILLS; FILLIBEN; DABROWSKI, 2011).

Com base nesta taxonomia, o método de alocação de VMs proposto nesta tese se classifica como: sem migração (camada 1), alocação global e alocação sob demanda (camada 2), foco no consumo de energia e qualidade da rede (camada 3) e métodos de Pesquisa Operacional, Algoritmos Genéticos, e outros métodos, referente à simulação da rede (camada 4).

A literatura mostra muitos grupos individuais de pesquisa que propõem diferentes algoritmos sobre condições distintas para um pequeno número de VMs e *hosts* físicos para alocação. Estratégias mais realistas utilizam matrizes de tráfego como fonte histórica de dados para as avaliações. Por outro lado, como a carga das VMs muda frequentemente, realocações contribuem para otimizar a qualidade dos serviços oferecidos, uma vez que é impraticável realizar esse controle individual à medida que se aumenta o número de instâncias (HYSER, 2008).

Uma proposta de classificação de algoritmos é feita pelo NIST (MILLS; FILLIBEN; DABROWSKI, 2011) com um método para comparar 18 algoritmos baseados em *bin-packing* para alocação de VMs em ambientes com milhares de *hosts* físicos. O método faz alocações sob demanda em um ambiente construído com a plataforma de nuvem Eucalyptus (EUCALYPTUS, 2012). A estratégia possui dois níveis: 1) escolha de um *cluster* para uma VM; e 2) escolha de um nó para a respectiva VM dentro do *cluster*. Os testes são feitos com análise estatística de variância (ANOVA) sobre o conjunto de dados resultante dos testes. São utilizadas 42 variáveis, chamadas de variáveis de resposta, que são agrupadas em 6 categorias. As informações são rotuladas de acordo com o critério de seleção do *cluster*, heurística de seleção do nó, e combinação de parâmetros que levaram às respostas. A avaliação tem o objetivo de avaliar as diferenças entre os grupos de variáveis de respostas. Essas diferenças dizem respeito às razões médias de VMs obtidas (instanciadas). Os resultados mostraram que existe uma significativa diferença resultante dos critérios de seleção do *cluster*. Os testes são feitos no simulador de eventos discretos Koala, que executa junto a uma ferramenta de simulação comercial, e define configurações de instâncias similares às da nuvem Amazon EC2. A estrutura interna do Koala utiliza três componentes: *cloud controller*, *cluster controller*, e *node controller*. A comunicação entre esses componentes também é simulada. Uma VM é alocada a um nó desde que este nó ofereça os requisitos suficientes para a instanciação. Os parâmetros requisitados pelas instâncias são: a) Tipo da VM (*small*, *large* e *extra-large*); b) quantidade de CPU *cores*; c) Clock da CPU (GHz); d) capacidade de armazenagem; e) quantidade de interfaces de rede; f) quantidade de memória RAM; g) tipo de arquitetura de *hardware* (32 ou 64-bits); h) preço cobrado por hora de uso.

Dentre os diversos trabalhos que se destacam na literatura, Hyser et al. (HYSER, 2008) utilizam um sistema de alocação de VMs com controle autonômico de alocações baseado em políticas. A partir de uma distribuição inicial de VMs, o controlador autonômico realiza realocações com o objetivo de se alcançar a configuração desejada, de acordo com políticas especificadas pelo usuário. Esse sistema considera o uso de migrações e rearranjos sob demanda para alcançar um estado estável de distribuição balanceada entre os *hosts* do *data center*. As avaliações são feitas com um algoritmo do tipo *Simulated Annealing* e se observa que o controlador faz o balanceamento de carga entre seus servidores. O sistema é a base de um protótipo para monitoramento contínuo da CPU, memória, tráfego de rede e armazenagem de dados em disco. Os resultados foram conduzidos em dois passos: com o uso de um simulador que possui um controlador autonômico e políticas de balanceamento de carga; e com o uso de servidores físicos dedicados. Os resultados apontam que soluções autonômicas promovem melhor uso dos recursos da nuvem, auxiliadas por políticas de balanceamento de carga.

Bobroff et al. (BOBROFF; KOCHUT; BEATY, 2007) buscam minimizar a quantidade de máquinas físicas utilizadas para alocar VMs, ao mesmo tempo em que fazem a alocação dinâmica de requisições em períodos de pico de demanda. O objetivo é ajustar dinamicamente as máquinas físicas para suportar a demanda das VMs que são periodicamente alocadas. O algoritmo de alocação utiliza dados históricos de utilização dos servidores, com a previsão da demanda futura, e realocação dinâmica de VMs para *hosts* físicos em períodos regulares. Para isso, o trabalho utiliza um conjunto de fórmulas estatísticas para avaliar o potencial ganho de desempenho, baseado em dados históricos de demandas, antes de efetivamente fazer realocações.

Tsakalozos et al. (TSAKALOZOS; ROUSSOPOULOS; DELIS, 2011) adotam um esquema de alocação de VMs em duas fases, onde a primeira fase consiste em identificar o *host* físico mais apropriado (por exemplo, que pertençam a um mesmo *rack* e/ou que suportem *live migration*) para realizar alocações, e a segunda fase em determinar o mapeamento de VMs para *hosts* físicos,

de acordo com requisitos definidos por usuários da nuvem e características dos *hosts* selecionados na primeira fase. O objetivo é escalar centenas de VMs para *hosts* físicos e, entre outros aspectos, melhorar a qualidade de planejamento das alocações em comparação com outros métodos de apenas uma fase. Os resultados mostram que a largura de banda média efetivamente disponível para VMs tende a aumentar quando elas são agrupadas em *hosts* físicos vizinhos (próximos quanto ao número de *hops* na rede).

Calcavecchia et al. (CALCAVECCHIA et al., 2012) estudaram a natureza dinâmica do ingresso de VMs em ambientes de nuvem. Os autores utilizam uma técnica conhecida como BSP (*Backward Speculative Placement*) para manipular requisições continuamente, enquanto um processo de otimização periódico das alocações é feito sob demanda. Os experimentos foram conduzidos em uma máquina virtual Ubuntu, instanciada em um servidor quad-core Intel Xeon E5530 com 8 GB de RAM. Os resultados referentes ao uso de CPU no ambiente obtidos com a técnica BSP são comparados com os resultados obtidos na alocação ótima de VMs, a cada instante, por meio da solução ótima do problema com o *solver* comercial CPLEX. O algoritmo BSP lida com o mapeamento de VMs, e alocação de recursos de CPU para instanciar a VM da atual requisição. A função objetivo consiste de muitas partes, definidas para oferecer um balanço entre a quantidade de CPUs alocadas de acordo com as múltiplas requisições. Nessa função objetivo também está inclusa a minimização do número de migrações de VMs. Finalmente, a função objetivo inclui um balanceamento de carga das instanciações em *hosts* físicos. Os experimentos foram simulados para uma nuvem de dimensões fixas de até 150 *hosts* físicos, dados históricos de demandas típicas de *data centers*, e uma alternância entre períodos de alta e baixa demanda de VMs. Os resultados mostraram que 95% da demanda regular é satisfeita com a técnica BSP e, quando comparada com o uso exclusivo da programação linear inteira, o BSP reduz o tempo computacional para promover as alocações sob demanda.

Mehdi et al. (MEHDI et al., 2011) exploram o mapeamento de tarefas para VMs, com a intenção de satisfazer os *deadlines* para completar essas tarefas. É utilizado um algoritmo genético, e os testes são feitos com o simulador CloudSim, com o uso de dados históricos. De acordo com os autores, algoritmos genéticos contribuem para melhorar a eficiência do mapeamento de tarefas se comparados com heurísticas como *Minimum Completion Time*, que busca satisfazer a tarefa tão logo ela seja requisitada. No entanto, nenhuma consideração é feita sobre as alocações de VMs, ou sobre como o desempenho da rede do *data center* influencia as alocações de tarefas.

Em geral, outros estudos sobre alocação de VMs em nuvens têm sido abordados com diversos modelos matemáticos, aproximações feitas com algoritmos gulosos (*greedy algorithms*), heurísticas ou uso de *solvers* tais como CPLEX (LI, 2012) e recentes trabalhos baseados em programação baseada em restrições (*Constraint Programming*) (BIN et al., 2011). Alguns trabalhos também tratam o problema de alocação de VMs como um problema de *bin packing* sob demanda, com a adição de atributos específicos para ponderar os servidores (MILLS; FILLIBEN; DABROWSKI, 2011).

A literatura de outras áreas cita diversos trabalhos que empregam estratégias híbridas de otimização combinando métodos bio-inspirados com métodos de otimização clássicos como programação linear e não-linear. Souza et al. (SOUZA et al., 2010) descrevem uma proposta para avaliar a resolução do problema de planejamento de uma agroindústria para maximizar o retorno econômico e atender às limitações da atividade canavieira. Em resumo, busca-se o planejamento na colheita de cana-de-açúcar para determinar a época do ano que forneça o melhor acúmulo de açúcar por lote da plantação. Na modelagem do problema, a função objetivo prioriza os teores de sacarose e fibra (biomassa) por lote. O AG determina quais lotes serão colhidos e quais não. Para esse problema, não houve uma combinação dos métodos de programação linear

e não-linear com o AG, mas sim uma comparação dos resultados obtidos por ambos.

Krapivka e Ostfeld (KRAPIVKA; OSTFELD, 2009) fazem uma combinação de AG e PL (Programação Linear). O objetivo é definir um sistema de distribuição de água com canos de menor custo. Para isso, é definida uma fase de decomposição, onde o PL utiliza uma função objetivo para minimizar o comprimento e o diâmetro dos canos com fluxos arbitrários (fixos). Na segunda etapa, busca-se encontrar a árvore da rede de distribuição de menor custo, relativa ao comprimento e diâmetro dos canos, ainda com o fluxo arbitrário. Na terceira etapa, o AG é utilizado para minimizar o fluxo capaz de escoar pela rede anteriormente obtida. Na população do AG, os cromossomos representam o quanto de fluxo de água deve escoar por cada aresta na árvore de distribuição. O *fitness* é definido como o valor obtido pela função objetivo, uma vez que cada cromossomo é avaliado pelo *solver* de programação linear. As soluções não-factíveis são penalizadas. O critério de parada é definido como o número máximo de iterações de experimento, ou segundo a verificação de que nenhuma melhora significativa seja obtida em determinado número de iterações.

Luo et al. (LUO; GUIGNARD; CHEN, 2001) apresentam um *framework* que combina diferentes métodos de otimização em um modelo complementar híbrido. A proposta tem o objetivo de reduzir o esforço para computação de problemas de programação inteira. Um AG é utilizado para rapidamente gerar boas soluções factíveis com LP, e a modelagem em MILP para resolver o problema que resulta de uma parte inteira da solução, obtendo soluções próximas à ótima. Finalmente, um mecanismo de ordenação é utilizado para ranquear as soluções. Outro trabalho que faz a combinação de métodos de MILP e AG é apresentado em (NIEMINEM; RUUTH; MAROS, 2003). Nele, o AG é utilizado para encontrar soluções iniciais para posterior avaliação com um modelo MILP. A motivação é reduzir o tempo computacional de processamento do MILP, com o uso de relaxamento linear nos passos intermediários do AG.

2.4 Modelagem da Rede de Comunicação

Nesta seção, são analisados quatro trabalhos recentes que consideram o desempenho do tráfego de rede das VMs instanciadas. Apenas em um dos trabalhos o impacto é avaliado na rede entre *data centers*. Nos demais, a rede considerada é a rede interna ao *data center*. Em um dos trabalhos, o impacto é avaliado em termos do consumo de energia na rede de comunicação, enquanto, nos demais o impacto é avaliado em termos de métricas de roteamento como o número de *hops* que um fluxo percorre de sua origem ao destino.

Meng et al. (MENG; PAPPAS; ZHANG, 2010) consideram a rede conectando servidores em um *data center*. A rede é modelada como um grafo onde os vértices representam servidores e *switches* de rede, e as arestas representam os *links* de rede. O objetivo é alocar as VMs de tal forma que o número de *switches* necessárias para rotear o tráfego gerado pelas VMs seja minimizado. Os autores empregam um algoritmo de corte mínimo em grafos para o problema de alocação de VMs e propõem dois modelos de tráfego, global e particionado, de forma a reduzir a complexidade do problema. No modelo de tráfego global cada VM se comunica com outras em uma taxa de tráfego constante, enquanto que no modelo particionado a comunicação ocorre apenas entre VMs na mesma partição (corte no grafo). Os autores apresentam um estudo de caso com a avaliação da alocação de 1024 VMs em blocos de 64 VMs para diferentes tipos de topologias de rede, tamanhos de partição e modelos de tráfego. Além disso, sua estratégia de alocação é comparada com uma estratégia baseada em alocações não-ordenadas de VMs. Os autores concluem que a alocação de VMs considerando o tráfego na rede como proposto aumenta

a escalabilidade da rede ao reduzir o tráfego encaminhado nos *switches* de nível mais alto (*core switches*) onde os congestionamentos na rede usualmente ocorrem.

Alicherry e Lakshman (ALICHERRY; LAKSHMAN, 2012) consideram o problema de alocação de VMs com o foco na rede que interconecta os *data centers*. A rede é modelada como um grafo onde os vértices são *data centers* e as arestas são os caminhos de rede que conectam os *data centers*. As arestas são ponderadas de acordo com a distância (número de saltos) no caminho de rede. O objetivo é: dado um bloco de VMs e um tráfego entre as VMs (matriz de tráfego), minimizar a distância máxima de rede entre quaisquer duas VMs comunicantes. Os autores modelam o problema de alocação de VMs como o problema de encontrar um subgrafo do grafo da rede cujos vértices correspondam aos *data centers* onde as VMs serão alocadas, enquanto o comprimento máximo de qualquer caminho mais curto (diâmetro) é minimizado. Como esse problema é NP-difícil, os autores derivam um algoritmo de aproximação para resolvê-lo. Experimentos foram conduzidos em 5 cenários de nuvem com o número de *data centers* variando de 10 a 100 e com 50 a 200 servidores uniformemente distribuídos entre os *data centers*. A rede consiste de uma grade 1000×1000 onde os *data centers* estão aleatoriamente distribuídos. O algoritmo proposto é comparado com um algoritmo de alocação aleatório e outro algoritmo de alocação simplificado para a alocação de 1000 VMs. O algoritmo simplificado seleciona o *data center* com o número máximo de recursos disponíveis e aloca quantas requisições forem possíveis neste *data center*. De acordo com as avaliações conduzidas, o algoritmo de alocação proposto pelos autores supera os algoritmos de alocação aleatório e simplificado em percentuais que variam de 32,6% a 86,4%.

Wu et al. (WU et al., 2012) tratam o problema de alocação de VMs considerando o consumo de energia nos servidores e na rede. Os autores apresentam um modelo de alocação onde a energia consumida pelo servidor é uma função linear de sua carga, mais a energia consumida quando o servidor está ocioso. A energia consumida pela rede é uma função da quantidade de dados trocados entre as VMs. Tais trocas de dados são aleatoriamente geradas nas simulações conduzidas pelos autores. O modelo de alocação assegura a unicidade da alocação de VMs em servidores, e restrições de recursos relacionadas a CPU e memória consumida pelas VMs em seus respectivos servidores. A função objetivo é dada pela soma do consumo de energia nos servidores e na rede. Um algoritmo genético é utilizado para resolver o problema com 100 servidores e 500 VMs. Os resultados são comparados com um algoritmo do tipo *First Fit Decreasing*, que é uma das heurísticas mais simples para resolver o problema da mochila (*bin-packing*). De acordo com os autores, o algoritmo genético obteve uma melhora entre 3,5% e 23,5% na redução do consumo de energia, quando comparada com o algoritmo *First Fit Decreasing*.

O trabalho de Biran et al. (BIRAN et al., 2012) consideram como requisitos das VMs CPU e memória, bem como requisitos de rede nos *data centers*. A rede é modelada como uma árvore onde os nós representam os *switches* da rede e as arestas representam os *links* que conectam os *switches*. O objetivo é alocar as VMs de tal forma que o balanceamento de carga na rede seja alcançado. Uma carga balanceada torna a rede capaz de acomodar rajadas do tráfego sem degradar a qualidade do serviço, principalmente a perda de pacotes. O problema é formulado de acordo com um modelo de programação inteira quadrática. A função objetivo minimiza o pior caso de *cut load ratio*. Em suma, o *cut load ratio* é uma métrica da capacidade restante em uma partição da rede (corte no grafo). Como o problema é NP-difícil, os autores propõem duas heurísticas para encontrar soluções sub-ótimas. A primeira heurística consiste de duas fases. A fase inicial emprega uma formulação de programação linear inteira (ao invés de quadrática) para derivar um conjunto de partições, enquanto a próxima fase aloca VMs em partições por meio de programação inteira quadrática, mas para um problema de dimensão reduzida. A segunda

heurística não emprega nenhum algoritmo de programação matemática. O simulador de rede NS-2 é utilizado para verificar o impacto da alocação de VMs na rede empregando modelos de tráfego mais realísticos. Os autores apresentam experimentos com variação da quantidade de VMs de 640 a 3430 VMs, mas as simulações de rede são restritas ao caso de 128 VMs e 64 servidores. Quando são comparadas as duas heurísticas propostas, os autores concluem que a primeira heurística alcança alocações de VMs com carga mais balanceada, o que economiza a capacidade da rede para futuras alocações, bem como acomoda melhor as flutuações no tráfego.

Os trabalhos citados acima apresentam algumas similaridades com a proposta desta tese no sentido que levam em conta o impacto da alocação de VMs na rede, seja em termos de parâmetros de roteamento (número de *hops* ou custo dos *links*), seja em termos do consumo de energia nos roteadores. Quando a simulação de rede é empregada, permitindo uma avaliação mais precisa do tráfego resultante da alocação, a mesma é realizada após o processo de alocação. Como consequência, as alocações não levam em consideração parâmetros de QoS da rede.

A principal diferença da proposta desta tese em relação aos trabalhos avaliados nesta seção é a capacidade da estratégia proposta de otimizar o consumo de energia nos *data centers* e na rede de comunicação concomitantemente com parâmetros de QoS da rede, ou seja, os parâmetros de QoS são processados *durante* o processo de alocação, e não *a posteriori*, como nas estratégias avaliadas. A estratégia proposta permite incluir no processo de alocação parâmetros de QoS tais como atraso fim-a-fim, perda de pacotes e balanceamento de tráfego nos *links*. Por exemplo, é possível definir que no processo de alocação de VMs o consumo de energia e a perda de pacotes devem ser minimizados *conjuntamente*. Nenhum trabalho avaliado é capaz de otimizar parâmetros de alocação de VMs e de rede simultaneamente.

2.5 Computação Verde em Nuvem

A Computação Verde em ambientes de nuvem tem sido considerada primariamente como uma alternativa para reduzir os gastos com refrigeração e provisão de energia, muito embora também sejam consideradas questões ambientais relativas ao aquecimento global, aumento do consumo de energia e a disseminação da tecnologia da informação. Nesse sentido, a definição do termo "Computação Verde" é ampla e envolve muitas áreas, das quais são citadas: otimização do uso dos recursos, projeto da infraestrutura, consumo de energia, uso de fontes energéticas renováveis, a refrigeração do ambiente interno, o descarte do *hardware*, o reaproveitamento dos recursos utilizados, o impacto ambiental da instalação do *data center*. Murugesan (MURUGESAN, 2008) define Computação Verde como as melhores práticas em sistemas de TIC para melhorar a eficiência energética durante a vida útil dos equipamentos, reduzir a emissão de gases poluentes, uso de recursos menos poluentes, e alternativas para reutilizar e reciclar lixo eletrônico (*e-waste*). Murugesan afirma que o problema do *e-waste* é global, e análises predizem que $\frac{2}{3}$ dos quase 870 milhões de PCs no mundo todo, em menos de 5 anos, serão descartados, e mais de 20 milhões de toneladas de *e-waste* são geradas por ano no mundo todo. Como consequência, é previsto um aumento da quantidade de produtos tóxicos descartados no ambiente. Por outro lado, as pessoas têm se tornado mais conscientes sobre sua própria posição diante das mudanças climáticas, e como elas podem ajudar a fazer o uso mais sustentável de recursos ao buscar utilizar produtos que tenham preocupação com a redução da poluição no ambiente. Alternativas convencionais para redução do consumo de energia em dispositivos eletrônicos são o uso de funções de *sleep*, *hibernate* e *wakeup* (HOW-TO GEEK, 2012). Na função *sleep* o estado de execução do *host* é armazenado na memória RAM e o *host* entre em estado de baixo consumo

de energia. Basicamente, a função *sleep* tem o mesmo papel da função *standby*. Na função *hibernate* o estado de execução do *host* é armazenado em disco, sendo que o *host* é desligado, e não há consumo de energia. Em ambos os casos, quando o *host* for novamente ligado (função *resume/wakeup*), as últimas informações salvas devem ser novamente apresentadas. Por outro lado, o selo *Energy Star* está presente na maioria dos dispositivos de TI de forma a identificar produtos energeticamente eficientes (ENERGY STAR, 2012).

No Brasil, a Resolução 382/2006 do Conama (Conselho Nacional do Meio Ambiente) (PORTAL BRASIL, 2012) define os limites máximos de emissão de poluentes pelas indústrias que iniciaram a partir de 2007, e a Resolução 436/2011 para as que iniciaram antes de 2007. Um pouco mais severa é a política climática da União Européia (EUROPEAN COMMISION, 2012), que estabeleceu um projeto de lei para reduzir sua emissão de gases poluentes em 20%, em relação aos níveis de 1990, até 2020, e uma redução de 80%, em relação ao níveis de 1990, até 2050. Por outro lado, Fettweis e Zimmermann (FETTWEIS; ZIMMERMANN, 2008) afirmaram que, em 2008, *server farms* e infraestruturas de telecomunicação seriam responsáveis por até 3% de todo o consumo mundial de eletricidade, e que o crescente aumento no tráfego da Internet e do número de telefones móveis tenderia a aumentar esse consumo ainda mais. Os autores também afirmaram que *data centers* e sistemas de comunicação móvel dobram o seu consumo de energia a cada 5 anos. No entanto, um estudo de Renzenbrink (RENZENBRINK, 2011) em 2011 afirmou que os *data centers* consumiram em torno 1,3% do consumo mundial de eletricidade. As razões para a redução foram a crise financeira de 2008, combinada com os esforços da indústria em criar servidores mais eficientes. Ainda assim, a Cisco (CISCO SYSTEMS, 2012c) prevê que, até 2017, o crescimento no tráfego de dados estará principalmente relacionado à telefonia móvel e aplicações de vídeo sob demanda.

Especificamente em *data centers*, a modernização da própria infraestrutura de TIC contribui para reduzir os impactos ambientais: técnicas como virtualização contribuem para consolidar muitos servidores virtuais em um mesmo *host*, o que potencialmente reduz o consumo de energia e encoraja a reutilização do *hardware*. Outras estratégias são o uso de abordagens DCIM (*Data Center Infrastructure Management*) para fornecer monitoramento com *software* especializado, sensores, e plataformas centralizadas de gerência em tempo-real; CHP (*Combined Heat and Power*), ou co-geração, é uma outra alternativa para o fornecimento de energia adicional. CHP é entendida como a geração combinada e simultânea de energia térmica (para aquecimento e/ou refrigeração) e elétrica ou mecânica, a partir de uma mesma fonte (SHIPLEY et al., 2008). De fato, CHP é uma alternativa para redução da emissão de gases poluentes, e os geradores que utilizam essa tecnologia podem utilizar tanto combustíveis fósseis (por exemplo: carvão, petróleo, gás natural) quanto renováveis (por exemplo: cana-de-açúcar, mamona, óleo de soja e outros). A motivação é utilizar a energia térmica que normalmente não é utilizada no processo de geração de energia elétrica para fornecer refrigeração à infraestrutura do *data center*. Outra alternativa é o uso de *Thermal Storage* que auxilia a manter a temperatura da infraestrutura com sistemas adicionais de água refrigerada em casos de falta de energia elétrica (INTEL, 2012). Essas soluções são o reflexo de iniciativas, tais como a ISO 14000 e normas relacionadas (NASCIMENTO; POLEDNA, 2002), que buscam estabelecer mecanismos para promover gerência ambiental em empresas. Melhorias na infraestrutura dos *data centers* contribuem para reduzir os custos com refrigeração e controle climático. Outra medida que tem sido adotada é a instalação de *data centers* em regiões geográficas onde o clima favorece a refrigeração, por exemplo, em regiões árticas (LADURANTAYE, 2012).

Um ambiente de *data center* típico possui um grande conjunto de equipamentos de TICs, sistemas de UPS (*Uninterruptible Power Supply*) para proteger os equipamentos da falta de

energia, sistemas de distribuição de energia elétrica, sistemas de ar-condicionado, umidificação e iluminação. Uma métrica utilizada para estimar o uso efetivo de energia em um *data center* é o PUE (*Power Usage Effectiveness*). Essa métrica é obtida pela divisão da quantidade de energia que entra em um *data center* pela energia utilizada na infraestrutura em um ano (READ, 2012b), que pode ser obtida com a medição das PDUs (*Power Distribution Units*). A medição das PDUs deve representar a energia total fornecida aos *racks* de servidores. Um PUE de $1,0$ equivale a um *data center* 100% eficiente, enquanto que um PUE de $2,0$ indica que o *data center* utiliza 2 vezes mais energia do que a necessária para manter os seus equipamentos de TIC. De acordo com Fontecchio (FONTECCHIO, 2012), um típico *data center* tem um PUE médio de $2,5$, o que significa que de cada $2,5$ watts oferecidos, apenas 1 é utilizado pela insfraestrutura. De acordo com Calvo (CALVO, 2013) no Brasil a média do PUE é de $2,4$. Fontecchio (FONTECCHIO, 2012) indica a possibilidade de reduzir o valor do PUE para até $1,6$ com equipamentos mais eficientes. Como exemplo, modelos de *data centers* verdes, tais como o Merlin (READ, 2012a), representam esforços para melhorar a eficiência energética de grandes centros de dados, com PUE estimado de 1,08 (READ, 2012b). Esse *data center* é um projeto piloto construído na Inglaterra que utiliza práticas verdes em todo o projeto de sua infraestrutura com o objetivo de otimizar ao máximo o uso de seus recursos. As principais preocupações são voltadas para segurança de acesso ao dados (física e lógica), eficiência energética para manter a estrutura de *hardware*, e sensoriamento inteligente da temperatura e qualidade do ar. A responsabilidade social é relativa a manter dados de seus clientes com segurança, alta disponibilidade e menor impacto ambiental quanto à emissão de gases poluentes.

De acordo com Glanz (GLANZ, 2012), os *data centers* consomem cerca de 30 bilhões de watts de eletricidade anuais, o equivalente à geração de energia de 30 usinas nucleares. Estima-se que a maior parte desse consumo, até $\frac{1}{3}$ dele, seja decorrente de instalações nos Estados Unidos. Além disso, as companhias normalmente mantêm todo o seu parque tecnológico em funcionamento, qualquer que seja a demanda. Como consequência, até 90% de toda a energia elétrica consumida pode ser desperdiçada. Os dados apresentados em pesquisa por Rajic (RAJIC, 2012) mostram que a maior parte do custo total de propriedade relativo à instalação, funcionamento e manutenção de um *data center* é referente à provisão de energia elétrica para a infraestrutura, como mostra a Fig. 2.2. Rajic também relata que a maior parte do consumo de energia nesses ambientes é destinada à refrigeração (climatização), como apresentado na Fig. 2.3. Barroso e Hölzle (BARROSO; HÖLZLE, 2009) afirmam que grande parte do consumo de energia é devido à provisão da climatização a longas distâncias, sendo a prática comum manter a temperatura do ambiente da infraestrutura computacional em torno dos 20°C.

Os diversos servidores são comumente organizados em *racks* e requerem cuidados quanto às fontes de suprimento de energia e refrigeração. Outro exemplo são os ambientes computacionais para computação de alto desempenho (SILICON GRAPHICS INTERNATIONAL, 2012) que possuem centenas de CPU *cores*, terabytes de memória, e petabytes de espaço de armazenamento em disco. Um único *rack* pode suportar mais de 500 processadores. Configurações dessa natureza exigem tanto o uso de ar-condicionado quanto sistemas de refrigeração alternativos (por exemplo, água refrigerada) em cada *rack*. Outra alternativa que vem sendo considerada é o uso de *data centers* modulares (LI et al., 2011), que oferecem custo mais baixo e uso mais eficiente de energia em relação às infraestruturas convencionais. Esses *data centers* são montados em *containers* com toda a infraestrutura de servidores (de 1000 a 4000 servidores), armazenamento e dispositivos de rede das infraestruturas convencionais, com a vantagem de poderem ser transportados para a localização mais conveniente, e permitirem a rápida integração com infraestruturas já existentes. A heterogeneidade dessas infraestruturas em *containers* tem gerado

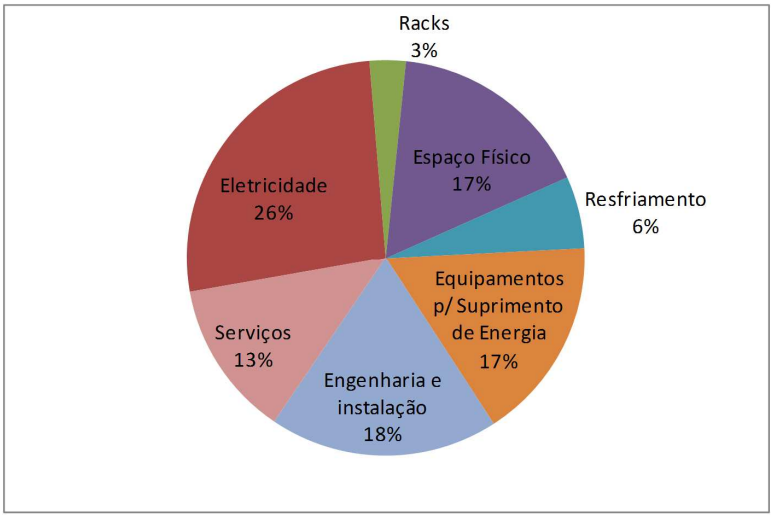

Figura 2.2: Estimativa de Custo Total de Propriedade de um *Data Center* (baseado em (RAJIC, 2012)).

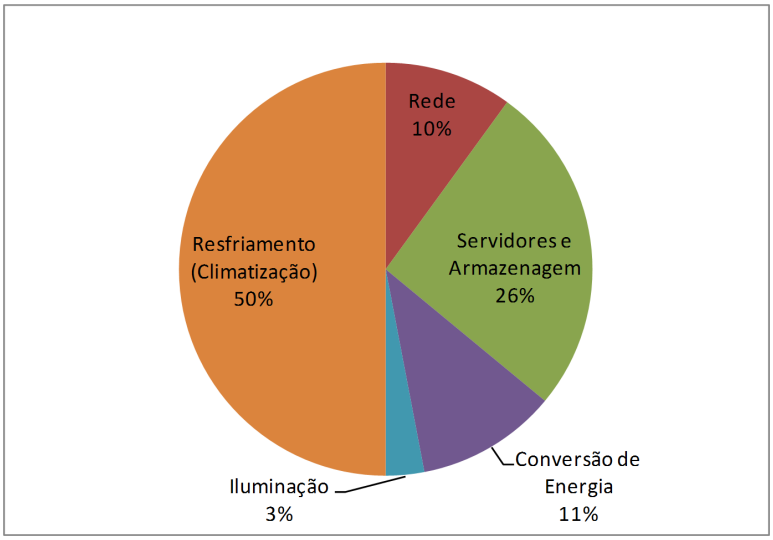

Figura 2.3: Estimativa de Consumo de Energia de um *Data Center* (baseado em (RAJIC, 2012)).

interesse na pesquisa de arquiteturas para integrá-los (LI et al., 2011).

O funcionamento de um *data center* requer considerável quantidade de energia. O grupo Gartner estima que esse consumo seja responsável por até 12% das despesas operacionais (PETTEY; GOASDUFF, 2012). Moore et al. (MOORE et al., 2005) afirmam que o alto consumo de energia também gera calor e o custo para prover o sistema de refrigeração alcança os milhões de dólares anuais em *data centers* clássicos de grande porte, como os da Google, HP, Cisco, e outros. Nesses ambientes é comum que existam diversas unidades CRAC (*Computer Room Air Conditioning*), que monitoram e mantêm a temperatura, a distribuição do ar e a umidade (ROUSE, 2012). Moore et al. também afirmam que é possível economizar milhões de dólares com sistemas de controle climático inteligentes nos CRACs. Além disso, essas condições sugerem a instalação de *data centers* onde o clima beneficia a manutenção da temperatura. *Data centers* típicos possuem área pouco menor a de um campo de futebol (em torno de 9 mil

m^2). De acordo com Rasmussen (RASMUSSEN, 2012), a maior parte do calor é gerado pela própria infraestrutura. Nesse sentido, a melhoria da organização e instalação da infraestrutura física e da otimização da interconexão entre os servidores é essencial.

Outro tema de pesquisa recente é a Computação Verde em redes (*Green networking*). Bianzino et al. (BIANZINO et al., 2012) definem a Computação Verde em redes como o *design* da arquitetura da rede que considera a economia do consumo de energia. Além disso, é considerada a manutenção da temperatura dos dispositivos da rede, uma vez que há um substancial gasto em *data centers* referente à manutenção da temperatura, com o objetivo de aumentar a vida útil de seus equipamentos. Nesse caso, a redução do consumo refere-se ao uso de equipamentos de baixo consumo com nível de desempenho similar (ou superior) ao de dispositivos sem essa preocupação.

Como alternativa para permitir o monitoramento e controle de energia na infraestrutura de rede, a Cisco desenvolveu a arquitetura EnergyWise (CISCO SYSTEMS, 2012a). A arquitetura EnergyWise é formada por três camadas. A camada 1 é a camada de gerência de aplicações, que inclui *softwares* específicos para o controle de energia da rede. A Cisco fornece um protocolo próprio nessa camada, conhecido como EnergyWise Toolkit MAPI (*Management API*) para monitorar, controlar e informar o consumo de energia dos recursos clientes conectados, mas também suporta o uso do protocolo SNMP (*Simple Network Management Protocol*), com a MIB (*Management Information Base*) CISCO-ENERGYWISE-MIB. Uma MIB é uma base de dados de objetos para gerenciar, atribuir ou ler informações de um dispositivo de rede (CISCO SYSTEMS, 2012b). A camada 2 da arquitetura é formada pelos *switches* e roteadores da rede interna que utilizam o protocolo Cisco EnergyWise. Nesse caso, esses dispositivos de rede devem ter a função EnergyWise habilitada. A camada 3 refere-se aos *endpoints* que utilizam o SDK (*Software Development Kit*) EnergyWise cliente. As aplicações clientes dessa camada adquirem a informação de consumo de energia dos *endpoints* conectados à rede e informam para as aplicações EnergyWise MAPI.

Na literatura, Baldi e Ofek (BALDI; OFEK, 2009) afirmam que os serviços que utilizam a Internet continuarão a a crescer exponencialmente, acompanhados pelo crescimento no consumo de energia elétrica para manter esses serviços operacionais. Os autores afirmam que a demanda por largura de banda na rede tende a crescer de 50 a 100 vezes, principalmente em função da demanda por serviços de vídeo baseado em *streaming* sob demanda e *downloads*. Dentre as alternativas consideradas para a Internet do futuro estão: redução do processamento por pacote IP, redução dos requisitos de memória nos dispositivos de rede, uso de elementos de comutação de pacotes mais eficientes, ampliação da capacidade de uso dos *links* de forma a colocar as linhas inativas em estado de baixo consumo de energia. Outra alternativa interessante é o uso do encaminhamento "em dutos" (*pipeline forwarding*) onde os elementos de roteamento na rede são sincronizados para permitir o encaminhamento periódico do tráfego, o que potencialmente gera economia de energia para o roteamento dos dados.

A literatura também contempla diversos trabalhos relacionados à economia de energia em redes *wireless* (SIMUNIC, 2005) (PERING et al., 2006) (TAKIGUCHI et al., 2009) (NOBAYASHI et al., 2011). Em resumo, são apresentadas propostas em todas as camadas de protocolos, sejam elas física, de enlace, de transporte ou de aplicação. Nordman e Christensen (NORDMAN; CHRISTENSEN, 2010) afirmam que grande parte do consumo anual de energia dos dispositivos de rede refere-se a torná-los presentes na rede, e não necessariamente em função da sua carga de processamento. Isso sugere a possibilidade de reduzir o consumo de energia gasto para manter esses dispositivos presentes na rede quando eles estão ociosos (quando não processam pacotes). Para isso, os autores indicam o uso de *proxy* de baixo consumo de energia

para manter a conectividade do *host* quando este estiver ocioso. O mecanismo de *proxy* possibilita, de forma transparente, que o *host* provedor de serviços na rede transite entre os estados de ocioso ou desligado para o estado de presença na rede, uma vez que o *proxy* mantém um *cache* da presença desse dispositivo. Esse sistema de *proxy* pode ser tanto interno quanto externo ao dispositivo de rede, ou seja, outro dispositivo pode atuar como *host* de presença, enquanto o *host* da rede permanece em estado de baixo consumo de energia ou desligado. Uma abordagem similar com o uso de *proxy* também é tratada por Nedevschi et al. (NEDEVSCHI et al., 2009). Em outro trabalho, Nedevschi et al. (NEDEVSCHI et al., 2008) também afirmam que os dispositivos presentes nas redes convencionais consomem substancial quantidade de energia mesmo quando estão ociosos. Diante disso, os autores propõem colocar esses dispositivos de rede em estado de *sleep* nos períodos em que esses dispositivos estão ociosos, e controlar a taxa de processamento de acordo com o *workload* atual. Os autores também afirmam que as primitivas de conservação de energia deveriam estar embutidas no *hardware*, e que os protocolos da rede deveriam fazer uso dessas primitivas.

Wang et al. (WANG et al., 2012) afirmam que o foco da maioria das redes móveis na atualidade está na capacidade, variedade e estabilidade dos serviços de comunicação, sem levar muito em consideração a eficiência energética. As técnicas utilizadas na comunicação móvel buscam em sua maior parte maximizar o desempenho relativo à vazão, qualidade do serviço, e confiabilidade. Outra consideração diz respeito ao uso da energia consumida por esses dispositivos a qual não é dinamicamente ajustada de acordo com as condições de tráfego. Os autores também afirmam que até o momento, a maioria das técnicas de Computação Verde em redes para Computação Móvel apresentam algum tipo de degradação da qualidade do serviço. Wang et al. também apresentam um resumo dos principais projetos de Computação Verde em redes móveis no mundo todo. No que diz respeito à computação em nuvem, os autores indicam que a maioria dos serviços de nuvem estão preocupados em melhorar o desempenho do transporte, processamento e armazenagem de dados, sem levarem muito em consideração a eficiência energética dos serviços oferecidos. Os autores propõem estudos em diversas áreas para melhorar a eficiência desses serviços: agregação de dados com transmissão esporádica, compressão, compartilhamento de réplicas de dados, além de algoritmos mais eficientes para a escolha do serviço de nuvem.

Roteadores com funções inovadoras são propostos por Yamada et al. (YAMADA et al., 2009), que identificam a possibilidade de se obter alta performance nos roteadores mesmo com baixo consumo de energia. Para isso, os autores criaram um roteador que integra FPGAs (*Field Programmable Gate Array*), com memórias de alta capacidade e interfaces de rede de alto desempenho. Os resultados indicam uma economia de até 50% no consumo de energia em comparação com roteadores convencionais. Outra consideração é a utilização de controle dinâmico de desempenho por porta ou módulo no roteador, com ajuste da frequência de operação. Nesse caso, a economia no consumo chega a alcançar de 10 a 20% em relação a roteadores convencionais.

2.6 Considerações Finais

Essa seção fez uma revisão bibliográfica de algumas das principais metodologias para alocação de VMs em nuvens. Foi proposta uma taxonomia para classificar as diferentes abordagens de alocação. Foi apresentado também um panorama de uma área de pesquisa recente, a Computação Verde em nuvens e nas redes de comunicação. A adição de funções de *sleep* e *wakeup* em servidores e roteadores possibilita a concepção de estratégias de alocação de VMs em nuvens

que tiram proveito destas funções, tal qual a estratégia proposta nesta tese. A estratégia de alocação de VMs proposta fornece o subconjunto mínimo de servidores e roteadores que devem se ativados para atender as demandas das VMs alocadas. Os demais servidores e roteadores podem ser postos em modo *sleep* a fim de economizar energia. Vale ressaltar ainda que a Computação Verde em nuvens e na infraestrutura de rede são temas recentes com muitas oportunidades de pesquisa em aberto.

Capítulo 3

Modelagem Linear para Alocação de VMs

Este capítulo descreve um modelo linear para alocação de VMs. Essa proposta considera os requisitos para as VMs serem alocadas, as capacidades do ambiente de nuvem e a topologia de rede entre *data centers*. A modelagem tem o objetivo de otimizar o consumo de energia entre os servidores e roteadores que encaminham o tráfego entre os domínios considerados. Entretanto, parâmetros de qualidade de serviço na rede tais como atraso fim-a-fim e taxa de perda de pacotes não são tratados pelo modelo linear.

3.1 Introdução

O problema de alocação de VMs em *data centers* consiste em encontrar um servidor em qualquer *data center* capaz de fornecer os recursos que a VM demanda. Estes recursos consistem tipicamente de:

- quantidade de CPUs (*cores*) físicas ou virtuais;
- quantidade de memória RAM (física);
- espaço em disco para a VM abrigar seu sistema de arquivos;
- largura de banda de rede para comunicação da VM com o mundo exterior.

A infraestrutura convencional de um *data center* é ilustrada na Fig. 3.1, baseada em Greenberg et al. (GREENBERG et al., 2008). A infraestrutura é composta de um grande conjunto de servidores (podendo chegar a centena de milhares) e uma rede de comunicação composta de roteadores IP, balanceadores de carga, *switches* de alta capacidade e enlaces tipicamente óticos. O *hardware* e *software* utilizados em um *data center* são especializados, por exemplo, virtualizadores, bases de dados e sistemas de arquivos (GREENBERG et al., 2008). Também é comum que subconjuntos de *racks* de servidores sejam agrupados em *containers* (NIEMANN, 2008). A rede de camada 2 (L2) é empregada na comunicação dentro do *data center*, sendo dimensionada para comportar o tráfego gerado pelos servidores sem qualquer restrição de desempenho definida, a priori, pelo provedor. A rede de camada 3 (L3), por outro lado, interliga os *data centers* entre si e com o *backbone* Internet mantido pelos grandes provedores de rede.

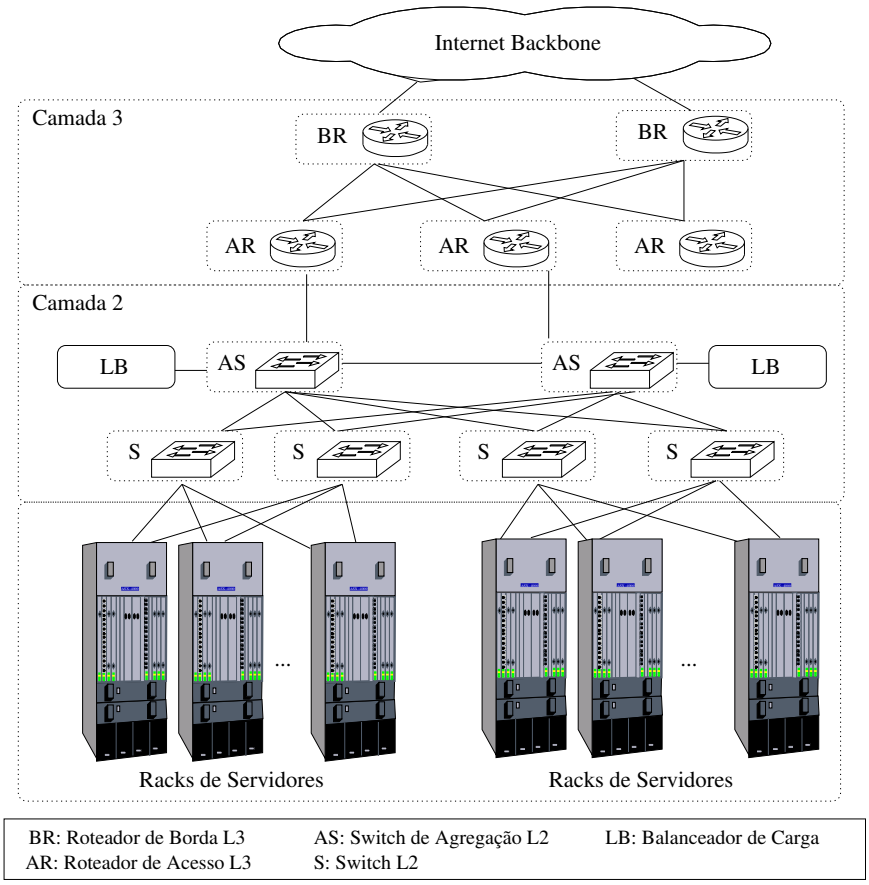

Figura 3.1: Arquitetura Convencional de um *Data Center* (baseada em (GREENBERG et al., 2008)).

Por ser baseada em roteadores, esta rede é sensível ao tráfego gerado pelos *data centers*. Portanto, a alocação de VMs em *data centers* deve levar em conta as capacidades e restrições desta rede. Mais detalhes da infraestrutura interna de *data centers* típicos são descritos por Barroso e Hölzle. (BARROSO; HÖLZLE, 2009). A estratégia de alocação de VMs proposta nesta tese leva em conta apenas a rede de camada 3 na escolha do *data center*, com a hipótese de que a comunicação *inter-data center*, por ser uma rede L2, fornece qualidade de serviço plena.

3.2 Alocação de VMs com Pesquisa Operacional

A Pesquisa Operacional é uma disciplina que agrupa técnicas de modelagem matemática com o objetivo de definir o melhor funcionamento para os sistemas modelados (GOLDBARG; PACCA; LUNA, 2001a). A programação (planejamento) matemático tem o objetivo principal de alocar de forma eficiente recursos limitados e que podem ser disputados por atividades alternativas. Nesse sentido, o seu uso é particularmente útil para auxiliar o processo de alocação de VMs.

Um problema de programação linear (GOLDBARG; PACCA; LUNA, 2001a) consiste em descobrir uma solução que satisfaça restrições para um conjunto de equações ou inequações lineares de forma a otimizar uma função linear. Um modelo de programação linear está na *forma padrão* se (MARINS, 2012): a) a função objetivo é de minimização ou maximização; b)

todas as restrições estão na forma de igualdade; c) todas as variáveis são não-negativas; d) as constantes de cada restrição são não-negativas. Dessa forma, representa-se o modelo matemático como segue:

$$\text{Min (ou Max) } Z = c^T X \tag{3.1}$$

$$\text{s.a: } A X = b \tag{3.2}$$

$$X \geq 0,\ b \geq 0 \tag{3.3}$$

onde $A \in \mathbb{R}^{m \times n}$ é a matriz de coeficientes, $X \in \mathbb{R}^n$ é o vetor das variáveis de decisão, $b \in \mathbb{R}^m$ é o vetor de demandas, e $c \in \mathbb{R}^n$ é o vetor de custos.

Em programação linear, uma *solução básica* (CHINNECKE, 2012b) corresponde a um ponto de interseção das retas descritas como restrições na modelagem, e essa solução pode ser factível ou não, ou seja, corresponder a um ponto que pertença à região viável (factível) do problema, definida a partir das restrições. Uma *solução básica viável* corresponde a um ponto de interseção das retas descritas como restrições na modelagem, e esse ponto encontra-se na região viável do modelo. O algoritmo Simplex (GOLDBARG; PACCA; LUNA, 2001b) é o algoritmo clássico utilizado para obter a solução ótima de um problema de programação linear, e baseia-se na busca de soluções básicas viáveis. Soluções básicas viáveis são aquelas onde todas as variáveis básicas são não-negativas, e estão associadas aos pontos extremos do conjunto de soluções viáveis do modelo em questão (CHINNECKE, 2012b).

Apesar de ser admissível a modelagem de um problema de alocação de VMs com programação linear, um modelo mais próximo da realidade deve considerar que frações de um mesmo recurso de uma VM não devem ser atribuídas a mais de um servidor por vez. Ou seja, requisitos tais como CPU, RAM e espaço para armazenagem de disco devem ser satisfeitos por um único servidor[1]. Essa consideração justifica o uso de variáveis binárias associadas a esses requisitos. No entanto, nem todas as variáveis do modelo precisam ser binárias: as variáveis de decisão relativas aos fluxos dos *links* de rede muito provavelmente assumem valores reais. Essa consideração justifica o uso de um modelo de programação linear inteira mista.

O método *Branch-and-Bound* é utilizado para solucionar problemas que lidam com programação linear inteira (CHINNECKE, 2012b). O método baseia-se na consideração de que a enumeração de soluções inteiras de um problema de programação linear inteira limitado possui uma estrutura de árvore (GOLDBARG; PACCA; LUNA, 2001a). Trata-se, portanto, de um método de enumeração implícita que, apesar de não tratar todas as soluções possíveis, assegura que a solução obtida pelo método seja ótima (KAWAMURA, 2006). A ideia é evitar o percurso de todos os nós da árvore em busca da melhor solução. Ao invés disso, o crescimento ocorre apenas nos nós que possivelmente contenham a melhor solução.

Referente à estratégia proposta, a modelagem matemática é baseada em um problema de atribuição, onde as VMs devem ser atribuídas a servidores. Na Fig. 3.2, essa atribuição deve ser feita de modo a:

1. Respeitar as capacidades de cada servidor;

2. Respeitar as demandas de cada VM.

[1]Nota: não é considerado o uso de sistemas de arquivos compartilhados, o que ampliaria a capacidade de armazenamento do ambiente virtualizado.

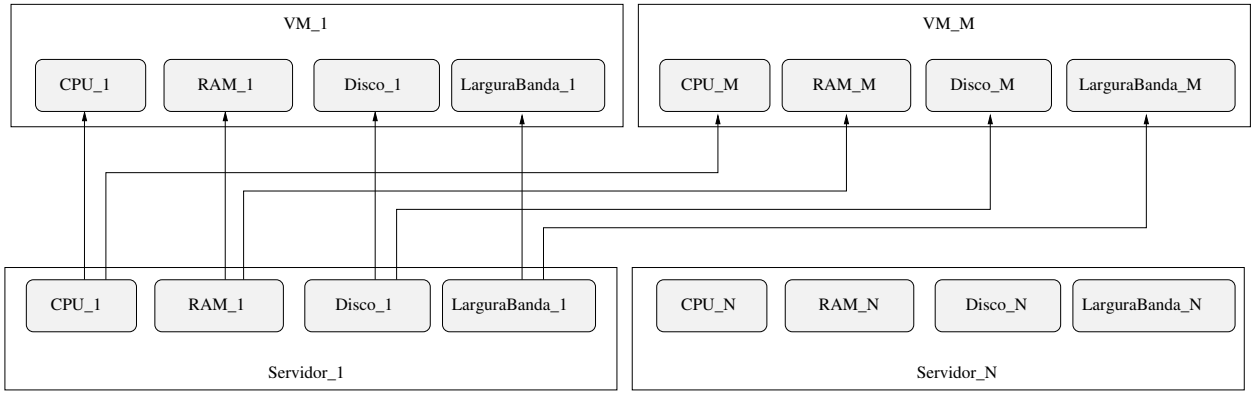

Figura 3.2: Problema de Atribuição Referente à Alocação de VMs em Servidores.

No exemplo da Fig. 3.2, cada servidor oferece recursos para suprir a demanda de uma ou mais VMs. Porém, nem todos os servidores precisam ser alocados porque cada VM precisa de um único servidor que satisfaça todos os seus requisitos. As seguintes considerações são feitas para esse modelo:

1. Um servidor é capaz de alocar nenhuma, uma ou mais VMs;
2. Uma VM é alocada e instanciada em apenas um servidor;
3. VMs podem ser alocadas em quaisquer servidores da nuvem;
4. Um servidor é capaz de alocar tantas VMs quanto puder, desde que respeite todos os requisitos de CPU, RAM, espaço de armazenamento em disco e largura de banda requisitada pela VM;
5. O sistema não promove realocações (migração de VMs) porque considera que a alocação seja ótima de acordo com a função objetivo do modelo.

A modelagem do problema de alocação de VMs a servidores de nuvem é feita em três etapas:

- Modelagem dos requisitos das VMs e capacidades dos servidores;
- Modelagem da rede entre os *data centers*;
- Modelagem do consumo de energia da rede e dos servidores.

3.2.1 Modelagem dos Requisitos das VMs e Capacidades dos Servidores

A modelagem dos requisitos das VMs e capacidades dos servidores é feita como segue[2]:

- Dados de entrada:
  - VM_i^{cpu}: requisito de CPU da $VM_i \mid i = 1 \ldots N$;
  - VM_i^{bw}: requisito de largura de banda para a $VM_i \mid i = 1 \ldots N$.

[2]Nota: Valores nos intervalos pertencem ao conjunto dos números naturais, isto é, $x \in \mathbb{N}$.

Interpretação: Os dados de entrada são referentes aos requisitos de CPU e largura de banda que cada VM solicita para ser alocada em um servidor. Esses requisitos devem ser satisfeitos em conjunto para que a alocação seja realizada. Essa modelagem também admite a inclusão de dados de entrada para outros requisitos das VMs, tais como: memória RAM, espaço disponível em disco, tipo de virtualização, e outros. Porém, a modelagem aqui descrita será restrita aos requisitos de CPU e largura de banda.

- Variáveis de decisão:
  - $a_{i,j} = 1$, caso a VM i seja alocada ao servidor j;
  - $a_{i,j} = 0$, caso contrário.

  Interpretação: As variáveis de decisão binárias indicam se a alocação da VM_i foi realizada, ou não, em um determinado servidor j.

- Seja S_j o servidor de índice j. A modelagem considera que os servidores são agregados em *data centers*. É definido $S_j \in D_k$ quando o servidor S_j está instalado no *data center* D_k. A representação das capacidades dos servidores é definida como segue:
  - S_j^{cpu}, quantidade de CPUs presentes no servidor j;
  - S_j^{bw}, capacidade (banda) da interface do servidor j.

  Interpretação: Cada servidor S_j possui capacidades para atender aos requisitos das VMs, referentes a CPU e largura de banda.

- Restrições das variáveis de decisão:

$$\sum_{i=1}^{N} a_{i,j} = 1 \mid j = 1 \ldots M \tag{3.4}$$

  Interpretação: a alocação da VM_i no servidor S_j deve ser única, isto é, apenas uma alocação de uma mesma VM_i é admitida no conjunto de M servidores.

- Restrições de Limite de Capacidade:

$$\sum_{i=1}^{N} a_{i,j}.VM_i^{cpu} \leq S_j^{cpu} \mid j = 1 \ldots M \tag{3.5}$$

$$\sum_{i=1}^{N} a_{i,j}.VM_i^{bw} \leq S_j^{bw} \mid j = 1 \ldots M \tag{3.6}$$

  Interpretação: cada alocação possui requisitos que devem ser satisfeitos em conjunto por um servidor até o limite de sua capacidade.

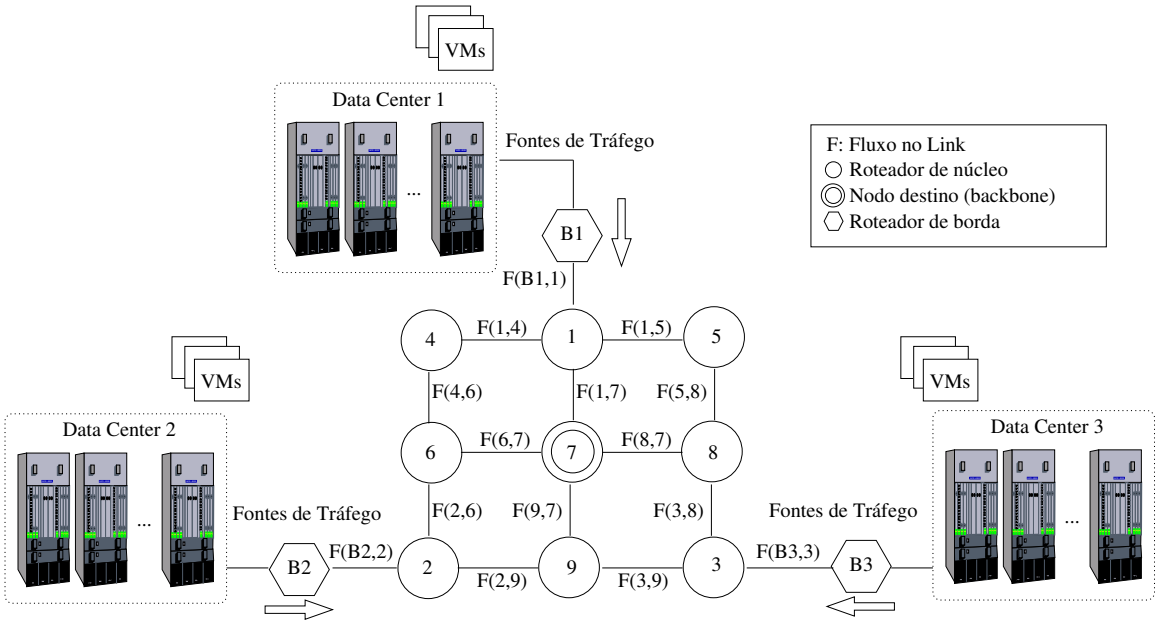

Figura 3.3: Exemplo de Modelagem da Rede de Comunicação entre *Data Centers*.

3.2.2 Modelagem da Rede entre os *Data Centers*

A modelagem da rede entre os *data centers* é feita como ilustrado na Fig. 3.3. Nessa modelagem, a rede é representada por um conjunto de nós e por um conjunto de arestas que representam os *links* por onde o tráfego das VMs pode ser encaminhado. As restrições referentes à rede são modeladas como segue:

- Restrição dos fluxos referentes às fontes de tráfego com origem nos *data centers*, e seus respectivos nós de destino (*sink nodes*):

$$\sum_{p} F_p^{src} - \sum_{q} F_q^{sink} = 0 \tag{3.7}$$

  Interpretação: Os fluxos do tráfego das VMs que saem dos *data centers* devem ser iguais aos fluxos que chegam em cada um dos seus respectivos nós destinos. A Fig. 3.4 ilustra a interpretação gráfica dessa restrição. Nessa figura, o tráfego de cada *data center* (*source*) deve ser consumido por um ou mais nós destino (*sink*) da topologia.

- Restrição do fluxo total gerado pelas VMs:

$$\sum_{j} \sum_{i=1}^{N} a_{i,j}.VM_i^{bw} - \sum_{k} F_k^{src} = 0 \mid S_j \in D_k \tag{3.8}$$

  Interpretação: Essa restrição tem o intuito de associar as alocações com os fluxos de rede. Os fluxos que saem de cada servidor j são o resultado do somatório do fluxo de todas as suas VMs. Portanto, a diferença entre o fluxo de todas as VMs de um *data center*, e o fluxo que sai do *data center*, deve ser igual a zero. A Fig. 3.5 ilustra a interpretação gráfica dessa restrição. Nessa figura, o fluxo total gerado pelas VMs alocadas no respectivo *data center* deve ser igual à quantidade de fluxo que sai do *data center*.

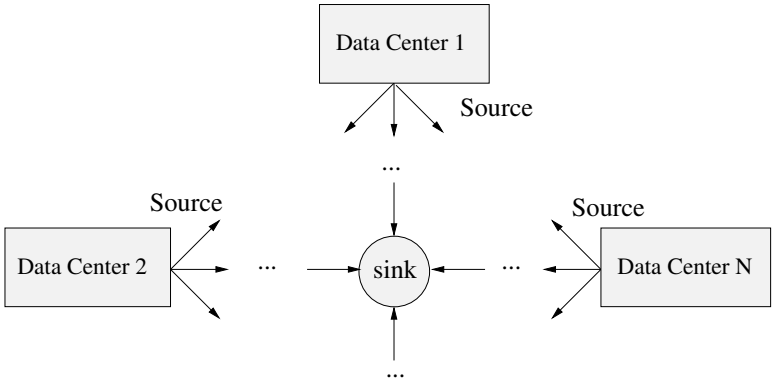

Figura 3.4: Interpretação da Restrição dos Fluxos Gerados pelas Fontes de Tráfego.

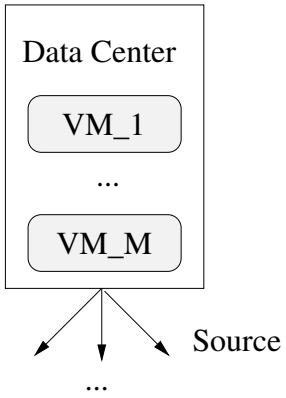

Figura 3.5: Interpretação da Restrição do Fluxo Total Gerado pelas VMs.

- Restrição de continuidade dos fluxos em nós intermediários:

$$\sum_{p} F_{p,c} - \sum_{q} F_{c,q} = 0, \ \forall p, q \tag{3.9}$$

Interpretação: Os fluxos que chegam a um nó intermediário são recebidos por suas interfaces de entrada. O somatório desses fluxos deve ser igual ao somatório dos fluxos de suas interfaces de saída para que o tráfego seja encaminhado ao longo da rede. A Fig. 3.6 ilustra a interpretação gráfica dessa restrição.

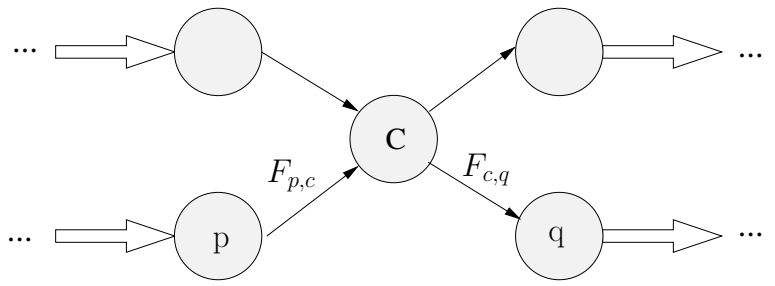

Figura 3.6: Interpretação da Restrição de Continuidade dos Fluxos.

- Também, para cada nó *sink* q:

$$\sum_{p} F_{p,q} - F_{q}^{sink} = 0, \forall q \tag{3.10}$$

- Restrição de capacidade dos *links* de rede:

$$\sum F_{p,q} \leq L_{p,q}^{bw}, \forall p, q \tag{3.11}$$

Interpretação: os fluxos não podem exceder a capacidade dos *links* que os transportam. A Fig. 3.7 ilustra a interpretação gráfica dessa restrição. O nó p gera um tráfego de pacotes com taxa de transferência de $xMbits/s$, e o nó q recebe esse tráfego na mesma taxa de transferência da origem. A capacidade das interfaces de envio e recepção desses nós garantem que o tráfego seja recebido sem perdas no trajeto fim-a-fim, com a consideração de que o enlace físico seja capaz de acomodar o tráfego entre os *end-points*.

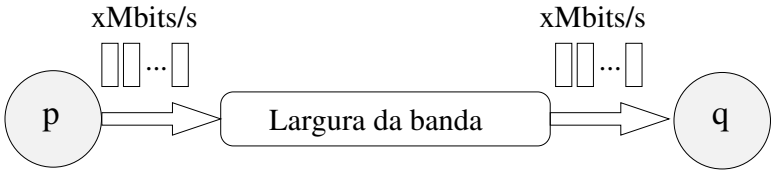

Figura 3.7: Interpretação da Restrição de Capacidade dos *Links* de Rede.

3.2.3 Modelagem do Consumo de Energia da Rede e dos Servidores

Em um *data center*, o consumo de energia dos servidores é tipicamente determinado por CPU, memória, armazenagem de dados, suprimentos de força, sistemas de ventilação, e quaisquer outros itens que estejam acoplados ao *hardware* desses servidores. Segundo (BELOGLAZOV; ABAWAJY; BUYYA, 2012), estudos recentes mostram que o consumo de energia desses servidores pode ser descrito, sem perda de generalidade, como uma função linear entre o consumo e a utilização da CPU, mesmo se mecanismos como DVFS (*Dynamic Voltage and Frequency Scaling*) forem aplicados.

A definição de métricas precisas é difícil de se obter em virtude dos muitos fatores que afetam o consumo, principalmente a arquitetura do *hardware* e os diversos componentes agregados a ela. Beloglasov et al. (BELOGLAZOV; ABAWAJY; BUYYA, 2012) apontam que, em média, um servidor ocioso consome aproximadamente 70% da energia relativa ao uso total da CPU. Isso justifica métodos de *sleep mode* para reduzir o consumo total de energia. Beloglasov et al. utilizam a seguinte modelagem linear para estimar o consumo de energia da CPU: $P(u) = k.P_{max} + (1-k).P_{max}.u$, onde k é a fração de energia consumida com o servidor ocioso, ou seja, 70% do consumo total de CPU. P_{max} é uma estimativa do consumo máximo. A variável u é a taxa de utilização da CPU. Para a estimativa do consumo total em um determinado período, utiliza-se uma soma contínua, dada pela seguinte fórmula: $\int_{t0}^{tn} P(u(t))dt$, onde t representa o instante de coleta dos dados.

Essa modelagem simplificada não aborda a arquitetura do sistema, e não considera o número de processadores lógicos comumente requeridos para a instanciação de VMs. Medições mais precisas são obtidas, por exemplo, com o uso de *benchmarks*, como os oferecidos pela SPECpower (SPECPOWER, 2009). Uma métrica geralmente utilizada é o TDP (*Thermal Design Power*), que indica o quanto de energia será necessária para que o sistema de resfriamento

dissipe o calor do processador para que este não exceda sua temperatura máxima. O consumo de energia irá depender da arquitetura mas, geralmente, um processador que produza mais calor irá exigir um consumo maior de energia. TDP é utilizado por fabricantes de processadores para indicar o quanto de calor pode ser dissipado pelo sistema de resfriamento. Por exemplo, um *host* com 30 watts TDP significa que ele pode dissipar, por meio do seu sistema de resfriamento, 30 watts de calor sem exceder a temperatura máxima do processador (CPU WORLD, 2012).

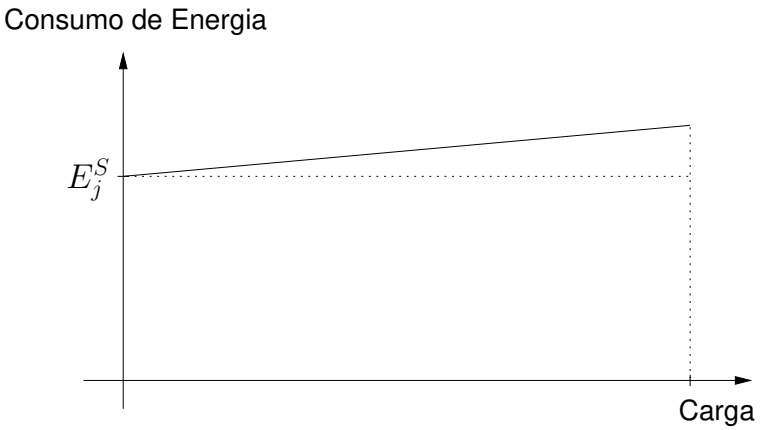

Figura 3.8: Modelagem do Consumo de Energia de Servidores e Roteadores.

Para o modelo linear em questão, é considerado um valor constante para o consumo de energia. A Fig. 3.8 ilustra que ocorre um pequeno aumento linear no consumo de energia à medida que a carga de processamento aumenta. No entanto, para a modelagem simplificada de consumo de energia proposta, é considerado que esse consumo seja constante, ou seja, a partir do momento em que ocorre instanciação de VMs no servidor, este é ligado, e há o consumo inicial de energia E_j^s. A restrição do modelo linear referente ao consumo de energia dos servidores é definida como segue:

$$\sum_{i=1}^{N} a_{i,j} - K_s.x_j \leq 0 \mid j = 1 \ldots M \tag{3.12}$$

Interpretação: Essa restrição é referente ao consumo de energia dos servidores no *data center*. Servidores que não recebem alocações podem ser desligados, ou entrar em estado de baixo consumo de energia. Nessa restrição, K_s é uma constante com valor maior ou igual ao número de alocações. Essa constante tem o objetivo de balancear o valor do somatório das alocações ($\sum_{i=1}^{N} a_{i,j}$), de forma a manter o valor da restrição menor ou igual a 0. A variável x_j recebe o valor 1 caso exista alguma instância de VM alocada no servidor j, o que significa que é necessário ligar o servidor para prover os recursos para a alocação. Isso também significa que existe o consumo de energia nesse servidor. Por outro lado, caso não haja alocação no servidor j, então x_j recebe o valor 0, ou seja, o servidor deve permanecer desligado. Para a modelagem, a variável x_j é definida como variável binária. A Fig. 3.9 ilustra a interpretação gráfica dessa restrição. Nessa figura, apenas os servidores 1 e 3 devem ser ligados para realizar as alocações da VM 1 no servidor 1 (alocação $a_{1,1}$), da VM 2 no servidor 3 (alocação $a_{2,3}$), e da VM 3 no servidor 3 (alocação $a_{3,3}$).

A modelagem linear referente ao consumo de energia dos roteadores que participam da comunicação entre os *data centers* é feita de forma semelhante. Adelin et

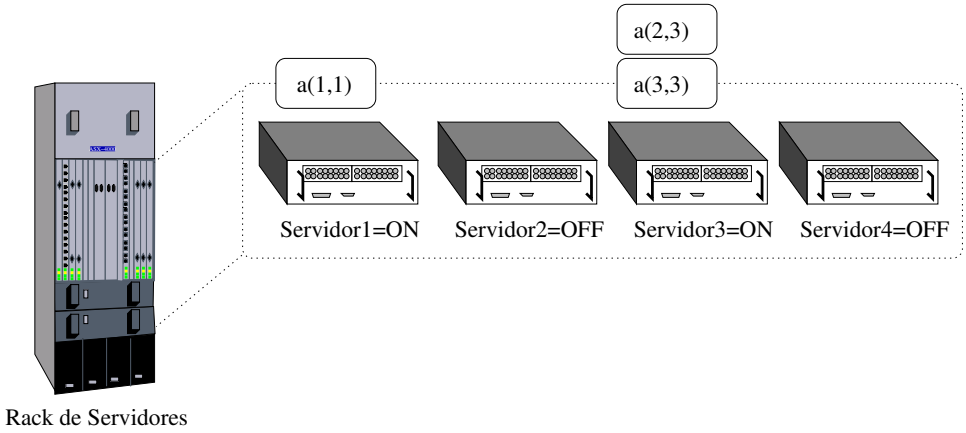

Figura 3.9: Interpretação da Restrição referente ao Consumo de Energia dos Servidores.

al. (ADELIN; OWEZARSKI; GAYRAUD, 2010) identificam que o consumo de energia de um roteador típico para grandes volumes de tráfego, com interfaces de rede Ethernet gigabit, tem um aumento de consumo próximo ao linear à medida que o tráfego aumenta. No entanto, para a modelagem simplificada do consumo de energia proposta, é considerado que esse consumo seja constante, ou seja, a partir do momento em que exista tráfego em quaisquer interfaces do roteador, este é ligado, e ocorre o consumo inicial de energia E_c^R. A restrição do modelo MILP referente ao consumo de energia dos roteadores é definida como segue:

$$\sum_{p} F_{p,c} - K_c.y_c \leq 0, \ \forall c \tag{3.13}$$

Interpretação: Essa restrição é referente ao consumo de energia dos roteadores entre os *data centers*. Roteadores que não recebem alocações podem ser desligados ou entrar em estado de baixo consumo de energia. Nessa restrição, K_c é uma constante maior do que a capacidade máxima de tráfego que cada roteador suporta ($K_c > \sum_p L_{p,c}^{bw}$). Essa constante tem o objetivo de balancear o somatório dos fluxos de entrada no roteador ($\sum_p F_{p,c}$), de forma a manter o valor da restrição menor ou igual a 0. A variável y_c recebe o valor 1 caso exista tráfego no roteador c, o que significa que é necessário ligar o roteador para prover o encaminhamento do tráfego. Isso também significa que existe consumo de energia nesse roteador. Por outro lado, caso não haja tráfego no roteador c, então y_c recebe o valor 0, ou seja, o roteador deve permanecer desligado. Para a modelagem, a variável y_c é definida como variável binária. A Fig. 3.10 ilustra a interpretação gráfica dessa restrição. Nessa figura, os nós representam os roteadores de núcleo entre os *data centers* e as arestas representam os *links* entre os roteadores. As linhas tracejadas indicam a rota do tráfego e apenas os roteadores que participam dessa rota devem permanecer ligados.

Finalmente, a função objetivo é definida como segue:

$$Min(E^{tot}) = \sum_{j=1}^{M} x_j.E_j^S + \sum_{c=1}^{R} y_c.E_c^R \tag{3.14}$$

Interpretação: A função objetivo busca minimizar o consumo de energia total dos servidores e roteadores (E^{tot}), com a intenção de reduzir a quantidade de recursos necessários para prover as alocações de VMs. No que se refere ao consumo de energia de servidores, a primeira parte da função objetivo apresenta E_j^S como o consumo de energia do servidor j, e a variável x_j

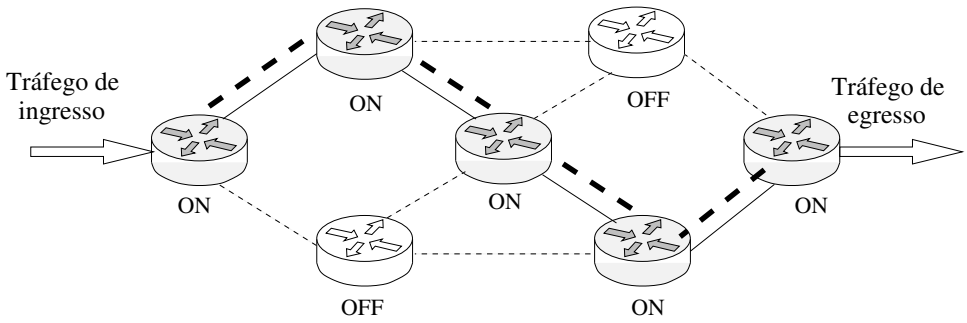

Figura 3.10: Interpretação da Restrição referente ao Consumo de Energia dos Roteadores.

como indicativo se esta quantidade de energia é consumida (servidor ligado) ou não (servidor desligado). No que diz respeito ao consumo de energia dos roteadores, a segunda parte da função objetivo apresenta E_c como o consumo de energia do roteador c, e a variável y_c como indicativo se esta quantidade de energia é consumida (roteador ligado) ou não (roteador desligado).

Portanto, o problema de minimização é um problema de programação linear inteira (binária) mista. A solução deste problema fornece o número mínimo de servidores e roteadores necessários para a alocação de VMs sem violar a capacidade dos *links* da rede. A solução tende a atribuir VMs nos servidores mais eficientes, além de minimizar a quantidade de roteadores para encaminhar o tráfego na rede.

3.3 Considerações Finais

Este capítulo abordou a proposta de alocação de VMs com o uso de MILP. Esse modelo contempla a descrição dos requisitos das VMs, capacidades dos servidores, e também a modelagem da rede de roteadores entre os *data center*. A modelagem da rede tem o objetivo de encaminhar o tráfego pelos *links* que geram menor consumo para as requisições de alocação de VMs. No entanto, alocações inteiras com o uso de variáveis binárias oneram o custo computacional para se obter a solução ótima, o que justifica o uso de técnicas de relaxamento linear, e o uso de heurísticas complementares. Deve-se observar que o modelo proposto não trata a rede interna (L2) dos *data centers*.

Capítulo 4

Modelagem Híbrida para Alocação de VMs

Este capítulo apresenta a proposta para alocação de VMs que leva em conta parâmetros de qualidade de serviço da rede de comunicação. A introdução de tais parâmetros na modelagem linear é inviável dada a dificuldade de estimá-los com base em expressões lineares apenas. De fato, a determinação de parâmetros de QoS de forma realista requer simuladores de rede que implementam os mesmos protocolos e técnicas de comutação presentes nos roteadores. A proposta desta tese consiste em empregar uma técnica de busca capaz de combinar o modelo linear de alocação de VMs apresentada no capítulo anterior com simuladores de rede capazes de estimar o impacto da alocação na rede de comunicação em termos de parâmetros de QoS. Esta estratégia de busca é realizada com um algoritmo genético capaz de evoluir um conjunto inicial de soluções otimizando concomitantemente o consumo de energia na nuvem e a qualidade de serviço na rede de comunicação. Estratégias de solução de problemas que empregam conjuntamente métodos exatos e métodos derivados da Computação Natural como algoritmos genéticos são denominadas Estratégias Híbridas.

4.1 Computação Natural

A Computação Natural é um conjunto de metodologias inspirada nos mecanismos naturais, que se baseia na aquisição de ideias da natureza para o desenvolvimento de sistemas artificiais. Também é uma terminologia empregada para se referir a: ferramentas computacionais (e seus respectivos modelos matemáticos e teóricos) baseadas em fenômenos naturais; modelos computacionais para simular processos naturais; e paradigmas emergentes para processar e armazenar informação (CASTRO, 2001).

A natureza é a principal fonte de inspiração para o desenvolvimento de técnicas de otimização bio-inspiradas. A grande diversidade de seres e as formas como eles se adaptaram às adversidades do meio onde vivem, ao longo de muitos anos, são fatores que motivam o estudo da evolução natural dos seres vivos. Dentre os muitos ramos de pesquisa da Computação Natural, podem ser enumerados:

- *Simulated Annealing*: É um método de otimização que simula o comportamento natural de recozimento (*annealing*) de sistemas físicos (SARAMAGO, 2003). Como exemplo, estruturas cristalinas de um metal a ser fundido podem ser agrupadas em um estado mais

ordenado pelo uso do processo de recozimento, que submete o sistema a uma alta temperatura e o resfria lentamente. Caso o resfriamento ocorra de forma brusca, um produto de qualidade inferior será obtido. Porém, caso o resfriamento ocorra de forma lenta e controlada, o próprio sistema buscará obter um ponto de equilíbrio, resultando em uma estrutura mais estável e de boa qualidade. A literatura destaca os trabalhos pioneiros descritos em (KIRKPATRICK; GELATT; VECCHI, 1983) e (METROPOLIS et al., 1953);

- Lógica Nebulosa (Lógica *Fuzzy*): a motivação da lógica nebulosa é o tratamento de problemas onde a classificação das soluções apresenta um determinado grau de incerteza (AGUADO; CANTANHEDE, 2010). Um exemplo clássico é o do copo que está preenchido com um líquido até a metade. A pergunta que se faz é: o copo está cheio ou vazio? Diante dessa incerteza, é possível afirmar que o copo está meio cheio e meio vazio. Através dessas afirmações incertas (variáveis linguísticas) foram geradas escalas de valores que representam estados sem fronteiras bem definidas no problema. Dessa forma, o grau de incerteza das variáveis linguísticas pode ser traduzido para um modelo matemático. Na lógica *Fuzzy* uma função de pertinência faz um mapeamento dos possíveis valores numéricos associados às variáveis linguísticas, de forma a ser possível gerar conclusões válidas;

- Redes Neurais: as redes neurais artificiais são compostas de unidades de processamento que simulam os neurônios do cérebro humano. Essas unidades recebem e processam operações simples e encaminham o resultado desse processamento para os neurônios com os quais estão conectados. Uma extensa literatura é descrita em (ZUBEN, 2006);

- Computação Evolucionária: a teoria darwiniana (DARWIN, 2003) oferece uma explicação para a grande diversidade de seres biológicos. Em ambientes onde os indivíduos compartilham recursos reduzidos de alimento e espaço físico, por exemplo, os indivíduos mais aptos possuem maiores chances de sobreviverem e se reproduzirem. Diz-se que a seleção natural ocorre de forma a manter um equilíbrio populacional, ao mesmo tempo em que elege os indivíduos mais bem adaptados ao meio onde vivem, a fim de perpetuarem a sua espécie. Baseada em conceitos darwinianos e na genética moderna, a computação evolutiva é uma abordagem que pode ser considerada quando métodos computacionais fortes (soluções clássicas) ou específicos não forem adequados para a resolução de determinado tipo de problema, seja por não existir uma solução com esses métodos, por não serem aplicáveis em um tempo computacional razoável, ou simplesmente por falharem na resolução de uma determinada classe de problemas (CASTRO; ZUBEN; ATTUX, 2009). Uma das grandes vantagens de se resolver um problema com computação evolutiva está na não-obrigatoriedade de se indicar explicitamente os passos para se atingir o objetivo esperado. No entanto, o processo de busca deve ser utilizado sobre um conjunto de informações, ou seja, sobre um espaço de busca que possivelmente contenha a solução. A solução ótima poderá ser obtida a partir desse conjunto de informações, o qual contém um conjunto de soluções candidatas. Além disso, nenhuma, uma ou mais soluções ótimas podem ser obtidas no mesmo espaço de busca (CASTRO; ZUBEN; ATTUX, 2009).

Especificamente em relação à computação evolutiva, um problema é definido como um conjunto de informações a partir das quais se deseja extrair alguma informação. A resolução de um problema corresponde à tomada de decisões que melhorem o desempenho de um conjunto de soluções candidatas. Em termos computacionais, um algoritmo evolutivo corresponde a um

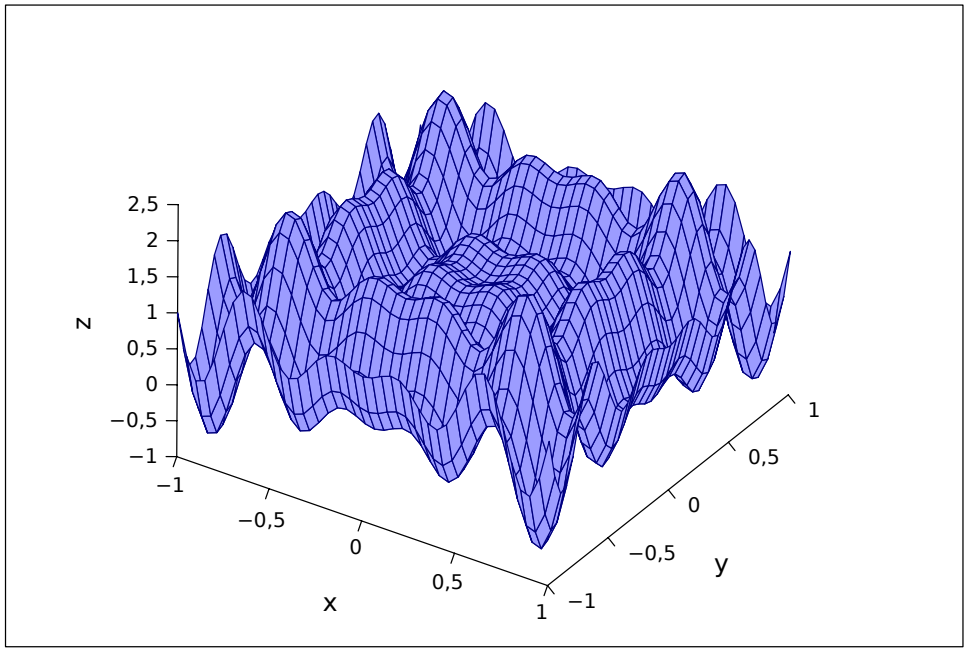

Figura 4.1: Exemplo de Superfície de *Fitness*.

processo iterativo de busca em uma superfície de adaptação, ou superfície de *fitness*, onde os extremos representam as melhores soluções candidatas no espaço de busca. Como exemplo, a Fig. 4.1 ilustra a superfície de *fitness* da função $g(x) = x.sen(4.\pi.x) - y.sen(4.\pi.y + \pi) + 1 \mid x \in [-1, 1], y \in [-1, 1]$. Para o processo de busca, são definidas as seguintes informações (CASTRO; ZUBEN; ATTUX, 2009):

- a representação das soluções candidatas (por exemplo, definir que os dados das possíveis soluções serão armazenados em cadeias binárias de comprimento $l \geq 1 \mid l \in \mathbb{N}^*$);
- a definição de um objetivo, por exemplo, maximizar a função que define a superfície de *fitness*;
- a definição da função de avaliação (ou função de *fitness*), que indica a qualidade relativa das soluções obtidas até o momento (por exemplo, uma função matemática que atue sobre $g(x)$ e retorne um valor qualitativo para cada uma das soluções candidatas do espaço de busca).

Os algoritmos genéticos (AGs) clássicos foram originalmente concebidos por John Holland (EIBEN; SMITH, 2007), com o intuito de estudar comportamentos naturais adaptativos. As principais características desses algoritmos são enumeradas a seguir:

- A busca por soluções ocorre sobre uma população de indivíduos;
- Os indivíduos são representados em estruturas binárias de tamanho fixo;
- O processo de seleção é estocástico e proporcional ao *fitness*;
- O processo de reprodução é simples, isto é, o operador de recombinação realiza as permutações de bits de uma cadeia para outra a partir de um ponto escolhido de uma das

cadeias. Esse processo é probabilístico, uma vez que as cadeias são escolhidas duas a duas de acordo com o seu *fitness*;

- O processo de mutação é pontual, isto é, o operador de mutação possui uma probabilidade, originalmente baixa, para escolher um bit em uma cadeia e substituir esse bit por outro. Assim, caso o valor do bit seja 0, seu valor é trocado para 1, e vice-versa.

Outros exemplos de computação evolutiva são: Programação Evolutiva, desenvolvida por Fogel et al. (EIBEN; SMITH, 2007); Estratégias Evolutivas, propostas por Rechenberg, Schwefel e Bienert (EIBEN; SMITH, 2007); e Programação Genética, onde destacam-se os trabalhos de Koza et al. (EIBEN; SMITH, 2007).

4.2 Estratégia Híbrida para Alocação de VMs

Diferentes alocações produzidas pelo modelo linear podem ser obtidas por meio da variação dos enlaces que compõem a rede de comunicação. A alocação de VMs é diretamente afetada pelas capacidades nos enlaces, posto que o tráfego produzido pela alocação deve ser capaz de fluir por tais enlaces. Portanto, uma solução de alocação pode ser representada por um vetor de pesos W, onde $W_{p,q} > 0$ altera a capacidade do enlace $p - q$ para:

$$L^{bw}_{p,q} = W_{p,q}.L^{bw}_{p,q},\ \forall p, q$$

Na terminologia de algoritmos genéticos (EIBEN; SMITH, 2007), aplicada ao problema em questão, o conjunto dos pesos $W_{p,q}$ atribuídos aos *links* determinam o fenótipo, ou seja, formam o conjunto das características observáveis no objeto resultante do processo de decodificação dos genes. O genótipo, por sua vez, é codificado nas estruturas dos cromossomos. No genótipo são caracterizados os indivíduos da população, representados por cromossomos. Cada posição na estrutura do cromossomo é denominada *locus*. O alelo é o valor que está presente em um *locus* de um cromossomo em um dado momento. No estudo em questão, os valores possíveis são 0 e 1. A Fig. 4.2 mostra o mapeamento do indivíduo (pesos dos *links*) no cromossomo (cadeia de bits que determina o peso).

A Fig. 4.3 mostra o diagrama geral da estratégia híbrida proposta. O processo inicia com o fornecimento da quantidade de *data centers*, capacidades dos servidores, requisitos das VMs e topologia da rede. Esses dados são fornecidos uma única vez ao AG, a partir de uma extensão ao formato BRITE (MEDINA et al., 2001), que é um formato textual para a representação de topologias físicas de redes de larga escala (acima de 100.000 nós). No passo 1, o *Componente Algoritmo Genético (AG)* inicia o processo evolutivo, com o mapeamento de diferentes capacidades de *links* em cromossomos, os quais são mapeados para o modelo MILP. No passo 2, o *Componente MILP* processa o modelo MILP e retorna o resultado obtido pelo *solver* de programação linear inteira. Esse resultado é referente ao consumo de energia dos servidores e roteadores, e é mapeado nos cromossomos do AG, como parte do valor da função de *fitness*. Os cromossomos com combinações de capacidades de *links* não factíveis recebem alto valor de *fitness*, de forma a terem maior probabilidade de serem substituídos nas próximas gerações do AG. No passo 3, o *Componente Simulador de Rede* recebe os dados dos cromossomos com soluções factíveis, e traduz para o modelo de rede as capacidades dos *links* de rede do cromossomo, e a estimativa do tráfego agregado das VMs obtido com o resultado das alocações do modelo em MILP. No passo 4, o simulador de rede mapeia métricas de rede, tais como perda de pacotes

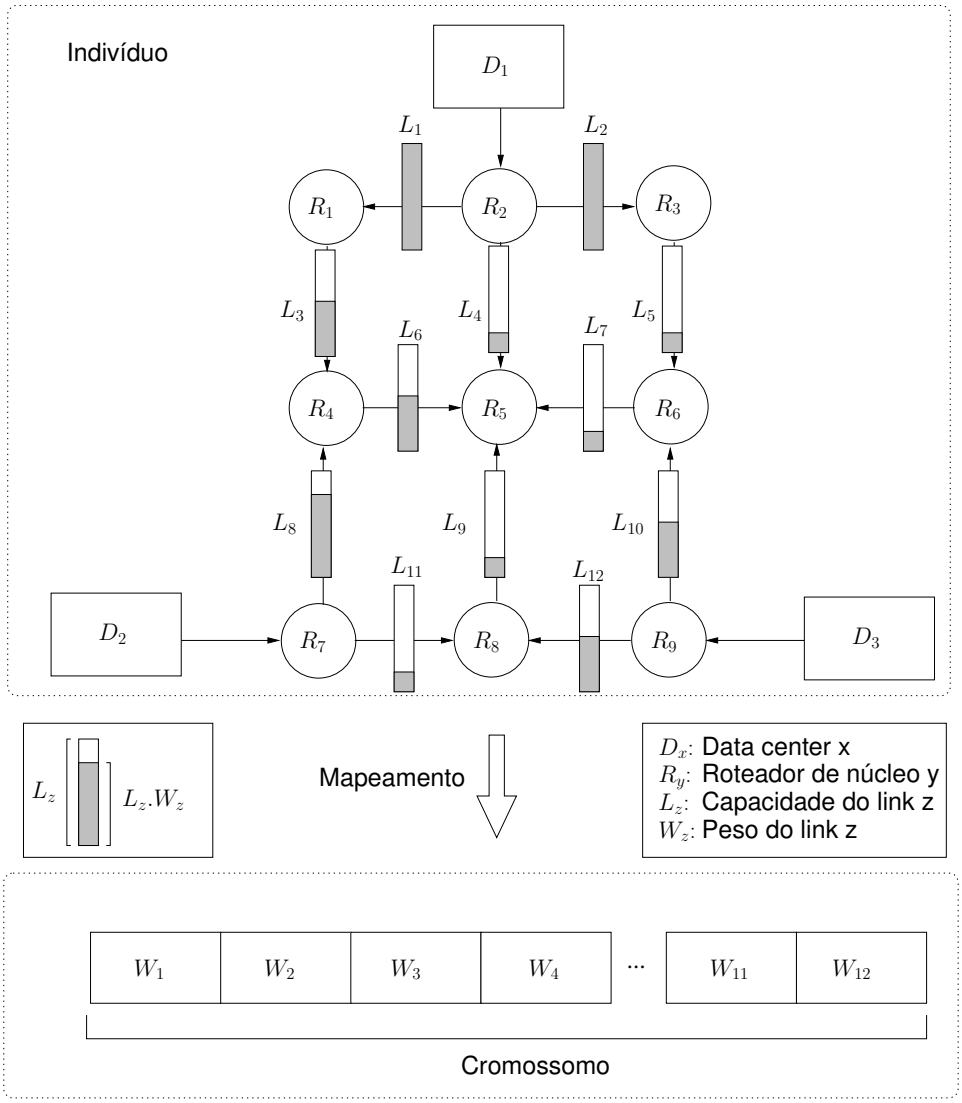

Figura 4.2: Mapeamento do Indivíduo no Cromossomo.

e atraso fim-a-fim, no valor do *fitness* de cada cromossomo. Portanto, o *fitness* de cada cromossomo é o resultado da combinação de métricas de consumo de energia (*solver* para MILP) e de métricas de rede (simulador rede). No passo 5, o cromossomo com o menor valor de *fitness* é eleito como a melhor solução obtida no final do processo evolutivo. No diagrama apresentado, o *Componente Computação de Rotas* é definido para o processo de pós-otimização para readequar o resultado do AG aos requisitos de rede de cada provedor, por exemplo, roteamento baseado em restrições e balanceamento de carga na rede.

O algoritmo 1 mostra o pseudocódigo do AG utilizado na estratégia híbrida proposta. Os principais métodos do algoritmo são descritos a seguir:

- *recuperarTopologia(documentoRequisitos):* método para recuperar as informações do *data center* referentes às capacidades dos servidores, requisitos das VMs, e topologia da rede;
- *inicializarPopulacaoCapLinks():* método para inicializar a população de cromossomos com capacidades de *links* inteiras e não-ordenadas. Essas capacidades são codificadas em sequências binárias nos cromossomos;

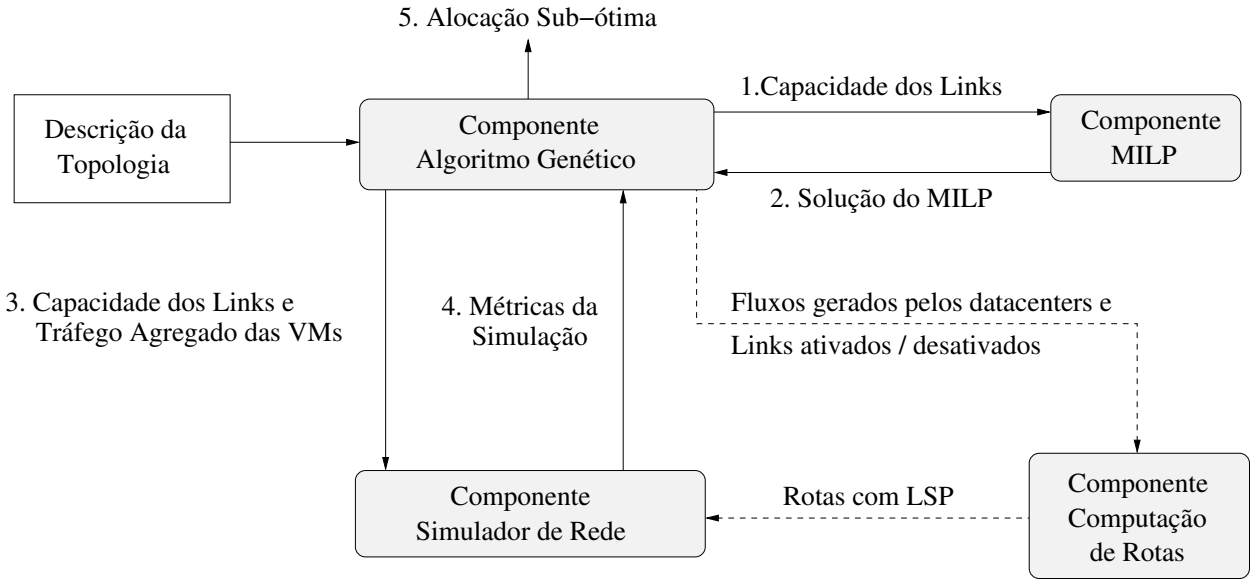

Figura 4.3: Diagrama da Estratégia Híbrida para Alocação de VMs.

Algoritmo 1: Algoritmo de Alocação da Proposta Híbrida.

1: $recuperarTopologia(documentoRequisitos)$
2: $P \Leftarrow inicializarPopulacaoCapLinks()$
3: $P \Leftarrow avaliar(P)$
4: **while** $(iteracao < numIteracoes) AND (!evolucaoFitness)$ **do**
5: $\quad P2 \Leftarrow selecionar(P)$
6: $\quad P2 \Leftarrow recombinar(P2)$
7: $\quad P2 \Leftarrow variar(P2)$
8: $\quad P2 \Leftarrow avaliar(P2)$
9: $\quad P \Leftarrow renovar(P2)$
10: $\quad iteracao = iteracao + 1$
11: **end while**

- *selecionar():* método de seleção de cromossomos. A partir da população original de tamanho P, são selecionados $P2 = \frac{P}{2}$ cromossomos por meio do algoritmo de roleta, ou seja, a seleção é estocástica e proporcional ao *fitness*. A roleta é gerada somando-se os valores de *fitness* de cada cromossomo até o momento, de forma a se ter uma escala que atinja o maior valor de *fitness*. Na verdade, a roleta mantém uma lista de valores que representam as probabilidades de cada indivíduo em relação aos demais, de acordo com a equação 4.1.

$$p_i = \frac{f_i}{\sum f_i} \tag{4.1}$$

Interpretação: a probabilidade p_i do cromossomo i ser selecionado é definida pelo valor do *fitness* desse cromossomo (f_i) em relação ao somatório dos *fitness* de todos os cromossomos ($\sum f_i$). Essa relação implica que a soma de todas as probabilidades parciais p_i seja igual a 1, ou seja, há uma escala de valores para a seleção de cada cromossomo de acordo com o seu *fitness*. No entanto, por se tratar de um processo de minimização, utiliza-se uma

roleta invertida, sendo que os indivíduos com menor valor de *fitness* tendem a ter uma maior probabilidade de serem selecionados;

- *recombinar():* método para recombinar cromossomos. O processo de *crossover* foi definido como pontual, com a escolha de um número aleatório na escala de $[0, l-1]$, onde l representa o comprimento da codificação, e então a troca de ambos os pais neste ponto, e criação de dois filhos pela troca das caudas, como mostra a Fig. 4.4;

- *variar():* método para variar os bits nos cromossomos selecionados. Esse processo é feito com mutação, sendo o operador de mutação pontual, com uma probabilidade intrinsicamente baixa de $\frac{1}{P}$ para substituir um bit de uma cadeia por outro bit. A motivação é adicionar pequenas perturbações no espaço de busca de forma a encontrar soluções alternativas, e escapar de mínimos locais. A taxa de mutação é mantida baixa porque o seu aumento pode vir a gerar uma alta variação do *fitness* dos indivíduos, o que não é desejado;

- *avaliar():* método para atribuir o valor do *fitness* dos cromossomos. O processo de avaliação consiste na geração e execução do modelo em MILP; a seguir, ocorre a geração e execução do modelo de rede no simulador de rede. Após a execução de ambos, o valor do *fitness* é atribuído. O valor do *fitness* para cada cromossomo é calculado como segue:

$$fitness = \alpha.(\sum_{j=1}^{M} E_j^S + \sum_{c=1}^{R} E_c^R) + \beta.(\sum_{i=1}^{D} \%perdaPacotes) \tag{4.2}$$

  Interpretação: O valor do *fitness* é o resultado do consumo de energia dos servidores (E_j^S), do consumo de energia dos roteadores (E_c^R), e da porcentagem total de perda de pacotes nos *links* da rede interna entre os *data centers*. Outras métricas de QoS como atraso fim-a-fim e balanceamento de carga podem ser igualmente incorporadas na função de *fitness*. Os valores de α e β são atribuídos para ponderar equitativamente o consumo de energia e a perda de pacotes;

- *renovar():* método para renovar a população P da iteração (época) atual. O processo de renovação da população é elitista, com os cromossomos com maiores *fitness* em P sendo substituídos pelos indivíduos com menores *fitness* em $P2$, por se tratar de um problema de minimização. A intenção é preservar os melhores indivíduos para aumentar a probabilidade de melhores alocações;

O processo do AG visa a tomada de decisões que melhorem o desempenho relativo das soluções candidatas. Uma vez que são definidos *links* com capacidades muito superiores às necessárias para fluir o tráfego das alocações, o algoritmo retorna a cada iteração um conjunto de soluções que correspondem a diferentes capacidades de *links* de rede suficientes para acomodar o tráfego.

A Fig. 4.5 ilustra o diagrama UML de sequência da implementação do modelo híbrido. Os passos são explicados a seguir:

- No passo 1, o AG recolhe os dados da topologia a partir de um documento BRITE com a descrição dos requisitos das VMs e capacidades dos servidores;

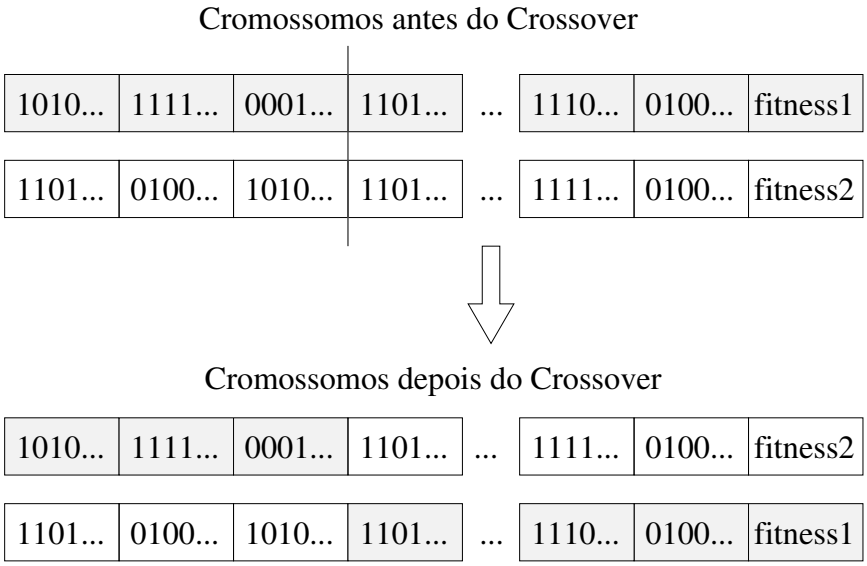

Figura 4.4: Exemplo do Processo de *Crossover* Pontual da Estratégia Híbrida.

- No passo 2, a população de cromossomos é inicializada com diferentes capacidades de valores binários para os *links* de rede;
- No passo 3, a avaliação inicial da população é feita para atribuir o valor inicial de *fitness* dos cromossomos. Essa avaliação é feita como segue: no passo 3.1 é gerado para cada cromossomo um modelo MILP. No passo 3.2, o modelo MILP é executado no *solver*. No Passo 3.3, a partir do resultado do *solver*, é gerado um modelo de rede equivalente para a simulação do tráfego na rede. Apenas se o modelo MILP for factível esse modelo será gerado. No passo 3.4, o modelo de rede é executado. No passo 3.5, é realizado o *parser* dos resultados para atribuir o novo valor de *fitness* dos cromossomos. No passo 3.6, o *fitness* é calculado a partir dos dados do *solver* MILP e da simulação da rede. Os cromossomos que possuem soluções infactíveis recebem alto valor de *fitness* de forma a serem penalizados. A motivação é manter o valor do *fitness* dessas soluções acima do máximo valor que seria obtido por uma solução factível. Dessa forma, os indivíduos com alto valor de *fitness* tendem a ser descartados durante o processo evolutivo;
- Os passos 4, 5, 6, 7 e 8 continuam o processo evolutivo da geração atual. Na próxima geração do AG, o processo evolutivo continua a partir do passo 3.1.

4.3 Aprimoramentos ao Modelo Híbrido Proposto

Esta seção propõe dois aprimoramentos para aumentar a eficiência e a escalabilidade da estratégia híbrida proposta: relaxamento linear e alocação incremental de VMs.

4.3.1 Relaxamento Linear

O modelo híbrido proposto gera soluções factíveis para quaisquer *solvers* de problemas em MILP. No entanto, o uso de variáveis binárias implica em aumento considerável do tempo de computação para se atingir a solução ótima, com tempo superior a 1 hora para problemas com

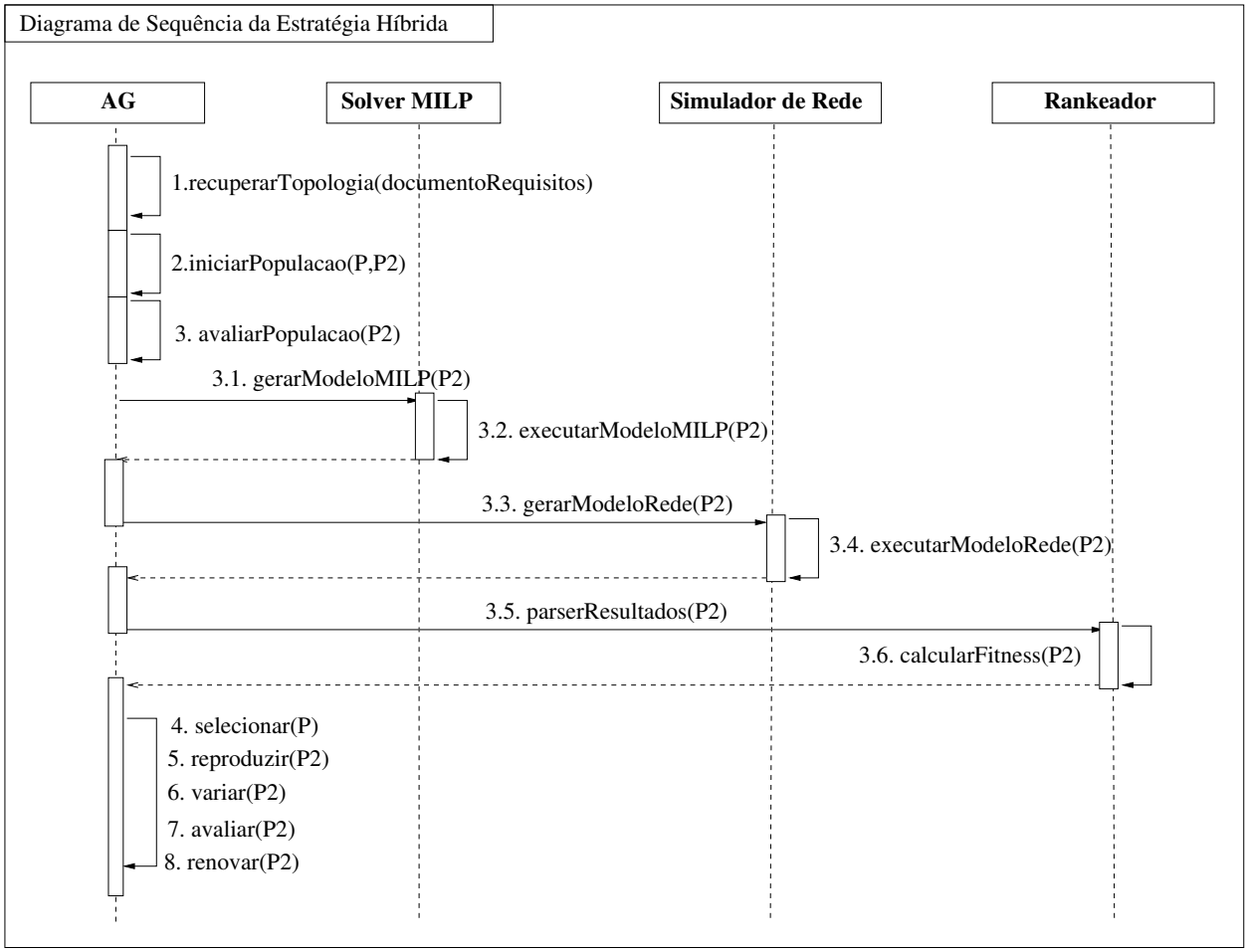

Figura 4.5: Diagrama UML de Sequência da Estratégia Híbrida.

pouco mais de 1000 variáveis. Por conta disso, introduzimos na estratégia híbrida técnicas de relaxamento linear (remoção das variáveis binárias). Uma abordagem similar também é adotada por Bonde (BONDE, 2011). Para isso foram identificadas heurísticas para o relaxamento das variáveis binárias a (variáveis de alocação), x (servidores ligados/desligados) e y (roteadores ligados/desligados).

O relaxamento linear das variáveis de alocação a é obtido pela heurísca abaixo:

$$0 \leq a_{i,j} \leq 1 \mid \forall i, j \tag{4.3}$$

Interpretação: As variáveis de decisão contínuas indicam que o valor da alocação será 1 caso a alocação seja feita no servidor j, e 0 caso contrário. Essa restrição é válida apenas se os requisitos das VMs forem múltiplos inteiros das capacidades dos servidores. Por exemplo, 2 *CPU cores* da VM_i são completamente alocadas em um servidor com 4 *CPU cores*, mas uma requisição de 3 *CPU cores* da VM_i, apesar de ser completamente satisfeita por um servidor com 4 *CPU cores*, gera um valor para $a_{i,j}$ entre 0 e 1, o que não é desejado.

O relaxamento linear das variáveis x (servidores ligados/desligados) se dá pela substituição destas variáveis pelas variáveis z conforme a expressão:

$$\sum_{i=1}^{N} a_{i,j}.VM_i^{cpu} + z_j = S_j^{cpu} \mid j = 1 \ldots M \tag{4.4}$$

Interpretação: A quantidade de *CPU cores* requisitadas por VMs mais a quantidade de *CPU cores* que ainda restam ser alocados (z_j) deve ser igual à quantidade de *CPU cores* disponíveis em j. A Fig. 4.6 ilustra a interpretação gráfica dessa restrição. Nessa figura, o servidor S_1 tem capacidade de alocar até 4 *CPU cores* e cada VM consome 1 *CPU core*. A variável z_1 indica a quantidade de *CPU cores* que ainda estão disponíveis para novas alocações. Uma variável $x_j = 0$ (servidor j desligado) é equivalente a $z_j = S_j^{cpu}$. Indiretamente, minimizando-se as variáveis z minimiza-se também a quantidade de servidores ligados.

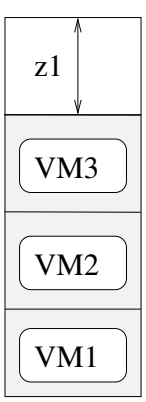

Figura 4.6: Interpretação da Restrição para Tratamento das Alocações.

O relaxamento linear das variáveis y (roteadores ligados/desligados) se dá pela substituição da parcela $\sum_{c=1}^{R} y_c.E_c^R$ na função objetivo (expressão) pela parcela $\sum_{k=1}^{R} RE_k.fin_k$, onde RE_k é uma constante referente ao consumo de energia por fluxo do roteador k, e fin_k é a quantidade de fluxo de ingresso no roteador k. Indiretamente, ao minimizar esta parcela minimiza-se também as variáveis y, ou o número de servidores ligados.

Com o relaxamento linear das variáveis a, x e y, a função objetivo do modelo linear é dada pela expressão abaixo:

$$MIN = \sum_{j=1}^{M} CE_j.z_j + \sum_{k=1}^{R} RE_k.fin_k \tag{4.5}$$

onde CE_j é uma constante inversamente proporcional ao número de *CPU cores* no servidor j. Desta forma, privilegia-se a alocação de VMs nos servidores com maior capacidade que tendem a ter um consumo por VMs menor que os servidores de menor capacidade. A razão é que, nos servidores de maior capacidade, os *CPU cores* compartilham refrigeração, periféricos, memória e outros dispositivos que consomem energia. A segunda parte da função objetivo é similar, ou seja, roteadores de maior capacidade apresentam um custo por fluxo transportado RE_k menor que roteadores de menor capacidade.

Desta forma, reduz-se o problema de alocação baseado em programação linear inteira em um problema de programação linear (contínua). Como este problema é resolvido a cada iteração do algoritmo genético, o tempo computacional é reduzido drasticamente. Ademais, como o modelo linear na estratégia híbrida se presta a avaliar o *fitness* da população (alocações) mantidas pelo

algoritmo genético, mesmo que o relaxamento linear produza soluções infactíveis do ponto de vista operacional (por exemplo, uma VM ocupando *CPU cores* de dois servidores), tais soluções são adequada para fins de evolução da população. Entretanto, comprovou-se na prática que o relaxamento linear proposto é capaz de produzir soluções com alocações corretas quando as demandas das VMs são submúltiplos inteiros das capacidades dos servidores.

4.3.2 Alocação Incremental de VMs

Data centers possuem de centenas a milhares de servidores, o que torna a modelagem da alocação única de uma grande quantidade de VMs e servidores pouco prática para cenários de grande escala. Nesse sentido, é proposta nessa seção uma variante da estratégia híbrida de alocação de VMs: a alocação incremental de grupos de VMs. A motivação é realizar a alocação de blocos de VMs conforme a demanda, em estágios. A vantagem é uma redução drástica da complexidade do problema pela diminuição de sua dimensão. Por exemplo, em vez de alocar 1000 VMs são alocadas 50 VMs por estágio. A desvantagem é que o número total de VMs possíveis de serem alocadas pode se tornar menor que as alocadas em um único estágio.

Na alocação incremental as seguintes adequações ao modelo linear são introduzidas:

- Os recursos dos servidores pré-alocados para VMs em estágios anteriores são descontados dos recursos disponíveis para o estágio atual;
- O fluxo de rede gerado pelas VMs alocadas nos estágios anteriores são acrescentados aos fluxos gerados pelas VMs alocadas no estágio atual.

Para qualquer servidor j no estágio t, depois da alocação de H VMs no estágio atual, a quantidade de recursos para o próximo estágio $t+1$ é definido como segue:

$$S_j^{cpu}(t+1) = S_j^{cpu}(t) - \sum_{i=1}^{H} a_{i,j}.VM_i^{cpu} \mid j = 1 \ldots M \tag{4.6}$$

$$S_j^{bw}(t+1) = S_j^{bw}(t) - \sum_{i=1}^{H} a_{i,j}.VM_i^{bw} \mid j = 1 \ldots M \tag{4.7}$$

Interpretação: Nas fórmulas anteriores, $S_j^{cpu}(t)$ e $S_j^{bw}(t)$ representam as quantidade de recursos de CPU e largura de banda disponíveis, respectivamente, no estágio t. Da mesma forma, para qualquer *data center* k no estágio t, o fluxo gerado pelas VMs neste *data center* após as alocações de H VMs neste estágio é definido como segue:

$$F_k^{src}(t) = \sum_{j} \sum_{i=1}^{H} a_{i,j}.VM_i^{bw} + F_k^{src}(t-1) \mid S_j \in D_k \tag{4.8}$$

Interpretação: No estágio t, a partir da alocação resultante do modelo, o simulador de rede adquire os parâmetros referentes à quantidade de VMs alocadas em cada *data center*, e do fluxo gerado em cada um dos *links* de rede. O fluxo residual gerado no instante $t-1$ é acrescentado ao fluxo das novas alocações no instante t.

A cada estágio, a qualidade de serviço na rede é avaliada com o simulador de redes. Se a qualidade estiver dentro dos limites, o estágio se completa com sucesso. Caso contrário, três possibilidades são possíveis:

1. Declarar que o número máximo de VMs foi atingido no estágio anterior;
2. Realizar uma alocação global das VMs alocadas até então;
3. Rearranjar o roteamento da rede.

Realizar uma alocação global das VMs alocadas até então pressupõe que VMs podem migrar de um servidor para outro. Esta alternativa é viável se a migração ocorrer no mesmo *data center*, pois, assim, a VM preserva sua conectividade (endereço IP). A terceira alternativa, rearranjar o roteamento da rede, é mais eficaz. O rearranjo do roteamento se dá pela execução do algoritmo genético como proposta na estratégia híbrida, exceto que a função de *fitness* agora leva em conta apenas a parcela relativa à rede (perda de pacotes na expressão 4.2). Neste caso, o modelo linear não é resolvido no processo de avaliação da população a cada iteração do algoritmo genético, pois as fontes de tráfego não se alteram, posto que as VMs já foram alocadas anteriormente.

4.4 Considerações Finais

Esse capítulo faz a descrição da estratégia híbrida de alocação de VMs. Foram propostas também heurísticas de relaxamento incremental para transformar o modelo linear com variáveis binárias (MILP) em um modelo puramente linear (LP). O capítulo também propôs uma variante do modelo híbrido onde a alocação de VMs se dá de acordo com um processo incremental. O próximo capítulo descreve a implementação e resultados visando validar a estratégia proposta e suas variantes. Além disso, a estratégia possui alto potencial de paralelismo como alternativa para otimizar o desempenho do processo evolutivo. No próximo capítulo, essa questão é tratada na avaliação concorrente do *fitness* de cada indivíduo (alocação) da população.

Capítulo 5

Avaliação e Resultados

Este capítulo descreve a implementação da estratégia híbrida de alocação de VMs descrita no capítulo anterior e os experimentos realizados para avaliar e validar esta estratégia. Esses experimentos buscam manter um compromisso entre o tempo computacional para obter as soluções e a qualidade da solução obtida.

5.1 Introdução

Experimentos com algoritmos em infraestruturas de nuvem reais como as da Amazon AWS ou Microsoft Azure são difíceis de reproduzir devido à rigidez da infraestrutura, dificuldade de reconfiguração de parâmetros em escala massiva, e elevado consumo de tempo para executar os testes. Por isso, o uso de simuladores é uma alternativa viável para avaliar o comportamento de diversos tipos de algoritmos de alocação de recursos em um tempo computacional razoável (CALHEIROS et al., 2012). A avaliação do modelo híbrido é feita com o uso da ferramenta REALcloudSim (ROCHA, 2012), desenvolvida durante a pesquisa. É claro que não será possível avaliar todos os cenários possíveis diante da grande diversidade de configurações que o modelo híbrido admite. Para a avaliação dos resultados, a seguinte metodologia foi adotada:

- Definir um ambiente de execução: o *hardware* a ser utilizado nos experimentos;
- Definir uma configuração de referência: essa configuração tem o intuito de ser um padrão para a avaliação dos experimentos, uma vez que a gama de possibilidades de configuração oferecidas pelo REALcloudSim são tão diversas quanto se queira. São definidas as capacidades dos *data centers* e seus servidores, os requisitos para que as VMs sejam alocadas, a topologia da rede, o protocolo de roteamento, e as diversas configurações do AG;
- Avaliar o uso de paralelismo na execução dos experimentos;
- Realizar o processamento das alocações de acordo com a configuração de referência. Os principais resultados a serem obtidos são o mapa de alocação de VMs, o consumo de energia, o tráfego na rede quanto ao roteamento e o percentual de perda de pacotes nos *links* da topologia.

5.1.1 Ambiente de Execução

Os experimentos foram realizados em um servidor Dell Power Edge R710 que possui 2 processadores Quad-Core Intel Xeon E5620, com 2,4 GHz e tecnologia Hyper-Threading. Essa configuração permite o uso de até 8 núcleos físicos ou até 16 núcleos lógicos. A plataforma oferece até 65 GB de memória física, e é gerenciada com o uso do *software* de virtualização Proxmox (PROXMOX, 2013) versão 1.9. Para a realização dos experimentos com execução paralela, foram utilizadas 4 VMs neste servidor. Cada VM possui a ferramenta REALcloudSim instalada, um *solver* de programação linear Lingo13 (LINDO SYSTEMS, 2012) versão para arquiteturas de 64 bits, um simulador de rede NS-2 versão 2.35. Cada VM é configurada com 30 GB de espaço em disco, 4 GB de memória RAM, e até 16 núcleos lógicos de CPU. Cada VM possui um sistema operacional Linux Ubuntu versão 64 bits. A execução dos experimentos é feita sobre um ambiente com virtualização assistida por *hardware*, com o uso do KVM (*Kernel-based Virtual Machine*) (KVM, 2012), que é uma solução de código aberto para virtualização em arquiteturas de processadores x86 e x64. Cada máquina virtual instanciada possui seus próprios recursos de *hardware* virtualizados. A virtualização com KVM exige o suporte à virtualização assistida por *hardware* na CPU do servidor, e utiliza o escalonador e o gerenciamento de memória regulares do Linux (JONES, 2012).

5.1.2 Configuração de Referência

A configuração de referência define um padrão para a avaliação dos resultados obtidos pelas diferentes políticas de alocação. Nesses experimentos, as capacidades dos servidores foram definidas de acordo com a Tab. 5.1. Cada VM a ser alocada possui requisitos que precisam ser inteiramente satisfeitos pelos servidores. A definição de diferentes requisitos das VMs é considerada na maioria dos provedores de computação em nuvem, com instâncias de pequeno, médio e grande porte. A configuração de referência dos requisitos das VMs é mostrada na Tab. 5.1.

A configuração de referência para a topologia da rede define uma estrutura composta de nós que representam roteadores de núcleo, sendo que o nó central é definido como o nó de destino de todo o tráfego (*backbone* da rede). As arestas definem as conexões de rede entre os nós. Roteadores redundantes nesta rede são inspirados em uma rede de ISPs que precisam de garantias contra falhas nos *links* entre os *data centers*. Nessa configuração, cada *data center* possui um roteador de borda para conectar o *data center* aos roteadores de núcleo da topologia. Admite-se que todo o tráfego gerado pelas VMs seja agregado ao roteador de borda do seu *data center*, para fins de simulação, como mostra a Fig. 5.1.

Relativamente às unidades de medida, não se tem o objetivo de avaliar a sua precisão quanto à carga de ambientes reais de grande porte. O objetivo é estimar, sem perda de generalidade, a quantidade de recursos necessários para prover a demanda, tanto no *solver* de programação linear quanto no simulador de rede, com o uso dos mesmos valores em ambos os modelos. Por exemplo, a largura de banda de 500 unidades representa a capacidade do *link* com o valor de 500 unidades para o *solver*, mas também representa a capacidade do *link* em 500 unidades no simulador de rede (onde as unidades são definidas em Kbps).

Quanto à rede entre os *data centers*, foi observado que o modelo é capaz de resolver alocações para quaisquer topologias desde que os grafos no modelo sejam direcionados (dígrafos) e acíclicos, porém não necessariamente fortemente conexos. Essas propriedades são necessárias para garantir a factibilidade do modelo quanto à rede. A modelagem da rede com grafos direcionados pode

Configuração de Referência para Servidores	
Quantidade de *data centers*	4 *data centers*
Quantidade de servidores por *data center*	200 servidores
Quantidade de CPU	4,8,32 cores
Servidores Tipo 1 (4 CPUs):	60% (120 servidores)
Servidores Tipo 2 (8 CPUs):	30% (60 servidores)
Servidores Tipo 3 (32 CPUs):	10% (20 servidores)
Largura de Banda	1000 unidades
Configuração de Referência para VMs	
Quantidade de CPU	4 cores
Largura de Banda Reservada	2 unidades
Quantidade de Fluxo Gerado	2 unidades
Configuração de Referência da Topologia	
Quantidade de Nós	25 nós
Quantidade de *Links*	40 *links*
Custo dos *Links*	Unitário
Largura de Banda dos *Links*	500 e 700 unidades

Tabela 5.1: Configuração de Referência.

ser vista como uma rede que indica caminhos potenciais para um roteamento baseado em restrições. Por exemplo, caminhos potenciais para o estabelecimento de LSPs (*Label Switch Paths*) em um domínio MPLS (*MultiProtocol Label Switching*) (ROSEN; VISWANATHAN; CALLON, 1999) (ALLAN et al., 2006) (KUROSE; ROSS, 2007).

Essa propriedade foi considerada para a criação de uma topologia com 25 nós e 40 arestas. Para garantir as propriedades anteriormente observadas foi utilizado o *software* GraphViz (GRAPHVIZ, 2012), que possui a ferramenta *acyclic* para validação e geração de dígrafos acíclicos. A partir da topologia BRITE foi utilizada a ferramenta *acyclic* para validar a topologia de rede. A topologia de referência da Fig. 5.1 foi traduzida para o software NAM (*Network Animator*) do NS-2, como mostra a Fig. 5.2.

A motivação é avaliar a distribuição do tráfego em uma topologia que permita diversas opções de encaminhamento do tráfego. Por outro lado, busca-se aproximar os resultados quanto a alocações de VMs em redes reais convencionais. Essa configuração busca representar uma grande quantidade de opções de roteamento entre as fontes e o destino do tráfego. Apesar de o modelo admitir múltiplos destinos, a obtenção do menor caminho é mais simples de ser avaliada com um destino comum. De acordo com Vaquero et al. (VAQUERO et al., 2012), grandes topologias de rede são difíceis de reproduzir em experimentos de rede. Por outro lado, a maioria dos ISPs mantém essa informação confidencial, e o acesso à informação interna dessas topologias é muito limitado. Alternativas são o uso de redes de pesquisa acadêmicas, como o PlanetLab (PLANETLAB, 2012), geradores aleatórios de topologias, ou mapas de topologias pré-definidas.

Para essa topologia, o uso do protocolo de roteamento padrão do NS-2 não é adequado para estabelecer as mesmas rotas indicadas pelo resultado do modelo LP. É importante notar que os *links* simulados são os mesmos indicados pela solução do modelo LP, a partir do fluxo que percorre esses *links*. Por conta disso, o protocolo de roteamento é definido entre cada par

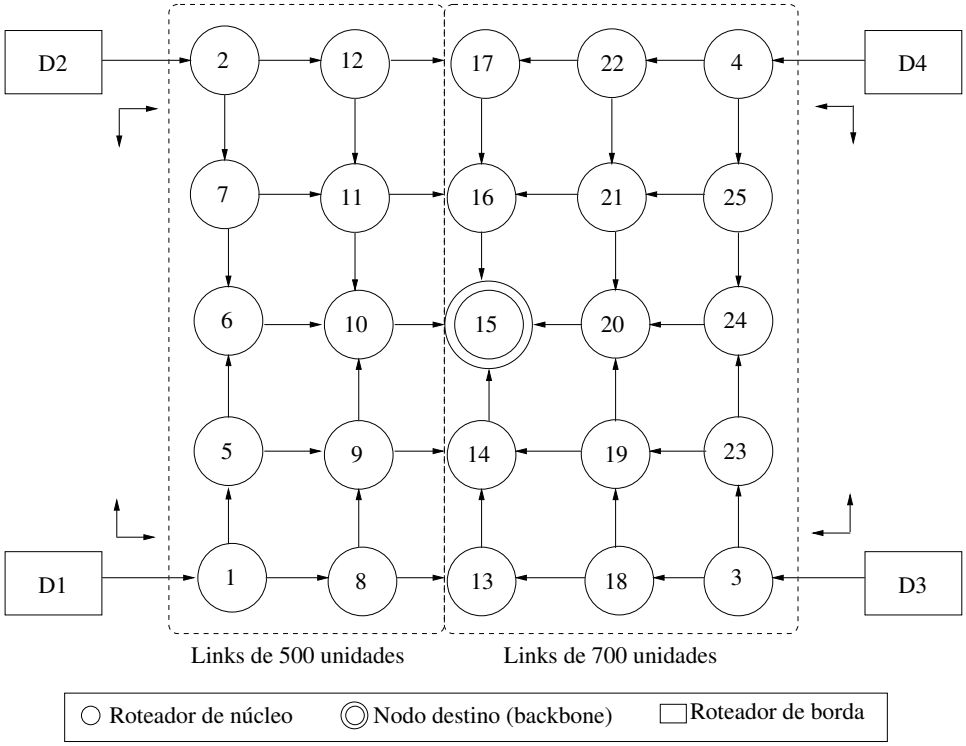

Figura 5.1: Topologia de Referência: Grafo Acíclico com 25 nós e 40 arestas.

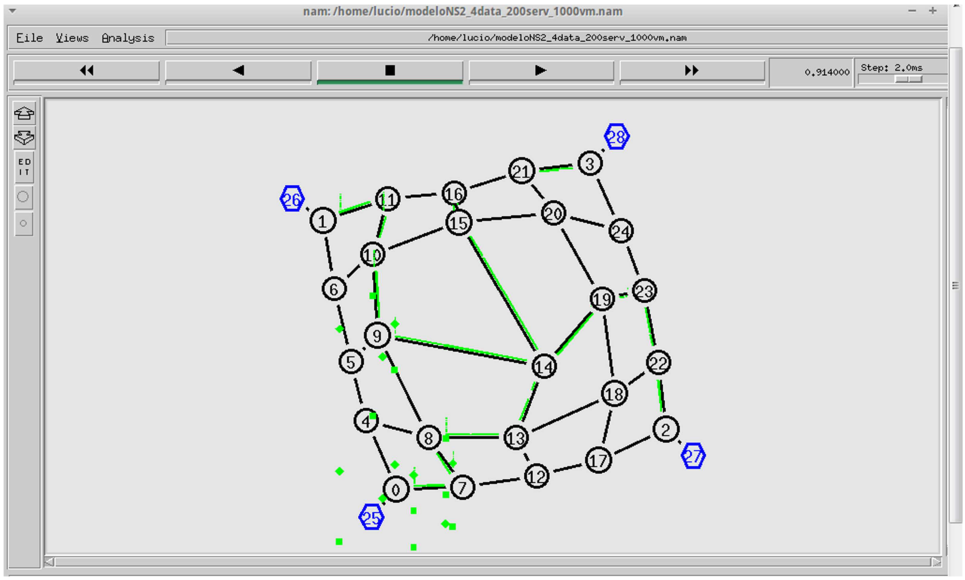

Figura 5.2: Topologia de Referência no *Software* NAM: Grafo Acíclico com 25 nós e 40 arestas.

de *links*, a partir do resultado obtido pela solução do modelo de programação linear. Caso contrário, o roteamento seria diferente do indicado pelo modelo LP.

5.2 Alocação Global Única com o Modelo Linear

Nos experimentos dessa seção, é feita a alocação global de todas as VMs e utilizada a técnica de relaxamento linear, motivo pelo qual esses experimentos seguem um modelo estritamente linear, ou seja, um modelo LP. O objetivo dessa seção é avaliar, inicialmente, o mapa de alocações por servidor e quantidade de alocações por *data center*. A avaliação feita busca promover a alocação de 1000 VMs, e segue a configuração de referência da Tab. 5.1. A Fig. 5.3 mostra o resultado da alocação. Nesta figura, os 200 primeiros servidores (índices de 1 a 200) correspondem ao *data center* D1, os 200 servidores subsequentes ao *data center* D2 e assim sucessivamente. Em um *data center*, os índices mais altos correspondem aos servidores de mais alta capacidade (número de *CPU cores*).

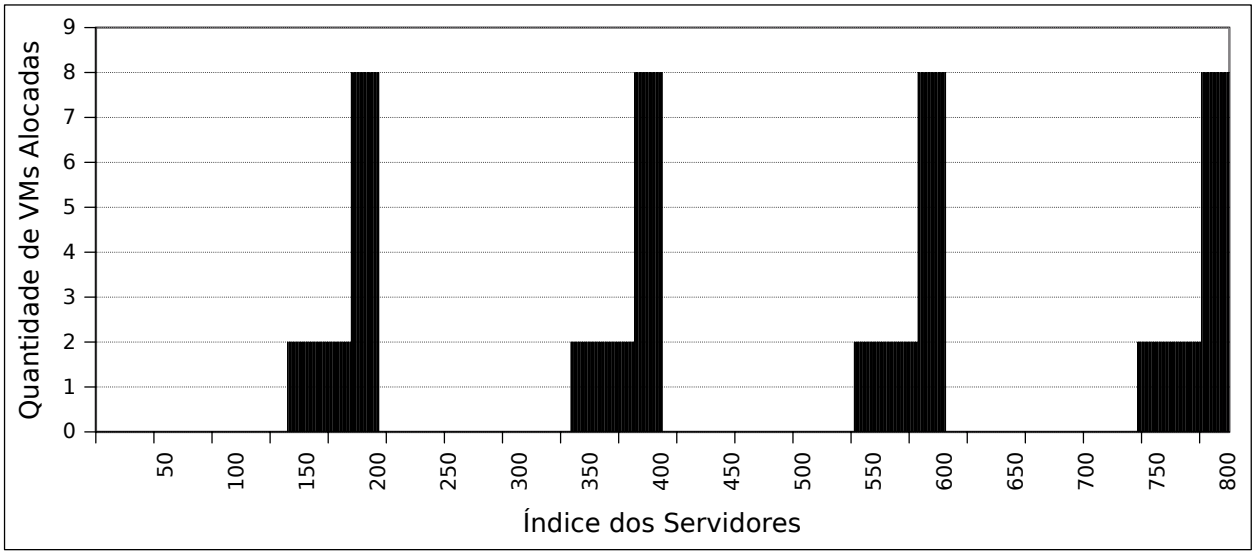

Figura 5.3: Mapa de Alocação das VMs com o Modelo LP.

Interpretação dos resultados: Os resultados da Fig. 5.3 mostram que os servidores com maior quantidade de CPUs recebem a maior quantidade de alocações. Isso é consequência da função objetivo, que indica que os servidores com maior quantidade de CPUs têm menor custo por VM alocada, com o objetivo de que uma quantidade maior de VMs seja alocada nesses servidores, o que resulta em uma quantidade menor de servidores necessários para acomodar todas as alocações. Os resultados da Tab. 5.2 indicam que todas as VMs foram alocadas e distribuídas igualmente entre todos os *data centers*.

A fim de avaliar o volume do tráfego resultante do modelo de referência, foi realizada a simulação da rede no *software* NS-2. Esse simulador de rede é implementado em linguagem C++, Object Tcl (OTcl) e Tcl/Tk. Os objetos em C++ servem para o processamento de dados, enquanto OTcl é usado para as funcionalidades de controle e para unir os objetos C++ em uma estrutura comum. Os *scripts* de simulação são gerados em OTcl (ISSARAIYAKUL; HOSSAIN, 2012) (BALANDIN; HEINER, 2002).

Embora simulações estejam longe de representar a realidade, elas conferem facilidade para a variação de parâmetros e facilitam a avaliação da escalabilidade do modelo. Para os experimentos com o NS-2, a vazão (*throughput*) é dada pelo valor do número total de bits que trafega no *link* em relação ao tempo da simulação, ou seja, a vazão é a taxa na qual o processo que envia dados é capaz de entregar bits ao processo que recebe dados (KUROSE; ROSS, 2007). A

	Serv. Tipo 1	Serv. Tipo 2	Serv. Tipo 3	Total
VMs alocadas no *data center* 1	0	90	160	250
VMs alocadas no *data center* 2	0	90	160	250
VMs alocadas no *data center* 3	0	90	160	250
VMs alocadas no *data center* 4	0	90	160	250
Total de VMs alocadas	0	360	640	1000/1000
Servidores alocados no *data center* 1	0	45	20	65
Servidores alocados no *data center* 2	0	45	20	65
Servidores alocados no *data center* 3	0	45	20	65
Servidores alocados no *data center* 4	0	45	20	65
Total servidores alocados	0	180	80	260/800
Consumo de energia dos servidores	804 unidades			
Consumo de energia dos roteadores	17 unidades			
Total roteadores internos ativos	17/25			
Tempo de processamento	120 segundos			

Tabela 5.2: Tabela de Alocação do Modelo LP.

vazão e a largura de banda (*bandwidth*) são conceitos distintos (KUROSE; ROSS, 2007). Assim, a vazão máxima entre a origem e o destino da conexão pode ser menor ou igual à largura de banda oferecida. Nesse sentido, a largura de banda é a taxa de transmissão máxima oferecida como resultado dos enlaces existentes na comunicação fim-a-fim. Por outro lado, quando o *buffer* do roteador está cheio, geralmente ocorre o descarte dos novos pacotes entrantes na fila desse *buffer*. A fração de perda de pacotes aumenta com a intensidade do tráfego. Isso faz com que o desempenho de um nó da rede também seja avaliado em função da sua probabilidade de perda de pacotes. Portanto, a perda de pacotes pode ocorrer quando são enviados mais pacotes do que a capacidade de processamento deles nos roteadores e/ou no *host* destino. A perda de pacotes dependerá da carga do tráfego, da velocidade relativa do elemento de comutação, da taxa de transmissão do enlace, de erros na transmissão dos pacotes, entre outros fatores. Esse é um fator essencial para a análise de congestionamento e do atraso fim-a-fim (KUROSE; ROSS, 2007).

Para esse experimento, as fontes de tráfego submetem tráfego do tipo UDP/*Exponential*, ou seja, um tráfego com variações entre os períodos de *send/receive* similar às do tráfego da Internet. O tráfego é modelado como o agregado de tráfego de todas as VMs de um dado *data center*. Os pacotes possuem tamanho de até 1500 bytes referentes à MTU (*Maximum Transmition Unit*) da rede Ethernet. A submissão do tráfego respeita a seguinte fórmula:

$$trafegoExponential = ((idle\_time + burst\_time)/burst\_time) * rate \tag{5.1}$$

Interpretação da fórmula: os períodos de *idle_time* e *burst_time* são os períodos de inatividade e de envio de rajadas, respectivamente. No tráfego UDP/*Exponential*, a submissão de rajadas é aleatória, o que pode vir a gerar picos de tráfego nos *links*, com o aumento do atraso fim-a-fim e da perda de pacotes. Ainda em função dessa aleatoriedade, a vazão final média tende a ser menor do que taxa inicial de transmissão (*rate*) fornecida, pois o tráfego não é constante.

Interpretação dos resultados: O objetivo desse experimento é avaliar se o tráfego das VMs flui as longo dos *links* entre os nós da topologia. A Fig. 5.4 mostra o resultado da distribuição do tráfego na topologia. É possível observar que o modelo é capaz de indicar quais roteadores

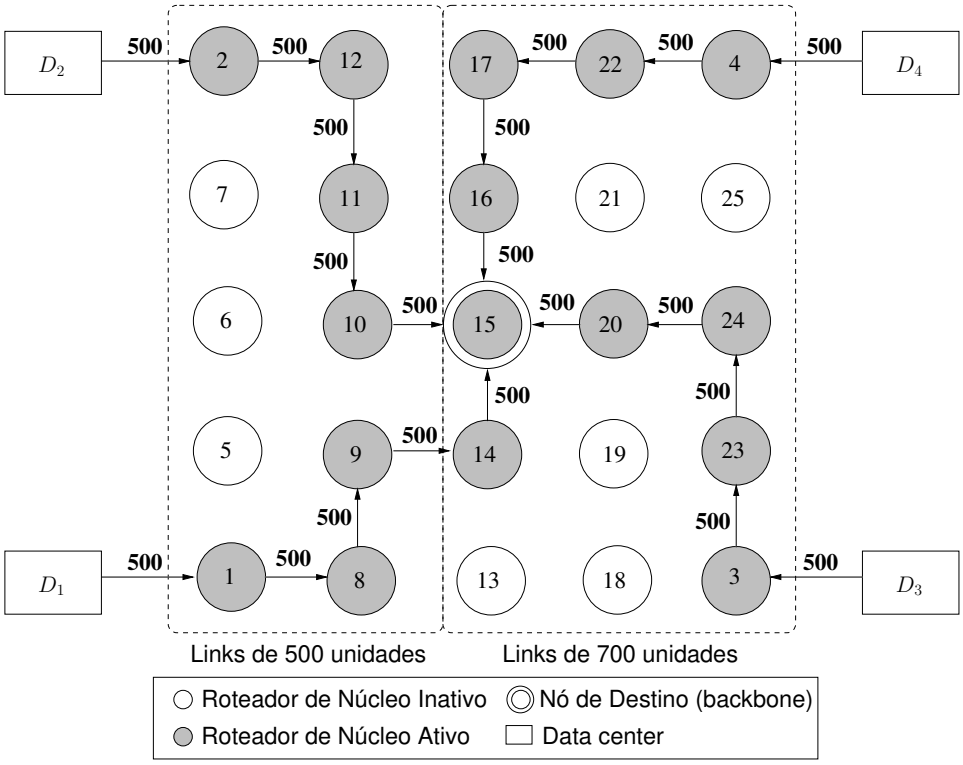

Figura 5.4: Fluxo nos *Links* com o Modelo LP.

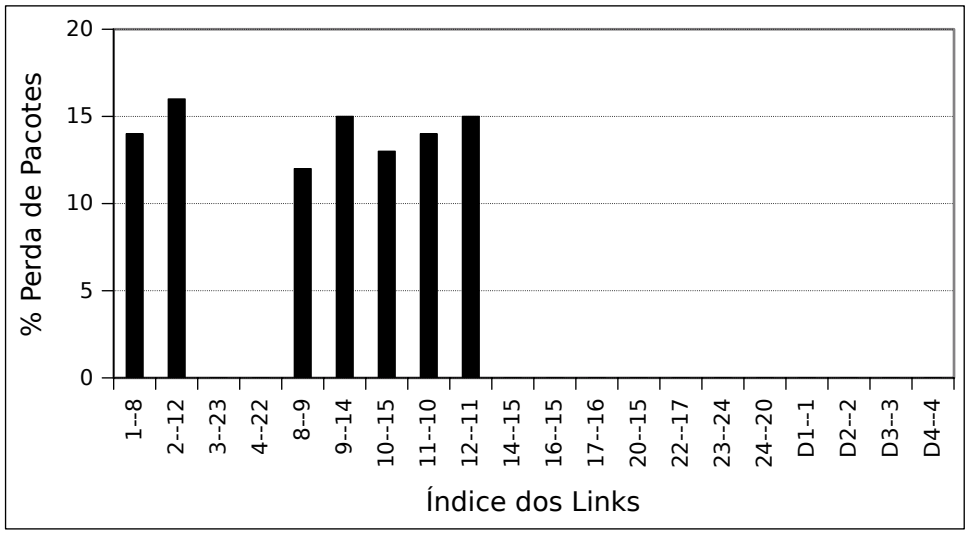

Figura 5.5: Perda de Pacotes nos *Links* da Topologia.

são necessários para comportar o tráfego das VMs. No experimento em questão, os roteadores que não apresentam passagem de fluxo em seus *links* de entrada são elegíveis para entrarem em estado de baixo consumo de energia (inativos).

A Fig. 5.5 mostra a perda de pacotes da distribuição. A porcentagem de perda de pacotes alcança mais de 5% se for considerado o tráfego que percorre todos *links*. Os roteadores de borda de cada *data center* possuem *links* com a maior largura de banda da topologia, motivo pelo qual a perda de pacotes não é significativa nesses *links*. É observado que o modelo satura

os *links* para encaminhar o tráfego, mesmo que existam rotas alternativas para encaminhar o tráfego. Os *links* de menor capacidade (500 unidades) apresentam as maiores perdas, como esperado. Isso porque os *links* de maior capacidade (700 unidades) são capazes de acomodar uma quantidade maior de tráfego nos períodos de rajada. A proximidade da largura de banda em relação à vazão do tráfego gera perda de pacotes nos *links* internos, o que é um indicativo de que o modelo em programação linear não é suficiente para modelar o tráfego na rede no sentido de reduzir a perda de pacotes e de encaminhar o tráfego por rotas menos congestionadas, o que justifica o uso de uma estratégia híbrida que trate essas deficiências. Os parâmetros de referência para a simulação do tráfego na rede são definidos conforme a Tab. 5.3.

Tempo de Simulação	60 segundos
Packet Size	1500
Rate	Taxa do Agregado do Fluxo das VMs
Idle_Time	100ms
Burst_Time	400ms
Tipo de Tráfego	UDP/*Exponential*

Tabela 5.3: Configuração de referência para simulação do tráfego.

Protocolos de roteamento típicos, tais como BGP (*Border Gateway Protocol*), RIP (*Routing Information Protocol*) ou OSPF (*Open Shortest Path First*), privilegiam *links* de menor custo (usualmente os de maior capacidade), o que causa alta taxa de descarte de pacotes nestes *links*. Nas simulações, considerou-se o roteamento baseado em restrições, por exemplo, com o uso da tecnologia MPLS. Desta forma, os caminhos dos fluxos fornecidos pelo modelo de programação linear são reproduzidos no simulador NS-2 como se fossem roteados por LSPs em uma rede MPLS. Tal reprodução não seria possível com o uso de protocolos de roteamento padrão, por exemplo, OSPF. A Fig. 5.6 ilustra a vazão obtida no NS-2 para a distribuição de VMs em questão. Esse resultado mostra que as rotas indicadas pelo modelo LP correspondem aos *links* que encaminham o tráfego na simulação.

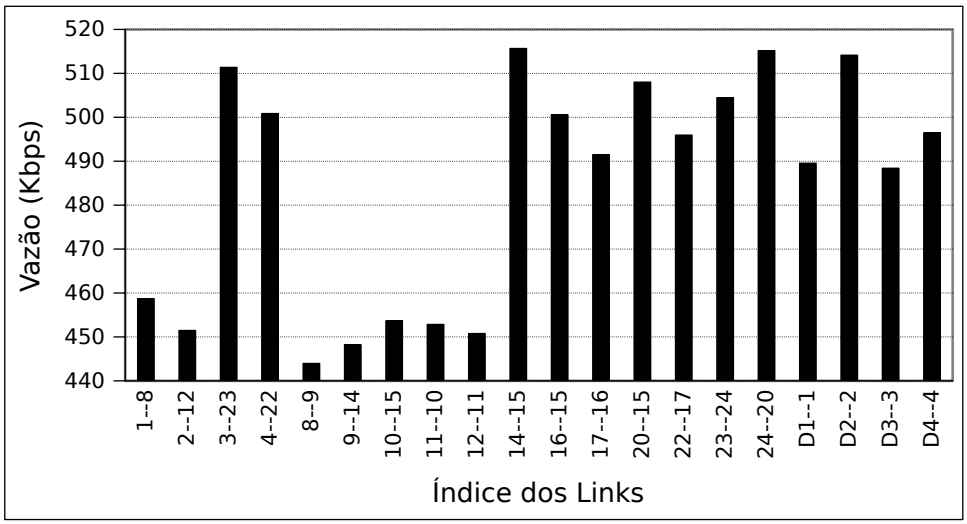

Figura 5.6: Vazão nos *Links* da Topologia.

5.3 Alocação com a Estratégia Híbrida

Diante da dificuldade de se modelar características de rede em problemas de programação linear, o AG complementa a solução do modelo LP ao buscar adequar as alocações em função das características do tráfego da VMs na rede entre os *data centers*.

5.3.1 Variação das Capacidades dos *Links*

A variação das capacidades dos *links* é controlada pelo AG, e os modelos LP utilizam esses valores como segue: a fim de reduzir o tamanho das estruturas de dados dos cromossomos, cada *locus* (campo do cromossomo) contém um conjunto de 4 bits. Assim, cada *locus* possui bits atribuídos para um único *link*, ou seja, um cromossomo é formado por um conjunto de *loci*, e o conjunto de bits em cada *locus* é mapeado para cada *link*. Para o cálculo da capacidade máxima de um *link*, os bits do *locus* são transformados em valor decimal. O máximo valor decimal dos bits de um *locus* é 15, com todos os bits iguais a 1. Assim, o valor da capacidade de cada *link* é regido pela fórmula:

$$capacidadeLink = (valorDecimalBitsLocus * MAX\_LINK)/fatorCorrecao \tag{5.2}$$

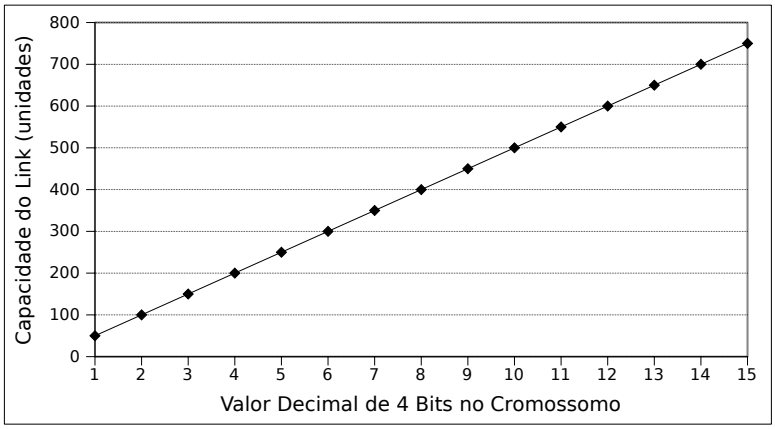

Figura 5.7: Variação da Capacidade dos *Links* na Topologia.

A Fig. 5.7 ilustra a variação da capacidade dos *links* quando $MAX\_LINK = 500\,unidades$. O AG busca encontrar uma distribuição de capacidades de *links* suficientes para encaminhar o tráfego com reduzida perda e reduzido atraso fim-a-fim. O processo evolutivo faz a seleção de pares de cromossomos, e essa seleção é inversamente proporcional ao *fitness*, pois trata-se de um problema de minimização. Os cromossomos que geram soluções não-factíveis recebem valor de *fitness* muito acima dos demais para aumentar as chances de serem substituídos nas próximas gerações. A motivação é promover o descarte desses cromossomos ao longo do processo evolutivo. Para isso, é atribuído um valor arbitrário de grandeza superior à estimativa do valor máximo do *fitness* da população.

5.3.2 Paralelismo com o Modelo *Publish/Subscribe*

Os resultados da execução sequencial da estratégia híbrida revelaram que é necessário um tempo computacional elevado para se alcançar uma solução factível com até 100 gerações do

AG. Em média, até 24 horas de processamento são necessárias para avaliar a convergência do *fitness* com 32 cromossomos. Inicialmente, optou-se por utilizar uma estratégia de paralelismo baseada em *threads*. Como resultado, o tempo computacional foi reduzido para pouco mais de 5 horas para se obter uma solução factível com até 100 gerações do AG e 32 cromossomos.

Em virtude do elevado tempo de execução da estratégia híbrida com *threads* foi proposta a avaliação do processamento distribuído dos cromossomos com um modelo *Publish/Subscribe*, empregando-se VMs como unidades de paralelismo. Nesse modelo, um ou mais *workers* executam nas VMs instanciadas no servidor Dell. Cada *worker* gera uma cópia própria dos arquivos da topologia da rede, e a identifica com um ID único (*rank*). O algoritmo genético publica tarefas com os valores das capacidades dos *links* dos cromossomos, em formato JSON (*JavaScript Object Notation*), no canal de eventos de tarefas. Cada *worker* recolhe (via HTTP GET), processa e publica (HTTP POST) o resultado de seu processamento no canal de eventos de resultados. O AG aguarda em espera ocupada até que todos os cromossomos da geração tenham sido processados. A Fig. 5.8 mostra a interação do AG com o canal de eventos e os *workers*. O canal de eventos é instanciado em uma das VMs na forma de um *servlet* Java que recebe conexões HTTP e executa em um servidor Apache Tomcat. Nos experimentos que seguem foi utilizada a configuração de referência da Tab. 5.1, porém com a redução da quantidade de VMs para 100 unidades, a fim de avaliar o desempenho do paralelismo.

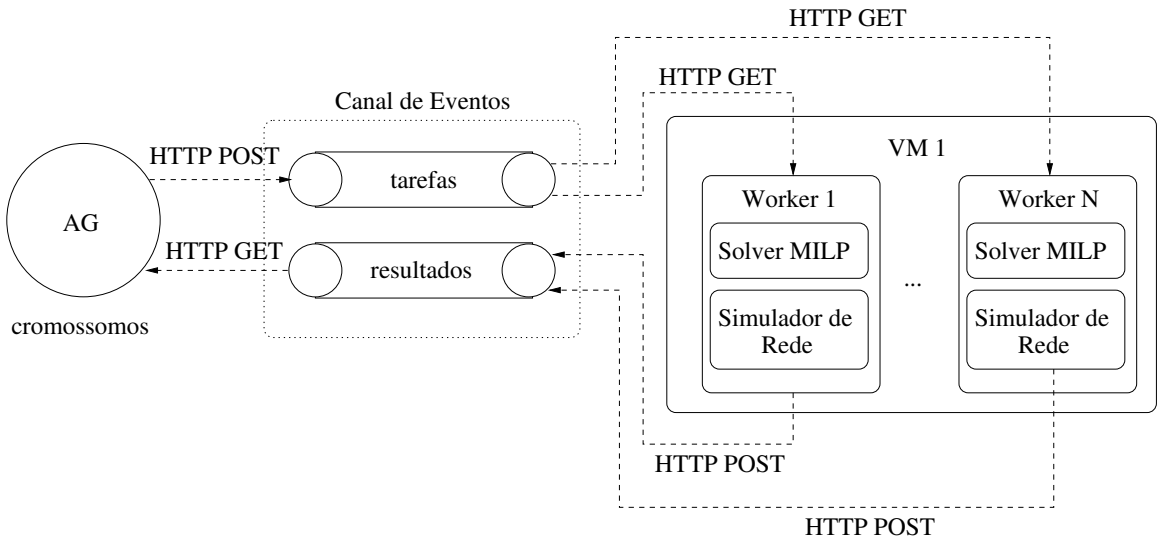

Figura 5.8: Modelo *Publish/Subscribe*.

Workers	8 cromossomos	16 cromossomos	32 cromossomos
1	46min:20s	1h:3min	1h:36min
2	25min:27s	42min:03s	1h:15min:55s
4	17min:27s	26min:35s	43min:43s
6	17min:12s	26min:12s	35min:58s
8	17min:07s	19min:12s	29min:13s
10	17min:15s	19min	29min:12s
12	17min:32s	19min:18s	30min:15s

Tabela 5.4: Tempo de Execução da Estratégia Híbrida com o modelo *Publish/Subscribe*.

Figura 5.9: *Speedup* com o Modelo *Publish/Subscribe*.

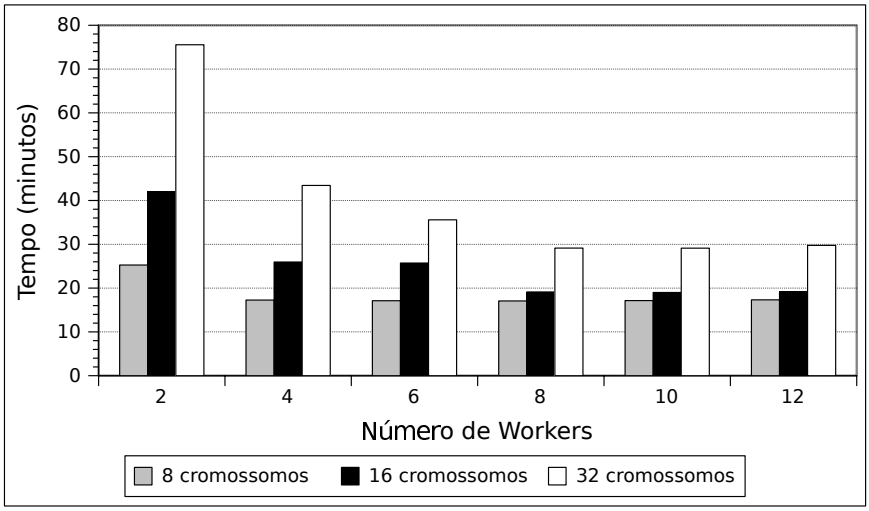

Figura 5.10: Tempo de Evolução × Número de *Workers* com o Modelo *Publish/Subscribe*.

O tempo de execução da estratégia híbrida para 100 gerações do AG, e com variação da quantidade de *workers*, é mostrado na Tab. 5.4. Esses resultados indicam que o modelo *Publish/Subscribe* é adequado para reduzir o tempo de processamento das informações contidas nos cromossomos. No entanto, o paralelismo é limitado pela quantidade física de CPUs disponíveis no servidor para efetuar o processamento, motivo pelo qual o ganho de desempenho passa a ser menos acentuado quando mais de 8 *workers* são alocados, uma vez que a plataforma possui 8 núcleos físicos, e cada *worker* é um processo que executa em uma VM no servidor e que consome recursos de um único *CPU core*, como mostram as Fig. 5.9 e 5.10. O *speedup* foi obtido pela divisão do tempo de execução de 1 *worker* pelo tempo de execução com N *workers*, no caso, $N = 12$, para uma população de 32 cromossomos. Uma possibilidade de aumento do ganho de desempenho é o uso de uma rede de servidores para escalar o processamento dos cromossomos. Nesse caso, o desempenho estaria limitado ao desempenho de rede entre esses servidores. Outro fator limitante do desempenho é a quantidade de conexões HTTP no canal de eventos, o que o torna um possível gargalo no processamento. Observou-se também que há ganho de desempenho quando os núcleos virtualizados são distribuídos entre as VMs, ao invés

de se alocar uma única VM com uma grande quantidade de *CPU cores*.

5.3.3 Execução da Estratégia Híbrida

O processo de alocação de VMs em servidores baseia-se no uso da função de *fitness* que envolve a otimização de critérios distintos, notadamente, o consumo de energia dos servidores e roteadores, e a qualidade de serviço da rede, representada pela perda de pacotes do tráfego gerado pelas VMs na simulação. O consumo de energia e a qualidade de serviço da rede são critérios que não são conflitantes entre si, mas precisam ser otimizados simultaneamente. Trata-se, portanto, de um problema de decisão com múltiplos critérios, em um problema de programação linear mista, sujeito a restrições.

Segundo (SILVA, 2007) o conjunto de soluções ótimas para um problema multi-objetivo recebe o nome de Fronteira de Pareto, que é uma forma de representar soluções de problemas com múltiplos objetivos. Os métodos clássicos adotados para resolver esses problemas são os métodos WSM (*Weighted Sum Method*) e *ε-constrained* (SARKER; RAY, 2009). O método WSM associa para cada função objetivo um coeficiente de peso, e minimiza/maximiza o somatório dos pesos dos objetivos. Os pesos representam a importância que se deseja atribuir a um dado objetivo. As múltiplas funções objetivo são agrupadas em uma única função objetivo. Por outro lado, o método *ε-constrained* define que uma das funções objetivo deve ser selecionada para ser otimizada, enquanto as outras funções objetivo são transformadas em restrições com limites superiores.

No entanto, para o problema em questão, o fato de não haver dependência entre os critérios de otimização dificulta o uso das abordagens clássicas citadas anteriormente. Por conta disso, optou-se por ponderar os pesos dos critérios baseado nas observações empíricas do comportamento do AG. Uma possibilidade futura de avaliação mais eficiente seria a otimização dos objetivos com algoritmos evolucionários (SILVA, 2007). Outra consideração é a de que a aleatoriedade do AG em encontrar soluções no espaço de busca não garante que sejam encontrados pontos na fronteira de Pareto, porém permite obter um conjunto de soluções não-dominadas.

No experimento dessa seção são avaliados os resultados obtidos com a função de *fitness* dada pela expressão 4.2. Nesse experimento, os valores de α e β são atribuídos para ponderar equitativamente o consumo de energia e a perda de pacotes. Atribuiu-se $\alpha = 1$ porque esse objetivo apresenta a maior variação (em torno de 800 unidades), e $\beta = 500$ para que pequenos percentuais da perda tenham a mesma ordem de influência no *fitness* da solução.

A avaliação que segue utiliza a técnica de relaxamento linear apresentada na seção 4.3.1. Apenas na última iteração do AG é executado o modelo MILP com as variáveis de alocação binárias, para até 150.000 iterações do *solver*. Além disso, os operadores genéticos são definidos como segue: taxa de mutação de $\frac{1}{P}$ onde P é o tamanho da população; *crossover* pontual com seleção probabilística por método da roleta; tamanho da população definido em 32 indivíduos, que se mostrou suficiente para avaliar a convergência do algoritmo na topologia considerada; população inicial gerada de forma aleatória. Nesses experimentos, é feita a alocação de 1000 VMs, e se segue a configuração de referência da Tab. 5.1.

A Fig. 5.11 ilustra a evolução da porcentagem de perda de pacotes na rede; a Fig. 5.12 ilustra a evolução do consumo de energia dos servidores e roteadores; e a Fig. 5.13 ilustra a evolução da quantidade de roteadores ativos. A interpretação desses resultados deve ser feita em conjunto, na mudança dos intervalos de cada um deles. Como existem vários intervalos, é feita a seguir uma descrição de intervalos significativos a partir desses resultados:

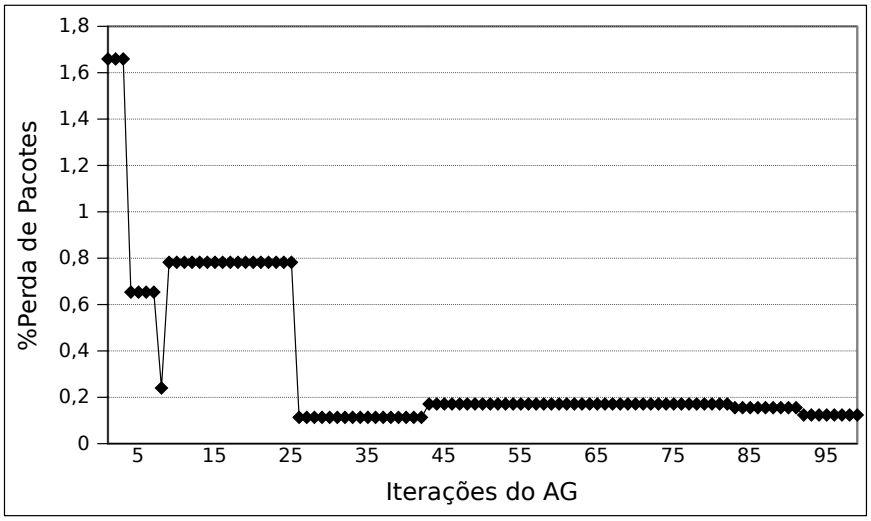

Figura 5.11: Evolução da Perda de Pacotes.

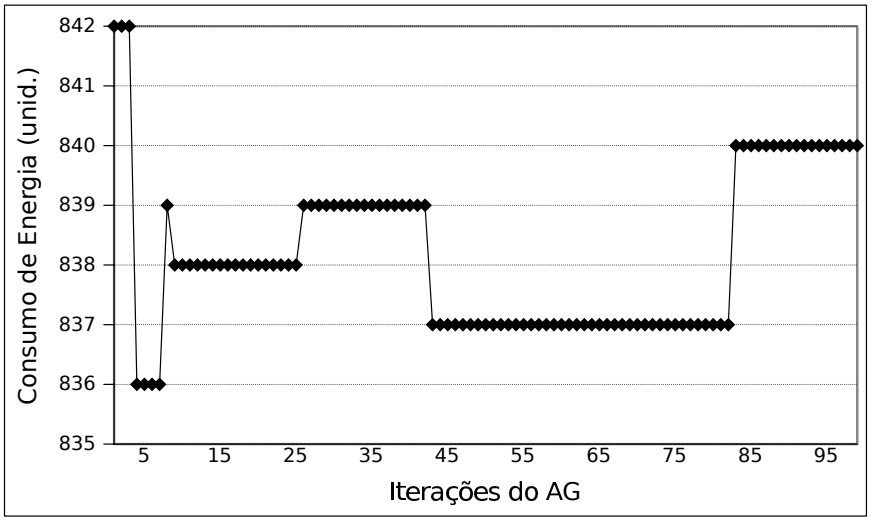

Figura 5.12: Evolução do Consumo de Energia.

- No intervalo em torno da iteração 5 observa-se o menor consumo de energia (836 unidades) e o menor número de roteadores ativos (20 roteadores). A perda de pacotes acompanha essa redução, e fica próxima de 0,7%.

- Em torno da iteração 35, obtém-se a menor taxa de perda de pacotes (em torno de 0,1%), porém é acompanhada por um aumento do consumo de energia (839 unidades) e de roteadores ativos (23 roteadores).

- Em torno da iteração 55, tem-se uma solução ponderada entre os objetivos, com taxa de perda de pacotes próxima de 0,2%, consumo de energia de 837 unidades e 21 roteadores ativos.

Baseado na definição de (SILVA, 2007), uma dada solução x^* domina uma solução x nos casos em que a solução $f(x^*) \leq f(x)$ em todos os objetivos do problemas e, para pelo menos um desses objetivos, x^* é uma solução melhor do que x.

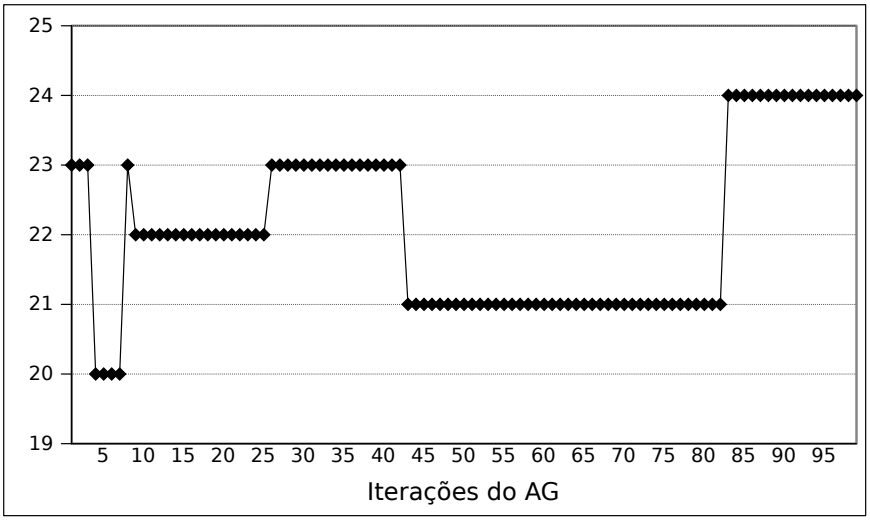

Figura 5.13: Evolução da Quantidade de Roteadores Ativos.

Os três pontos acima são soluções não-dominadas (dentre as encontradas) para a minimização conjunta do consumo de energia e da perda de pacotes para o cenário em questão, conforme ilustra a Fig. 5.14. No entanto, não é possível afirmar que essas soluções delineiam a fronteira de Pareto, uma vez que o método empregado pela estratégia híbrida possui aleatoriedade na busca por soluções.

Os resultados mostram que não é necessário que a variação da perda seja acompanhada pela variação do consumo porque existem inúmeras possibilidades de roteamento com o mesmo consumo, mas que geram diferentes perdas (por exemplo, em torno da iteração 95).

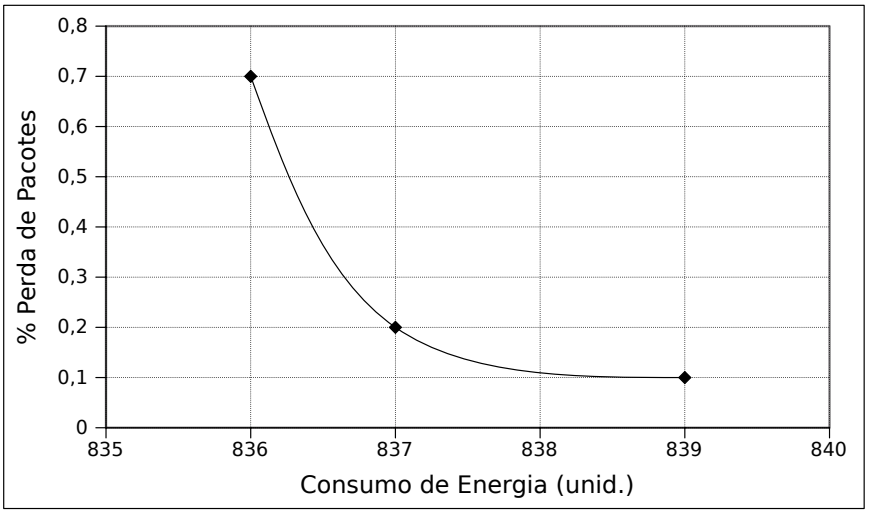

Figura 5.14: Soluções não-dominadas: Consumo de Energia × Perda de Pacotes.

Os resultados que seguem mostram o resultado final obtido com a estratégia híbrida. A Fig. 5.15 ilustra a quantidade de fluxos que percorre os *links*, como solução do modelo de programação linear. É observado que os fluxos de uma mesma origem são capazes de alcançar o destino por rotas diferentes. Isso porque a estratégia híbrida indica a cada iteração por quais *links* o tráfego deve fluir com a menor perda. Também é importante notar que a variação da capacidade dos *links* ocorre apenas para que o *solver* do modelo de programação linear

gere distribuições alternativas. Na simulação da rede, as capacidades originais dos *links* são preservadas. Isso acontece porque a variação da capacidade dos *links* afeta o roteamento do tráfego, e são encontradas rotas alternativas em função da variação da capacidade dos *links* fornecida pelo AG.

A Fig. 5.16 mostra o mapa de alocação obtido ao final da evolução do AG. Para essa função de *fitness*, é observado que a distribuição utiliza preferencialmente os servidores com maior capacidade, mas não é balanceada entre os quatro *data centers*. Por conta disso, é observado nesse experimento que a ponderação não é suficiente para reduzir totalmente a perda de pacotes, uma vez que a otimização também considera o consumo de energia com igual peso. A Tab. 5.5 mostra os resultados da alocação.

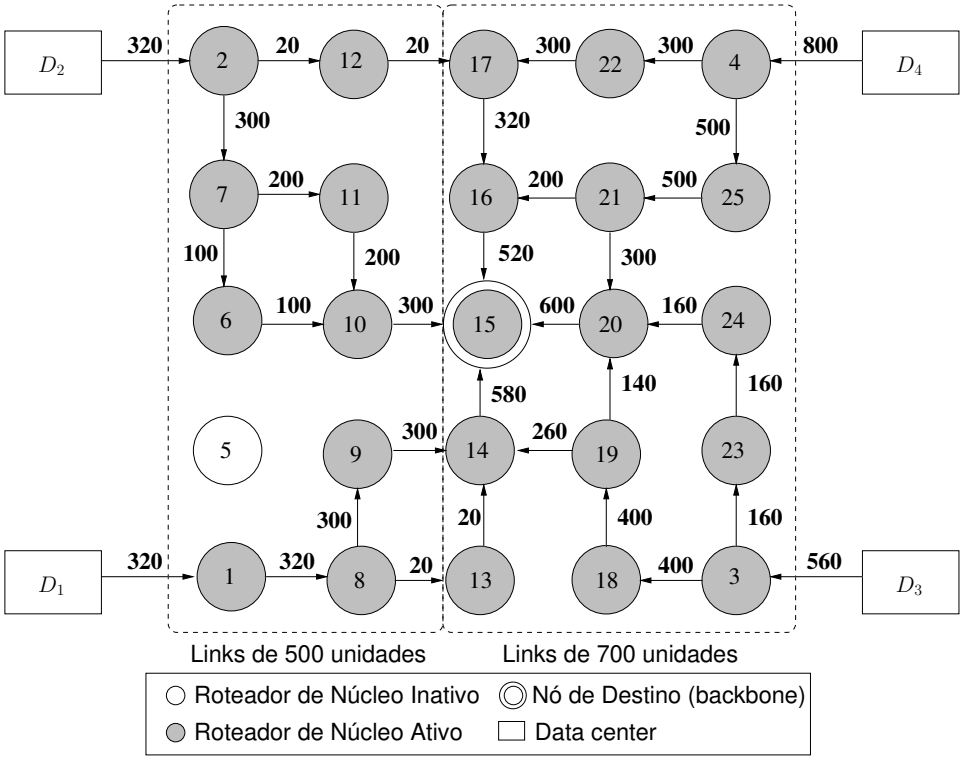

Figura 5.15: Fluxo nos *Links* com a Estratégia Híbrida.

As primeiras iterações das Fig. 5.17 e Fig. 5.18 geram soluções não-factíveis, motivo pelo qual não são mostradas nos gráficos. A Fig. 5.17 mostra a ocorrência de redução dos melhores *fitness* ao longo do processo evolutivo, ou seja, foram exploradas melhores opções de distribuição de VMs em relação à qualidade do tráfego da rede. A Fig. 5.18 mostra o *fitness* médio da população. Nessa figura, os cromossomos com soluções não-factíveis recebem alto valor de *fitness*, ou seja, um valor de grandeza superior à estimativa de maior *fitness* das soluções factíveis. As diversas soluções não-factíveis a cada iteração influenciam o fitness médio da população, motivo pelo qual esses valores se mantêm distantes do melhor *fitness* a cada geração. No entanto, a alta taxa de variação indica que a população não converge para um mínimo local, e o espaço de busca é explorado com diversidade.

A conclusão que se faz desse experimento é que a estratégia híbrida realiza a distribuição de VMs com a otimização do consumo de energia em função do tráfego das VMs na rede. Esses resultados são melhores se comparados à solução exclusiva da distribuição sem considerar os critérios de qualidade do tráfego na rede. Essa otimização depende de como serão atribuídos

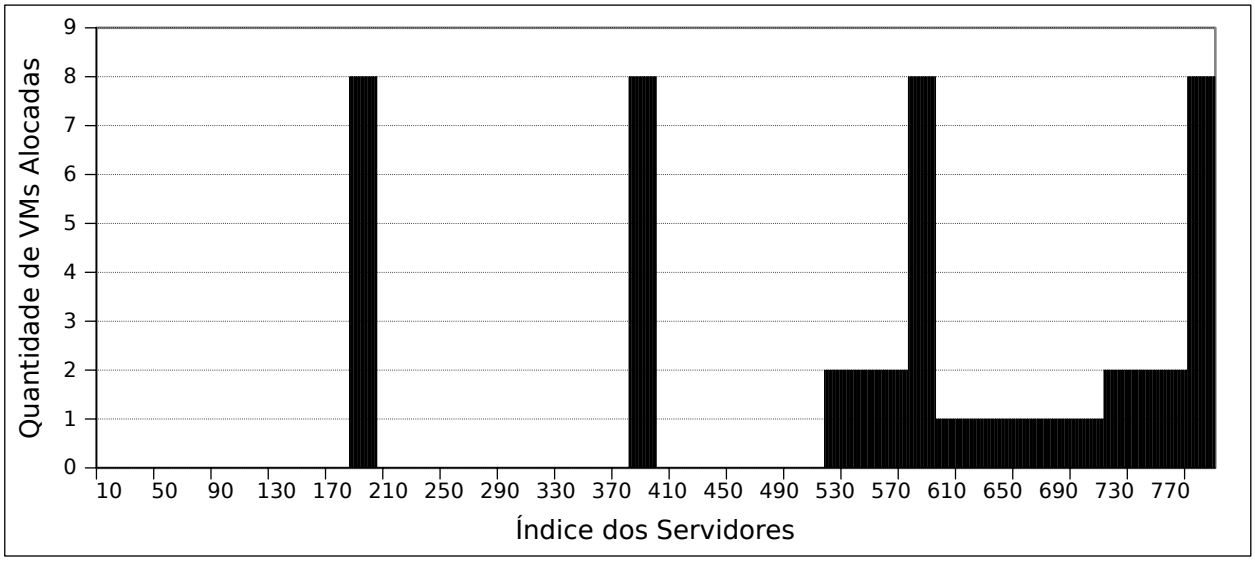

Figura 5.16: Mapa da Alocação com a Estratégia Híbrida.

pesos para os objetivos. Os resultados mostram que essa atribuição de pesos contribui para balancear a distribuição de VMs com melhor qualidade no tráfego da rede. No entanto, é útil que esse resultado seja apoiado pela avaliação da convergência do *fitness* para um número maior de amostras, como apresentado na próxima seção.

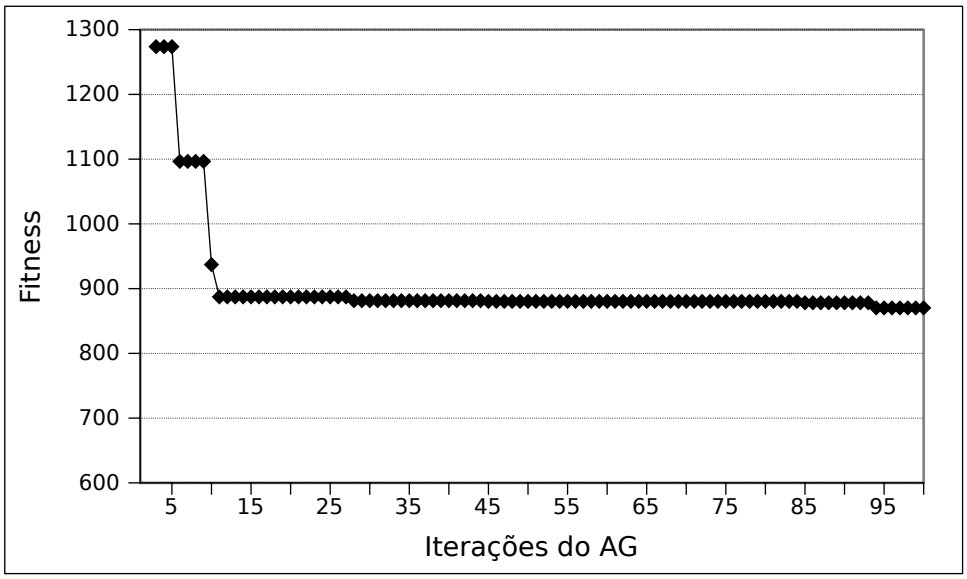

Figura 5.17: Evolução dos Melhores *Fitness*.

5.3.4 Avaliação da Convergência do Fitness

O tempo computacional necessário para executar simulações com a configuração de referência é superior a 5 horas, o que dificulta a avaliação da convergência do *fitness*. Por conta disso, essa seção busca avaliar o processo de convergência em um modelo reduzido, porém com parâmetros de entrada do modelo equivalentes aos da configuração de referência.

	Serv. Tipo 1	Serv. Tipo 2	Serv. Tipo 3	Total
VMs alocadas no *data center* 1	0	0	160	160
VMs alocadas no *data center* 2	0	0	160	160
VMs alocadas no *data center* 3	0	120	160	280
VMs alocadas no *data center* 4	120	120	160	400
Total VMs alocadas	120	240	640	1000/1000
Servidores alocados no *data center* 1	0	0	20	20
Servidores alocados no *data center* 2	0	0	20	20
Servidores alocados no *data center* 3	0	60	20	80
Servidores alocados no *data center* 4	120	60	20	200
Total servidores alocados	120	60	20	320/800
Consumo de energia dos servidores	816 unidades			
Consumo de energia dos roteadores	24 unidades			
Total roteadores internos ativos	24/25			
Tempo de processamento	20401s (∼6h)			

Tabela 5.5: Resultado da Alocação das VMs com a Estratégia Híbrida.

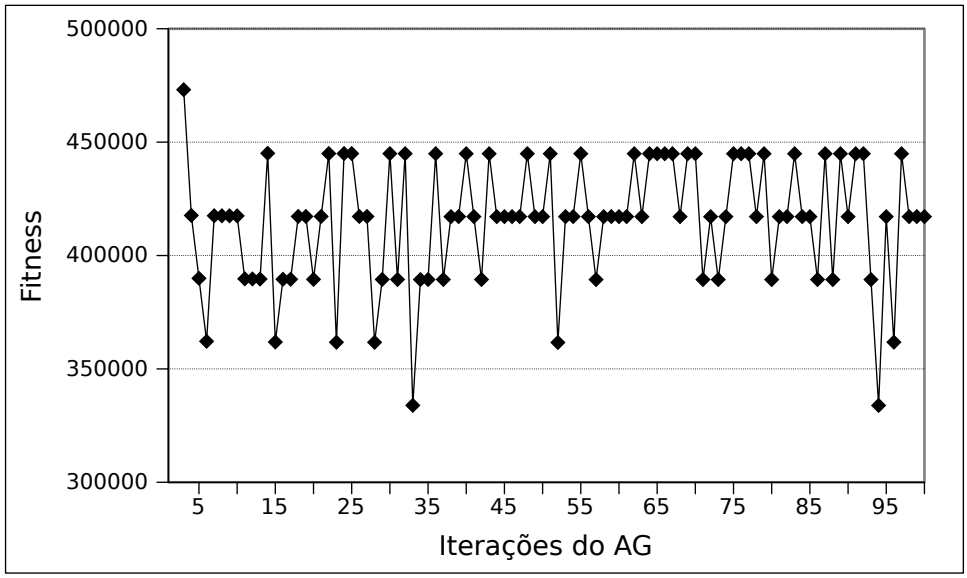

Figura 5.18: Evolução do *Fitness* Médio da População.

A configuração de menor escala utilizada é mostrada na Tab. 5.6. Foi utilizada a mesma topologia de referência. A redução é feita em três pontos: nas capacidades dos *links* de rede, na capacidade dos servidores, e na quantidade de VMs. A motivação é manter a equivalência entre a quantidade de tráfego e avaliar possibilidades de mudança de distribuição de VMs em função da rede. Nos resultados apresentados, em cada iteração do AG foram obtidas 30 amostras para a alocação de 100 VMs, onde cada amostra executa 100 iterações da estratégia híbrida, em paralelo. Entende-se que os resultados obtidos nessa amostragem sejam muito próximos aos que seriam obtidos com a configuração de referência para a alocação de 1000 VMs. A avaliação é feita sobre a média e o desvio-padrão das amostras obtidas.

A Fig. 5.19 mostra a média e o desvio-padrão do *fitness*. Esse resultado mostra uma evolução

Configuração para Servidores	
Quantidade de *data centers*	4 *data centers*
Quantidade de servidores por *data center*	20 servidores
Quantidade de CPU	4,8,32 cores
Servidores Tipo 1 (4 CPUs):	60% (12 servidores)
Servidores Tipo 2 (8 CPUs):	30% (6 servidores)
Servidores Tipo 3 (32 CPUs):	10% (2 servidores)
Largura de Banda	1000 unidades
Configuração para VMs	
Quantidade de VMs	100 VMs
Quantidade de CPU	4 cores
Largura de Banda Reservada	2 unidades
Quantidade de Fluxo Gerado	2 unidades
Configuração da Topologia	
Quantidade de Nós	25 nós
Quantidade de *Links*	40 *links*
Custo dos *Links*	Unitário
Largura de Banda dos *Links*	50 e 70 unidades

Tabela 5.6: Configuração de Menor Escala.

lenta ao longo da iterações, mas ainda é possível observar que o AG contribui para melhorar a qualidade das alocações de VMs. Outra consideração é que o desvio-padrão é maior nas primeiras iterações do AG, o que é esperado, em razão da maior diversidade inicial dos cromossomos. Também é mostrado que a média e o desvio-padrão do *fitness* médio da população mantém-se próximo dos melhores *fitness*. Isso porque o tráfego das VMs encontra muitos *links* que não são sobrecarregados. No entanto, para modelos com maior quantidade de VMs é esperado que exista um maior número de soluções não-factíveis em virtude das capacidades de *links* inferiores ao tráfego que é submetido. Neste caso, os cromossomos recebem alto valor de *fitness*, o valor médio do *fitness* da população será influenciado por esse conjunto, e tende a ficar distante dos melhores *fitness*.

A Fig. 5.20 ilustra as soluções não-dominadas da otimização conjunta do consumo de energia e da perda de pacotes na topologia. Esse resultado mostra que a evolução do AG contribui para reduzir a perda de pacotes até um limiar mínimo (próximo de 100 unidades de consumo) a partir da qual o aumento do consumo é acompanhado pelo aumento da perda de pacotes na rede.

5.4 Alocação Sob Demanda com AG

Data centers possuem de centenas a milhares de servidores, o que torna a modelagem da alocação única de uma grande quantidade de VMs e servidores pouco prática para cenários de grande escala. Nesse sentido, foi proposta nesta seção uma derivação da estratégia híbrida onde VMs são alocadas incrementalmente. O AG atua quando a perda de pacotes ultrapassa o limiar de perda pré-definido. A avaliação desta estratégia é feita na topologia de referência da Fig. 5.1, de acordo com a configuração de referência da Tab. 5.1.

Na primeira etapa, a alocação é feita em grupos de 50 VMs, e a cada alocação com sucesso

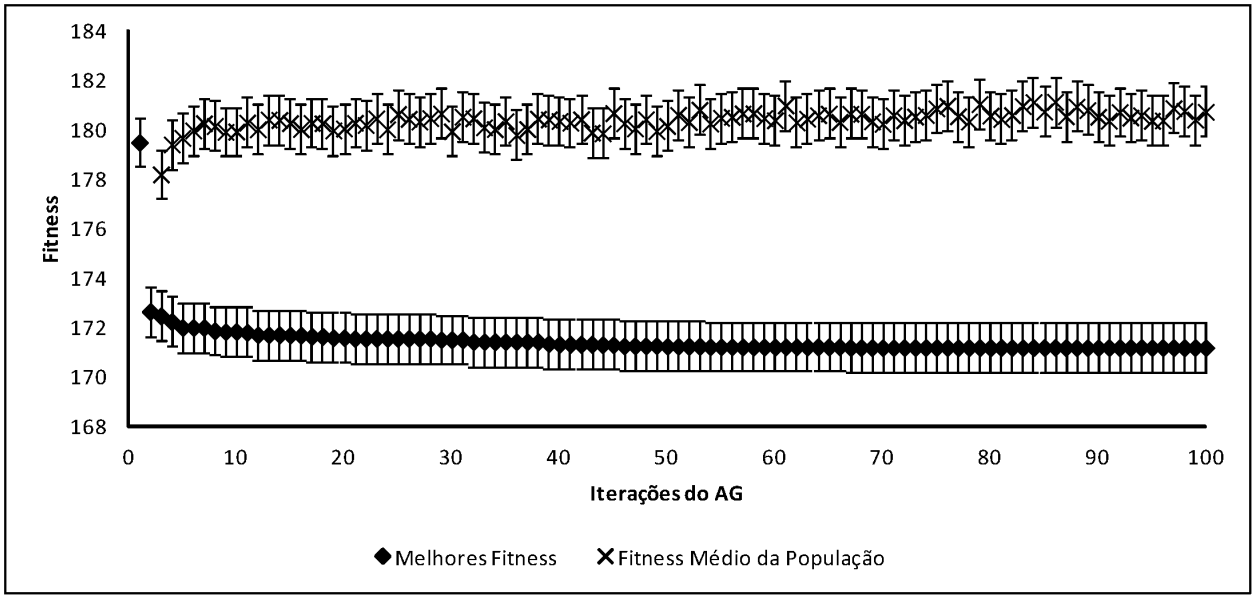

Figura 5.19: Configuração de Menor Escala: Desvio-padrão e Média do Fitness.

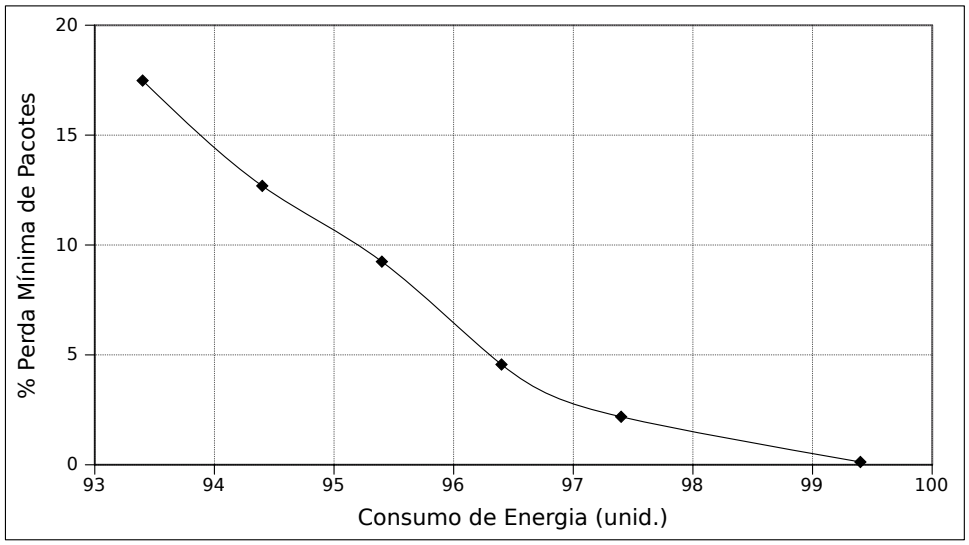

Figura 5.20: Soluções não-dominadas: Consumo de Energia × Perda Mínima de Pacotes.

são decrementados os respectivos recursos dos servidores do *data center* atual. Nota-se que o critério de parada é definido pelo limite de capacidade dos recursos dos servidores e também pelo limite de capacidade da rede. É importante destacar que os limites de capacidade não são conhecidos a priori. O pseudocódigo do algoritmo da política de alocação do modelo sob demanda é mostrado no Algoritmo 2. Nesse algoritmo, para cada alocação de um grupo de VMs, é gerado um modelo de programação linear, com execução de simulação do tráfego na rede de todas as outras VMs previamente alocadas. Os recursos dos servidores são decrementados se a alocação do grupo atual foi realizada com sucesso. O AG atua apenas quando o limiar de perda de pacotes do tráfego gerado pelo novo grupo alocado ultrapassa o limiar de 1,5% de todo o tráfego.

A Tab. 5.7 mostra que é respeitada a propriedade de manter a maior parte das alocações nos

Algoritmo 2: Pseudocódigo da Política de Alocação Sob Demanda com AG.

```
1: while ( grupoVMsRequisitadas ∧ ) do
           ( alocacaoDisponivel     )
2:   alocacaoDisponivel = false
3:   gerarModeloLP()
4:   executarModeloLP()
5:   if (solucaoFactivel) then
6:     alocacaoDisponivel = true
7:     gerarModeloNS2()
8:     executarModeloNS2()
9:     perdaPacotes = parserResultadoNS2()
10:    if (perdaPacotes > LIMIAR_PERDA) then
11:      while (i < MAX_ITERACOES ∧ perdaPacotes > 0) do
12:        executarAG()
13:        i = i + 1
14:      end while
15:    end if
16:    atualizarRecursosConsumidosServidor()
17:    avaliarProximoGrupoVMs()
18:  end if
19: end while
```

servidores de maior capacidade, antes de preencher os demais servidores, e que todas as VMs são alocadas. A quantidade de servidores alocados também manteve-se baixa, mesmo com o aumento da quantidade de requisições, assim como a quantidade de roteadores necessários para prover o tráfego. Finalmente, o tempo de processamento por estágio mostra que essa política continua a ser admissível para a provisão de uma grande quantidade de VMs em topologias de maior escala, mesmo com a atuação do AG. A Fig. 5.21 apresenta o mapa de alocação correspondente aos dados da Tab. 5.7.

A Fig. 5.22 indica o percentual de perda total obtida em todos os *links*. O AG atua apenas quando o limiar de perda ultrapassa 1,5% de todo o tráfego. Esse resultado é importante porque efetivamente mostra que a atuação sob demanda do AG contribui para reduzir a perda que seria obtida com a alocação exclusiva com LP. Como consequência, a alocação até a iteração 19 foi realizada com perdas abaixo de 1% (com exceção da iteração 18). A partir da iteração 19 a atuação do AG não é capaz de reduzir a perda das alocações para abaixo de 1%. No entanto, em praticamente todo o processo de alocação sob demanda a atuação do AG melhorou a qualidade do tráfego na rede. A Fig. 5.23 ilustra os fluxos nos *links* obtidos na última iteração da alocação sob demanda.

A Fig. 5.24 mostra o pequeno aumento gradual da quantidade de servidores alocados à medida que grupos de VMs são recebidos, o que mostra o potencial da alocação sob demanda para reduzir o consumo de energia referente aos servidores necessários para prover a demanda. O desempenho no tráfego da rede é auxiliado pela atuação do AG.

	Serv. Tipo 1	Serv. Tipo 2	Serv. Tipo 3	Total
VMs alocadas no *data center* 1	0	0	160	160
VMs alocadas no *data center* 2	0	50	160	210
VMs alocadas no *data center* 3	0	120	160	280
VMs alocadas no *data center* 4	70	120	160	350
Total VMs alocadas	70	290	640	1000/1000
Servidores alocados no *data center* 1	0	0	20	20
Servidores alocados no *data center* 2	0	25	20	45
Servidores alocados no *data center* 3	0	60	20	80
Servidores alocados no *data center* 4	70	60	20	150
Total servidores alocados	70	145	80	295/800
Consumo de energia dos servidores	811 unidades			
Consumo de energia dos roteadores	25 unidades			
Total roteadores internos ativos	25/25			
Tempo de Processamento	2h26min segundos (∼12 minutos/estágio do AG)			

Tabela 5.7: Resultado da Alocação Sob Demanda de 1000VMs com AG.

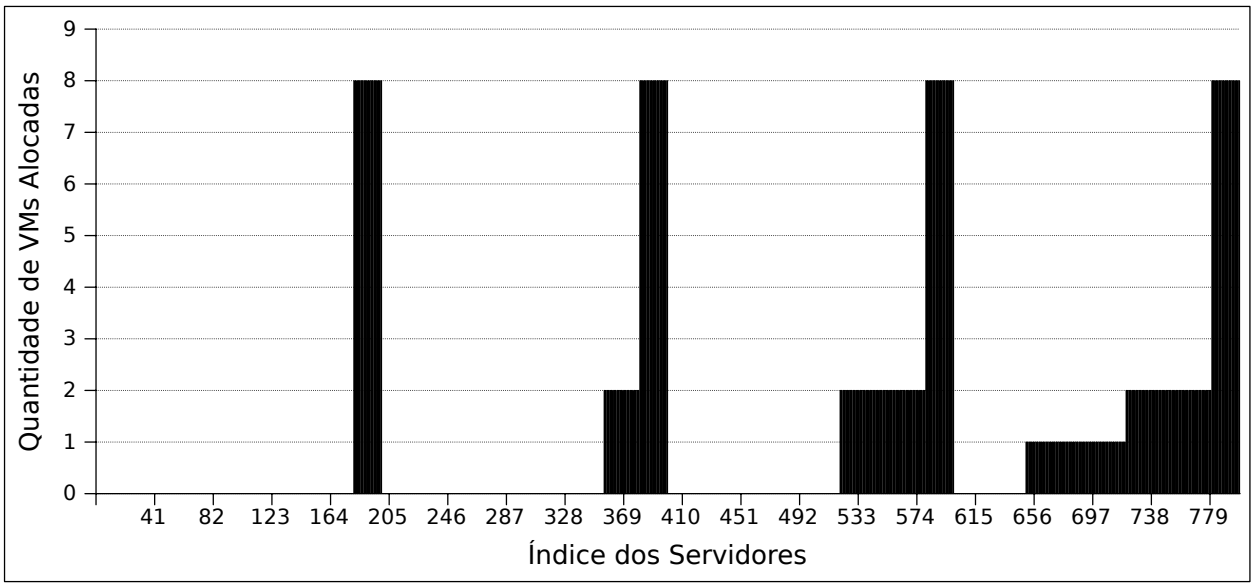

Figura 5.21: Alocação Sob Demanda de 1000VMs: Mapa da Alocação das VMs.

5.5 Alocação com Política Next Fit

Nessa seção é apresentada uma política de alocação mais simples, porém competitiva em termos de desempenho com as demais políticas anteriormente apresentadas. O roteamento utiliza o protocolo OSPF, uma vez que não se tem o modelo de programação linear para indicar os *links* por onde o tráfego deve fluir. A política de alocação NF (*Next Fit*) realiza a alocação sob demanda, promovendo a alocação nos servidores de acordo com a seguinte metodologia:

1. Para cada VM do grupo requisitado, faça a alocação no servidor de maior capacidade do *data center* atual. Este é o servidor candidato;

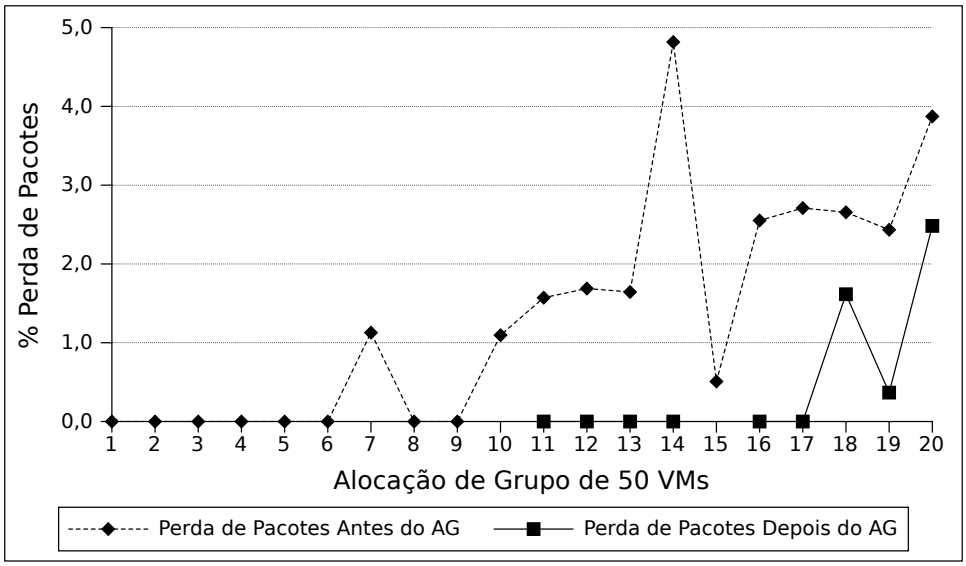

Figura 5.22: Alocação Sob Demanda de 1000VMs: Perda de Pacotes.

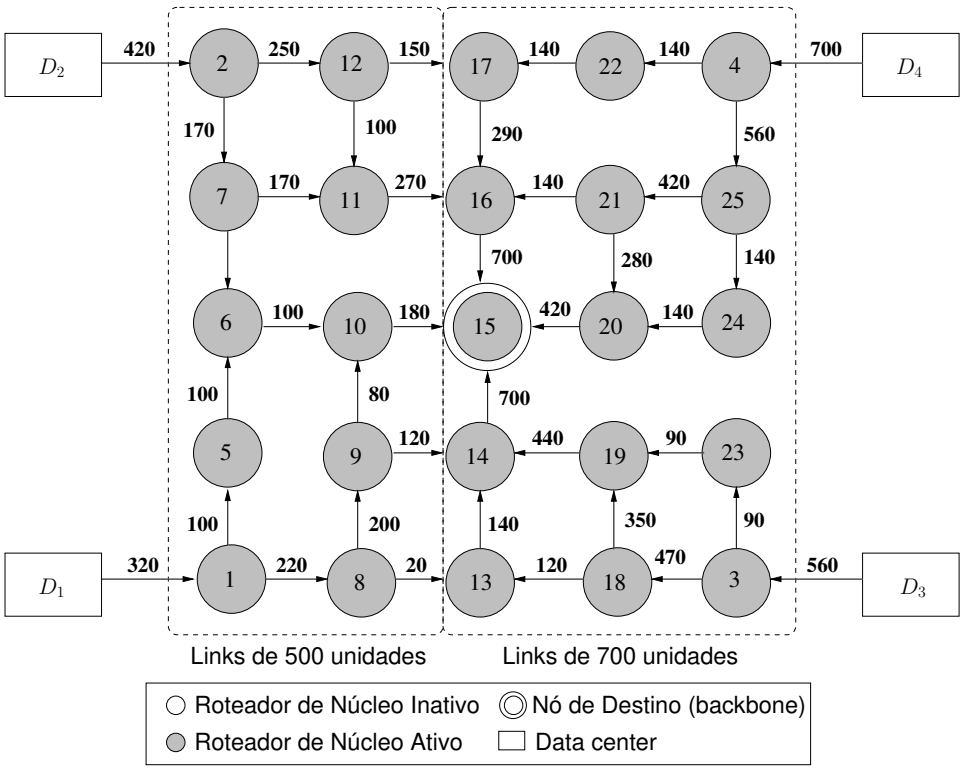

Figura 5.23: Fluxo nos *Links* com a Alocação Sob Demanda e AG.

2. A próxima alocação deve ser feita no próximo (*next*) *data center*, e no servidor candidato de maior capacidade deste;

3. Repita os passos anteriores para as próximas alocações, até o limite de capacidade do servidor candidato de cada *data center*;

4. Ao atingir o limite de capacidade do servidor candidato, eleja o próximo (*next*) servidor

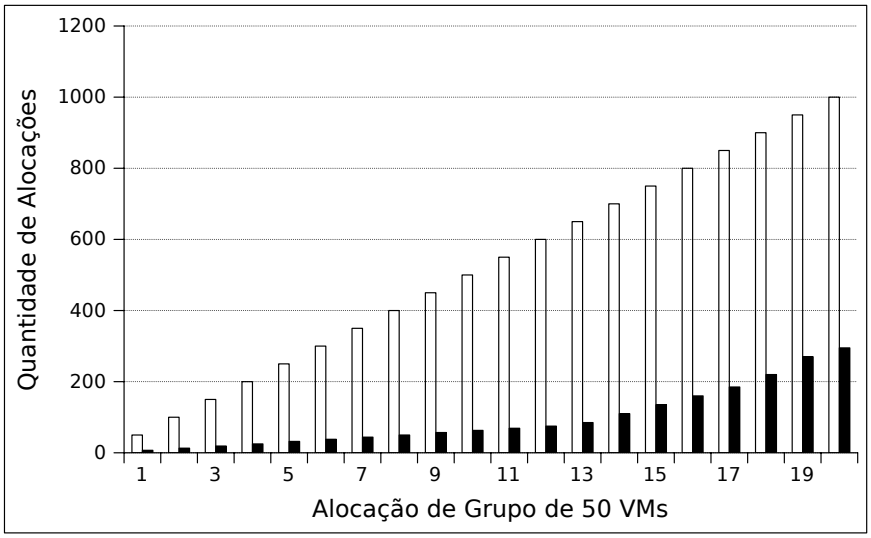

Figura 5.24: VMs Alocadas × Servidores Alocados na Alocação Sob Demanda de 1000VMs.

candidato de maior capacidade do *data center* atual.

5. A partir do item 1, repita o procedimento para os próximos grupos de requisições.

A configuração utilizada busca realizar a alocação de todas as 1000 VMs em grupos de 50 VMs sob demanda. Os requisitos das VMs quanto a CPU, largura de banda e quantidade de fluxo gerado são os mesmos da configuração de referência da Tab. 5.1. Na Tab. 5.8 são apresentados os resultados da distribuição das VMs nos diferentes tipos de servidores, com a política NF. Os resultados apontam uma distribuição balanceada entre os *data centers*.

	Serv. Tipo 1	Serv. Tipo 2	Serv. Tipo 3	Total
VMs alocadas no *data center* 1	0	90	160	250
VMs alocadas no *data center* 2	0	90	160	250
VMs alocadas no *data center* 3	0	90	160	250
VMs alocadas no *data center* 4	0	90	160	250
Total de VMs alocadas	0	360	640	1000/1000
Servidores alocados no *data center* 1	0	45	20	65
Servidores alocados no *data center* 2	0	45	20	65
Servidores alocados no *data center* 3	0	45	20	65
Servidores alocados no *data center* 4	0	45	20	65
Total servidores alocados	0	180	80	260/800
Consumo de energia dos servidores	804 unidades			
Consumo de energia dos roteadores	25 unidades			
Total roteadores internos ativos	25/25			
Tempo de processamento	120 segundos			

Tabela 5.8: Resultado da Alocação de 1000 VMs com Política NF.

Os resultados apresentados nas Fig. 5.25 e Fig. 5.26 são semelhantes aos resultados anteriormente apresentados com a política de alocação sob demanda. Novamente, o pequeno aumento

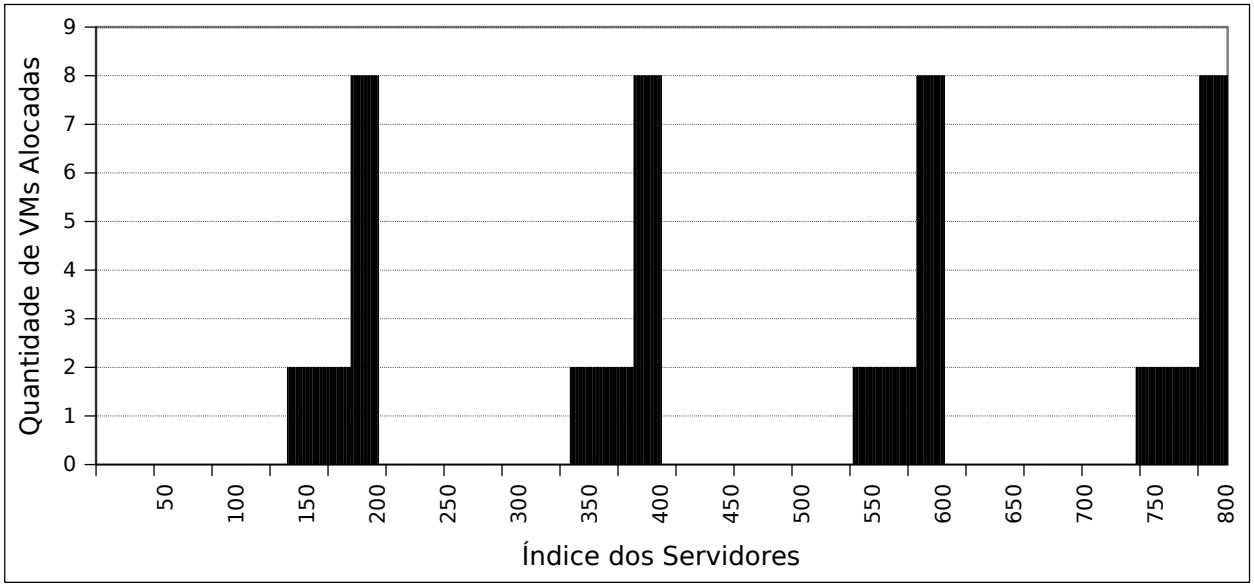

Figura 5.25: Política NF: Mapa da Alocação Sob Demanda de 1000VMs.

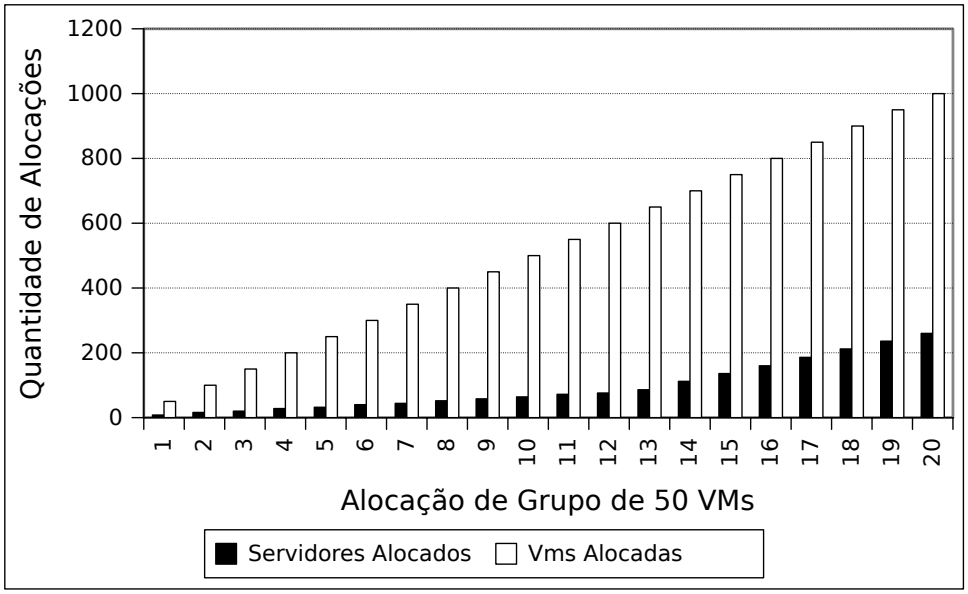

Figura 5.26: Política NF: VMs Alocadas × Servidores Alocados.

gradual da quantidade de servidores alocados à medida que grupos de VMs são recebidos é desejável, o que mostra o potencial da política NF para reduzir o consumo de energia referente aos servidores necessários para prover a demanda. No entanto, o ganho de desempenho é penalizado com a degradação da qualidade do tráfego na rede. Esses resultados mostram que a política é viável em termos da quantidade de VMs distribuídas por servidor e balanceamento da distribuição. À primeira vista esse resultado é melhor ao obtido pela alocação híbrida.

Ao avaliar a qualidade do tráfego na rede, essa abordagem gera resultado com qualidade inferior ao da alocação híbrida. O resultado apresentado na Fig. 5.27 mostra o percentual de perda total de pacotes na topologia. Apesar do tempo de processamento ser relativamente baixo, em torno de 1 minuto, a desvantagem é apresentada quanto à qualidade do tráfego

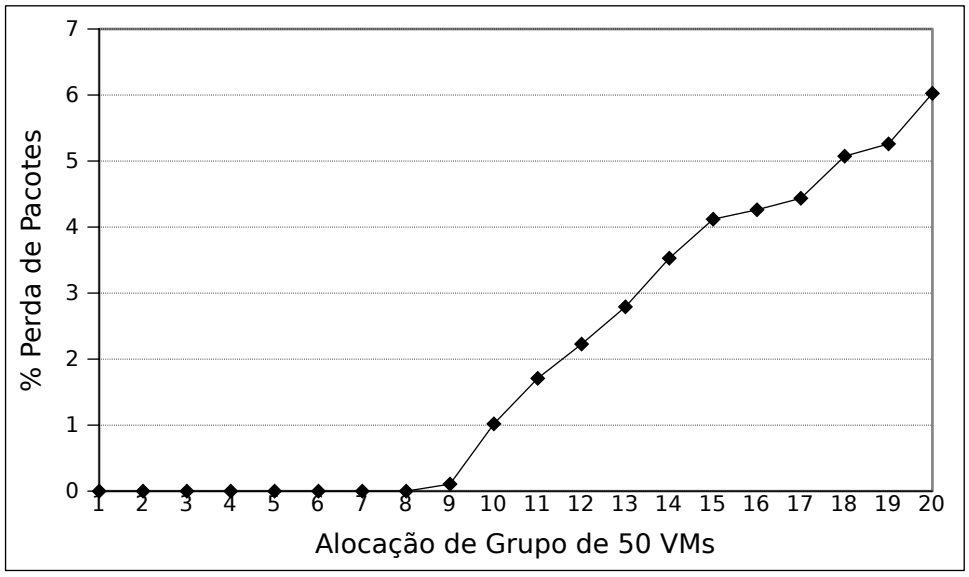

Figura 5.27: Política NF: Resultado da Perda de Pacotes.

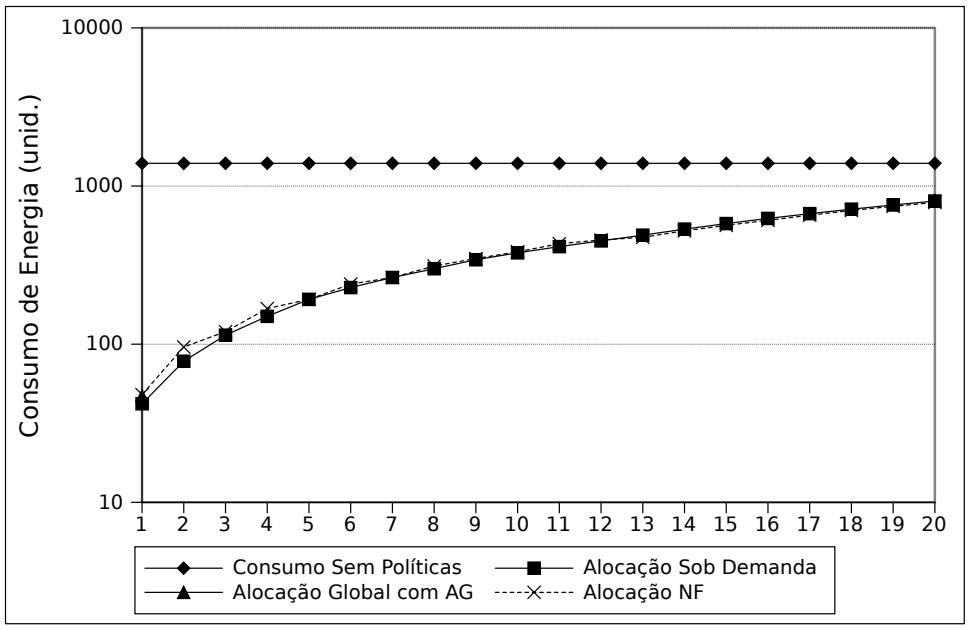

Figura 5.28: Estimativa do Consumo de Energia das Diferentes Políticas (Escala em log).

na rede. Ou seja, à medida que mais VMs são alocadas, a política NF não é suficiente para acomodar o tráfego com o objetivo de reduzir a perda de pacotes na rede. Como consequência, a porcentagem de perda de pacotes aumenta progressivamente ao longo das alocações. A Tab. 5.9 ilustra os valores utilizados para estimar o consumo de energia das diferentes abordagens.

	Servidor Tipo 1	Servidor Tipo 2	Servidor Tipo 3
Valor de Consumo (unid.)	1	1.8	6

Tabela 5.9: Valores utilizados para Estimar o Consumo de Energia.

	Servidores Ativos			VMs por Data Center			
Estratégia	T_1	T_2	T_3	D_1	D_2	D_3	D_4
Alocação Global com LP	0	180	80	250	250	250	250
Estratégia Híbrida	120	60	20	160	160	280	400
Sob Demanda com AG	70	145	80	160	210	280	350
Next-Fit	0	180	80	250	250	250	250

Tabela 5.10: Resumo da Alocação de Servidores e VMs das Estratégias Avaliadas.

Estratégia	Roteadores Ativos	Links Ativos	Energia (unid.)	Perda Pacotes	Tempo Execução
Alocação Global com LP	17	27	821	4.5%	120 s.
Estratégia Híbrida	24	34	840	0.1%	10,200 s.
Sob Demanda com AG	25	39	811	2.6%	744 s./estágio do AG
Next-Fit	25	44	829	6.0%	60 s.

Tabela 5.11: Resumo do Desempenho das Estratégias Avaliadas.

A Fig. 5.28 mostra a comparação das estimativas de consumo de energia das diferentes políticas abordadas. Observa-se que o consumo de energia estimado tende a ser o mesmo. Esse resultado é esperado, uma vez que a distribuição segue balanceada nessas abordagens. A diferença está no roteamento entre as fontes e destinos do tráfego. Nesse contexto, a abordagem híbrida é a estratégia que apresentou o melhor ganho em relação à qualidade do tráfego. A política NF produz um consumo próximo às demais abordagens, porém com menor qualidade quanto ao tráfego na rede.

5.6 Considerações Finais

Este capítulo descreve a implementação e os resultados obtidos com a estratégia híbrida de alocação para ambientes de nuvem. Os resultados indicam diversas possibilidades de uso da estratégia híbrida em ambientes de nuvem e faz a interpretação dos resultados obtidos em cada uma delas.

As Tab. 5.10 e Tab. 5.11 fazem um resumo dos resultados obtidos pelas 4 estratégias de alocação de 1000 VMs. Em termos de consumo as 4 estratégias são similares. Isso é esperado devido à simetria dos *data centers* e ao fato de que todas as estratégias favorecem servidores com alta capacidade, com baixos custos por VM alocada. No entanto, as diferenças em termos de tráfego na rede são consideráveis. A política Next-Fit alcança a pior desempenho, em torno de 6,0% de perda de pacotes. As estratégias de alocação global e sob demanda são capazes de reduzir a perda de pacotes fornecendo um melhor roteamento, se comparado com o protocolo OSPF. Com a estratégia híbrida é possível escolher soluções não-dominadas que fornecem baixa perda de pacotes e baixo consumo de energia.

A alocação única com o modelo de programação linear mostra que a modelagem é adequada para distribuir de forma balanceada as VMs entre os servidores. Porém, ao considerar o tráfego na rede, essa distribuição não é suficiente para acomodar o tráfego sem reduzir a qualidade de serviço na rede.

O uso da estratégia híbrida é auxiliada com o paralelismo no processamento dos cromossomos. Isso é necessário para reduzir o tempo de processamento da distribuição fornecida. O

Estratégia	Principais Pontos Positivos	Principais Pontos Negativos
Alocação Global Única	• Alocação simultânea de todas as VMs. • Otimização do consumo global de energia.	• Não considera qualidade de serviço na rede.
Alocação com a Estratégia Híbrida	• Otimização combinada de energia e qualidade de serviço na rede; • Alto potencial de paralelismo.	• Tempo computacional de solução elevado.
Alocação Sob Demanda com AG	• Tempo computacional reduzido por estágio do AG; • Otimização do consumo global de energia com estimativa da qualidade de serviço na rede.	• Exige a criação de um novo modelo a cada iteração, para reduzir os recursos consumidos na iteração anterior.
Alocação com a Política NF	• Tempo computacional reduzido; • Política de implementação simples.	• Não considera a qualidade de serviço na rede; • Considera apenas o consumo de energia nos servidores.

Tabela 5.12: Resumo das Principais Características das Estratégias Abordadas.

ganho indicado pela evolução do *fitness* é obtido por novos roteamentos avaliados no processo evolutivo, ao mesmo tempo em que considera a minimização do consumo de energia em função da distribuição de VMs. Ainda assim, a estratégia híbrida mantém o balanceamento da distribuição. No entanto, a função de *fitness* apresenta objetivos que precisam ser ponderados. Além disso, deve ser atribuído um peso relativo à importância de cada objetivo. Como resultado, a mudança da função de *fitness* afeta a distribuição de VMs. Em função do tempo elevado de simulação, a simulação em um modelo com menor quantidade de VMs mostra que o *fitness* evolui lentamente, com maior desvio-padrão no início da simulação, em razão da maior diversidade da população.

Os métodos de alocação sob demanda promovem a alocação de VMs tão logo elas sejam requisitadas, além de simplificar o tratamento das requisições. Os resultados mostram que a atuação do AG é importante para otimizar o tráfego das VMs. A política de alocação sob demanda NF apresenta o menor tempo de execução por ser a política de alocação mais simples em termos de implementação. Porém, a qualidade de serviço na rede é prejudicada em função dessa simplicidade.

A estimativa do consumo de energia das abordagens contempladas é muito semelhante. Isso é esperado porque a estratégia híbrida e a estratégia sob demanda consideram um modelo base semelhante de programação linear. Esse consumo é inferior ao consumo que seria obtido com todos os servidores ativos. Quanto à política NF, o consumo tende a ser próximo à estratégias anteriores, o que mostra o potencial dessa técnica quanto à distribuição de VMs com foco apenas no consumo de energia. No entanto, essa simplicidade não leva em consideração os parâmetros de rede, o que não a torna adequada quanto ao critério de qualidade de serviço na rede. Dentre as abordagens estudadas, a estratégia híbrida é a que melhor contribui para otimizar a qualidade das distribuições em função do consumo de energia e da qualidade do tráfego da rede. A Tab. 5.12 faz um resumo das principais características de cada estratégia abordada. O próximo capítulo faz as considerações finais e indica trabalhos futuros.

Capítulo 6

Considerações Finais e Trabalhos Futuros

O crescimento dos mercados de computação em nuvem tem despertado a atenção para otimizar a alocação de recursos nos *data centers*. Uma visão que tem sido difundida é de se ter a infraestrutura do *data center* como um sistema computacional remoto (BARROSO; HÖLZLE, 2009). Em um cenário onde diferentes provedores de nuvem podem adquirir e oferecer recursos entre domínios parceiros, as restrições de rede devem ser consideradas no processo de alocação de recursos, ainda que esse requisito aumente a complexidade do problema. Nesse sentido, as técnicas de alocação de recursos em nuvem devem considerar a dinâmica da rede, ao mesmo tempo em que se busca melhorar a eficiência no uso dos recursos oferecidos.

6.1 Contribuições

Nesta tese foi desenvolvida uma nova estratégia para a alocação de VMs com o objetivo de otimizar a eficiência do consumo de energia em infraestruturas de computação em nuvem. As principais contribuições dessa tese são enumeradas como segue:

- Uma estratégia híbrida que contempla os agrupamentos de servidores de *data centers* e as redes de comunicação entre esses *data centers*. O objetivo é realizar a alocação de VMs para que a energia consumida nos servidores e na rede seja minimizada, mantendo-se ainda níveis aceitáveis de QoS na rede. A abordagem proposta na tese combina Algoritmos Genéticos, Programação Linear Inteira Mista e simulação das redes de comunicação.

- Especificamente em relação aos métodos de alocação de recursos, a literatura mostra que esse é um problema em aberto, uma vez que cada *data center* possui características diferentes dos demais. Em virtude disso, a taxonomia proposta nesta tese simplifica o entendimento das principais características de uma ou outra abordagem.

- Desenvolvimento do *software* de otimização REALcloudSim para simular as alocações de VMs de forma a combinar a modelagem em MILP com algortimos genéticos, e a simulação do tráfego de rede, de forma a simplificar o estudo e análise da viabilidade das alocações em redes de referência. O *software* permite a inclusão de novas políticas de alocação a partir do componente Política de Alocação (vide Anexo A).

6.2 Trabalhos Futuros

Nesta tese, algumas possibilidades de trabalhos futuros na linha de pesquisa são descritas como segue:

- Uso da estratégia híbrida em redes reais. Como exemplo, a aplicação de um usuário de instrumentação robótica poderia estar dentro de uma VM. Nesse sentido, qual seria o *host* com a melhor qualidade de serviço de rede para alocar essa VM do usuário? O autor dessa tese verificou que a alocação de VMs próxima a recursos robóticos, no nível de rede, contribui para melhorar o desempenho de aplicações sensíveis à carga da rede (ROCHA et al., 2012). No entanto, múltiplos usuários podem vir a requisitar diferentes recursos robóticos de maneira concorrente. O uso da estratégia híbrida poderia ser usada para indicar a melhor alocação, no nível de rede, para migrar VMs, de forma que a comunicação fim-a-fim entre o usuário e o recurso robótico tenha a melhor qualidade de serviço quanto ao tráfego de rede, ainda que existam diversos usuários competindo por diferentes recursos.

- Integração da estratégia de otimização proposta com métodos de engenharia de tráfego em redes. A estratégia híbrida indica quais LSPs devem ser estabelecidos na rede para que o tráfego possa fluir com a melhor qualidade de serviço obtida ao final do processo evolutivo. Ao menos duas possibilidades de uso desse resultado são possíveis: em redes reais convencionais via MPLS; e em redes SDN (*Software Defined Networking*). Nesse último caso, os LSPs poderiam ser estabelecidos com o protocolo OpenFlow (OPENFLOW.ORG, 2012) que é um padrão aberto adicionado como um protocolo a muitos *switches* Ethernet comerciais, roteadores, e pontos de acesso, sem precisar expor as funcionalidades internas dos dispositivos de rede. O protocolo OpenFlow separa as opções de encaminhamento de pacotes (*data path*) e decisões de roteamento (*control path*). O encaminhamento de pacotes é mantido no *switch*: uma tabela de fluxos define ações a serem realizadas, tais como encaminhar pacotes, modificar campos nos pacotes, ou descartá-los. Porém as decisões de roteamento são feitas em um servidor separado: os pacotes que não se enquadram nas tabelas do *data path*, são enviados para o *control path*, que pode decidir continuar com o encaminhamento, modificar campos, descartar os pacotes, ou adicionar entradas no *switch* sobre como encaminhar pacotes similares que venham a surgir.

- O resultado do AG pode ser aprimorado com o acréscimo de parâmetros de engenharia de tráfego no *rankeamento* da população. Por exemplo, balanceamento de carga na rede e estabelecimento de caminhos de *backup*.

- A grande quantidade de variáveis que precisam ser processadas na definição de problemas de alocação de VMs sugere o uso de mecanismos mais eficientes para reduzir o tempo computacional necessário para se obter a solução ótima de alocação. O uso de GPUs (*Graphics Processing Unit*) (NVIDIA, 2012) é uma alternativa que pode ser considerada para acelerar esse processamento, com a utilização de centenas de *cores* para processamento paralelo. No caso da estratégia híbrida, uma possibilidade é o uso de paralelismo junto ao *solver* de programação linear inteira.

Produção Bibliográfica

1. ROCHA, L. A. et al. Heurísticas para Reserva e Negociação de Recursos Intra-Domínio em WebLabs com Suporte a DiffServ. In: XXXV CONFERÊNCIA LATINO-AMERICANA DE INFORMÁTICA. Pelotas: CLEI, 2009.

2. ROCHA, L. A. et al. Uma Infraestrutura para Experimentos Robóticos Bio-inspirados em Grades Colaborativas. In: VIII WORKSHOP EM CLOUDS, GRIDS E APLICAÇÕES - XXVIII SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS. Gramado: SBRC, 2010.

3. CARDOZO, E. et al. A Platform for Networked Robotics. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS. Taipei, Taiwan: IROS, 2010.

4. ROCHA, L. A. et al. NetLab Web Lab: um laboratório de experimentação remota para o ensino de redes de computadores baseado em SOA. IEEE Latin America Transactions, Revista IEEE America Latina, v.8, n.5, 2010.

5. SOUZA, R. et al. Control of Mobile Robots Through Wireless Sensor Networks.In: XXIX SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS. Campo Grande: SBRC, 2011.

6. FELICIANO, G. O. et al. Uma Arquitetura para Gerência de Identidades em Nuvens Híbridas. In: IX WORKSHOP EM CLOUDS, GRIDS E APLICAÇÕES - XXIX SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS. Campo Grande: SBRC, 2011.

7. FELICIANO, G. O. et al. Gerência de Identidades Federadas em Nuvens: Enfoque na Utilização de Soluções Abertas. In: XI SIMPÓSIO BRASILEIRO EM SEGURANÇA DA INFORMAÇÃO E DE SISTEMAS COMPUTACIONAIS. Brasília: SBSEG, 2011.

8. ROCHA, L. A. et al. A Cloud Computing Environment for Supporting Networked Robotic Applications. In: 3RD INTERNATIONAL WORKSHOP ON WORKFLOW MANAGEMENT IN SERVICE AND CLOUD COMPUTING. Sydney, Austrália: WMSC, 2011.

9. ROCHA, L. A. et al. A Bio-inspired Approach to Provisioning of Virtual Resources in Federated Clouds. In: INTERNATIONAL CONFERENCE ON DEPENDABLE, AUTONOMIC, AND SECURE COMPUTING (DASC). Sydney - Austrália: CGC, 2011. p.598-604.

10. ROCHA, L. A. et al. Advances in Educational Robotics in Cloud with Qualitative Scheduling in Workflows. In: MAHMOOD, Z.; HILL, R. Computer Communications and Networks. University of Derby, London: Springer, 2012. cap.7, p.135-157.

Resumos Expandidos Publicados em Workshops

1. ROCHA, L. A. et al. Um Modelo Federado em Cloud Computing para Infraestruturas Robóticas. In: IV ENCONTRO DOS ALUNOS E DOCENTES DO DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL. Campinas, SP: Faculdade de Engenharia Elétrica e de Computação. Unicamp, 2011.

Bibliografia

ADELIN, A.; OWEZARSKI, P.; GAYRAUD, T. On the impact of monitoring router energy cansumption for greening the internet. In: *11TH IEEE/ACM INTERNATIONAL CONFERENCE ON GRID COMPUTING*. Brussels: GRID, 2010. p. 298–304.

AGUADO, A.; CANTANHEDE, M. *Lógica Fuzzy*. Universidade Estadual de Campinas, 2010. Acesso em: dezembro de 2011. Disponível em: `http://www.ft.unicamp.br/liag/wp/monografias/monografias/2010_IA_FT_UNICAMP_logicaFuzzi.pdf`.

AHRONOVITZ, M. et al. *Cloud Computing Use Cases, Version 4.0*. 2010. Disponível em: <http://cloudusecases.org/Cloud_Computing_Use_Cases_Whitepaper-4_0.odt>. Acesso em: dezembro de 2010.

ALICHERRY, M.; LAKSHMAN, T. V. Network aware resource allocation in distributed clouds. In: *IEEE CONFERENCE ON COMPUTER COMMUNICATIONS*. Orlando: IEEE INFOCOM, 2012. p. 963–971.

ALLAN, D. et al. *A Framework for Multiprotocol Label Switching (MPLS) Operations and Management*. 2006. Internet Draft. RFC 4378. Acesso em: dezembro de 2011. Disponível em: `http://tools.ietf.org/html/rfc4378`.

AMAZON WEB SERVICES. *Amazon Elastic Compute Cloud (Amazon EC2)*. 2012. Disponível em: <http://aws.amazon.com/ec2>. Acesso em: dezembro de 2012.

ARMBRUST, M. et al. *Above the Clouds: A Berkley View of Cloud Computing*. EECS Department, University of California, 2009. Acesso em: dezembro de 2009. Disponível em: `http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf`.

BALANDIN, S.; HEINER, A. P. Ospf protocol and statistical tools for network simulations in ns-2. *24TH INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY INTERFACES*, Cavtat, v. 1, p. 465–470, 2002.

BALDI, M.; OFEK, Y. Time for a greener internet. In: *IEEE GREEN COMMUNICATIONS (GreenComm'09) - IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS*. Dresden: IEEE ICC, 2009.

BARROSO, L. A.; HÖLZLE, U. Energy and power efficiency. In: _____ (Ed.). *The Datacenter as a Computer - An Introduction to the Design of Warehouse-scale Machines*. University of Wiscosin, Madison: Morgan & Claypool Publishers, 2009. p. 50.

BELOGLAZOV, A.; ABAWAJY, J.; BUYYA, R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, [S.l.], v. 8, n. 5, p. 755–768, 2012.

BIANZINO, A. P. et al. A survey of green networking research. *IEEE Communications Surveys and Tutorials*, [S.l.], v. 14, n. 1, p. 3–20, 2012.

BIN, E. et al. Guaranteeing high availability goals for virtual machine placement. In: *31ST INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS*. Minneapolis: ICDCS, 2011. p. 700–709.

BIRAN, O. et al. A stable network-aware vm placement for cloud systems. In: *IEEE/ACM INTERNATIONAL SYMPOSIUM ON CLUSTER, CLOUD AND GRID COMPUTING*. Ottawa: IEEE Computer Society, 2012. p. 498–506.

BOBROFF, N.; KOCHUT, A.; BEATY, K. Dynamic placement of virtual machines for managing sla violations. In: *10TH IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT*. Munich: IFIP, 2007. p. 119–128.

BONDE, D. V. *Adaptative Provisioning of Virtual Machines*. 56 p. Dissertação (Mestre de Tecnologia em Ciência da Computação e Engenharia) — Indian Institute of Technology. Department of Computer Science and Engineering, Bombay, 2011.

BREITMAN, K.; VITERBO, J. Computação na nuvem - uma visão geral. In: *III CONGRESSO INTERNACIONAL SOFTWARE LIVRE E GOVERNO ELETRÔNICO. Amãpytuna - Computação em Nuvem: serviços livres para a sociedade do conhecimento.* Brasília: Fundação Alexandre de Gusmão, 2010. cap. 1, p. 17–45.

BUNGEE LABS. *Bungee Connect*. 2010. Acesso em: dezembro de 2012. Disponível em: `http://www.bungeeconnect.com`.

CALCAVECCHIA, N. M. et al. Vm placement strategies for cloud scenarios. In: *IEEE 5th INTERNATIONAL CONFERENCE ON CLOUD COMPUTING*. Honolulu: IEEE CLOUD, 2012. p. 852–859.

CALHEIROS, R. N. et al. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience (SPE)*, New York, v. 41, n. 1, p. 23–50, 2012.

CALVO, S. *Máxima Eficiência Energética é a Meta dos Data Centers Brasileiros*. 2013. Acesso em: junho de 2013. Disponível em: `http://computerworld.uol.com.br/tecnologia/2012/06/01/maxima-eficiencia-energetica-e-a-meta-dos-data-centers-brasileiros/`.

CANONICAL. *Ubuntu Enterprise Cloud*. [S.l.], 2012. Acesso em: dezembro de 2012. Disponível em: `http://www.ubuntu.com/system/files/UEC_Brochure.pdf`.

CASTRO, L. N. de; ZUBEN, F. J. V.; ATTUX, R. R. F. *Tópico 1: Introdução à Computação Natural. Apostila do Curso de Computação Evolutiva*. 2009. Acesso em: dezembro de 2009. Disponível em: `ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia006_03/topico1_03.pdf`.

CASTRO, R. E. de. *Otimização de Estruturas com Multi-objetivos via algoritmos Genéticos de Pareto*. Tese (Doutorado em Ciências em Engenharia Civil) — Universidade Federal do Rio de Janeiro, Coordenação dos Programas de Pós-Graduação de Engenharia (COPPE/UFRJ). Rio de Janeiro, 2001.

CHINNECKE, J. W. *Chapter 13: Binary and Mixed-Integer Programming. Practical Optimization: A Gentle Introduction*. 2012. Acesso em: janeiro de 2011. Disponível em: `http://www.sce.carleton.ca/faculty/chinneck/po.html`.

CHINNECKE, J. W. *Chapter 3: Towards the Simplex Method for Efficient Solution of Linear Programs. Practical Optimization: A Gentle Introduction*. 2012. Acesso em: janeiro de 2011. Disponível em: `http://www.sce.carleton.ca/faculty/chinneck/po.html`.

CISCO SYSTEMS. *Cisco EnergyWise Design Guide*. 2012. Acesso em: dezembro de 2012. Disponível em: `http://www.cisco.com/en/US/docs/solutions/Enterprise/Borderless-Networks/Energy-Management/energywisedg.pdf`.

CISCO SYSTEMS. *Cisco IOS MIB Tools*. 2012. Acesso em: dezembro de 2012. Disponível em: `http://tools.cisco.com/ITDIT/MIBS/servlet/index`.

CISCO SYSTEMS. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2012 - 2017*. 2012. Acesso em: dezembro de 2012. Disponível em: `http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html`.

CLOUD-STANDARDS.ORG. *Cloud Storage for Cloud Computing*. 2010. Disponível em: <http://cloud-standards.org/wiki/index.php?title=Demos>. Acesso em: dezembro de 2010.

CPU WORLD. *Thermal Design Power*. 2012. Acesso em: dezembro de 2012. Disponível em: `http://www.cpu-world.com/Glossary/T/Thermal_Design_Power_(TDP).html`.

CUNNINGHAM, L. *Defining Cloud Computing: Part 6 - IaaS*. 2009. Disponível em: <http://clouddb.info/2009/02/23/defining-cloud-computing-part-6-iaas/>. Acesso em: dezembro de 2009.

DARWIN, C. *A Origem das Espécies (versão traduzida)*. Porto, Portugal: Lello & Irmão, 2003. E-book baseado na tradução de Joaquim da Mesquita Paul. Disponível em: <http://ecologia.ib.usp.br/ffa/arquivos/abril/darwin1.pdf>. Acesso em: dezembro de 2009.

EIBEN, A. E.; SMITH, J. E. Genetic algorithms. In: _____ (Ed.). *Introduction to Evolutionary Computing*. 1. ed. New York: Springer-Verlag Berlin Heidelberg, 2007. cap. 3.

ENDO, P. T. et al. A survey on open-source cloud computing solutions. In: *VIII WORKSHOP EM CLOUDS, GRIDS E APLICAÇÕES. XXVIII SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS*. Gramado: UFRGS, 2010.

ENERGY STAR. *How a Product Earns the ENERGY STAR Label*. 2012. Acesso em: dezembro de 2012. Disponível em: `http://www.energystar.gov/index.cfm?c=products.pr_how_earn`.

EUCALYPTUS. *Eucalyptus Cloud Documentation. Eucalyptus Documentation*. 2012. Acesso em: dezembro de 2012. Disponível em: `http://www.eucalyptus.com/docs/eucalyptus/3.3/faststart-guide/`.

EUROPEAN COMMISION. *Roadmap for moving to a low-carbon economy in 2050*. 2012. Acesso em: dezembro de 2012. Disponível em: http://ec.europa.eu/clima/policies/roadmap/index_en.htm.

FACEBOOK. *Facebook is on Facebook*. 2012. Acesso em: dezembro de 2012. Disponível em: http://www.facebook.com/facebook.

FETTWEIS, G.; ZIMMERMANN, E. Ict energy consumption - trends and challenges. In: *11TH INTERNATIONAL SYMPOSIUM ON WIRELESS PERSONAL MULTIMEDIA COMMUNICATIONS*. Lapland: WPMC, 2008. p. 2006–2009.

FLEXIANT. *FlexiScale public cloud*. 2010. Acesso em: dezembro de 2010. Disponível em: http://www.flexiant.com/products/flexiscale.

FONTECCHIO, M. *Power Usage Effectiveness (PUE)*. 2012. Acesso em: dezembro de 2012. Disponível em: http://searchdatacenter.techtarget.com/definition/power-usage-effectiveness-PUE.

FRANGULYAN, K. *Next Generation is for the Cloud?* 2009. Acesso em: dezembro de 2009. Disponível em: http://www.searchenginejournal.com/next-generation-is-for-the-cloud/13539/.

GLANZ, J. The cloud factories: Power, pollution and the internet. *The New York Times*, Set. 2012. Acesso em: julho de 2013. Disponível em: http://www.nytimes.com/2012/09/23/technology/data-centers-waste-vast-amounts-of-energy-belying-industry-image.html?_r=1.

GOLDBARG, M. C.; PACCA, H.; LUNA, L. Modelagem de problemas. In: _____ (Ed.). *Otimização Combinatória e Programação Linear*. 2. ed. Rio de Janeiro: Campus, 2001. cap. 1.

GOLDBARG, M. C.; PACCA, H.; LUNA, L. Método simplex. In: _____ (Ed.). *Otimização Combinatória e Programação Linear*. 2. ed. Rio de Janeiro: Campus, 2001. cap. 3.

GONÇALVES, G. E. et al. Resource allocation in clouds: Concepts, tools and research challenges. In: *XXIX SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS*. Campo Grande: UFMS, 2011.

GOOGLE APPS. *Google Apps*. 2010. Acesso em: dezembro de 2012. Disponível em: http://www.google.com/apps.

GRAPHVIZ. *Graph Visualization Software*. 2012. Acesso em: dezembro de 2012. Disponível em: http://www.graphviz.org.

GREENBERG, A. et al. Towards a next generation data center architecture: scalability and commoditization. In: *ACM WORKSHOP ON PROGRAMMABLE ROUTERS FOR EXTENSIBLE SERVICES OF TOMORROW*. Seattle: PRESTO, 2008. p. 57–62.

HOW-TO GEEK. *What's the Difference Between Sleep and Hibernate in Windows?* 2012. Acesso em: dezembro de 2012. Disponível em: http://www.howtogeek.com/102897/whats-the-difference-between-sleep-and-hibernate-in-windows/.

HYSER, C. *Autonomic Virtual Machine Placement in the Data Center*. [S.l.], 2008. 11 p. Acesso em: dezembro de 2012. Disponível em: http://www.hpl.hp.com/techreports/2007/HPL-2007-189.pdf.

IBM. *WebSphere CloudBurst Appliance*. 2012. Acesso em: dezembro de 2012. Disponível em: http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp.

INTEL. *Thermal Storage System Provides Emergency Data Center Cooling*. 2012. Acesso em: dezembro de 2012. Disponível em: http://www.intel.com/content/dam/doc/white-paper/intel-it-thermal-storage-system-provides-emergency-data-center-cooling-paper.pdf.

ISSARAIYAKUL, T.; HOSSAIN, E. *NS2 Ultimate*. 2012. Acesso em: dezembro de 2012. Disponível em: http://www.ns2ultimate.com/post/5240359082/post-processing-ns2-result-using-ns2-trace-ex3.

JONES, M. T. *Anatomia de uma nuvem de software livre*. 2012. Acesso em: dezembro de 2012. Disponível em: http://www.ibm.com/developerworks/br/opensource/library/os-cloud-anatomy/.

KAWAMURA, M. S. *Aplicação do Método Branch-and-Bound na Programação de Tarefas em uma Única Máquina com Data de Entrega Comum sob Penalidades de Adiantamento e Atraso*. 71 p. Dissertação (Mestrado em Engenharia de Produção) — Escola Politécnica da Universidade de São Paulo, São Paulo, 2006.

KIRKPATRICK, S.; GELATT, C.; VECCHI, M. Optimization by simulated annealing. *Science*, n. 220, p. 671–680, 1983.

KRAPIVKA, A.; OSTFELD, A. Coupled genetic algorithm-linear programming scheme for least-cost pipe sizing of water-distribution systems. *Journal of Water Resources Planning and Management*, Faculty of Civil and Environmental Engineering, Technion - Israel Institute of Technology, v. 135, n. 4, p. 298–302, 2009.

KUROSE, J. F.; ROSS, K. W. The network core. In: _____ (Ed.). *Computer Networking - A Top Down Approach*. 4. ed. Boston: Pearson - Addison Wesley, 2007. cap. 1, p. 25–40.

KVM. *Kernel-based Virtual Machine*. 2012. Acesso em: dezembro de 2012. Disponível em: http://www.linux-kvm.org.

LADURANTAYE, S. *Canada called prime real estate for massive data computers*. 2012. Acesso em: dezembro de 2012. Disponível em: http://www.intel.com/content/dam/doc/white-paper/intel-it-thermal-storage-system-provides-emergency-data-center-cooling-paper.pdf.

LANAMARK. *Lanamark Suite Overview*. 2012. Acesso em: dezembro de 2012. Disponível em: http://www.lanamark.com/product/overview.

LI, D. et al. Building mega data center from heterogeneous containers. In: *19TH IEEE INTERNATIONAL CONFERENCE ON NETWORK PROTOCOLS*. Vancouver: ICNP, 2011.

LI, W. *Virtual Machine Placement in Cloud Environments*. [S.l.], 2012. 74 p. Acesso em: dezembro de 2012. Disponível em: `http://www8.cs.umu.se/research/uminf/reports/2012/013/part1.pdf`.

LINDO SYSTEMS. *LINDO Systems - Optimization Software: Integer Programming, Linear Programming, Nonlinear Programming, Stochastic Programming, Global Optimization*. 2012. Acesso em: dezembro de 2012. Disponível em: `http://www.lindo.com/`.

LUO, Y.; GUIGNARD, M.; CHEN, C. A hybrid approach for integer programming combining genetic algorithms, linear programming and ordinal optimization. *Journal of Intelligent Manufacturing*, Netherlands, Klumer Academic Publisher, v. 12, p. 509–519, 2001.

MARINS, F. A. S. *Pesquisa Operacional - Capítulo 2 - Programação Linear*. 2012. Acesso em: dezembro de 2012. Disponível em: `http://www.feg.unesp.br/~fmarins/po/slides/`.

MARTINS, A. Computação na nuvem - uma visão geral. In: *III CONGRESSO INTERNACIONAL SOFTWARE LIVRE E GOVERNO ELETRÔNICO. Amãpytuna - Computação em Nuvem: serviços livres para a sociedade do conhecimento.* Brasília: Fundação Alexandre de Gusmão, 2010. cap. 2, p. 47–65.

MEDINA, A. et al. *Brite: Universal Topology Generation from a User's Perspective*. 2001. Acesso em: dezembro de 2012. Disponível em: `http://www.cs.bu.edu/brite/`.

MEHDI, N. A. et al. Impatient task mapping in elastic cloud using genetic algorithm. *Journal of Computer Science*, [S.l.], v. 7, n. 6, p. 877–883, 2011.

MELL, P.; GRACE, T. *The NIST Definition of Cloud Computing*. Department of Commerce. Gaithersburg, USA., 2011. 7 p. Disponível em: <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>. Acesso em: dezembro de 2012.

MENG, X.; PAPPAS, V.; ZHANG, L. Improving the scalability of data center networks with traffic-aware virtual machine placement. In: *IEEE CONFERENCE ON COMPUTER COMMUNICATIONS.* San Diego: IEEE INFOCOM, 2010. p. 1–9.

METROPOLIS, N. et al. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, n. 21, p. 1087–1092, 1953.

MICROSOFT. *About Virtual Machine Placement*. 2012. Disponível em: <http://technet.microsoft.com/en-us/library/bb740817.aspx>. Acesso em: dezembro de 2012.

MICROSOFT. *Windows Azure Platform*. 2012. Acesso em: dezembro de 2012. Disponível em: `http://www.microsoft.com/windowsazure`.

MILLS, K.; FILLIBEN, J.; DABROWSKI, C. Comparing vm-placement algorithms for on-demand clouds. In: *THIRD IEEE INTERNATIONAL CONFERENCE ON CLOUD COMPUTING TECHNOLOGY AND SCIENCE.* Atenas: CloudCom, 2011. p. 91–98.

MISHRA, M.; SAHOO, A. On theory of vm placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach. In: *IEEE 4th INTERNATIONAL CONFERENCE ON CLOUD COMPUTING.* Washington: CLOUD, 2011. p. 275–282.

MOORE, J. et al. Making scheduling "cool": temperature-aware. In: *USENIX ANUAL TECHNICAL CONFERENCE*. [S.l.]: USENIX, 2005. p. 61–75.

MURPHY, A. *Enabling Long Distance Live Migration with F5 and VMware vMotion*. [S.l.], 2011. 10 p. Acesso em: dezembro de 2012. Disponível em: `http://www.f5.com/pdf/white-papers/cloud-vmotion-f5-wp.pdf`.

MURUGESAN, S. Harnessing green it: Principles and practices. *IEEE IT Professional*, [S.l.], v. 10, n. 1, p. 24–33, 2008.

NASCIMENTO, L.; POLEDNA, S. O processo de implantação da ISO 14000 em empresas brasileiras. In: *XXII ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO*. Curitiba: ENEGEP, 2002.

NEDEVSCHI, S. et al. Reducing network energy consumption via sleeping and rate-adaptation. In: *5TH USENIX SYMPOSIUM ON NETWORKED SYSTEMS DESIGN AND IMPLEMENTATION*. San Franscisco: NSDI, 2008.

NEDEVSCHI, S. et al. Skilled in the art of being idle: reducing energy waste in networked systems. In: *6TH USENIX SYMPOSIUM ON NETWORKED SYSTEMS DESIGN AND IMPLEMENTATION*. Boston: NSDI, 2009. p. 381–394.

NETIQ. *PlateSpin Recon*. 2012. Acesso em: dezembro de 2012. Disponível em: `https://www.netiq.com/products/recon`.

NIEMANN, J. *Hot Aisle vs. Cold Aisle Containment*. [S.l.], 2008. 13 p. Acesso em: dezembro de 2012. Disponível em: `http://www.apcmedia.com/salestools/DBOY-7EDLE8_R0_EN.pdf`.

NIEMINEM, K.; RUUTH, S.; MAROS, I. *Genetic Algorithm for finding a good first integer solution for MILP*. Department of Computing, Imperial College, London, 2003. 16 p. Acesso em: dezembro de 2012. Disponível em: `https://www.doc.ic.ac.uk/research/technicalreports/2003/DTR03-4.pdf`.

NING. *What is NING?* 2010. Acesso em: dezembro de 2012. Disponível em: `www.ning.com/what-is-ning`.

NOBAYASHI, D. et al. Performance evaluation of power saving scheme for wireless lan with station aggregation. In: *THIRD INTERNATIONAL CONFERENCE ON INTELLIGENT NETWORKING AND COLLABORATIVE SYSTEMS*. Fukuoka: INCoS, 2011. p. 338–339.

NORDMAN, B.; CHRISTENSEN, K. Proxying: The next step in reducing it energy use. *Computer*, [S.l.], v. 43, n. 1, p. 91–93, 2010.

NVIDIA. *Popular GPU-accelerated Applications*. 2012. Acesso em: dezembro de 2012. Disponível em: `http://www.nvidia.com/docs/IO/123576/nv-applications-catalog-lowres.pdf`.

OPEN MPI. *Open MPI: Open Source High Performance Computing*. 2012. Acesso em: dezembro de 2012. Disponível em: `http://www.open-mpi.org/`.

OPENFLOW.ORG. *OpenFlow Tutorial*. 2012. Acesso em: dezembro de 2012. Disponível em: `http://www.openflow.org/wp/learnmore/`.

PERING, T. et al. Coolspots: reducing the power consumption of wireless mobile devices with multiple radio interfaces. In: *4TH INTERNATIONAL CONFERENCE ON MOBILE SYSTEMS*. Uppsala: MobiSys, 2006. p. 220–232.

PETTEY, C.; GOASDUFF, L. *Gartner Says Energy-Related Costs Account for Approximately 12 Percent of Overall Data Center Expenditures*. 2012. Acesso em: dezembro de 2012. Disponível em: `http://www.gartner.com/newsroom/id/1442113`.

PINAL, K. D. *Introduction to Cloud Computing*. 2010. Disponível em: <http://dotnetslackers.com/articles/sql/Introduction-to-Cloud-Computing.aspx>. Acesso em: dezembro de 2012.

PLANETLAB. *PlanetLab - An Open Platform for Developing, Deploying, and Accessing Planetary-scale services*. 2012. Acesso em: dezembro de 2012. Disponível em: `http://www.planet-lab.org`.

PORTAL BRASIL. *Meio Ambiente*. 2012. Acesso em: dezembro de 2012. Disponível em: `http://www.brasil.gov.br/sobre/meio-ambiente/climas`.

PROXMOX. *Proxmox*. 2013. Acesso em junho de 2011. Disponível em: `www.proxmox.com`.

RAJIC, J. *Evolving Toward the "Green"Datacenter*. 2012. Acesso em: julho de 2013. Disponível em: `http://stack.nil.si/ipcorner/GreenDC/`.

RASMUSSEN, N. *Calculating total cooling requirements for data centers*. Schneider Electric - Data Center Science Center, 2012. 7 p. Acesso em: dezembro de 2012. Disponível em: `http://www.ptsdcs.com/whitepapers/23.pdf`.

READ, K. *A closer look at Merlin - Technical specifications for the world's most sustainable data centre*. [S.l.], 2012. 4 p. Acesso em: dezembro de 2012. Disponível em: `http://www.capgemini.com/insights-and-resources/by-publication/a-closer-look-at-merlin`.

READ, K. *Merlin - The World's Most Sustainable Data Centre*. [S.l.], 2012. 12 p. Acesso em: dezembro de 2012. Disponível em: `http://www.capgemini.com/sites/default/files/resource/pdf/Merlin___%__the_World___s_Most_Sustainable_Data_Center.pdf`.

RENZENBRINK, T. *Data centers Use 1.3% of World's Total Electricity. A Decline in growth*. 2011. Acesso em: junho de 2013. Disponível em: `http://www.techthefuture.com/energy/data-centers-use-1-3-of-worlds-total-electricity-a-decline-in-growth/`.

RIED, S. et al. *Sizing The Cloud - Understanding And Quantifying The Future Of Cloud Computing*. 2010. Acesso em: dezembro de 2011. Disponível em: `http://www.forrester.com/Sizing+The+Cloud/fulltext/-/E-RES58161?objectid=RES58161`.

ROCHA, L. *Introdução à Computação em Nuvem*. Departamento de Computação e Automação Industrial. Campinas, 2013. 37 p. Disponível em: <http://sourceforge.net/projects/realcloudsim/files/RealCloudSim/introcloud.pdf/download>. Acesso em: junho de 2013.

ROCHA, L. A. *REALcloudSim*. 2012. Acesso em: dezembro de 2012. Disponível em: `http://sourceforge.net/projects/realcloudsim/`.

ROCHA, L. A. et al. A bio-inspired approach to provisioning of virtual resources in federated clouds. In: *INTERNATIONAL CONFERENCE ON DEPENDABLE, AUTONOMIC, AND SECURE COMPUTING (DASC)*. Sydney: CGC, 2011. p. 598–604.

ROCHA, L. A. et al. Advances in educational robotics in cloud with qualitative scheduling in workflows. In: MAHMOOD, Z.; HILL, R. (Ed.). *Computer Communications and Networks*. London: Springer, 2012. cap. 7, p. 135–157.

ROSEN, E. C.; VISWANATHAN, A.; CALLON, R. *Multiprotocol Label Switching Architecture*. 1999. Internet Draft. RFC 3031. Acesso em: dezembro de 2011. Disponível em: `www.ietf.org/rfc/rfc3031.txt`.

ROUSE, M. *Computer Room Air Conditioning Unit (CRAC)*. 2012. Acesso em: dezembro de 2012. Disponível em: `http://searchdatacenter.techtarget.com/definition/computer-room-air-conditioning-unit`.

SALESFORCE. *Sobre a Salesforce*. 2010. Acesso em: dezembro de 2010. Disponível em: `http://www.salesforce.com/br/company/`.

SARAMAGO, S. F. P. *Métodos de Otimização Randômica: algoritmos genéticos e "simulated annealing". Notas em Matemática Aplicada.* São Carlos, 2003. 37 p. Acesso em: dezembro de 2011. Disponível em: `http://www.sbmac.org.br/boletim/pdf_2003/livro_06_2003.pdf`.

SARKER, R.; RAY, T. An improved evolutionary algorithm for solving multi-objective crop planning models. *Computers and Electronics in Agriculture*, v. 68, n. 2, p. 191 – 199, 2009. Acesso em: dezembro de 2011. Disponível em: `http://www.sciencedirect.com/science/article/pii/S0168169909000969`.

SCHIFF, J. L. *Grid Computing and the Future of Cloud Computing*. 2010. Acesso em: dezembro de 2010. Disponível em: `http://www.enterprisestorageforum.com/outsourcing/features/article.php/3859956/Grid-Computing-and-the-Future-of-Cloud-Computing.htm`.

SHANG, S. et al. An intelligent capacity planning model for cloud market. *Journal of Internet Services and Information Security*, [S.l.], v. 1, n. 1, p. 37–45, 2011.

SHANKAR, A.; BELLUR, U. *Virtual Machine Placement in Computing Clouds*. Department of Computer Science and Engineering. Indian Institute of Technology Bombay, 2010. 24 p. Acesso em: dezembro de 2010. Disponível em: `http://www.cse.iitb.ac.in/synerg/lib/exe/fetch.php?media=public:students:dhaval:anjana.pdf`.

SHIPLEY, A. et al. *Combined Heat and Power. Effective Energy Solutions for a Sustainable Future.* [S.l.], 2008. 38 p. Acesso em: junho de 2013. Disponível em: `http://www1.eere.energy.gov/manufacturing/distributedenergy/pdfs/chp_report_12-08.pdf`.

SILICON GRAPHICS INTERNATIONAL. *SGI Ice X*. [S.l.], 2012. 2 p. Acesso em: dezembro de 2012. Disponível em: `http://www.sgi.com/pdfs/4330.pdf`.

SILVA, A. E. A. da. *Uma Abordagem Multi-objetivo e multimodal para Reconstrução de Árvores Filogenéticas*. Tese (Doutorado em Engenharia Elétrica) — Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas. Campinas, 2007.

SIMUNIC, T. Power saving techniques for wireless lans. In: *CONFERENCE ON DESIGN, AUTOMATION AND TEST IN EUROPE*. Washington: IEEE Computer Society, 2005. v. 3, p. 96–97.

SOUZA, C. C. d. et al. Uso de algoritmos genéticos como ferramenta auxiliar no processo decisório em atividades de gestão agroindustrial. *Informe GEPEC. Revista de Desenvolvimento Regional e Agronegócio*, Toledo, v. 14, n. 1, p. 113–126, 2010.

SPECPOWER. *Standard Performance Evaluation Corporation.SPECpower_ssj2008 Reporter Version: [SSJ 1.2.8, June 24, 2009]*. 2009. Acesso em: dezembro de 2012. Disponível em: `http://www.spec.org/power_ssj2008/results/res2011q1/power_ssj2008-20110209-00353.html`.

TAKIGUCHI, T. et al. A novel wireless wake-up mechanism for energy-efficient ubiquitous networks. In: *IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS WORKSHOPS*. Dresden: IEEE ICC, 2009.

TSAKALOZOS, K.; ROUSSOPOULOS, M.; DELIS, A. Vm placement in non-homogeneous iaas-clouds. In: *9TH INTERNATIONAL CONFERENCE ON SERVICE ORIENTED COMPUTING.* Paphos: ICSOC, 2011. p. 172–187.

VAQUERO, L. M. et al. A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, New York, v. 39, n. 1, p. 50–55, 2009.

VAQUERO, L. M. et al. Sampling ISP backbone topologies. *IEEE Communications Letters*, [S.l.], v. 16, n. 2, p. 4–20, 2012.

VERDI, F. et al. Novas arquiteturas de data center para cloud computing. In: *XXVIII SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS*. Gramado: UFRGS, 2010.

VMWARE. *VMware Capacity Planner*. 2012. Acesso em: dezembro de 2012. Disponível em: `http://www.vmware.com/products/capacity-planner/overview.html`.

WANG, X. et al. A survey of green mobile networks: Opportunities and challenges. *Mobile Network Applications*, Springer-Verlag New York, v. 17, n. 1, p. 4–20, 2012.

WLADAWSKY-BERGER, I. The industrial imperative. In: FOSTER I.; KESSELMAN, C. (Ed.). *The Grid 2, Second Edition: Blueprint for a New Computing Infrastructure*. 2. ed. San Francisco: Elsevier, 2003. cap. 3, p. 25–34.

WU, G. et al. A hybrid approach for integer programming combining genetic algorithms, linear programming and ordinal optimization. In: HUANG, T. et al. (Ed.). *Neural Information Processing. Lecture Notes in Computer Science*. Berlin: Springer Berlin Heidelberg, 2012. v. 7665, p. 315–323.

YAMADA, M. et al. Power efficient approach and performance control for routers. In: *IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS WORKSHOPS*. Dresden: NSDI, 2009. p. 1–5.

YAZIR, Y. O. et al. Dynamic resource allocation in computing clouds through distributed multiple criteria decision analysis. In: *IEEE 3RD INTERNATIONAL CONFERENCE ON CLOUD COMPUTING*. Miami: IEEE CLOUD, 2010. p. 91–98.

ZHANG, Y. et al. Integrating resource consumption and allocation for infrastructures resources on-demand. In: *IEEE 3RD INTERNATIONAL CONFERENCE ON CLOUD COMPUTING*. Miami: IEEE CLOUD, 2010. p. 75–82.

ZOHO. *Zoho CRM - Affordable On-demand CRM*. 2010. Acesso em: dezembro de 2010. Disponível em: `http://www.zoho.com`.

ZUBEN, F. von. *Modelos e Aplicações de Redes Neurais Artificiais*. 2006. Acesso em junho de 2013. Disponível em: `ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ea072_2s06/notas_de_aula/`.

Apêndice A

Aspectos da Ferramenta de Alocação de VMs desenvolvida

Este apêndice detalha a ferramenta de otimização REALcloudSim utilizada na tese. Essa ferramenta de otimização contribui com o estado da arte porque generaliza, de forma simplificada, a representação de requisitos de VMs, capacidades de servidores, e topologia da rede de um cenário típico de *data centers*. Para os requisitos informados o *software* informa o consumo de energia por servidores e *data centers*, e gera relatórios descritivos com os resultados.

A ferramenta implementa os algoritmos de alocação propostos na tese. O *software* é formado por um conjunto de componentes, cada um dos quais responsável por implementar uma funcionalidade específica da estratégia híbrida.

A.1 Especificação da Ferramenta

A ferramenta é organizada em um conjunto componentes que são descritos a seguir:

1. Descritor da Alocação;
2. Processador de requisições;
3. Política de alocação;
4. *Engines* de Execução: Sequencial, com *Threads* e *Publish/Subscribe*;
5. Gerador de modelos MILP;
6. Processador de modelos MILP;
7. Gerador de modelos de simulação de rede;
8. Processador de modelos de simulação de rede.

A organização desses componentes na ferramenta, e os detalhes das funções de cada um são explicados a seguir.

A.1.1 Organização dos Componentes da Ferramenta

O principal objetivo da ferramenta de otimização proposta é criar um ambiente para processar diferentes requisitos de VMs, de acordo com um conjunto de capacidades de servidores organizados em *data centers*, e conectados por meio de redes de alto desempenho. A Fig. A.1 mostra o diagrama UML dos principais componentes dessa ferramenta.

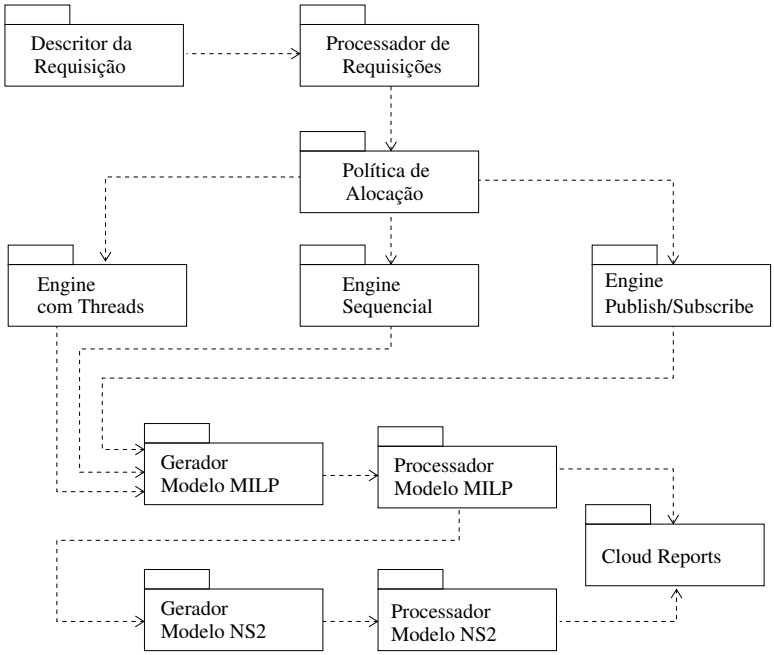

Figura A.1: Diagrama UML do Otimizador REALcloudSim.

A.1.2 Descritor da Requisição

O descritor da requisição mantém o documento que especifica a alocação requisitada. Esse documento, mantido em uma base de dados, é uma extensão do formato BRITE (MEDINA et al., 2001) especificamente para representar de forma genérica e textual, as seguintes características de uma alocação de VMs em nuvem:

- Topologia da rede entre os *data centers*: nodos roteadores, *links* físicos de conexão, largura de banda dos *links*;
- Roteadores de acesso à topologia entre os *data centers*;
- Servidores e suas capacidades;
- VMs e seus requisitos;

O formato do arquivo de descrição da alocação é mostrado a seguir, bem como a descrição de seus campos:

```
Topology: ( <numeroNodos> Nodes, <numeroArestas> Edges )
Model (<descriçãoModelo>): <parâmetrosModelo>
```

```
Nodes: ( <numeroNodos> )
#NodeId xpos ypos indegree outdegree ASid type
...

Edges: ( <numeroEdges> )
#EdgeId From To Length Delay Bandwidth ASfrom ASto Type Other
...

Datacenters: ( <numeroDatacenters> )
#DatacenterId BandwidthBorderRouter
...

Servers: ( <numeroServidores> )
#datacenterId serverId cap_cpu cap_ram cap_disk cap_bandwidth /
virtualizer cost_cpu cost_ram cost_disk cost_bw xpos ypos
...

VMs: ( <numeroVMs> )
#vmId initialServer cpu_req ram_req disk_req bw_req /
fluxo_gerado virtualizer xpos ypos
...
```

Descrição dos campos do cabeçalho:

- <numeroNodos>: quantidade de nodos representados na topologia de rede;
- <numeroArestas>: quantidade de arestas entre os nodos da topologia de rede;
- <descriçãoModelo>: indica o modelo utilizado para a geração da topologia de rede (exemplos: personalizado, Barabasi, Waxman);
- <parâmetrosModelo>: parâmetros utilizados pelo gerador da topologia;

Descrição dos campos para nodos:

- NodeId: identificador único do nodo da rede;
- xpos, ypos: coordenadas espaciais dos nodos;
- indegree: quantidade de arestas que entram no nodo;
- outdegree: quantidade de arestas que saem do nodo;
- ASid: indice do AS (*Autonomous System*) ao qual esse nodo pertence (se hierarquico)
- type: tipo atribuido ao nodo (por exemplo: BORDER_ROUTER, AS)

Descrição dos campos para arestas:

- EdgeId: ID único atribuido à aresta;
- From: ID do nodo de origem;
- To: ID do nodo de destino;

- length: distância Euclidiana entre os nodos;
- delay: atraso de propagação;
- bandwidth: largura de banda do *link*;
- ASfrom: se a topologia for hierárquica, o ID do AS nodo de origem;
- ASto: se a topologia for hierárquica, o ID do AS nodo de destino;
- type: tipo atribuido à aresta pela rotina de classificação.

Decrição dos campos para *data center*:

- DatacenterId: ID único do *data center*;
- BandwidthBorderRouter: largura de banda do roteador de borda.

Descrição dos campos para servidores:

- datacenterId: ID único do *data center* ao qual pertence o servidor;
- serverId: ID único do servidor;
- cap_cpu, cap_ram cap_disk cap_bandwidth: capacidades do servidor;
- virtualizer: tipo de virtualizador;
- cost_cpu cost_ram cost_disk cost_bw: custos dos recursos do servidor;
- xpos, ypos: coordenadas espaciais dos nodos.

Descrição dos campos para VMs:

- vmId: ID único da VM;
- initialserver: servidor inicial onde a VM foi criada;
- cpu_req, ram_req, disk_req, bw_req: requisitos da VM;
- fluxo_gerado: estimativa de fluxo de pacotes a ser gerado;
- virtualizer: tipo de virtualizador;
- xpos, ypos: coordenadas espaciais dos nodos.

A.1.3 Processador de Requisições

Esse componente utiliza expressões regulares para recuperar as informações sobre a requisição de alocação de VMs da base de dados. Essas informações são mantidas em memória para o processamento pelos outros componentes.

A.1.4 Política de Alocação

Esse componente implementa as diferentes políticas de alocação do REALcloudSim. Uma estrutura *Cloud* é definida na ferramenta como um conjunto de VMs que possuem diferentes requisitos que precisam ser satisfeitos serem instanciadas adequadamente. Na ferramenta, a estrutura *Cloud* é definida como uma matriz, onde cada linha possui a atribuição de uma VM a um servidor. Para cada atribuição são definidos os recursos de que a VM requisita para ser instanciada. Assume-se que cada VM é criada e mantida em um servidor, porém a sua instanciação depende da carga atual do servidor, ou seja, caso o servidor atual onde a VM foi criada não satisfaça os requisitos, a VM não será instanciada no servidor corrente, e deve ser realocada para outro servidor. A estrutura é definida em uma matriz, conforme descrito a seguir:

Servidor_n	**VM_n**	**req_cpu**	**req_ram**	**req_disk**	**req_link**

Tabela A.1: Estrutura Cloud

A estrutura dos servidores é definida como uma matriz, onde cada linha representa um servidor e seus recursos disponíveis. A cada nova alocação de uma VM, os recursos do respectivo servidor são reduzidos/incrementados conforme os requisitos da VM. Essa estrutura é definida como segue:

Servidor_n	**cap_cpu**	**cap_ram**	**cap_disk**	**cap_link**
	cost_cpu	**cost_ram**	**cost_disk**	**cost_link**

Tabela A.2: Estrutura Servidores

A.1.5 *Engines* de Execução: Sequencial, com *Threads* e *Publish/Subscribe*

A ferramenta implementa as *engines* sequencial, com *threads* e o modelo *publish/subscribe*. A *Engine Sequencial* executa todo o processo de simulação no mesmo *host* ou VM onde o componente *Processador de Requisições* foi iniciado. Cada cromossomo é processado e executado um após o outro. A *Engine com Threads* dispara *threads* independentes para cada cromossomo da política de alocação híbrida. Cada *thread* realiza a geração e processamento do modelo MILP, e a geração e execução da simulação do tráfego. A *engine* principal do REALcloudSim é responsável por sincronizá-las. A *Engine com Threads* foi considerada porque executa em menor tempo computacional quanto comparada à alternativa de paralelismo inicialmente proposta com o uso da biblioteca de passagem de mensagens OpenMPI (OPEN MPI, 2012). A proposta de uso de OpenMPI possui muitas limitações: dificuldade de configuração das relações de confiança entre os *hosts* (com necessidade de autenticação sem senha), dificuldade de sincronização de tarefas de grande porte, limpeza do *buffer* de execução de tarefas, além da manutenção de um sistema de arquivos compartilhado. A *Engine Paralela* com OpenMPI define que cada cromossomo é processado em uma VM da nuvem. Essa *engine* implementa um modelo MapReduce apropriado para o problema de alocação de VMs: cada cromossomo é mapeado para um processo MPI que executa os passos da simulação MILP e simulação de rede em VMs distintas da nuvem. É interessante notar que não é necessário que exista a mesma quantidade de VMs por cromossomo,

pois um ou mais processos podem executar na mesma VM. No entanto, quanto maior o número de VMs, maior será o ganho de desempenho.

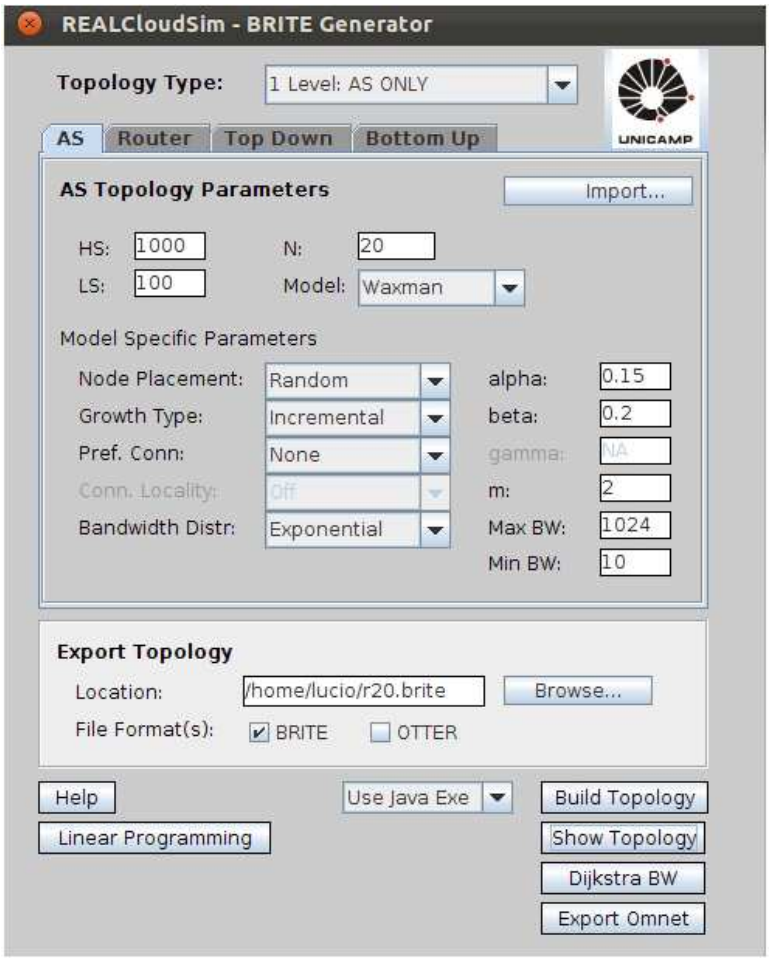

Figura A.2: Interface para Criação da Topologia BRITE no REALcloudSim.

A *Engine Publish/Subscribe* envia mensagens HTTP POST para uma fila do canal de eventos do *servlet* da plataforma. Cada mensagem HTTP POST contém as informações do cromossomo que precisam ser processadas. Essas informações são recuperadas de forma assíncrona pelos diversos *workers* que possuem recursos suficientes para processar essa informação, via HTTP GET. É interessante notar que os *workers* podem estar em redes distintas, uma vez que o canal de eventos do *servlet* é mantido em um *host* com endereço IP roteável. Cada *worker* executa o *solver* MILP e a simulação do tráfego indicado pelo cromossomo, e envia a mensagem de resposta, via HTTP POST, no mesmo *servelt*, mas na fila de resultados do canal de eventos.

A.1.6 Gerador e Processador do Modelo MILP

O componente *Gerador do Modelo MILP* traduz as informações do documento de alocação no respectivo modelo de programação linear. O componente *Processador do Modelo MILP* invoca a execução do *software* Lingo para executar o modelo anterior. Após a execução, é realizado um *parser* com expressões regulares para adquirir as informações para o relatório final.

A.1.7 Gerador e Processador do Modelo NS2

O componente *Gerador do Modelo MILP* traduz as informações do documento resultante do componente *Processador do Modelo MILP* no respectivo modelo NS-2 para a simulação da rede. O componente *Processador do Modelo NS-2* invoca a execução do *software* NS-2 para executar o modelo anterior. Após a execução, é realizado um parser com expressões regulares para adquirir as informações para o relatório final.

A.1.8 Cloud Reports

O componente *Cloud Reports* armazena o resultado do processamento da simulação no REALcloudSim. A simulação gera um conjunto de arquivos com relatórios que descrevem todo o experimento. Na Tab. A.3 é descrito o conteúdo de cada um dos relatórios. A *tag* <*config*> indica o tipo da simulação realizada. Como exemplo, <*config*>=*5data_10serv_300vm* indica uma simulação com 5 *data centers*, cada um com 10 servidores, com requisição para alocação de 300 VMs entre esses servidores.

A.2 Uso da Ferramenta de Otimização

Simulações são feitas como segue:

1. Criação da topologia de rede: os parâmetros são fornecidos na própria interface gráfica do REALcloudSim;

2. Criação do arquivo de alocação: definição, no REALcloudSim, das capacidades dos servidores, requisitos das VMs, e parâmetros específicos do algoritmo genético;

3. Simulação: REALcloudSim simula o processo de alocação com os dados fornecidos pelo usuário;

4. Análise dos resultados: REALcloudSim gera os arquivos com os relatórios em formato texto para análise de cada simulação realizada.

A Fig. A.2 mostra as extensões feitas à interface gráfica original do gerador BRITE. O comando *Show Topology* exibe a interface gráfica da nuvem; *Dijkstra BW* calcula os menores custos de rede com relação à largura de banda da topologia gerada; *Linear Programming* executa o algoritmo de alocação híbrido.

Arquivos da Modelagem em MILP	**Descrição**
modeloLingo<config>.lg4	Modelagem em MILP da função de otimização da alocação, das variáveis de decisão, das restrições do modelo referente à rede aos recursos, e ao consumo de energia dos servidores e roteadores.
modeloLingo<config>.brite	Descrição em formato BRITE sobre a topologia de rede original antes da simulação.
modeloLingo<config>.datacenter	Descrição em formato BRITE sobre a topologia de rede durante a simulação. Alterações nos pesos dos *links* são armazenadas nesse arquivo.
Arquivos de Relatório MILP	**Descrição**
modeloLingo<config>.lgr	Relatório com o resultado da execução do modelo MILP.
modeloLingo<config>.range	Relatório com o resultado da análise de sensibilidade do modelo MILP.
modeloLingo<config>.banda	Relatório sobre a largura de banda entre os roteadores da rede durante a simulação.
modeloLingo<config>.parser	Relatório que descreve os parâmetros fornecidos e as saídas resultantes referentes a: 1. capacidades dos servidores; 2. requisições de VMs; 3. configuração original da topologia de rede; 4. parâmetros do AG, parâmetros da simulação; 5. resultado do consumo de energia dos servidores; 6. resultado do consumo de energia dos roteadores.
modeloLingo<config>.evolucaoFitness	Relatório sobre a evolução do *fitness* durante a simulação.
Arquivos da Modelagem da Rede	**Descrição**
modeloNS2<config>.tcl	Modelagem em linguagem NS2 a partir do modelo definido em MILP. Adicionalmente, são definidos os parâmetros da simulação do tráfego na rede: tipo de aplicações geradoras de tráfego, por exemplo, tráfego CBR sobre UDP; tempo da simulação do tráfego na rede; largura de banda dos enlaces; e custo dos enlaces.
Arquivos de Relatório da Rede	**Descrição**
modeloNS2<config>.vazao	Relatório sobre a vazão obtida em cada *link* de rede.
modeloNS2<config>.perda	Relatório sobre a perda de pacotes em cada *link* de rede.

Tabela A.3: Descrição dos Arquivos de Modelagem e de Relatórios.