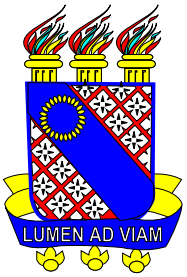


UNIVERSIDADE ESTADUAL DO CEARÁ

SILVIO ROBERTO MARTINS AMARANTE

UTILIZANDO O PROBLEMA DE MÚLTIPLAS MOCHILAS PARA MODELAR O PROBLEMA DE ALOCAÇÃO DE MÁQUINAS VIRTUAIS EM COMPUTAÇÃO NAS NUVENS

FORTALEZA, CEARÁ

2013

SILVIO ROBERTO MARTINS AMARANTE

UTILIZANDO O PROBLEMA DE MÚLTIPLAS MOCHILAS PARA MODELAR O PROBLEMA DE ALOCAÇÃO DE MÁQUINAS VIRTUAIS EM COMPUTAÇÃO NAS NUVENS

Dissertação apresentada no Curso de Mestrado Acadêmico em Ciência da Computação do Centro de Ciências e Tecnologia da Universidade Estadual do Ceará, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Joaquim Celestino Júnior

Co-Orientador: Prof. Dr. André Ribeiro Cardoso

FORTALEZA, CEARÁ

2013

Dados Internacionais de Catalogação na Publicação

Universidade Estadual do Ceará

Biblioteca Central Prof. Antônio Martins Filho

Bibliotecário Responsável – Francisco Welton Silva Rios – CRB-3 / 919

A485u Amarante, Silvio Roberto Martins

Utilizando o problema de múltiplas mochilas para modelar o problema de alocação de máquinas virtuais em computação nas nuvens / Silvio Roberto Martins Amarante. -- 2013.

CD-ROM. 80 f. ; il. (algumas color.) : 4 ¾ pol.

"CD-ROM contendo o arquivo no formato PDF do trabalho acadêmico, acondicionado em caixa de DVD Slim (19 x 14 cm x 7 mm)".

Dissertação (mestrado) – Universidade Estadual do Ceará, Centro de Ciências e Tecnologia, Curso de Mestrado Acadêmico em Ciência da Computação, Fortaleza, 2013.

Área de Concentração: Ciência da Computação.

Orientação: Prof. Dr. Joaquim Celestino Júnior.

Co-orientação: Prof. Dr. André Ribeiro Cardoso.

1. Computação nas nuvens. 2. Alocação de recursos. 3. Algoritmo de colônia de formigas. 4. Problema de múltiplas mochilas. I. Título.

CDD: 005

SILVIO ROBERTO MARTINS AMARANTE

UTILIZANDO O PROBLEMA DE MÚLTIPLAS MOCHILAS PARA MODELAR O PROBLEMA DE ALOCAÇÃO DE MÁQUINAS VIRTUAIS EM COMPUTAÇÃO NAS NUVENS

Dissertação apresentada no Curso de Mestrado Acadêmico em Ciência da Computação do Centro de Ciências e Tecnologia da Universidade Estadual do Ceará, como requisito parcial para obtenção do grau de Mestre.

Aprovada em: 04/06/2013.

BANCA EXAMINADORA

Prof. Dr. Joaquim Celestino Júnior (Orientador)
Universidade Estadual do Ceará – UECE

Prof. Dr. André Ribeiro Cardoso (Co-orientador)
Universidade Estadual do Ceará – UECE

Prof. Dr. Marcial Porto Fernandez
Universidade Estadual do Ceará – UECE

Prof. Dr. Paulo Henrique Mendes Maia
Universidade Estadual do Ceará – UECE

Prof. Dr. Javam de Castro Machado
Universidade Federal do Ceará – UFC

AGRADECIMENTOS

Aos meus familiares pelo apoio dado durante o desenvolvimento desse trabalho, tendo sempre investido na minha formação.

À minha namorada, Natália Batista, pela compreensão e motivação dada nos meses de escrita da dissertação.

Em especial, ao professor e amigo Filipe Maciel, por seu papel fundamental no desenvolvimento dos estudos e formulação do problema a ser resolvido.

Em especial, aos professores Joaquim Celestino e André Cardoso pela orientação desenvolvida para a conclusão dos estudos, além das diversas oportunidades oferecidas durante a graduação e mestrado.

Ao professor Francisco Oliveira, pelos conselhos e apoio dados durante o mestrado.

Aos meus amigos da UECE, pelos momentos de descontração e aprendizado proporcionados durante toda a minha formação. Em especial, aos colegas Rudy Matela, Luis Ribeiro, Sávio Moreira, Filipe Fernandes e Fábio Cerqueira.

Aos meus amigos de empresa júnior, por me proporcionarem crescimento tanto profissional como pessoal.

RESUMO

Computação nas nuvens (*Cloud Computing*) é um novo paradigma computacional onde tem-se a oferta dos recursos computacionais como serviços acessados através da Internet. Nesse paradigma, os recursos computacionais são agrupados e alocados de acordo com a demanda do cliente. A crescente procura por esse novo tipo de serviço ocasionou maior utilização de energia por parte dos provedores de serviços, devido a necessidade de manter a infraestrutura computacional, passando a ser um dos principais fatores de custo dos provedores. Nesse contexto, tem-se buscado soluções onde seja possível atender aos requisitos por recursos dos clientes tendo, entretanto, menor consumo de energia possível. Adotando esse objetivo, é possível obter um melhor planejamento da arquitetura do *data center* a ser utilizada. Desse modo, este trabalho consiste de uma modelagem do problema de alocação de máquinas virtuais como um problema de múltiplas mochilas, e de um algoritmo baseado em colônia de formigas para solucionar o problema abordado. Além disso, é feita uma análise comparativa da solução proposta com outros mecanismos existentes, visando avaliar o desempenho.

Palavras-chave: computação nas nuvens, alocação de recursos, algoritmo de colônia de formigas, problema de múltiplas mochilas.

ABSTRACT

Cloud computing is a new computing paradigm which has the provision of computational resources like services accessed over the internet. In this paradigm, computing resources are pooled and allocated according to customer demand. The growing demand for this new type of service has led to increased use of energy on the part of service providers, due to the need to maintain the computing infrastructure, becoming one of the leading providers of cost factors. In this context, solutions have been tried wherever possible to meet the requirements of customers for resources consuming minimum power required. Adopting this goal, it's possible get a better planning of data center architecture to be used. Thus, this work consists of modeling the allocation problem of virtual machines as a multiple knapsack problem, and an algorithm based on ant colony to solve the problem presented. Furthermore, a comparative analysis is made of the proposed solution with other existing mechanisms, to evaluate the performance.

Keywords: cloud computing, resource allocation, ant colony optimization, multiple knapsack problem.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

O aumento do uso de sistemas distribuídos, em especial o novo modelo computacional denominado Computação nas Nuvens (*Cloud Computing*), tem motivado o estudo desses sistemas de modo a identificar as vantagens e desvantagens na sua implantação, manutenção e utilização. Atualmente, os estudos estão voltados para a área de computação nas nuvens, que refere-se às aplicações entregues como serviço, bem como o *hardware* e *software* do sistema nos *data centers* que fornecem esse serviço (ARMBRUST et al.,). Um *data center* pode ser definido como um prédio onde são instalados diversos servidores e componentes de manutenção (por exemplo, as centrais de resfriamento e equipamentes de rede) (WIKIPEDIA, 2013).

Antes do surgimento de computação nas nuvens, utilizava-se o modelo tradicional, onde cada empresa possuía um departamento que funcionava como um *data center*. Esse modelo está sendo substituído por um modelo onde existem provedores de serviço que alugam recursos, tanto *hardware* como *software*, para que as empresas possam executar suas aplicações e hospedar seus serviços, acessando-os através da Internet. Uma das vantagens desse novo modelo é a possibilidade de acesso a recursos sob demanda, reduzindo assim os gastos das empresas em seus *data centers*, onde os recursos eram, muitas vezes, subutilizados.

A criação e adoção desse novo paradigma só foram possíveis devido aos avanços na área de comunicação de dados, pois os recursos passaram a ser acessados remotamente através da internet. Devido a esse fato, tornou-se necessário a utilização de recursos de rede mais eficientes para atender a essa nova demanda, pois os recursos utilizados pelas aplicações, antes executadas na mesma máquina e agora dispersas geograficamente, trafegam pela internet. Apesar da evolução dos sistemas de comunicação ter conseguido suprir os requisitos do paradigma de computação nas nuvens, surgiram novas exigências como, por exemplo, o tempo de resposta das aplicações deve ficar o mais próximo possível do tempo de resposta da aplicação no modelo tradicional, pois, caso contrário, a adoção desse novo paradigma não seria atraente para os clientes.

Com uma infraestrutura de rede atendendo os requisitos de velocidade e qualidade da transmissão de dados desse novo paradigma, passou-se a analisar outro problema: a subutilização da infraestrutura do *data center*. Antigamente, em um *data center*, serviços eram implantados diretamente e exclusivamente em um servidor, ou seja, um servidor possuía, exclusivamente, uma aplicação executando sobre ele. Desse modo, se a aplicação não utilizasse todos os recursos do servidor, os recursos não utilizados ficariam ociosos, ocasionando uma perda de lucros para o provedor do *data center*.

Para solucionar esse problema, empregou-se a tecnologia de virtualização (ZHANG;

CHENG; BOUTABA, 2010). O objetivo dessa tecnologia é maximizar o uso dos recursos físicos dos servidores, através da possibilidade de implantação de diversas aplicações em um mesmo servidor, sem que uma aplicação afete o funcionamento das outras em execução no mesmo servidor. Para atender esse objetivo, os recursos físicos são mapeados e distribuídos em grupos de recursos denominados máquinas virtuais (*Virtual Machine*-VM). Essas máquinas virtuais são criadas com as configurações desejadas, tendo como restrição a soma dos recursos alocados a todas as máquinas virtuais do servidor ultrapassando a quantidade total de recursos físicos do servidor. Cada máquina virtual funciona como se fosse uma máquina física independente, compartilhando os recursos físicos do servidor com as outras VMs presentes no mesmo servidor.

Logo após o cliente definir a aplicação a ser hospedada na nuvem e quais recursos essa aplicação irá requisitar, faz-se necessário o desenvolvimento de um contrato para especificar os direitos e deveres do provedor de serviços (*data center*) e dos clientes. Esse contrato é denominado acordo de nível de serviço (*Service Level Agreement*-SLA) (C.GUO et al.,). Após a definição desse acordo, o provedor de serviços de computação nas nuvens aloca os recursos requisitados em forma de máquina virtual e entrega o controle ao cliente. Além disso, devido à propriedade de elasticidade, um serviço pode ter a quantidade de recursos alocados incrementados ou decrementados, de acordo com a utilização desse serviço.

Devido à crescente popularização e utilização desse paradigma, pesquisadores tem estudado os impactos de sua adoção. Um desses estudos está relacionado com o consumo de energia elétrica em um *data center* seguindo o modelo de computação nas nuvens. Como exemplo, Beloglazov, Abawajy e Buyya (2012) comparam o consumo de um *data center* de médio porte ao consumo de 25000 casas. Os provedores consideram os gastos com energia como o principal fator de custo para manutenção de um *data center*.

Esse custo elevado é decorrente tanto da energia consumida pelos equipamentos de TI (servidores, equipamentos de redes etc) como a consumida pelo sistema de resfriamento, visto que todos os equipamentos de TI dissipam calor durante o seu funcionamento. A questão do resfriamento do *data center* é primordial, pois, sob temperaturas mais elevadas, os equipamentos de TI tem a sua confiabilidade e tempo de vida reduzidos. Uma falha em algum equipamento de rede, por exemplo, pode colocar servidores indisponíveis, dependendo da arquitetura de rede implementada.

Um dos modos de amenizar o consumo de energia é através da tecnologia de virtualização, quando utilizada juntamente com o desligamento dos servidores não utilizados. Como já destacado, a virtualização consegue diminuir a subutilização dos recursos através da consolidação de máquinas virtuais no menor número possível de servidores. Com isso, servidores não utilizados no momento podem ser colocados em um estado onde ele consuma uma quantidade menor de energia e colocados em operação rapidamente para atender a alguma demanda por re-

cursos, conseguindo, assim, obter economia de energia. As regras adotadas para a realização da alocação das máquinas virtuais são especificadas no mecanismo de alocação, responsável por realizar o mapeamento delas com os servidores onde serão alocadas dependendo dos critérios que foram descritos na política de alocação. No caso citado, o critério utilizado foi minimizar o número de servidores utilizados.

1.2 Motivação

Com a crescente utilização de computação nas nuvens e de estudos realizados na área, os provedores de serviços identificaram os gastos decorrentes do aumento de consumo de energia. Como alternativa, houve uma crescente busca de soluções na redução do consumo de energia. Um dos ramos de estudo nesse problema é o dos mecanismos de alocação. No contexto de economia de energia, são especificados critérios para que o mecanismo de alocação consiga dispor as máquinas virtuais de modo que consuma a menor quantidade de energia. Surge, então, a necessidade de pesquisar soluções de viabilidade econômica de energia, de critérios a serem utilizados pelos mecanismos de alocação, de determinar os impactos da abordagem proposta, assim como de comparar o desempenho das soluções existentes com a solução proposta.

1.3 Objetivos

1.3.1 Objetivo geral

Esse trabalho tem como objetivo geral realizar a modelagem matemática do problema de alocação de máquinas virtuais em um *data center* utilizando como critério o consumo de energia elétrica e a taxa de utilização dos processadores dos servidores que compõem o *data center*. Esse problema será modelado como um problema de mochilas múltiplas (*multiple knapsack problem*) possibilitando a utilização de soluções já existentes para o problema e aplicando-as no ambiente de computação nas nuvens.

1.3.2 Objetivos específicos

Como objetivos específicos desse manuscrito, serão abordados:

- Realizar a modelagem do problema descrito como um problema de múltiplas mochilas;
- Utilizar uma abordagem baseada em algoritmos bioinspirados para solucionar o problema modelado;
- Implementar a abordagem proposta no simulador CloudSim;

- Realizar a simulação da abordagem proposta com outras existentes;
- Analisar comparativamente a modelagem proposta com outras modelagens existentes;
- Identificar, com base nos resultados, as vantagens e desvantagens no uso desse tipo de modelagem.

1.4 Metodologia

Nesse trabalho será feito um levantamento dos conceitos e características de computação nas nuvens destacando a virtualização e os mecanismos de migração de máquinas virtuais.

A fim de obter o embasamento necessário para a proposição de uma solução, será feito um estudo sobre o problema de múltiplas mochilas, bem como de soluções empregadas para esse problema, de modo a avaliar a aplicabilidade dessas soluções no contexto do problema a ser abordado nesse trabalho.

Em seguida, será feito um estudo sobre uma das soluções propostas nos trabalhos relacionados, visando identificar as vantagens e desvantagens da abordagem, e propor uma melhor solução utilizando a modelagem como um problema de múltiplas mochilas para isso. Essa solução proposta utilizará como base os algoritmos bioinspirados, os quais tem apresentado bons resultados na solução de problemas combinatórios.

Por fim, a solução proposta nesse trabalho e a utilizada como referência serão implementadas no simulador CloudSim e realizada uma análise comparativa dos resultados obtidos nas duas abordagens.

1.5 Organização do trabalho

O restante deste manuscrito está organizado da seguinte forma. A fundamentação teórica sobre computação nas nuvens, sobre o problema de múltiplas mochilas e algoritmo de colônia de formigas é mostrada no capítulo 2. O capítulo 3 descreve os trabalhos relacionados com a problemática abordada. O capítulo 4 apresenta a descrição do algoritmo proposto juntamente com modelagem matemática do problema. O capítulo 5 contém informações sobre a avaliação de desempenho da proposta. Por fim, o capítulo 6 mostra as conclusões obtidas pelos testes realizados, bem como os possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta toda a fundamentação teórica necessária para a construção do algoritmo proposto, incluindo computação nas nuvens, problema de alocação de máquinas virtuais para servidores, problema de múltiplas mochilas e algoritmo de colônia de formigas.

2.1 Computação nas nuvens

Apesar do uso de computação nas nuvens (*Cloud Computing*) ser considerado um paradigma novo, o conceito é relativamente antigo. Na década de 1960, John McCarthy utilizou pela primeira vez o termo computação nas nuvens (XING; ZHAN,), onde ele afirmou que "a computação poderia algum dia ser organizada como uma utilidade pública do mesmo modo que o telefone". Ou seja, a computação disponível para os usuários tarifada de acordo com a utilização dos recursos.

Em computação nas nuvens são utilizados os avanços das tecnologias e da infraestrutura de rede com o intuito de prover diversos recursos computacionais (*hardware*, plataforma de desenvolvimento e *software*) ao cliente como serviços que são acessados através da internet e utilizam um modelo de tarifação denominado de *pay-per-use*, ou seja, você pagará apenas pelos recursos requisitados durante o tempo que estiver utilizando. Além dessas características, os clientes não necessitam possuir máquinas de grande desempenho para obter os resultados das aplicações, como ocorreria se a aplicação fosse executada localmente, pois a maioria do trabalho computacional é executado na máquina onde o serviço está hospedado (no *data center*). Assim, a finalidade do computador do cliente seria somente para a entrada dos dados e para a exibição dos resultados.

Algumas empresas que já fornecem/utilizam os serviços no modelo de computação nas nuvens podem ser vistas na figura 1. Como exemplo de aplicação, pode-se citar o conjunto de aplicativos de escritório oferecido pela Google, o Google Apps (GOOGLE, 2013). Com essas ferramentas, você pode criar documentos de textos, planilhas e apresentações sem a necessidade de instalar nenhum componente no seu computador, realizando acesso a essas aplicações através do navegador de internet.

Os serviços implantados em computação nas nuvens são armazenados em um ou mais servidores de *data centers*. Como esses serviços estão sendo executados nas máquinas dos *data centers*, a tarefa de manutenção do serviço é deslocada dos clientes, que estão pagando por esse serviço, para os *data centers*, agora denominados provedores de serviços.

A figura 2 mostra como é realizada a interação do cliente com o serviço hospedado na nuvem. Nessa arquitetura, pode ser feita uma divisão em duas camadas: *front-end* e *back-end*.

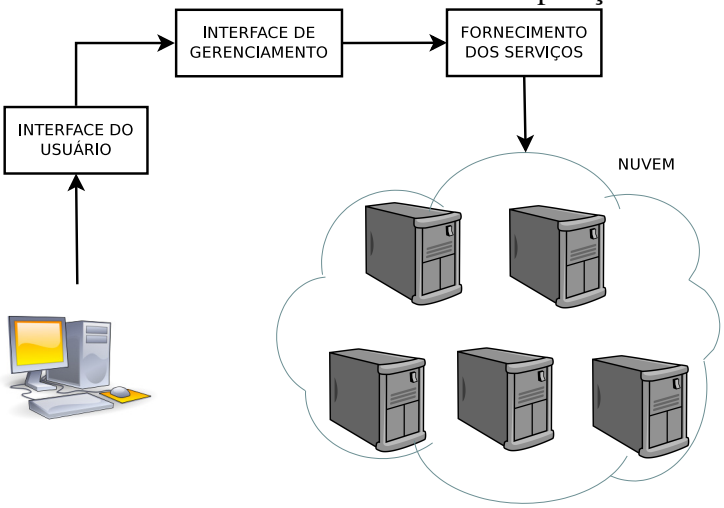
Figura 1: Empresas que já utilizam computação nas nuvens

Fonte: Telovations (2012).

A camada *front-end* tem a responsabilidade de realizar a interação com o cliente (interface do usuário), de modo que tenha como funções a entrada de dados, envio desses dados para o *back-end* e exibição dos resultados. Já no *back-end*, a requisição do cliente enviada pelo *front-end* é processada de acordo com os dados contidos na requisição. Finalizado o processamento, o resultado é enviado para o *front-end* para que o cliente visualize o resultado. Percebe-se que a maior parte do processamento concentra-se na nuvem, ou seja, no *back-end*. A interface de gerenciamento serve para controlar o acesso aos recursos da nuvem, de modo que um cliente não cause interferência na requisição de outro cliente.

Como exemplo tem-se a aplicação Dropbox (DROPBOX, 2013). O *front-end* dessa aplicação é acessada através do navegador de Internet, onde são mostrados todos os arquivos do cliente. O *back-end* trata do armazenamento desses arquivos em algum servidor e o processamento de requisições de recuperação/atualização desses arquivos.

Figura 2: Funcionamento de um ambiente computação nas nuvens



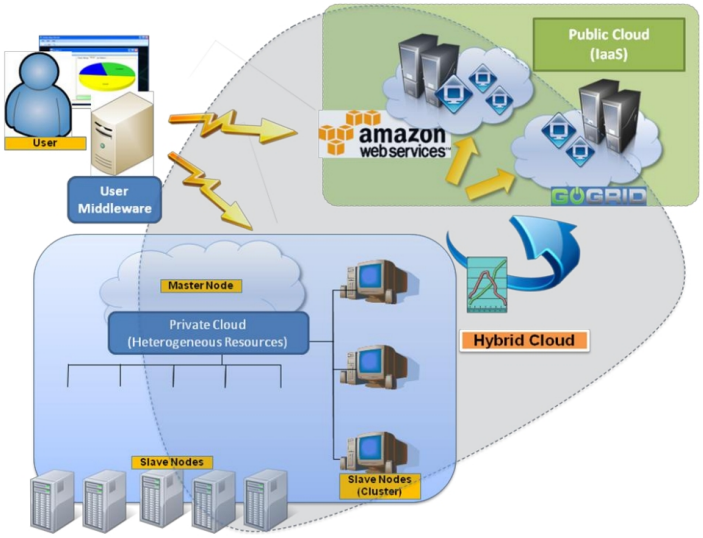
Dada a inviabilidade de hospedar cada serviço/aplicação em um servidor dedicado ocasionado tanto pelo custo de manutenção e subutilização da infraestrutura, como pela necessidade de atender a característica de elasticidade oferecida pelo paradigma de computação nas nuvens, foi adotado um modelo baseado na tecnologia de virtualização.

A virtualização nada mais é do que mapear os recursos físicos de uma máquina real (servidor) em recursos virtuais e realizar o agrupamento desses recursos virtuais com o intuito de criar uma máquina virtual (*Virtual Machine* - VM). Uma máquina virtual fornece a ilusão ao usuário de possuir uma máquina física dedicada a ele, mas, na realidade, encontra-se executando em um servidor juntamente com outras máquinas virtuais.

Além da preocupação de como os serviços serão implantados nos servidores, faz-se necessário definir os critérios de acesso à nuvem, ou seja, basicamente responder a pergunta: "Quem vai poder acessar a nuvem?". Adotando esse critério, pode-se dividir os modos de implantação em (ilustrados na Figura 3) (KUMAR; BUYYA, 2012):

- Nuvem pública - Na nuvem pública, os serviços estão disponíveis a todos os interessados pelos serviços, seguindo o modelo *pay-per-use*. Esses serviços geralmente são oferecidos por companhias que possuem grande quantidade de recursos (VERDI et al.,).
- Nuvem privada - esse modelo é geralmente implantado dentro do ambiente de uma organização, ou seja, esse modo é gerenciado e só está disponível para a organização.
- Nuvem híbrida - a abordagem híbrida é a junção de dois ou mais modelos de nuvem. Tal modelo foi concebido com o intuito de tentar evitar os problemas pertinentes a cada modelo (ZHANG; CHENG; BOUTABA, 2010).

Figura 3: Modelos de implantação para computação nas nuvens



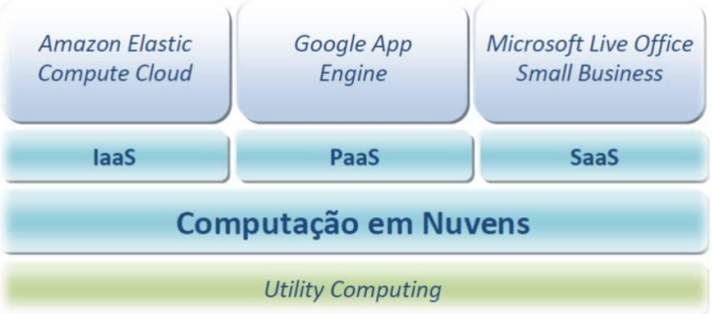
Fonte: Kumar, Buyya (2012).

Por fim, deve ser definido o nível de abstração ao qual o serviço será inserido. Esses níveis servem para especificar quais recursos serão fornecidos ao usuário e como eles serão utilizados. Os níveis de abstração mais utilizados são (ZHANG; CHENG; BOUTABA, 2010; VOORSLUYS; BROBERG; BUYYA, 2011):

- *Infrastructure as a Service* (IAAS) - nesse modelo, os recursos de *hardware* (processador, memória e comunicação) são disponibilizados como serviço. O provisionamento desses recursos é usualmente feito em termos de máquinas virtuais. Quem disponibiliza tais recursos é denominado de provedor de IAAS (*IAAS Provider*). Com exemplo, pode-se citar a *Amazon Web Services* (AMAZON, 2013) e *Cloud Sigma* (CLOUDSIGMA, 2013).
- *Plataform as a Service* (PAAS) - é oferecido um ambiente (plataforma) de desenvolvimento para os clientes. Nesse ambiente, o desenvolvedor não necessita se preocupar com o *hardware* sobre o qual está desenvolvendo e executando suas aplicações. Por exemplo, no *Google App Engine* (GOOGLE, 2011) é oferecida uma *Application Program Interface* (API) para que o usuário acesse os serviços do servidor.
- *Software as a Service* (SAAS) - no nível mais alto de abstração, os usuários acessam o serviço (aplicativo) através da internet, geralmente por navegadores de internet, não sabendo onde realmente o aplicativo está sendo executado. *Google Docs*, *Microsoft Office 2010* e *Salesforce.com* são exemplos de aplicativos.

Na figura 4 são ilustrados os níveis de abstração em computação nas nuvens. Pode ser observada na camada mais inferior o conceito de computação utilitária (PARKHILL, 1966), onde tem-se os serviços sendo tarifados de acordo com a utilização. Sobre a camada seguinte, computação em nuvens, há a divisão nos três níveis de abstração definidos anteriormente e exemplos de serviços ofertados.

Figura 4: Representação dos modelos de serviços oferecidos em computação nas nuvens



Fonte: adaptado de Langley (2012).

Quando uma máquina virtual é alocada para um usuário implantar e/ou usar serviços, espera-se, normalmente, que ela possua as seguintes características fornecidas pelo paradigma de computação nas nuvens (VERDI et al., 2010; ZHANG; CHENG; BOUTABA, 2010):

- Recursos sob demanda: de acordo com as necessidades do cliente, é realizada a alocação de mais recursos ou desalocação de alguns recursos não utilizados, sendo essas ações executadas de maneira automatizada. Isso é possível pela utilização da tecnologia de virtualização de recursos.
- Amplo acesso aos recursos: para que os recursos possam ser acessados de qualquer dispositivo, garantindo assim maior portabilidade, a internet é utilizada como interface padrão de acesso aos recursos.
- Elasticidade: realiza a liberação de recursos rapidamente de acordo com a demanda da aplicação.
- Medição dos serviços: para fins de contabilidade e tarifação, a utilização dos recursos pelos clientes deve ser monitorada constantemente de modo que se possa medir a quantidade e a duração do uso dos recursos. Essa tarefa é geralmente realizada pelo *software* de gerenciamento do *data center*.
- Auto-organização: Para uma maior confiabilidade do serviço, faz-se necessário um sistema de gerenciamento de recursos automatizados. Ele tem como objetivo de reagir rapidamente a mudanças na demanda do serviço.
- Rede de acesso ubíqua: A internet é utilizada como rede de acesso e de provisão de recursos. Como é amplamente difundida, qualquer dispositivo com conexão com a internet pode utilizar os serviços oferecidos.

2.2 Virtualização

Na década de 1960, um projeto da IBM foi desenvolvido com o intuito de dividir os recursos pertencentes a um *mainframe* para múltiplos usuários (XING; ZHAN,). Nesse contexto, surgiu o conceito de virtualização, definido como o particionamento dos recursos físicos de uma máquina e seu mapeamento em recursos virtuais a possibilitar a formação de máquinas virtuais (VM) utilizando os recursos virtuais. Tais máquinas virtuais funcionam como se fossem uma máquina real e são independentes umas das outras, apesar de estarem alocadas na mesma máquina física.

Para realizar o gerenciamento das VMs, existe um *software* denominado de Monitor de Máquinas Virtuais (MMV) ou *hypervisor*, responsável por fornecer o isolamento entre as máquinas virtuais que estão executando na mesma máquina. Conforme Soares (2010), algumas propriedades obtidas pelo isolamento e pela abstração oferecidas pelas VMs são:

- Consolidação de servidores - visando resolver o problema de subutilização dos servidores que utilizavam o paradigma tradicional, a consolidação de máquinas virtuais é utilizada

para hospedar várias máquinas virtuais em um único servidor. Desse modo, é possível executar múltiplos serviços simultaneamente, utilizando uma quantidade menor de servidores.

- Consolidação de aplicações: cada aplicação pode requisitar um *hardware* específico para poder executar. Com a virtualização é possível criar máquinas virtuais com configurações específicas de *hardware*, independente do *hardware* sobre o qual a máquina virtual está sendo hospedada.

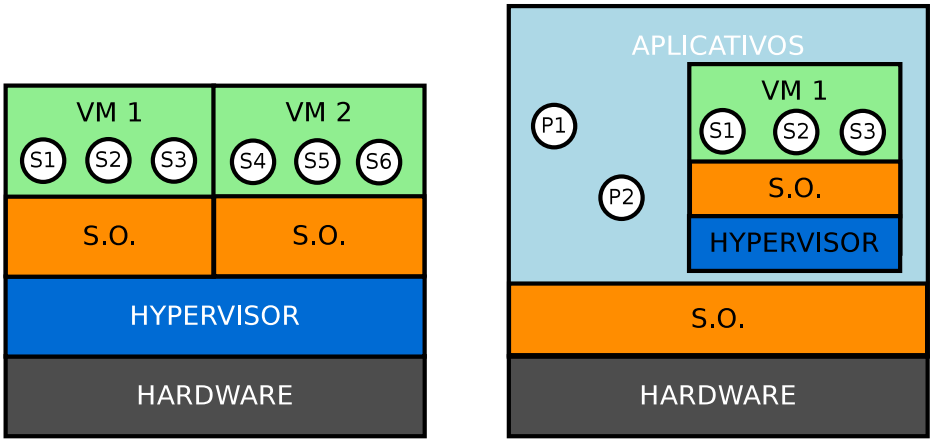
- *Sandboxing* - como uma máquina virtual é totalmente independente das outras hospedadas no mesmo servidor, ela pode ser utilizada para executar aplicações não confiáveis de modo que, caso essa aplicação cause algum dano ao sistema da máquina virtual, isso não afetará as outras máquinas virtuais, nem mesmo o servidor no qual a máquina virtual está.

- Suporte a múltiplos ambientes de execução - como dito anteriormente, em um mesmo servidor podem ser hospedadas várias máquinas virtuais, e cada uma pode possuir configurações distintas bem como executar aplicações totalmente diferentes uma das outras, com diferentes requisitos de *hardware* e *software*.

Com relação ao monitor de máquinas virtuais, existem dois tipos (LAUREANO; MAZIERO,): em nível de *hardware* (tipo 1) e em nível de *software* (tipo 2). No monitor tipo 1, o MMV executa diretamente sobre o *hardware*. Assim, o MMV tem como função controlar o acesso das máquinas virtuais ao *hardware* real fazendo com que cada máquina virtual tenha a ilusão de estar executando sobre uma máquina dedicada. Essa é a forma original adotada na década de 1960 pela IBM. Como exemplo desse tipo de virtualização pode-se citar o Xen (XEN, 2008), VMware ESXi (VSPHERE, 2011) e IBM Power VM (IBM, 2013). Já no monitor tipo 2, os MMVs são instalados e executados dentro de um sistema operacional. Para que as máquinas virtuais tenham acesso ao *hardware* da máquina real, faz-se necessário a utilização de chamadas ao sistema operacional sobre o qual está sendo executado. São exemplos o VirtualBox (ORACLE, 2013), QEMU (QEMU, 2013) e VMware Workstation (WORKSTATION, 2013).

Nas figuras 6(a) e 6(b), S1, S2, S3, S4, S5 e S6 representam serviços executados nas máquinas; P1 e P2 são programas executados em paralelo com a VM; e S.O. representa o sistema operacional usado. Em 6(a), não há um sistema operacional abaixo do hypervisor, sendo este responsável por gerenciar quais recursos serão alocados e utilizados por cada máquina virtual. Isso não acontece em 6(b), onde o monitor de máquinas virtuais executa como um processo do sistema operacional. Para acessar os recursos de *hardware*, ele utiliza chamadas ao sistema do sistema operacional sobre o qual está executando.

Figura 5: Tipos de Monitores de Máquinas Virtuais (*hypervisor*)



(a) *Hypervisor tipo 1* (b) *Hypervisor tipo 2*

Fonte: Laureano e Maziero, 2008.

2.2.1 *LIVE MIGRATION*

Devido ao ambiente com demanda variável, uma máquina virtual hospedada em um servidor pode necessitar de mais recursos do que o que está atualmente alocado para ela. Mas como fazer isso quando a capacidade do servidor está no limite? A saída encontrada foi realizar a migração da máquina virtual para outro servidor. A migração de máquinas virtuais é utilizada também para realizar o balanceamento de carga, tolerância a falhas e gerenciamento de energia (JIN et al.,).

Uma maneira intuitiva em se pensar na migração de uma máquina virtual seria parar sua execução, transferi-la para o servidor de destino, e, ao finalizar a transferência, reiniciar a execução da máquina. O problema dessa abordagem é que o tempo no qual a máquina virtual ficou parada (*downtime*) tem como consequência a indisponibilidade do serviço hospedado na máquina virtual, tornando-se indesejável para o cliente. Além disso, os SLAs acordados entre cliente e provedor de serviço poderão ser violados, ocasionando penalidades para o provedor de serviço.

Com objetivo de tornar a técnica de migração de máquinas virtuais mais eficiente, Clark et al. (2005) propuseram a técnica *live migration*. Ao se utilizar essa técnica, os clientes ou aplicações conectadas à máquina virtual a ser migrada não precisam se desconectar para realizar a migração.

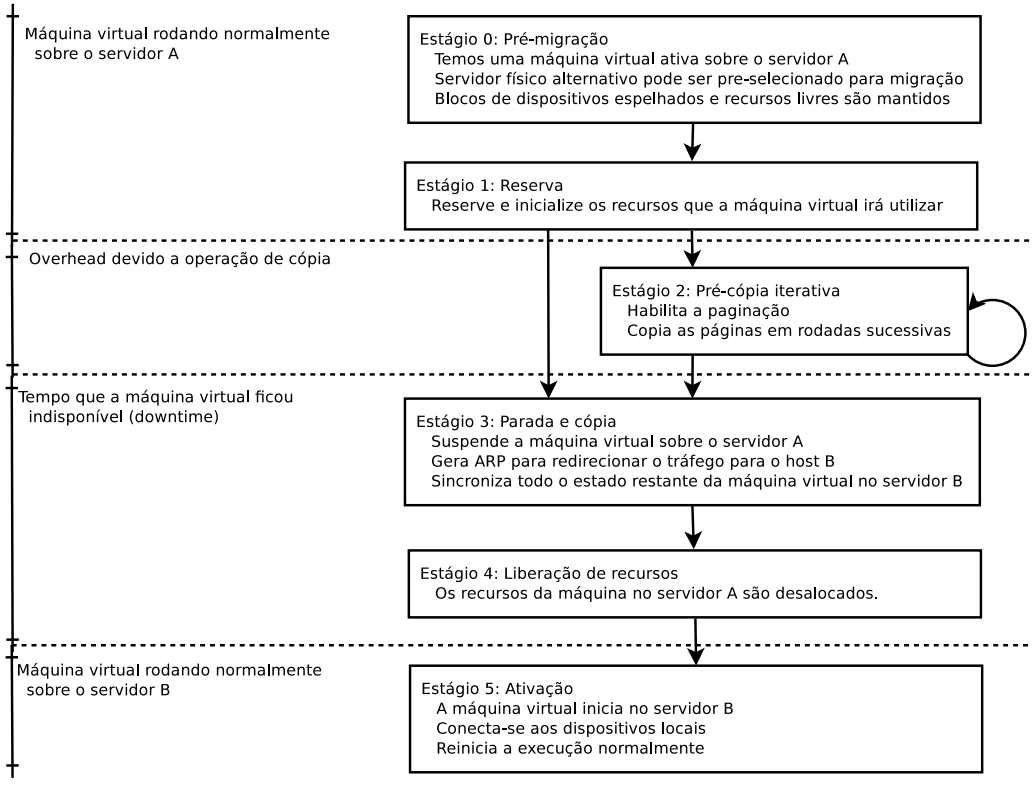
Quando a *live migration* é realizada, todas as informações pertinentes à máquina virtual devem também ser migradas, como os estados da memória física, conexões de redes e arquivos de sistema. Como as informações presentes na memória são muito dinâmicas, elas são as principais impactantes sobre a migração, visto que é necessário manter todas as alterações realizadas durante a migração sincronizadas nos dois servidores.

Com relação aos aspectos temporais da migração, o tempo total de uma migração é medida no momento de início do processo de migração até o momento de conclusão da migração. Nesse intervalo, haverá um período de tempo em que será necessário parar a máquina virtual para transferir as últimas porções de memória. Se comparada com a abordagem inicialmente descrita, esse tempo que a aplicação ficará indisponível pode ser considerada desprezível.

Com relação às abordagens utilizando *live migration*, pode-se considerar (SHRIBMAN; HUDZIA,):

- Pré-cópia → Na migração, com a máquina virtual em execução, primeiramente é realizada, dinamicamente, a cópia das páginas de memória para o servidor de destino. Para que a migração não degrade o desempenho das aplicações, essa etapa de cópia é gerenciada por um algoritmo adaptativo. Quando for concluída a cópia do conteúdo de memória para a máquina destino, para-se a máquina virtual em execução no servidor original, migra algum resquício de memória alterado entre a última verificação e o momento da parada da máquina virtual e desaloca os recursos utilizados nesse servidor. Em seguida, a máquina virtual no servidor destino da migração reinicia a máquina virtual, passando a funcionar como se nada tivesse acontecido. O esquema de funcionamento é mostrado na figura 6.

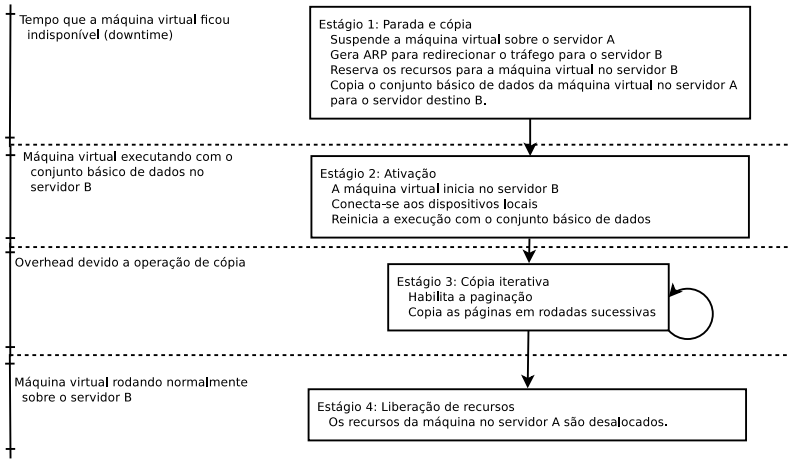
Figura 6: Funcionamento da *live migration* utilizando a abordagem de pré-cópia



Fonte: baseado em Clark et al. (2005).

- Pós-cópia → Nessa abordagem, a migração é realizada inicialmente parando a execução da máquina virtual. Após isso, identifica-se o conjunto básico de informações para o funcionamento da máquina virtual e migra esses dados para o servidor de destino. No servidor de destino, a máquina virtual é reiniciada com esse conjunto básico de informações. Os restantes das informações são transferidas em segundo plano. Caso seja feita uma requisição por algum dado que ainda não foi migrado, gera-se uma *trap* para informar ao servidor de origem que o servidor de destino está precisando de determinada informação para a máquina virtual sendo migrada. A desvantagem é a possibilidade de degradar o desempenho da aplicação, visto que para cada dado necessário não encontrado, é necessário consultar o servidor de origem, causando um atraso no tempo de resposta da máquina virtual. O esquema de funcionamento é mostrado na figura 7.

Figura 7: Funcionamento da *live migration* utilizando a abordagem de pós-cópia



2.3 Computação verde

A partir da popularização dos computadores pessoais, tem-se estudado os impactos do crescimento na utilização desses dispositivos. O principal impacto observado foi o aumento na demanda por energia ocasionando a necessidade de ampliação do sistema atual. Nesse contexto, surgiu o termo Computação Verde (*Green Computing*). Basicamente, quando se fala o termo "Verde" está relacionado em como reduzir os impactos no meio ambiente ocasionados pela utilização de dispositivos. Por exemplo, o termo pode está relacionado em como reduzir o consumo de energia de uma empresa com equipamentos de tecnologia da informação. Essa redução no consumo estará ajudando o meio ambiente a diminuir a quantidade de energia gerada para atender aos usuários.

2.3.1 Consumo de energia em um *data center*

Para atender toda a demanda dos usuários, os *data centers* possuem, basicamente, um grande número de servidores e equipamentos de rede, além de uma infraestrutura de resfriamento para dissipar o calor gerado por todos esses equipamentos (MOORE et al.,). A grande quantidade de energia consumida por toda essa infraestrutura é considerada como um dos principais custos de manutenção dos *data centers* (BELOGLAZOV; ABAWAJY; BUYYA,).

Com a consolidação da infraestrutura, ou seja, uma quantidade maior de dispositivos por unidade de área, a quantidade de calor gerada é grande. Para tratar desse calor gerado, são implantadas unidades de resfriamento encarregadas de retirar o calor e deixar o ambiente em uma temperatura tolerável. Caso esse calor gerado não seja tratado corretamente, ele pode diminuir consideravelmente a confiabilidade dos dispositivos (ANDERSON; DYKES; RIEDEL,).

Para um *data center* que ocupa uma área de 30000 pés^2 com 1000 *racks* de computação padrão, cada um consumindo 10kW, o custo inicial para a compra e instalação da infraestrutura de resfriamento é de $2-$5 milhões; bem como o custo anual para realizar o resfriamento é, aproximadamente, $4-$8 milhões (PATEL et al.,). Devido a esse fato, várias pesquisas estão direcionadas a descobrir métodos que consigam reduzir esse custo com sistema de resfriamento, entre elas a de Buyya, Beloglazov e Abawajy (2012), D. Abimannan (2011) e Sharma et al. (2005).

Entre vantagens de se utilizar o gerenciamento baseado na temperatura em um *data center* (MOORE et al.,) visando a manutenção do ambiente e a economia de energia, pode-se citar:

- Redução dos custos com resfriamento - Como consequência direta da implementação de um mecanismo para economia de energia baseado na temperatura, tem-se a redução dos gastos com energia. Como observado na literatura (MOORE et al., 2005; SHARMA et al., 2005), qualquer economia nesse sentido tem um grande impacto sobre os custos.
- Aumento da confiabilidade de *hardware* - Como os componentes eletrônicos presentes nos servidores são sensíveis a temperatura alta, sua confiabilidade diminui com o aumento da temperatura. Utilizando o gerenciamento baseado na temperatura, é possível manter o ambiente na temperatura ideal para o funcionamento dos dispositivos.

Analisando no contexto dos servidores, já existem técnicas que auxiliam na tarefa de reduzir o consumo de energia. Uma dessas técnicas é a virtualização, que ajuda a reduzir a subutilização dos servidores através da consolidação dos recursos em uso no *data center*. Desse modo, é possível hospedar uma quantidade maior de serviços e máquinas virtuais utilizando

uma quantidade menor de servidores, aumentando, assim, a taxa de utilização e reduzindo o consumo, já que haverá uma quantidade menor de servidores ativos (ARMBRUST et al.,).

No aspecto relacionado aos componentes internos do servidor, o processador é considerado como o principal consumidor de energia (BELOGLAZOV; ABAWAJY; BUYYA,). Por esse motivo, fabricantes de processadores estão projetando processadores com uma maior eficiência energética, além de fornecer mecanismo que habilitem o administrador do sistema a configurar o dispositivo para que tenha um gasto menor. Uma dessas técnicas é denomidada *Dynamic Voltage and Frequency Scaling* (DVFS), na qual é possível reduzir a voltagem e a frequência de clock do processador, gastanto assim uma quantidade menor de energia (SUEUR; HEISER,).

O estudo desses fatores influentes no consumo de energia são motivados tanto pela economia que os provedores de serviço terão, pois terão gastos menores com a energia, como pelo aspecto ambiental, pois serão consumidos menos recursos naturais à geração da energia necessária para a operação do *data center*. Quando há essa preocupação com os impactos ambientais ocasionados por um *data center* em computação nas nuvens, é dado o nome de *Green Cloud Computing* (KUMAR; BUYYA, 2012).

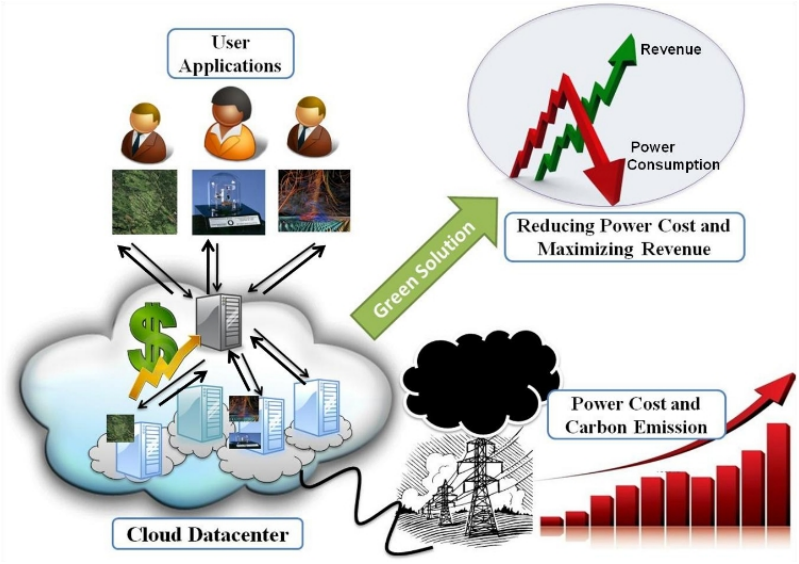
2.3.2 *Green cloud computing*

O termo *Green Cloud Computing* surgiu no momento da grande adoção do paradigma de Computação nas Nuvens. A partir dessa grande procura, surgiram novos *data centers* e a consequente expansão dos já existentes. Como o consumo de energia elétrica aumentou, fez-se necessário a ampliação de suas fontes atuais de fornecimento de energia elétrica, com intervenções que causaram e causam impactos no meio ambiente, como a construção de hidrelétricas e termoelétricas (KUMAR; BUYYA, 2012). *Green Cloud Computing* aborda os mesmos pontos de computação verde, mas em um ambiente de computação nas nuvens.

Comparando o gasto de um *data center* em computação nas nuvens com outro no modelo tradicional, já observa-se a existência de uma economia de energia na adoção de computação nas nuvens, obtida pela arquitetura baseada por ela. Essa economia (figura 8) é obtida devido a utilização da técnica de virtualização, onde é possível realizar uma maior consolidação nos servidores, reduzindo assim a subutilização dos recursos.

Os proprietários dos *data center* reconhecem que o custo gasto com energia é alto. Para tornar o projeto viável, é preferível instalá-los perto das fontes geradoras de energia, de modo a conseguir preços mais baixos a instalá-los no meio de uma cidade, mais distantes das fontes geradoras de energia. Ou seja, muitas vezes, os *data centers* são instalados em lugares onde a energia tem um preço menor ao invés de se utilizar técnicas para que o consumo de sua infraestrutura seja menor.

Figura 8: *Green Cloud Computing*



Fonte: Kumar e Buyya (2012).

Com isso, os governos e entidades não governamentais, como o *Greenpeace*, começam a pressionar os provedores de serviços para adotarem métodos que reduzam o consumo de energia dos *data centers*. Algumas das grandes empresas do setor já se mobilizaram e criaram o consórcio global *Green Grid* com o intuito de prover a eficiência no consumo de energia para os *data centers* e, consequentemente, redução no impacto ambiental (BELOGLAZOV; ABAWAJY; BUYYA,).

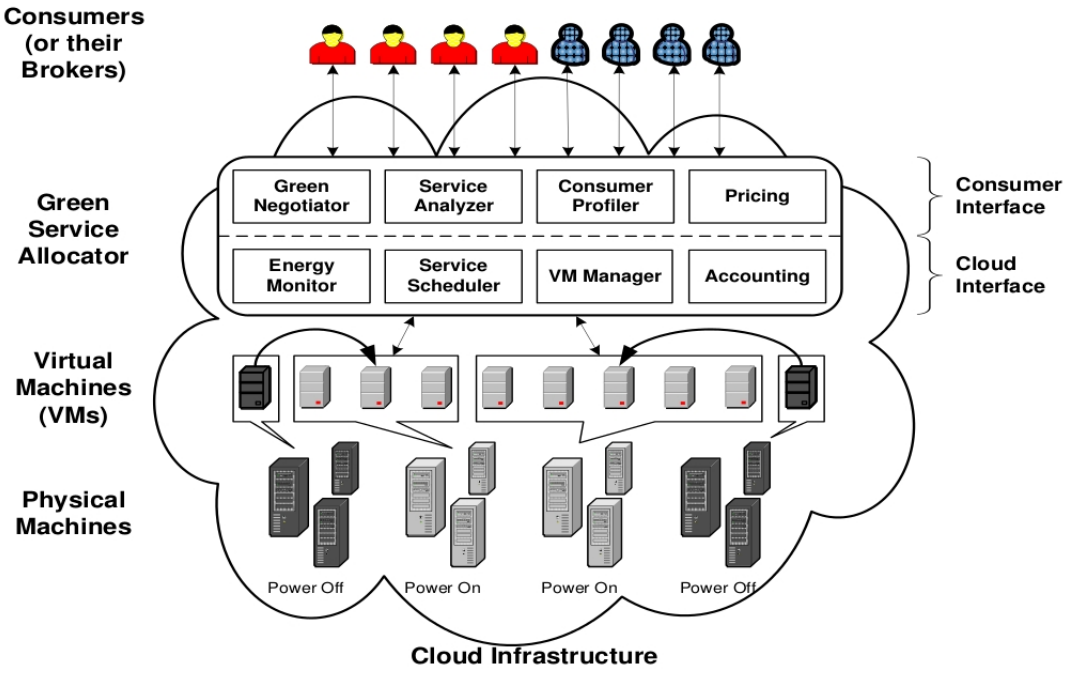
Na literatura (LAGO; MADEIRA; BITTENCOURT, 2011; VALENTINI et al., 2011; BELOGLAZOV; BUYYA, 2010; D.; ABIMANNAN, 2011) pode-se encontrar diversos trabalhos que tratam a questão da redução do consumo de energia em um *data center*. Tais soluções trazem benefícios não somente para o meio ambiente, mas também para os provedores de serviços, uma vez que o custo de energia diminuirá com a redução do consumo de energia.

2.3.3 Elementos de uma arquitetura *Green Cloud Computing*

O projeto da arquitetura de um mecanismo de alocação depende do critério que será utilizado como base para realizar a alocação das máquinas virtuais. Este trabalho foca na questão de alocação utilizando o critério de consumo de energia. Desse modo, foi utilizada a arquitetura apresentada em (BELOGLAZOV; BUYYA,). Como mostrado na figura 9, essa arquitetura é composta pelos seguintes elementos:

- Clientes (*Consumers*): são os interessados em implantar os seus serviços no *data center*. Conforme especificado por Beloglazov e Buyya (2010), é importante diferenciar os clientes dos usuários dos serviços implantados. Os clientes são aqueles que estão implantando

Figura 9: Arquitetura *Green Cloud Computing*



Fonte: Beloglazov e Buyya (2010).

o serviço na nuvem para atender a demanda dos usuários e estes utilizam esse serviço implantado.

- Alocador de Serviços Verde (*Green Service Allocator*): funciona como uma interface entre os clientes (*Consumer Interface*) e a nuvem (*Cloud Interface*). Para atender a requisição de alocação de recursos de um modo em que se consiga a economia de energia, os componentes dessa interface trabalham em conjunto. Os componentes são:

  – Negociador Verde (*Green Negociator*): antes da implantação de um serviço na nuvem, deve haver uma interação entre o cliente e o provedor de serviços para a definição dos recursos que serão utilizados, bem como as garantias que o serviço deve ter, as penalidades que serão aplicadas caso uma das partes não cumpra o especificado e de como serão tarifados os recursos oferecidos. Ao final desse processo, o Negociador Verde gera um SLA com todas essas informações a serem atendidas tanto pelo provedor de serviços como pelos clientes.

  – Analisador de Serviços (*Service Analyser*): antes do estabelecimento do SLA, o provedor de serviços deve saber se ele possui recursos suficientes para atender a requisição para decidir se aceita ou rejeita a requisição do cliente. Essa função é realizada pelo Analista de Serviços. Para o correto funcionamento desse módulo, ele deve ser capaz de obter as informações de estado dos recursos do *data center* assim como obter os dados de consumo de energia. Essas informações são obtidas através dos módulos Gerenciador de Máquinas Virtuais e Monitor de Energia

respectivamente.

  - Perfis de Consumidores (*Consumer Profile*): este módulo estabelece métodos de categorização dos clientes. Para isso, esse módulo utiliza as características das máquinas virtuais e uma medida de prioridade daquele serviço que diz o quão importante é aquele serviço comparado com os outros.
  - Tarifação (*Pricing*): de acordo com a informação de utilização dos recursos pela máquina virtual do cliente e pelos valores acordados via SLA, esse módulo calcula o valor a ser pago pelo cliente pelos recursos utilizados.
  - Monitor de Energia (*Energy Monitor*): Módulo responsável pelo monitoramento da infraestrutura física do *data center*, de modo a determinar o consumo das máquinas físicas (servidores) bem como determinar se as máquinas estão ligadas ou desligadas.
  - Escalonador de Serviços (*Service Scheduler*): Uma vez que a requisição do cliente tenha sido aceita pelo Analisador de Serviços, esse módulo é responsável pela alocação da máquina virtual em um servidor assim como a vinculação da máquina virtual com os recursos físicos especificados no SLA. Além dessa responsabilidade de implantação de máquinas virtuais, ele pode adicionar ou remover máquinas virtuais de um determinado serviço de modo a atender a alta na demanda ou liberar recursos que não estão sendo utilizados, respectivamente.
  - Gerenciador de Máquinas Virtuais (*VM Manager*): responsável por manter os dados de utilização dos recursos dos servidores pelas VMs. Além disso, de acordo com a política especificada pelo mecanismo de alocação, ele pode determinar a migração de máquinas virtuais entre servidores.
  - Contagem (*Accounting*): responsável por manter a medida atual e histórico de utilização dos recursos do *data center*. Estas informações são importantes quando o *data center* deseja fazer predição da utilização dos seus recursos em um momento futuro.

- Máquinas Virtuais (*Virtual Machines - VMs*): como definido anteriormente, uma máquina virtual é o conjunto de recursos fornecendo a ilusão de que aqueles recursos formam uma máquina física independente. A sua alocação ou remoção pode ser realizada a qualquer momento de acordo com as necessidades do cliente. Uma máquina virtual pode estar alocada a somente um servidor, mas sobre um servidor podem estar alocadas diversas máquinas virtuais.

- Máquinas Físicas: infraestrutura de recursos do provedor de serviços, onde é fornecido o *hardware* necessário para que seja feita a alocação das máquinas virtuais.

2.4 Alocação de máquinas virtuais em computação nas nuvens

Em um ambiente de computação nas nuvens, muito se fala sobre a questão da alocação de máquinas virtuais nos servidores de um *data center*, onde usuários enviam requisições ao *data center* para alocação de recursos em seus servidores. Para isso, o *data center* cria máquinas virtuais para atenderem a demanda do cliente através de suas alocações em algum servidor de sua infraestrutura. O problema está em determinar o servidor para alocar as máquinas virtuais. Como se trata de um ponto central na utilização de computação nas nuvens, essa área possui muitos estudos relacionados à melhoria dos mecanismos de alocação das máquinas virtuais (BUYYA; BELOGLAZOV; ABAWAJY, 2010; BELOGLAZOV; ABAWAJY; BUYYA, 2012; SOARES, 2010; LAGO; MADEIRA; BITTENCOURT, 2011; K.MUKHERJEE; G.SAHOO, 2010).

Um dos problemas nos mecanismos de alocação de máquinas virtuais está no fato da complexidade temporal para alocação de um conjunto de n ($n \in \{1, \cdots, n\}$) máquinas virtuais em um conjunto de m servidores ($m \in \{1, \cdots, m\}$). Uma solução intuitiva conduz a utilização de um algoritmo guloso, onde, para cada máquina virtual, percorre-se todos os servidores para selecionar em qual deles deve ser feita a alocação. Esse algoritmo tem complexidade temporal da ordem O(n x m). Para uma grande quantidade de máquinas virtuais e de servidores, o tempo necessário para achar uma solução seria inviável. Desse modo, faz-se necessário a utilização de técnicas mais eficientes para solucionar o problema em tempo viável.

Esse problema de alocação é semelhante ao problema de alocação de recursos. Nesse problema, há um conjunto de recursos disponíveis e é necessário particioná-los e alocá-los entre um conjunto de entidades de forma a atender a algum objetivo. Na maioria dos casos, utiliza-se como objetivo atender a todas as requisições do conjunto de entidades por recursos de modo a obter o menor custo. Fazendo a equivalência, os recursos são os servidores, as entidades são as máquinas virtuais e o objetivo é originado da política de alocação adotada pelo *data center*.

Para determinar a escolha do servidor, podem ser utilizados diversos critérios. Os principais deles são (BELOGLAZOV; ABAWAJY; BUYYA,):

- Servidor com menor quantidade de recursos não utilizados - analisa-se qual servidor possui a menor quantidade de recursos não utilizados e realiza-se a alocação, desde que esses recursos sejam suficientes para a alocação da máquina virtual. Nesse caso, busca-se a consolidação das máquinas virtuais na menor quantidade de servidores possível.

- Servidor com maior quantidade de recursos não utilizados - mesmo procedimento do item anterior, diferenciando-se apenas pela escolha do servidor que possui a maior quantidade de recursos não utilizados. Desse modo, tenta-se alocar as máquinas virtuais do modo mais disperso possível.

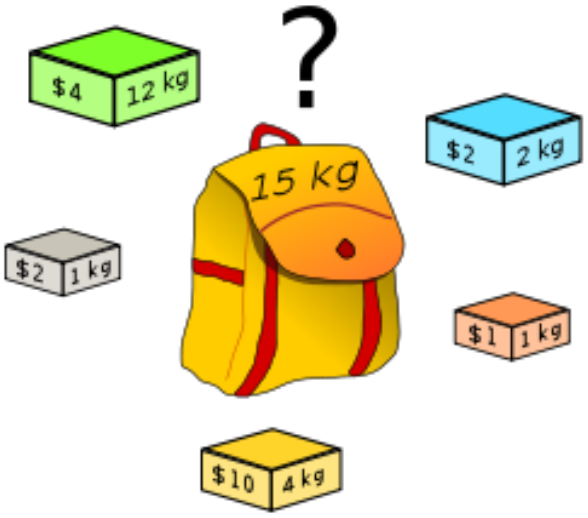
- Aleatório - seleciona-se um servidor aleatoriamente e aloca a máquina virtual, isso se o servidor possuir recursos suficientes para receber a máquina virtual.
- Menor consumo de energia - com esse critério, busca-se o servidor onde o impacto da alocação da máquina virtual seja o menor.

Este trabalho visa abordar o problema de alocação de recursos (máquinas virtuais) em um ambiente de computação nas nuvens tendo como objetivo a economia de energia. Como técnica para solucionar esse problema, será modelado como um problema da mochila (*knapsack problem*), adequado à alocação de recursos. Dentre os problemas de mochila, o de alocação será modelado como um problema de múltiplas mochilas (*multiple knapsack problem*) (KELLERER; PFERSCHY; PISINGER,), abordado nas seções subsequentes.

2.5 Problema de múltiplas mochilas

Um cenário onde se tem um conjunto de itens a serem colocados em uma mochila é denominado de problema da mochila. Nesse problema, existe uma determinada quantidade de itens, cada qual com o seu peso e valor, onde deseja-se colocar esses itens em uma mochila com uma capacidade predefinida (figura 10). O objetivo é colocar os itens na mochila de modo a se obter o maior valor (composto pela soma de valores de cada item inserido na mochila), desde que não ultrapasse o peso total suportado pela mochila. O problema da mochila 0-1 (também chamado de problema da mochila binária) é considerado o mais simples deles (MARTELLO; TOTH, 1990).

Figura 10: Ilustração do problema da mochila



Definindo o problema de um modo mais formal, tem-se um conjunto com n itens, onde cada item j (j=1,$\cdots$,n) possui um peso w_j e um lucro v_j associados. O lucro é uma medida que

define o quão importante é aquele objeto, utilizado para a priorização dos itens a serem inseridos na mochila. A mochila possui uma capacidade C, determinando o peso máximo suportado por ela. A variável x_j é utilizada para especificar se um determinado item j foi colocado na mochila. Se o item j foi escolhido para ser colocado na mochila, a variável x_j será preenchida com valor 1; caso contrário, será preenchida com valor 0. Como objetivo para a solução desse problema, deseja-se maximizar o lucro obtido pelos itens inseridos na mochila. Matematicamente, temos (MARTELLO; TOTH, 1990):

$$Maximizar\ z\ =\ \Sigma_{j=1}^{n}\ v_j x_j, \tag{2.1}$$

$$sujeito\ a$$

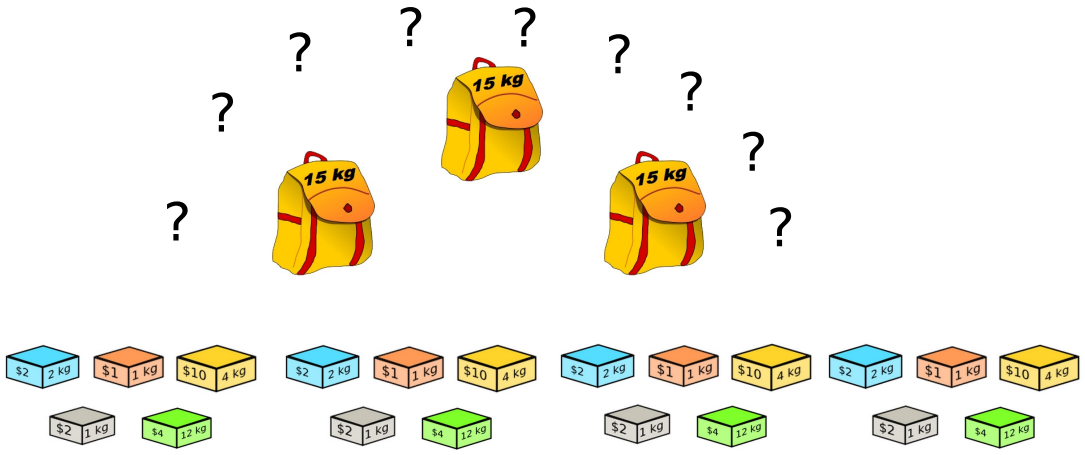
$$\Sigma_{j=1}^{n}\ w_j x_j\ \leq\ C, \tag{2.2}$$

$$x_j\ =\ 0\ ou\ 1,\ j\ \in\ N\ =\ \{1,\ \cdots,\ n\} \tag{2.3}$$

Em 2.1, é definido o objetivo de maximizar o lucro obtido pelos itens inseridos na mochila. Já na restrição 2.2, é especificada a soma de pesos dos itens inseridos, não podendo ultrapassar a capacidade C da mochila. Por fim, na restrição 2.3 são especificados os valores que a variável x_j pode receber (0 ou 1), informando se o item foi ou não inserido, respectivamente.

Um outro tipo de problema da mochila envolve múltiplas mochilas. O problema de múltiplas mochilas (figura 11) é muito similar ao problema da mochila simples. A diferença está no número de mochilas utilizadas: enquanto no problema simples só utiliza uma mochila, no de múltiplas mochilas são utilizadas várias mochilas. Esse problema foi proposto para ser aplicado em problemas de tolerância a falhas (SINHA; ZOLTNERS,), esquema de criptografia pública (DIFFIE; HELLMAN,), problemas de alocação e escalonamento, entre outros. A sua formulação matemática é descrita por Martello e Toth (1990):

Figura 11: Ilustração do problema de múltiplas mochilas



$$Maximizar\ z\ =\ \Sigma_{i=1}^{m}\ \Sigma_{j=1}^{n}\ v_j x_{ij}, \tag{2.4}$$

$$sujeito\ a$$

$$\Sigma_{j=1}^{n}\ w_j x_{ij}\ \leq\ c_i, \qquad i \in M\ =\ \{1,\ \cdots,\ m\}, \tag{2.5}$$

$$\Sigma_{i=1}^{m}\ x_{ij}\ \leq\ 1, \qquad j \in N\ =\ \{1,\ \cdots,\ n\}, \tag{2.6}$$

$$w_j \leq max_{i \in M}\{c_i\}, \qquad para\ j \in N, \tag{2.7}$$

$$c_i \geq min_{j \in N}\{w_j\}, \qquad para\ i \in M, \tag{2.8}$$

$$v_j\ e\ c_i\ =>\ inteiros\ positivos, \tag{2.9}$$

$$x_{ij}\ =\ 0\ ou\ 1, \qquad i \in M,\ j\ \in\ N \tag{2.10}$$

onde:

- n é a quantidade total de itens;
- m é a quantidade total de mochilas;
- w_j e v_j representam, respectivamente, o peso e o valor do item j;
- c_i representa a capacidade da mochila i;
- x_{ij} informa se o item j está presente (x_{ij} = 1) ou não (x_{ij} = 0) na mochila i.

O objetivo desse problema (equação 2.4) é maximizar o valor total obtido por inserir os elementos nas mochilas. Como restrições, tem-se as equações 2.5 e 2.6 estabelecendo, respectivamente, que a soma dos pesos dos itens inseridos em uma mochila não pode exceder a capacidade total da mochila, e um item só pode está inserido em uma mochila. É necessário também estabelecer condições mínimas para se realizar a inserção de um item nas mochilas, sendo para isso aplicadas regras restringindo o peso máximo de um item a ser menor ou igual à maior capacidade entre todas as mochilas (2.7), e que a capacidade de uma mochila deve ser maior ou igual ao menor peso entre os itens (2.8). Por fim, a restrição 2.9 limita o problema a só trabalhar com valores inteiros e positivos.

Dada a modelagem do problema conforme especificado anteriormente, faz-se necessário a utilização de uma técnica para resolver esse problema. Para isso, foi escolhido o algoritmo baseado em colônia de formigas.

2.6 Otimização por colônia de formigas

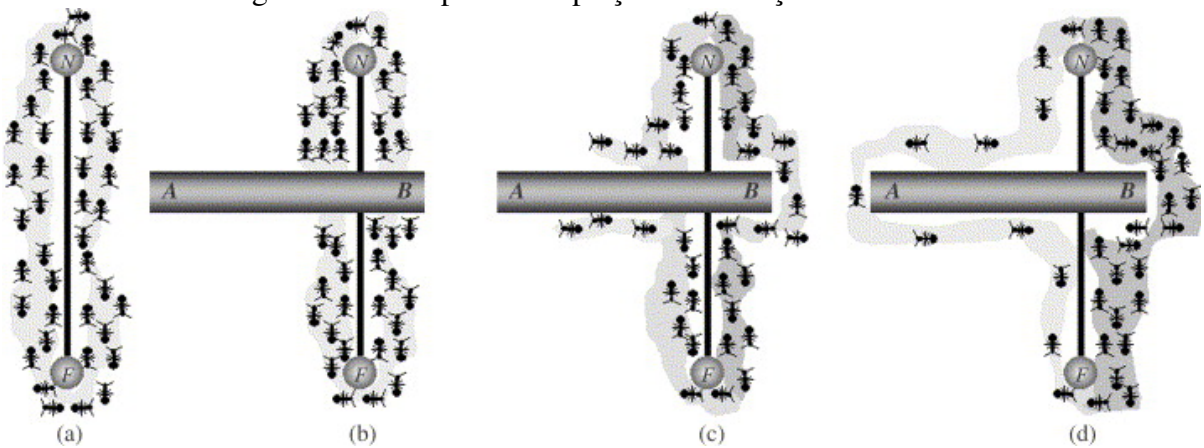
Dada a modelagem acima, é necessário selecionar um método eficiente para a resolução do problema de alocação das tarefas para as máquinas virtuais. Para isso, foi escolhida

a otimização por colônia de formigas, pois seu desempenho na solução de outros problemas mostrou-se eficiente quando comparado com outras metaheurísticas (BOZDOGAN; YILMAZ; EFE, 2010). A seguir, será mostrado o comportamento das formigas reais e, em seguida, das artificiais.

2.6.1 As colônias de formigas

Procurando otimizar o tempo na resolução de problemas de otimização combinatorial, Dorigo, Maniezzo e Colorni (1996) criaram uma metaheurística baseada no comportamento das colônias de formigas forrageiras denominada sistema de formigas (*Ant System Colony*). Uma das primeiras aplicações desse método foi na resolução do problema do caixeiro viajante (DORIGO; GAMBARDELLA,), onde foram obtidos resultados satisfatórios tendo como objetivo a escolha do menor caminho. A escolha da colônia de formigas foi motivada pelo seu comportamento cooperativo, onde os indivíduos se comunicam indiretamente entre si para otimizar o processo de coleta de alimentos (BECKERS; DENEUBOURG; GOSS, 1992). Outra característica importante é a questão da adaptabilidade das formigas à mudança no ambiente (HÖLLDOBLER; WILSON, 1998). Como pode ser visto no exemplo na figura 12, caso uma das rotas entre o formigueiro e a fonte de alimento seja bloqueada/rompida (figura 12 (b)), as formigas iniciam novamente o processo para determinar um novo caminho mais curto para a fonte de alimento (figura 12 (c)).

Figura 12: Exemplo de adaptação a mudanças no ambiente



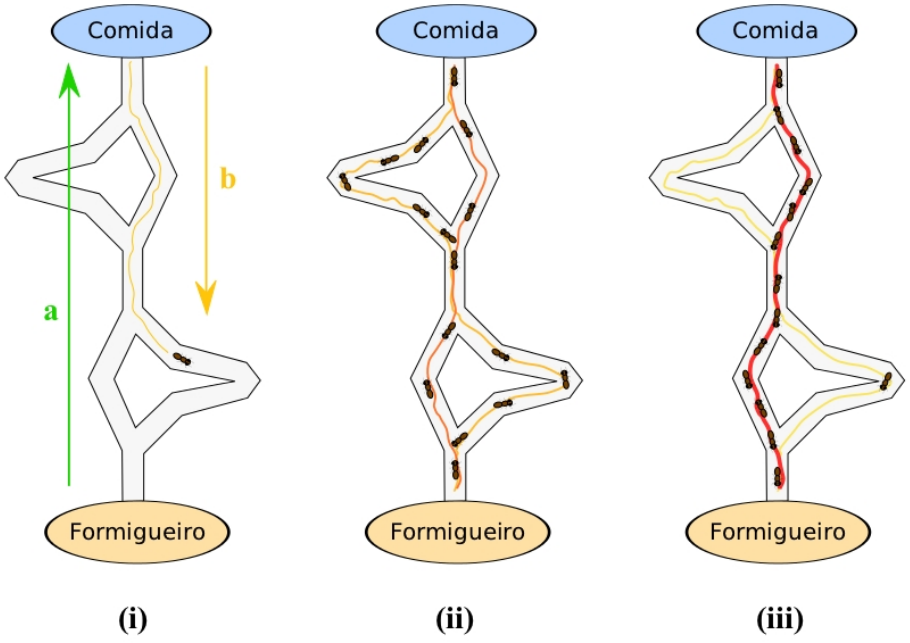
Fonte: Zecchin et al. (2006).

O procedimento adotado pelas formigas para determinar qual o caminho mais curto é muito simples. Inicialmente, elas buscam aleatoriamente fontes de alimento ao redor do formigueiro. Quando uma formiga encontra uma fonte de alimento, ela volta ao formigueiro depositando no percurso uma substância química denominada de feromônio. Alguns fatores que influenciam na quantidade de feromônio depositada no percurso são a qualidade e a quantidade de alimento (CORREIA, 2011). Assim, o feromônio serve para que as formigas se

comuniquem indiretamente entre si, de modo a determinar qual é a melhor fonte de alimento e qual o melhor caminho até ela. Essas informações são obtidas de acordo com a quantidade de feromônio depositada em cada trilha, de modo que, quanto maior a quantidade de feromônio, menor é o caminho até a fonte. Isso é consequência de um número maior de formigas seguindo por esse caminho, ocasionando maior acúmulo de feromônio depositado por cada uma delas. Outra característica do feromônio é que ele evapora com o tempo. Assim, quando uma fonte de alimento se esgota, aquela trilha deixará de ser utilizada (consequentemente, o feromônio evaporará) e as formigas irão repetir o processo para encontrar outra fonte de alimento.

O comportamento descrito pode ser observado na figura 13. Resumidamente, temos em 13 (i) uma formiga voltando ao formigueiro deixando uma trilha de feromônio após achar uma fonte de alimento na fase de busca aleatória. Em 13 (ii), outras formigas começam a explorar a mesma fonte de alimento. Como a taxa e feromônio ainda está baixa, elas exploram outras alternativas de caminho para a fonte de alimento. Por fim, em 13 (iii) o menor caminho é estabelecido, pois as formigas desse caminho conseguem coletar o alimento e retornar ao formigueiro mais rápido do que as formigas utilizando os caminhos alternativos. Desse modo, a quantidade de feromônio concentrou-se no menor caminho.

Figura 13: Funcionamento da colônia de formigas na busca por alimentos



Fonte: Correia (2011).

Baseado nesse comportamento, a metaheurística de colônia de formigas foi criada através da modelagem matemática desse comportamento. Essa metaheurística será abordado na seção a seguir.

2.6.2 Metaheurística de otimização por colônia de formigas

Para a utilização da metaheurística de Otimização por Colônia de Formigas (*Ant colony Otimization*, ACO) é necessária a modelagem do problema abordado em um grafo, de modo a transformar o problema em encontrar os melhores caminhos dentro do grafo utilizando as características de auto-organização e cooperação apresentadas pelas formigas (DORIGO; BLUM,). As metaheurísticas têm como principal característica o fato de não se basear em pressupostos do problema abordado, além de realizar uma busca eficiente no espaço de soluções. São empregadas principalmente nos problemas *NP-Hard*, onde não existem algoritmos para as soluções com tempo polinomial. Aplicadas nesses tipos de problemas, as metaheurísticas não garantem encontrar a solução ótima, mas gera boas soluções, próximas da ótima.

O comportamento da ACO na obtenção dos melhores caminhos em um grafo são similares ao comportamento observado nas formigas reais. Dado um vértice de partida (formigueiro), as formigas percorrem as arestas que ligam dois vértices depositando feromônio, podendo ser considerado como o peso da aresta ou valor de um vértice. Em cada vértice, a formiga avalia qual o vértice lhe é mais vantajoso (dependendo do objetivo do problema o qual se deseja resolver) de acordo com o feromônio presente nos vértices com o intuito de determinar o próximo movimento da formiga. Após a escolha do vértice, a formiga visita aquele o vértice escolhido e o feromônio do vértice é atualizado. Ao fim da iteração, representada pelo movimento de um vértice à outro no grafo, é feita a atualização global do feromônio das arestas, podendo ser incrementado, caso seja o vértice escolhido pela formiga ou decrementado de acordo com a taxa de evaporação do feromônio.

Como exemplo, será utilizada uma modelagem genérica (DORIGO; BIRATTARI; STüTZLE,). Será utilizado um grafo conexo e não orientado G = (V, E), onde cada vértice possui um lucro e cada aresta possui um peso. O objetivo é encontrar o percurso no grafo onde se tenha o maior lucro global, respeitando as restrições impostas. O valor do lucro global é calculado pela soma individual do lucro de cada vértice visitado pela formiga. Para evitar que um vértice seja visitado mais de uma vez, cada formiga possui uma memória onde são armazenados os vértices já visitados, de modo que, quando uma formiga está decidindo para qual vértice ela irá, busca-se o vértice de destino na memória para que não haja duplicações na solução. Ao final da iteração, é aplicada a fórmula de atualização dos feromônios presente em cada aresta.

Matematicamente, teremos a seguinte abordagem: cada vértice i $\in$ V é modelado como uma variável de decisão x_i. Estando em um vértice i, o lucro obtido para ir ao vértice j é representado por v_j. Se a variável de decisão x_i for configurada com valor 1, o vértice i pertence ao conjunto solução S; caso contrário, será utilizado o valor 0. Assim, o conjunto das soluções do problema é formado por pares x_i (determinando qual vértice foi selecionado) e o respectivo lucro v_j do vértice. Por fim, tem-se o conjunto representando as trilhas de feromônio no grafo

Γ, com o valor τ_i associado a cada componente da solução.

2.6.3 Algoritmo ACO simples

Dorigo, Birattari e Stützle (2006) descrevem genericamente o algoritmo da metaheurística de colônia de formigas na forma de etapas de modo que toda formulação baseada nessa abordagem deve possuir. Isto deve-se ao fato da existência de diversas implementações diferentes, sendo cada qual especializada para o problema que se está tentando resolver.

O funcionamento do algoritmo definido por Dorigo, Birattari e Stützle (2006) (mostrado no algoritmo 3) é descrito como segue: A cada iteração, n_a formigas são criadas para explorar o ambiente de soluções do problema de acordo com um determinado modelo de feromônio, onde cada formiga gera uma solução. Sobre cada solução gerada por uma formiga, pode-se, opcionalmente, realizar uma busca local a partir da solução encontrada a fim de tentar melhorar a solução. Por fim, antes que seja realizada uma nova iteração, é realizada a atualização dos feromônios de acordo com a solução encontrada. Esse processo continua até as condições de saída serem satisfeitas. Desse modo, o algoritmo pode ser dividido em quatro partes: inicialização das variáveis (feromônio), a construção da solução, uma busca local e, por fim, a atualização do feromônio. A seguir, os procedimentos serão detalhados de maneira mais profunda.

Algoritmo 1: Algoritmo ACO básico

```
1 Entrada: Uma instância P de um problema de otimização combinatória
2 Configura os parâmetros, Inicializa os feromônios;
3 while condições de saída não forem satisfeitas do
4 |   Constrói Solução;
5 |   Aplica Busca Local; // OPCIONAL
6 |   Atualiza Feromônio;
```

Fonte: Dorigo, Birattari e Stützle (2006)

Configura os parâmetros, Inicializa Feromônio. Antes das formigas iniciarem a busca por soluções, é necessário realizar a inicialização das variáveis do ambiente. No caso, é feita a inicialização de alguns parâmetros e dos valores de feromônio.

Constrói Solução. Nesse procedimento, as soluções são construídas utilizando métodos probabilísticos. Uma solução, no caso do ACO, é composta por um conjunto de vértices percorridos sobre um grafo, sendo representada por S. A solução é montada incrementalmente, respeitando as restrições, utilizando-se os vértices visitados a partir de um percurso sobre um

grafo completamente conectado G = (V, A). Para determinar quais serão os vértices visitados, em cada iteração (ou seja, em cada vértice), utiliza-se uma probabilidade de seleção. Essa probabilidade é calculada para todos os vértices ainda não visitados pela formiga. Clemente (2010) adota a probabilidade de seleção baseada na ponderação da quantidade de feromônio (τ_j) e da informação heurística (η_j) associada ao vértice. Desse modo, tem-se que a probabilidade de seleção (p_j) de um item é definida como:

$$p_j = \frac{(\tau_j^\alpha) * (\eta_j^\beta)}{\Sigma_{\forall y \in Cand}(\tau_j^\alpha) * (\eta_j^\beta)} \tag{2.11}$$

$$0 < p_j \leq 1; \tau_j, \eta_j > 0; \alpha, \beta \geq 0 \tag{2.12}$$

onde:

- τ_j é a quantidade de feromônio no vértice;
- η_j representa a informação heurística, definida como a relação lucro/custo associado ao item j. É também denominada de visibilidade, pois são capturadas informações do ambiente utilizadas para guiar as formigas para as regiões com boas soluções, melhorando, assim, a velocidade de convergência das soluções.
- α e β são utilizados para determinar quais dos parâmetros utilizados na probabilidade de seleção tem mais influência na geração da solução. No caso, os parâmetros influenciados são o feromônio e a informação heurística, respectivamente. Os valores possíveis para os expoentes estão na faixa de valores entre 0 e 1. Desse modo, quanto mais próximo de 1 estiver o expoente, menor será a influência do parâmetro em questão.
- *Cand* representa uma lista contendo todos os vértices não visitados, ou seja, vértices candidatos. O elemento y é um elemento genérico dessa lista.

Busca Local: são feitas pequenas variações na solução encontrada com o intuito de se encontrar uma solução melhor, sem ter que realizar todo o procedimento de construção de solução novamente. Apesar desse procedimento geralmente melhorar a solução encontrada diminuindo o tempo necessário para a convergência para uma solução ótima, ele é opcional (DORIGO; BLUM,).

Atualiza Feromônio. Os objetivos ao se atualizar o feromônio são dois: primeiramente, aumentar o nível de feromônio nos caminhos onde existem boas soluções, tornando-os

mais atraentes; o outro é decrementar globalmente uma parte do feromônio de modo a diversificar as soluções encontradas pela metaheurística, de modo que não fique presa em ótimos locais. Existem diferentes formas de se realizar a atualização de feromônio (DORIGO; BLUM, 2005; DORIGO; BIRATTARI; STüTZLE, 2006). Como exemplo, tem-se a utilizada por Dorigo, Birattari e Stützle (2006):

$$\tau_j = (1-\rho) * \tau_j + \Delta\tau_j \tag{2.13}$$

onde:

- ρ é a taxa de evaporação de feromônio, estando no intervalo entre 0 e 1. Como dito anteriormente, esse parâmetro é utilizado para fugir de ótimos locais de modo a explorar outras áreas no espaço de soluções.
- $\Delta\tau_j$ é a taxa de atualização de feromônio presente no vértice j, sendo proporcional a qualidade da solução.

2.7 Sumário do capítulo

Neste capítulo foram apresentados conceitos relativos à computação nas nuvens, à computação verde, ao problema de múltiplas mochilas e à otimização por colônia de formigas, onde foram abordados pontos relevantes ao desenvolvimento desse trabalho.

No capítulo a seguir será apresentada uma fundamentação sobre os trabalhos relacionados com a alocação de máquinas virtuais em computação nas nuvens considerando o consumo de energia.

3 TRABALHOS RELACIONADOS

Este capítulo apresenta uma revisão de alguns trabalhos sobre o problema de alocação de recursos levando em consideração o consumo de energia ocasionado pela alocação no *data center*.

3.1 Mecanismos de alocação

Analisando os diversos trabalhos na área de mecanismos de alocação, percebe-se o foco da grande maioria no uso da taxa de utilização dos servidores como base para o processo de tomada de decisão visando a alocação das máquinas virtuais. Geralmente, esses mecanismos tinham como objetivo o gerenciamento das ocorrências de violações de SLA (TAI et al., 2011; BOBROFF; KOCHUT; BEATY, 2007), adaptação à carga de trabalho ao qual o serviço está sendo submetido (ZHANG et al., 2009; WANG; VARELA, 2011), entre outros (LI; LO, 2012; LI, 2009; HU; WU; HUANG, 2012). Desse modo, essas soluções buscavam aumentar o lucro do provedor de serviço através da redução das violações de SLA.

Uma tendência recente tem incentivado a abordagem de outro critério para se alcançar a redução de gastos do provedor de serviços, bem como prover o aumento do lucro: o consumo de energia (FANG et al.,). A busca por mecanismos de alocação baseados no consumo de energia foi motivada tanto pela questão econômica, como também pela pressão de entidades não governamentais e de governos para a adoção de técnicas que proporcionassem a redução no consumo de energia (BELOGLAZOV; ABAWAJY; BUYYA,). Tal pressão foi motivada pela constatação da grande quantidade de energia necessária para manter toda a infraestrutura de um provedor de serviço e o eventual impacto no meio ambiente ocasionado por esse consumo, além da grande quantidade de *data centers* criados para atender essa nova demanda pelos serviços oferecidos em computação nas nuvens. Com isso, as políticas dos mecanismos de alocação passaram a adotar o consumo de energia em conjunto com a questão de atendimento do SLA, visando obter o menor consumo de energia sem, contudo, interferir nos SLAs acordados com os clientes.

Abordando o consumo de energia nos *data centers*, pode-se dividir os trabalhos em três vertentes principais:

- Consumo da infraestrutura de rede: A alocação de máquinas virtuais considerando o consumo da infraestrutura de rede vem ganhando uma atenção maior, indo contra a concentração dos trabalhos na área de alocação de máquinas virtuais que consideram apenas o consumo dos servidores ou o consumo da infraestrutura de resfriamento (FANG et al.,).

Esse fato é motivado por resultados de estudos recentes que mostram que o consumo dos elementos de rede em um *data center* consomem de 10% a 20% da energia total, surgindo o desafio de se conseguir efetuar uma redução nesse consumo sem degradar o desempenho da rede. Soluções nesse sentido (BOLLA et al., 2011; MENG; PAPPAS; ZHANG, 2010)tentam concentrar as máquinas virtuais de modo a utilização de uma menor quantidade de equipamentos de rede, a fim de que os não utilizados sejam desligados. Para tornar isso possível, faz-se necessário a monitoração constante do tráfego de dados para a adequação dinâmica da localização das VMs de acordo com o tráfego atual (BELOGLAZOV; ABAWAJY; BUYYA,). Entretanto, tais abordagens não analisam o consumo adicional gasto pela infraestrutura de resfriamento devido a consolidação das máquinas virtuais nos servidores em uma determinada área. Visando atender a esse objetivos, Fang et al. (2013) propõem o *VMPlanner*, um mecanismo de alocação de máquinas virtuais com o objetivo de otimizar a localização das máquinas virtuais e do fluxo de roteamento dos elementos de rede de modo a colocar no modo sleep os elementos de rede desnecessários o quanto possível para poupar energia. Em outro trabalho, os autores possuem o objetivo de diminuir a quantidade de elementos de rede utilizados, através de técnicas de agregação de tráfego e alocação de máquinas virtuais, e desligar os que não são utilizados com o intuito de economia de energia.

- Consumo da infraestrutura de refrigeração - Em um ambiente de um *data center*, parte da energia consumida pela infraestrutura computacional é convertida em calor, o que, se não for tratada, pode reduzir a confiabilidade e a durabilidade dos dispositivos. Desse modo, estudos visando abordar esse problema em um ambiente de computação nas nuvens tentam efetuar uma alocação das máquinas virtuais de modo que o calor seja tratado utilizando a menor quantidade de energia possível. Ainda há pouco estudo da influência da localização das máquinas virtuais na taxa de utilização da infraestrutura de resfriamento devido a pouca quantidade de simuladores existentes, a complexidade inerente para a obtenção de todos os dados necessários e além do *hardware* específico neceessário para controlar as unidades de resfriamento para regular o funcionamento. Novos desafios incluem como e quando realocar VMs para minimizar a energia consumida pelo sistema de resfriamento, enquanto preservando uma temperatura segura dos recursos e minimizando o overhead de migração e degradação de perfomance (MOORE et al.,). Um outro modo para obter economia com a infraestrutura de resfriamento é baseada na localização otimizada de cargas de trabalho a fim de amenizar os hotspots locais (FAKHIM et al., 2011; MEIJER, 2010; WANG et al., 2009).

- Consumo dos servidores: Em um ambiente de computação nas nuvens, uma grande quantidade de servidores são utilizados para armazenar os serviços oferecidos por esse novo paradigma. Entretanto, em qual servidor alocar uma determinada máquina virtual depende do objetivo. Visando atender o objetivo da computação verde, um mecanismo

de alocação considerando o consumo de energia busca alocar as máquinas virtuais nos servidores com o objetivo de utilizar a menor quantidade de energia. Além disso, a consolidação dos servidores pode ser estática ou dinâmicas (BOBROFF; KOCHUT; BEATY, 2007). Na consolidação estática, é utilizado o histórico de utilização dos recursos para realizar o mapeamento, sendo a primeira fase realizada pelos algoritmos de alocação. Após algum período de tempo, esse mapeamento pode não refletir a melhor solução, sendo necessária a realização de um novo mapeamento. Devido a esse fato, é necessário realizar um mapeamento de tempos em tempos para se adequar as mudanças nas configurações desde o último mapeamento realizado, sendo esse mecanismo de alocação denominada de alocação dinâmica.

Esse trabalho foca em um mecanismo de alocação considerando o consumo de energia dos servidores de acordo com a taxa de utilização baseada na alocação estática. Desse modo, a seguir serão descritas algumas técnicas utilizadas para a alocação de máquinas virtuais baseado no consumo de energia de cada servidor.

3.2 Políticas simples de alocação

Com a finalidade de medir o desempenho dos mecanismos de alocação, Jansen e Brenner (2011) implementaram e testaram um conjunto de políticas de alocação em um ambiente real em uma nuvem federada, ou seja, uma nuvem composta por *data centers* em localizações geográficas distintas. Para isso, foram utilizadas duas ferramentas de código aberto para virtualização de *data centers* em computação nas nuvens: OpenNebula (OPENNEBULA, 2013) e Eucalyptus (EUCALYPTUS, 2013). A aplicação escolhida para executar o teste foi a da rede social de código aberto *Reddit.com* (REDDIT, 2013).

Jansen e Brenner (2011) não especificaram no artigo os algoritmos em si, mas as políticas utilizadas no processo de alocação. Desse modo, o objetivo foi detalhar o desempenho das políticas com relação a diversos critérios, sendo um deles o consumo de energia. Além disso, nenhum embasamento matemático foi considerado no desenvolvimento das abordagens de avaliação das métricas utilizadas.

A única política que aborda o consumo de energia é a denominada *Watts* por núcleo. Nessa política, inicialmente é avaliado se o servidor possui recursos suficientes para alocar a máquina virtual, e, caso tenha, é medido o aumento no consumo de energia por núcleo de CPU depois da alocação. Esse procedimento é aplicado em todos os servidores para determinar qual teve o menor incremento no consumo. Assim, o único fator considerado é o consumo dos núcleos de CPU, não possuindo uma visão mais geral fornecida quando se utiliza o consumo por

servidor, ou seja, não analisa um servidor contendo diversas máquinas virtuais como um todo para o cálculo do consumo de energia, mas sim como diversas partes compondo esse todo. As outras políticas abordadas são de uso específico, utilizadas para analisar outros critérios como o custo por CPU e balanceamento de carga.

Uma característica observada é a medição de consumo de energia de servidores não ociosos. Para isso, são utilizados o consumo mínimo e máximo do servidor (P_{min} e P_{max}), e a quantidade de núcleos utilizada e a total ($Core_{alloc}$ e $Core_{total}$). A fórmula utilizada, também utilizada por Beloglazov, Abawajy e Buyya (2012) , foi:

$$P = P_{min} + (P_{max} - P_{min}) * (Core_{alloc}/Core_{total}) \tag{3.1}$$

Os resultados obtidos mostram a política considerando o consumo de energia obtendo economia de energia entre 7 a 14 %.

Divergindo dos autores dessa abordagem, este trabalho utilizará o consumo de energia de um servidor como um todo, de modo a ter uma abstração maior por trabalhar com consumo de servidores ao invés do consumo por núcleo de processamento em cada servidor. Além disso, será feita a modelagem matemática do problema de alocação como um problema de múltiplas mochilas, bem como a aplicação do algoritmo de colônia de formigas para resolver esse problema. Para a obtenção de melhores resultados, uma maior quantidade de fatores e métricas será utilizada.

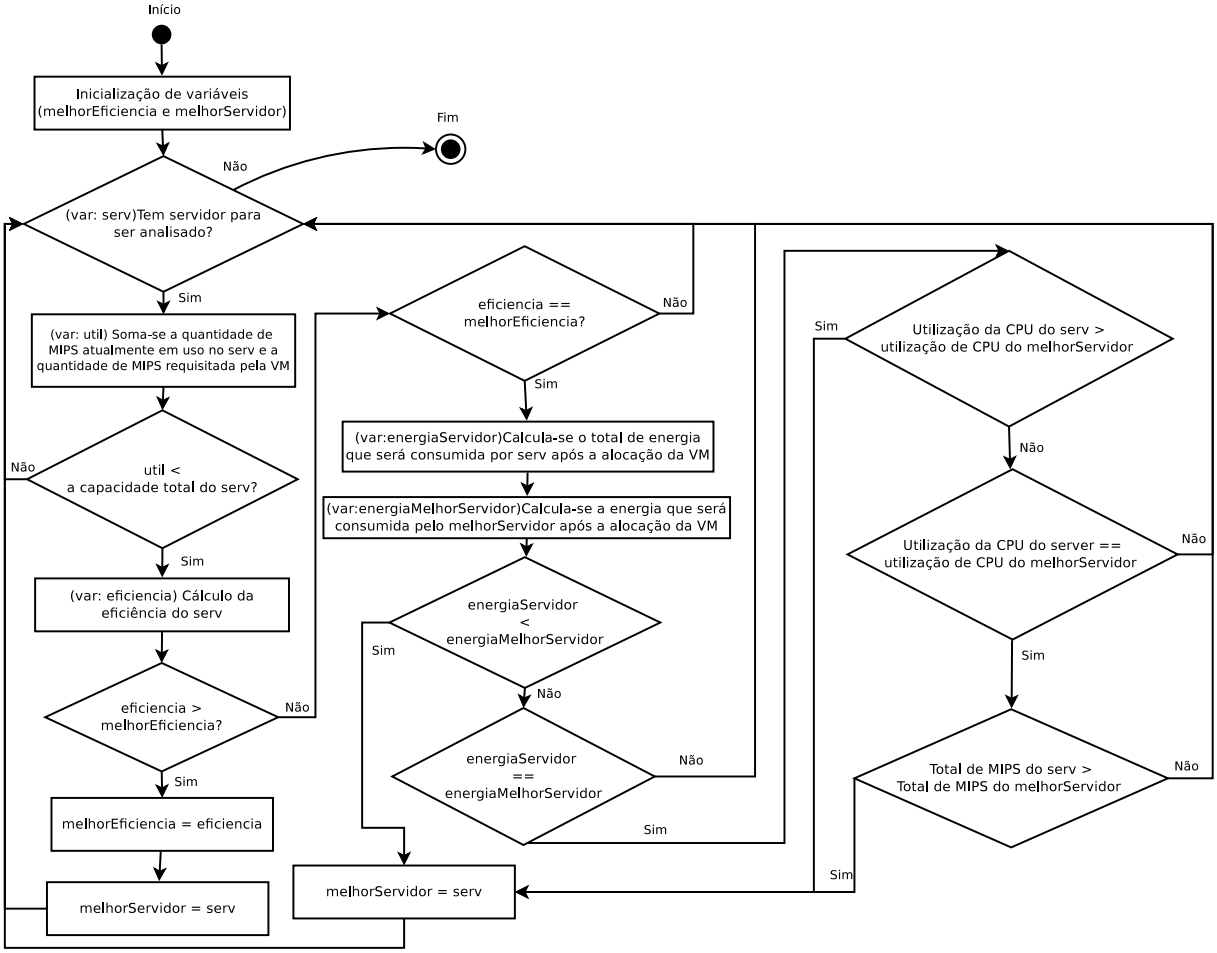
3.3 Lago Allocator

Lago, Madeira e Bittencourt (2011) propõem um mecanismo de alocação de busca de um servidor com a maior eficiência energética, considerando-se a quantidade de MIPS por energia consumida. Para fins de economia de energia, são utilizados dois critérios: a utilização de CPU e a temperatura do computador (através do mecanismo *fan control*, onde são feitos ajustes no funcionamento dos *coolers* de acordo com a temperatura). A alocação baseia-se no valor de energia consumida após a alocação dos recursos feita através de previsão. Desse modo, a solução proposta leva em conta apenas a utilização de CPU e a regulagem da voltagem aplicada aos *coolers* dos servidores para a economia de energia.

Como explicado anteriormente, o algoritmo *Lago Allocator* é utilizado para realizar a alocação de máquinas virtuais à servidores com o objetivo de obter um menor consumo ener-

gético. Para isso, os autores propuseram um algoritmo de alocação seguindo o fluxograma mostrado na figura 14, baseada na arquitetura do simulador Cloudsim (CALHEIROS et al.,). O procedimento *findHostForVM(vm)*, descrito na figura, é aplicado a toda máquina virtual a ser alocada em algum servidor. Ou seja, o algoritmo define em qual servidor deve alocar a máquina virtual passada como parâmetro.

Figura 14: Procedimento *findHostForVm*(vm) do algoritmo *Lago Allocator*



O algoritmo funciona do seguinte modo:

1. Para uma dada máquina virtual (vm1) passada como parâmetro, é realizada uma varredura em todos os servidores para identificar aqueles detentores de recursos suficientes para atender a requisição. A condição utilizada, no caso do algoritmo, é a quantidade de MIPS do servidor. Ou seja, para um servidor ser capaz de receber a vm1, ele deve possuir uma quantidade maior ou igual a quantidade de MIPS requisitada pela máquina virtual a ser alocada.

2. Verificada essa condição, é realizado o cálculo da eficiência energética do servidor, de modo a identificar o servidor com o melhor desempenho energético. A eficiência é cal-

culada pela divisão da quantidade máxima de MIPS do servidor pela quantidade máxima de energia consumida por esse servidor.

3. Pode ocorrer o caso onde dois ou mais servidores possuam o mesmo desempenho energético. Quando isso ocorrer, o consumo de energia do servidor é avaliado, de modo a escolher o de menor consumo de energia.

4. Pode ocorrer empate. Nesse caso, é analisado o percentual de utilização de CPU para escolher o servidor com maior percentual de utilização. O motivo da escolha de um servidor com essa característica se deve à necessidade de diminuir o consumo de energia; portanto, objetiva-se concentrar as máquinas virtuais no menor número possível de servidores.

5. Ainda assim, pode ocorrer empate. Como último critério de desempate, o total de MIPS suportado pelo servidor é utilizado. O servidor com maior capacidade de MIPS será escolhido.

Inicialmente, na simulação, é criado um conjunto de máquinas virtuais para serem alocadas nos servidores. Após todas as máquinas virtuais terem sido alocadas utilizando o procedimento descrito, é feita a submissão da carga de trabalho sobre a alocação e a coleta dos resultados.

Como as máquinas virtuais são submetidas ao algoritmo de alocação em uma ordem aleatória, pode ocorrer um aumento no consumo de energia. Suponha um cenário onde há uma lista com três máquinas virtuais *vm1*, *vm2* e *vm3* requisitando 200, 100, e 400 MIPS, respectivamente. Elas devem ser alocadas em uma infraestrutura composta por 2 servidores *serv1* e *serv2*, ambos com 400 MIPS de processamento. Suponha que a eficiência energética do *serv1* seja maior do que a do *serv2*. A ordem de submissão das máquinas virtuais ao procedimento de alocação segue a sequência numérica. Ao final da alocação, o servidor *serv1* terá alocado as máquinas virtuais *vm1* e *vm2* e o servidor *serv2* terá a *vm3*. Nesse caso, percebe-se o servidor menos eficiente com capacidade total de processamento, enquanto o servidor de melhor desempenho não está. Consequentemente, isso ocasiona um maior consumo de energia. Esse cenário pode ser visto nas figuras 15 e 16.

Outra desvantagem encontrada se dá quando é calculada a energia adicional consumida pelos servidores, após a alocação de uma máquina virtual. Na comparação realizada pelo algoritmo, é feita uma análise do impacto sobre os servidores, não sendo realizado uma análise mais global. Por exemplo, suponha o cenário onde deseja-se alocar uma máquina virtual em um servidor. Suponha, também, que a alocação da máquina virtual vm1 no servidor ocasionará um aumento de energia em 30 Watts. Há dois servidores disponíveis, serv1 e serv2, estando em serv1 alocada uma outra máquina virtual, consumindo 220 Watts, e serv2 no modo *sleep*, consumindo apenas 10W. Assume-se que quando o servidor passa do modo *sleep* para o de

Figura 15: Problema no mecanismo de alocação *Lago Allocator* - antes da alocação

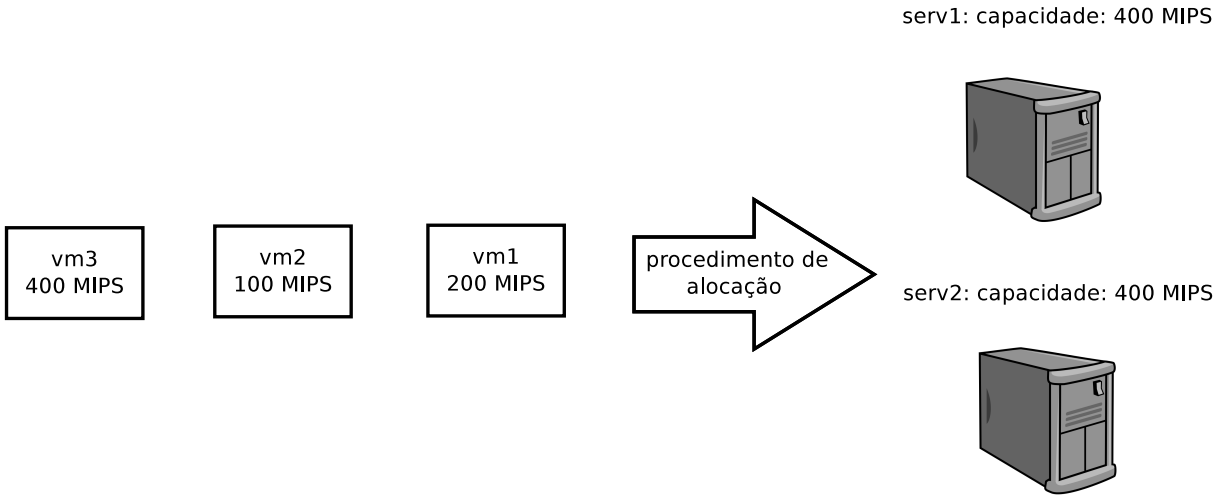
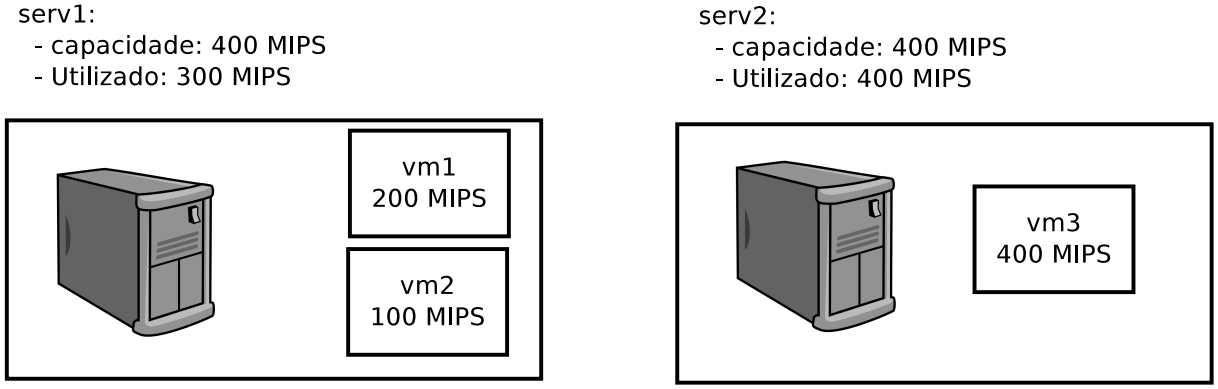


Figura 16: Problema no mecanismo de alocação *Lago Allocator* - após da alocação

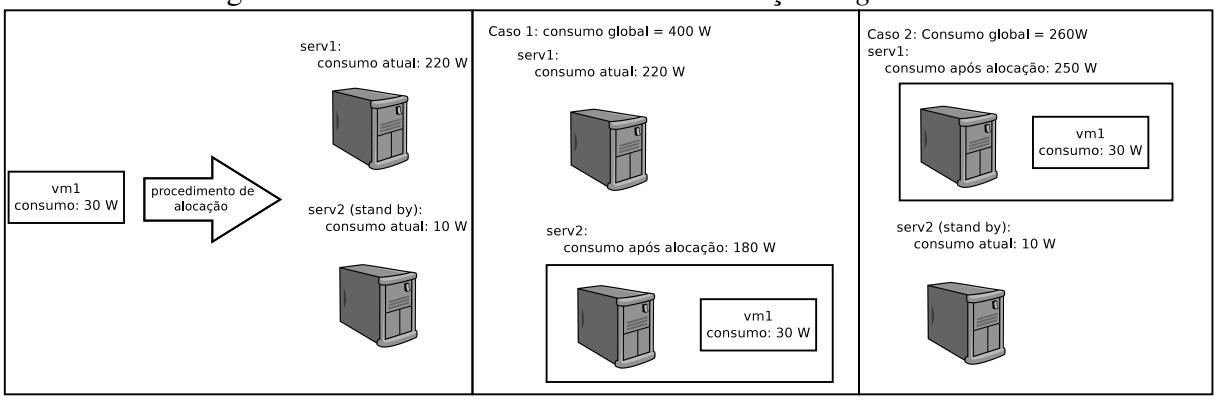


operação normal, passa-se a consumir 150 Watts. Seguindo o algoritmo, a vm1 será alocada no serv 2, pois comparando a energia consumida após a alocação, o menor será em serv2. Ou seja, o servidor serv1 consumiria 250 W, enquanto em serv2 teria o consumo de 180W, sendo, desse modo, o serv2 escolhido para receber a máquina virtual. Mas analisando o consumo global de energia, alocando em serv1 (serv1 + serv2 = [220W + 30W] + 10W = 260W) gastará uma quantidade menor de energia se comparado com a alocação em serv2 (220W + [150W + 30W] = 400W). Esse comportamento pode ser observado, respectivamente, nos caso 1 e 2 da figura 17.

3.4 *Modified Best Fit Decreasing*

O problema de alocação de máquinas virtuais considerando o consumo de energia é abordado por Beloglazov, Abawajy e Buyya (2012). Ele é dividido em duas partes: escolha do

Figura 17: Problema no mecanismo de alocação *Lago Allocator*



servidor que irá receber a máquina virtual, e otimização da localização das máquinas virtuais alocadas. Para abordar a primeira parte, é feita a modelagem do problema como um problema do empacotamento (*bin packing problem*). Nesse problema, tem-se um conjunto de objetos os quais devem ser alocados em um conjunto de caixas. A solução tem como objetivo a utilização da menor quantidade de caixas possíveis. No caso de computação nas nuvens, as caixas são os servidores e os objetos a serem colocados nas caixas são as máquinas virtuais.

A economia obtida ao fazer a modelagem como um problema de empacotamento é devido ao fato da utilização de uma quantidade menor de servidores a serem utilizados para atender as requisições por máquinas virtuais. Como são utilizados uma quantidade menor de servidores, os que permanecerem ociosos podem ser colocados no modo *sleep*, de modo a consumir uma quantidade menor de energia.

Como solução para o problema, utilizou-se o algoritmo *Modified Best Fit Decreasing* (baseado no *Best Fit Decreasing* (MARTELLO; TOTH, 1990)). Nesse algoritmo, a lista de máquinas virtuais a serem alocadas são ordenadas de maneira decrescente de acordo com a taxa de utilização de CPU. Em seguida, cada máquina virtual é alocada no servidor no qual haverá o menor aumento de consumo de energia após a alocação.

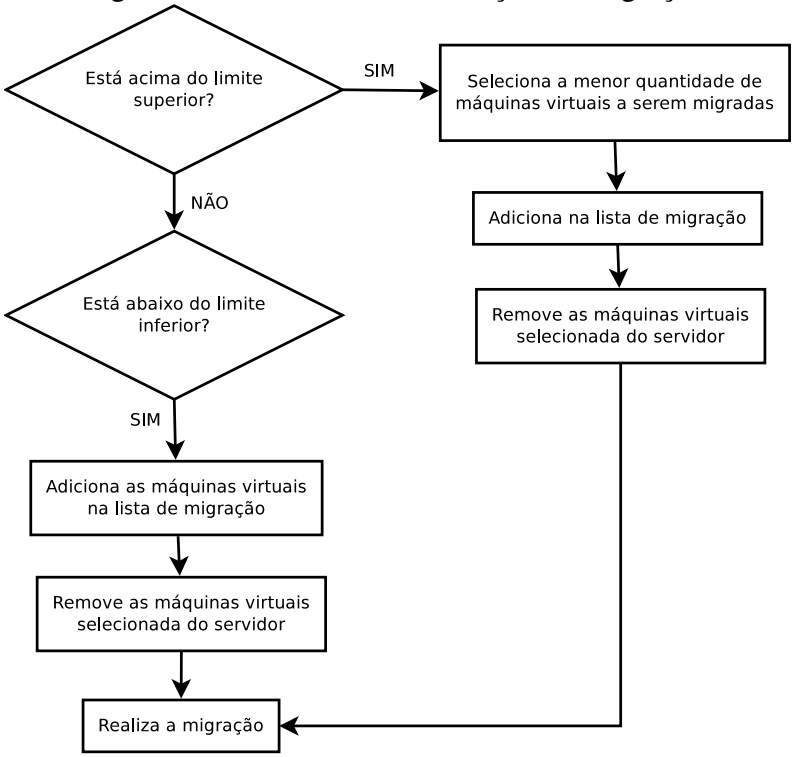
Além do mecanismo de alocação, políticas de migração de máquinas virtuais são propostas. Nessas políticas, são estabelecidos o limite inferior e o superior de utilização dos recursos, onde a taxa de utilização dos servidores deve estar entre esses limites. Há dois casos que podem ocorrer:

- Taxa de utilização de CPU do servidor está abaixo do limite inferior: no caso, as máquinas virtuais presentes nesse servidor são migradas, e o servidor é posto no modo *sleep*, reduzindo o consumo de energia.
- Taxa de utilização de CPU do servidor está acima do limite superior: seleciona-se um conjunto de máquinas virtuais para serem migradas, de modo a diminuir a taxa de utili-

zação do servidor para ficar entre os limites. Com isso, as máquinas virtuais presentes no servidor podem ter mais recursos disponíveis, sem a necessidade de migrar para outro servidor.

A política de migração que apresentou o melhor desempenho foi a de Minimização de Migrações. Para cada servidor no *data center*, o procedimento da Figura 18 é executado, ilustrando o comportamento dessa política.

Figura 18: Política de Minimização de Migrações



Na política de minimização de migrações, é feita constantemente uma avaliação sobre o estado de utilização dos servidores no *data center* com relação ao consumo de CPU. Quando algum servidor está acima do limite superior pré-estabelecido, é feita a seleção da menor quantidade de máquinas virtuais possível para serem migradas de modo que a taxa de utilização de CPU dos servidores fique abaixo do limite. É selecionada a menor quantidade de máquinas virtuais a serem migradas pelo fato da migração impactar na menor quantidade de usuários possível, evitando a violação no SLA do cliente.

No outro caso, onde a taxa de utilização de CPU está abaixo do limite inferior, as máquinas presentes nesse servidor são migradas para outro servidor, possibilitando ao servidor com baixa utilização entrar no modo *sleep*. Com isso, é possível realizar a concentração das máquinas virtuais em uma quantidade menor de servidores, e os servidores não utilizados são postos no modo *sleep*, possibilitando, assim, um consumo menor de energia.

Apesar de apresentar bons resultados para os cenários testados, essa abordagem não di-

ferencia os servidores com diferentes capacidades de processamentos e diferentes consumos de energia. A quantidade de processamento realizado por uma máquina dividida pelo consumo de energia para a realização desse processamento é denominada eficiência energética. Desse modo, em um cenário heterogêneo, onde as máquinas possuem diferentes eficiências energéticas, o algoritmo pode consolidar as máquinas virtuais em um servidor onde a eficiência energética é baixa, ocasionando um consumo maior de energia do que o esperado.

Visando atender os mais variados cenários, nessa dissertação a eficiência energética de cada servidor será levada em conta, possibilitando obter uma alocação mais eficiente quando considerando o consumo de energia. Dessa forma, os testes serão realizados em *data centers* homogêneos, onde todos os servidores possuem a mesma eficiência energética, e em *data centers* heterogêneos, onde os servidores possuem diferentes valores para eficiência energética.

3.5 *Energy-aware Epidemic Virtual Machine Consolidation Protocol*

Visando abordar o problema de alocação de máquinas virtuais em um *data center*, Esnault (2012) propôs um protocolo para gerenciar a consolidação de máquinas virtuais nos servidores. Devido a questão de escalabilidade, o protocolo proposto é descentralizado, utilizando uma rede *peer-to-peer* para comunicação entre os servidores. Um ponto importante a se observar é a aplicabilidade do protocolo, onde é utilizado apenas depois da alocação das máquinas virtuais nos servidores. Assim, sua finalidade é realizar a otimização de um estado de alocação das máquinas virtuais no *data center*. Na solução proposta, os autores preocupam-se com a influência do número de migrações a serem realizadas, podendo ocasionar uma sobrecarga na infraestrutura de rede.

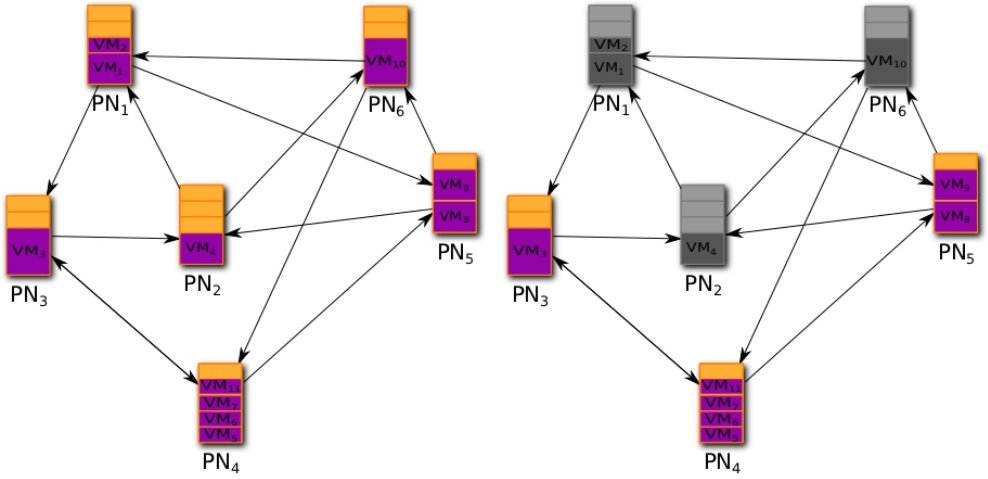
O problema de consolidação de máquinas virtuais é descrito como um conjunto de máquinas virtuais e um conjunto de servidores onde deseja-se realizar o melhor mapeamento possível das máquinas virtuais com os servidores, respeitando as restrições impostas. Essa otimização das máquinas virtuais é modelada como um problema do empacotamento multidimensional (SHAW, 2004). Esse problema é descrito como um conjunto de objetos que possuem diferentes volumes que devem ser empacotados em uma quantidade mínima de caixas, com diferentes capacidades, sem exceder a capacidade total da caixa. Como solução para esse problema, utiliza-se uma abordagem distribuída do algoritmo de colônia de formigas, diferentemente das aplicadas nas técnicas tradicionais centralizadas.

Antes da aplicação do protocolo, é executado um protocolo *peer-to-peer* para estabelecer a vizinhança de cada nó da rede. Após de ser feito o mapeamento da vizinhança, executa-se sobre cada nó o algoritmo de colônia de formigas somente para o nó e a sua vizinhança. A finalidade no uso desse algoritmo é consolidar as máquinas virtuais na menor quantidade de

servidores possível. Após a aplicação do algoritmo em cada nó, espera-se obter uma infraestrutura consolidada e os servidores não utilizados são desligados.

O comportamento do protocolo pode ser observado na Figura 19. Nesse exemplo, o servidor PN_4 foi escolhido para executar o algoritmo de colônia de formigas sobre a sua vizinhança, PN_3 e PN_5. Ao fim do processo, as máquinas virtuais são realocadas para a menor quantidade de servidores. Esse procedimento é aplicado para todos os servidores da rede, ocasionando uma certa sobrecarga de processamento.

Figura 19: Exemplo de aplicação do protocolo de consolidação de máquinas virtuais



Fonte: Esnault (2012).

Pela descrição apresentada, a formulação do problema como problema do empacotamento multidimensional é similar ao problema de múltiplas mochilas, com exceção do objetivo, que no problema do empacotamento é a utilização da menor quantidade possível de caixas para armazenar uma determinada quantidade de objetos.

Como desvantagem, tem-se que o protocolo proposto não trata de questões relativas a primeira alocação das máquinas virtuais, sendo, portanto, um *daemon* que verifica quais as VMs que devem ser migradas. Uma possível melhoria a ser realizada seria a utilização de algoritmo de alocação inicial das máquinas virtuais, podendo obter resultados melhores.

3.6 Resumo e comparação entre os trabalhos relacionados

As características gerais das abordagens utilizadas nos trabalhos relacionados são descritas na Tabela 1.

Relacionadas com as características descritas anteriormente e contidas na tabela, este trabalho:

Tabela 1: Comparativos dos trabalhos relacionados

Abordagem	**Modelagem matemática**	**Distribuição**	**Algoritmo**
Política simples	não possui	Centralizado	-
MBFD	Problema do empacotamento	Centralizado	*MBFD*
Lago Allocator	Problema do empacotamento	Centralizado	-
VM Consolidation Protocol	Problema do empacotamento multidimensional	Distribuído	Algoritmo de colônia de formigas

- Modelagem matemática: será utilizada umas das variantes do problema do empacotamento denominada de problemas de múltiplas mochilas, onde tenta-se alocar os objetos nas diversas mochilas disponíveis visando obter o maior retorno. A escolha dessa modelagem foi devido a semelhança do problema abordado (minimização do consumo de energia e consolidação das máquinas virtuais) com essa modelagem, possibilitando a utilização de técnicas já desenvolvidas em problemas de múltiplas mochilas no problema de alocação de máquinas virtuais.

- Distribuição: será utilizada uma abordagem centralizada, onde um servidor no *data center* ficará responsável por realizar a alocação das máquinas virtuais, bem como de saber o estado de utilização de cada servidor no *data center*. Tal fato é necessário para prover os dados necessários para a execução do algoritmo de colônia de formigas.

- Algoritmo: o algoritmo de colônia de formigas foi escolhido devido a sua aplicação no trabalho desenvolvido por Esnault (2012), onde foram obtidos bons resultados, apesar do foco ter sido somente a consolidação das máquinas virtuais no *data center*. Além disso, outro estudo comparando diversas heurística avaliou o algoritmo de colônia de formigas como apresentando o melhor desempenho (BOZDOGAN; YILMAZ; EFE, 2010). Devido a esses fatores, o algoritmo de colônia de formigas será utilizado com parâmetros avaliando o consumo de energia dos servidores e a consolidação das máquinas virtuais, sendo implementado como uma abordagem centralizada.

4 ALGORITMO PROPOSTO

Este capítulo apresenta a proposta do mecanismo de alocação de máquinas virtuais considerando o consumo de energia. Com a modelagem do problema de múltiplas mochilas, foi utilizado a metaheurística de colônia de formigas para gerar a solução. A seguir, são apresentados, primeiramente, os conceitos relativos a metaheurística para depois ser apresentada a solução utilizada.

4.1 Modelagem do problema de alocação de máquinas virtuais

O problema abordado nesse trabalho é a elaboração de um mecanismo de alocação de máquinas virtuais em um *data center* em computação nas nuvens considerando o consumo de energia. O objetivo é minimizar a quantidade de energia utilizada pelos servidores para atender a demanda das máquinas virtuais a serem inseridas. Esse problema pode ser modelado matematicamente como um problema de múltiplas mochilas. O objetivo em se fazer essa modelagem é utilizar soluções já existentes para o problema de múltiplas mochilas na solução do problema abordado.

Desse modo, pode-se fazer a seguinte equivalência entre os parâmetros utilizados nesses problemas:

- As mochilas são os servidores, representando a quantidade de recursos disponíveis.
- As máquinas virtuais são os itens a serem inseridos nas mochilas (servidores).
- O peso é o conjunto de requisitos computacionais da máquina virtual (processador, memória, largura de banda).
- O lucro é o consumo de energia gerado pela alocação de uma máquina virtual em um servidor.

Como o objetivo do mecanismo de alocação é minimizar o consumo de energia, foi realizada uma adequação na função objetivo do problema de múltiplas mochilas: o problema passou a ser de minimização. Por questões de nomenclatura, a propriedade do item que representava o lucro foi renomeada para consumo.

De maneira formal, o problema é definido do seguinte modo: tem-se um conjunto com n máquinas virtuais para serem alocadas em m servidores, onde cada máquina virtual j (j=1,$\cdots$,n) possui um vetor peso w_j composto pela quantidade de processamento p_j, memória s_j e largura de banda b_j, e um consumo de energia d_j associado ao consumo da máquina virtual

utilizando esses recursos. Cada servidor possui um vetor de limite de recursos c_i (composto pela quantidade de processamento p_max_i, memória s_max_i e largura de banda b_max_i), informando a quantidade máxima de recursos físicos do servidor. A variável x_{ij} é utilizada para especificar se uma máquina virtual j foi alocada no servidor i. Se isso acontecer, a variável x_{ij} será preenchida com valor 1; caso contrário, será configurado com valor 0. O objetivo desse problema é minimizar o consumo de energia gerado pela alocação das máquinas virtuais nos servidores. Desse modo, o problema é formulado matematicamente da seguinte forma:

$$Minimizar\ z\ =\ \Sigma_{i=1}^{m}\ d_i, \tag{4.1}$$

$$sujeito\ a$$

$$\Sigma_{j=1}^{n}\ w_j x_{ij}\ \leq\ c_i, \qquad i \in M\ =\ \{1,\ \cdots,\ m\}, \tag{4.2}$$

$$\Sigma_{i=1}^{m}\ x_{ij}\ \leq\ 1, \qquad j \in N\ =\ \{1,\ \cdots,\ n\}, \tag{4.3}$$

$$w_j \leq max_{i \in M}\{c_i\}, \qquad para\ j \in N, \tag{4.4}$$

$$c_i \geq min_{j \in N}\{w_j\}, \qquad para\ i \in M, \tag{4.5}$$

$$w_j,\ c_i\ =>\ inteiros\ positivos, \tag{4.6}$$

$$x_{ij}\ =\ 0\ ou\ 1, \qquad i \in M,\ j\ \in\ N \tag{4.7}$$

Dada a formulação, tem-se:

- A equação 4.1 representa a função objetivo. No caso, tenta-se minimizar a quantidade de energia consumida pela alocação das máquinas virtuais nos servidores.
- A restrição 4.2 limita a quantidade de recursos de processamento, memória e largura de banda das máquinas virtuais alocadas em um servidor e não podem superar os seus recursos físicos.
- A restrição 4.3 determina que uma máquina virtual só pode ser alocada em um único servidor;
- As restrições 4.4 e 4.5 são utilizadas para formatação do problema. No caso, essas regras evitam a existência de uma máquina virtual com uma capacidade superior à máxima oferecida pelos servidores (4.4), ou a existência de um servidor com quantidade de recurso menor do que o mínimo exigido pela máquina virtual utilizando a menor quantidade de recursos (4.5). Em ambos os casos, caso ocorram, não seria possível alocar algumas máquinas virtuais nos servidores.

- A restrição 4.6 limita o problema para o uso somente de valores inteiros positivos.
- Por fim, a restrição 4.7 limita os valores 0 ou 1 os quais a variável x_{ij} pode assumir.

Formulado o problema, faz-se necessário definir qual é a melhor forma para resolver o problema de alocação de máquinas virtuais nos servidores considerando energia. Na seção seguinte será abordada o algoritmo de colônia de formigas utilizado para solucionar esse problema.

4.2 Algoritmo proposto

A maioria dos algoritmos de colônia de formigas possui o mesmo formato, sendo diferenciados pelas fórmulas referentes a probabilidade de seleção, informação heurística e modelo de feromônio utilizados. Desse modo, as subseções a seguir definem como essas informações serão calculadas e, ao final, mostra-se o algoritmo final para ser aplicado ao problema de múltiplas mochilas.

4.2.1 Informação heurística

Tendo como objetivo a economia de energia, busca-se um modo de consolidar as máquinas virtuais no menor número de servidores possível. Ao se considerar esse objetivo, podem ser observados dois fatos. O primeiro está relacionado com o cenário onde a consolidação é completa, ou seja, a alocação de máquinas virtuais no servidor é feita até que não haja mais recursos disponíveis. O problema dessa abordagem encontra-se no fato das máquinas virtuais executarem aplicações com carga de trabalho dinâmica. Desse modo, com o aumento na demanda por recursos realizada pela grande quantidade de acessos à aplicação, faz-se necessário realizar uma realocação, de modo a fornecer mais recursos para as máquinas virtuais. Mas ela não pode ser realizada em um servidor sem recursos disponíveis por causa da alta taxa de consolidação, sendo necessário migrar a aplicação para outro servidor com mais recursos disponíveis. Desse modo, um *data center* com uma política de sempre ter obter uma alta taxa de consolidação irá possuir um grande número de migrações de máquinas virtuais, podendo degradar o desempenho da infraestrutura de rede. No outro extremo, tem-se o cenário onde há um servidor hospedando uma máquina virtual de baixo consumo energético. Nesse caso, seria interessante migrar essa máquina para alguma outra em utilização, liberando o servidor para ser colocado no modo *sleep*, consumindo, portanto, uma quantidade menor de energia.

Para tratar dessa questão da taxa de consolidação das máquinas virtuais aplicadas aos servidores, é utilizada a informação heurística do algoritmo de colônia de formigas. Dessa forma, utiliza-se a equação 4.8 para a informação heurística:

$$\eta_{i,j} = \frac{(C_i - (b_i + r_j))}{C_i} \tag{4.8}$$

Para um servidor i e uma máquina virtual j, utiliza-se para o cálculo a capacidade do servidor (C_i), a quantidade de recursos utilizados no servidor (b_i) e a quantidade de recursos requisitada pela máquina virtual (r_j). A equação 4.8 penaliza os servidores com a taxa de utilização dos recursos alta, de modo a obter um valor de informação heurística próximo de zero. Já quando a taxa de utilização é baixa, a informação heurística estará próxima de 1.

4.2.2 MODELO DE FEROMÔNIO

No algoritmo, um parâmetro utilizado para informar o quão bom (menor consumo de energia) é um mapeamento de uma máquina virtual em um determinado servidor, baseado no histórico de alocações, é denominado de feromônio. Tal informação é levada em conta no momento do cálculo da probabilidade de seleção, onde é calculada a probabilidade de uma determinada máquina virtual ser alocada em um determinado servidor.

No caso do feromônio, é necessário analisar dois processos: o de depósito e o de evaporação. O primeiro ocorre quando uma formiga encontra uma fonte de comida e volta para o formigueiro depositando feromônio no caminho para saber o caminho para a fonte de comida novamente. No caso do algoritmo ACO, o depósito serve como uma valorização da melhor solução encontrada durante uma rodada do algoritmo através do incremento da quantidade de feromônio. O segundo caso trata da evaporação do feromônio, processo onde as soluções que não são tão boas passam a ter uma probabilidade reduzida de serem escolhidas na rodada seguinte do algoritmo. A atualização dos valores de feromônio acontece logo após a finalização de um ciclo de busca da solução.

Fazendo o mapeamento para o contexto de alocação de máquinas virtuais, o processo de depósito de feromônio é a etapa onde se analisa o consumo de energia da alocação de uma máquina virtual em um servidor. Quando uma determinado mapeamento de uma máquina virtual em um servidor é inserido na solução, a quantidade de feromônio é atualizada, aumentando a sua taxa de acordo com o consumo de energia da solução. Já no processo de evaporação, o valor do feromônio dos mapeamentos de máquinas virtuais em servidores não escolhidos são penalizadas por não terem o consumo tão baixo como os mapeamentos presentes na solução, decrementando as taxas de feromônio de acordo com uma taxa de evaporação. Desse modo, boas soluções, ou seja, mapeamentos de máquinas virtuais em servidores com o menor consumo de energia, possuem taxa de feromônio maiores ocasionando uma maior probabilidade de serem inseridas em soluções futuras geradas pelo algoritmo.

Para determinar quais formigas influenciarão no processo de atualização de feromônio, foi utilizada o Sistema de colônia de formiga MAX-MIN (DORIGO; BIRATTARI; STüTZLE, 2006; STüTZLE; HOOS, 1996), onde são estabelecidos limites inferior (τ_{min}) e superior (τ_{max}) para a taxa de feromônio. Assim, quando uma taxa de feromônio estiver acima de τ_{max}, a taxa de feromônio do vértice será configurada com o valor de τ_{max}. Analogamente, quando uma taxa de feromônio estiver abaixo de τ_{min}, a taxa de feromônio do vértice será configurada com o valor de τ_{min}. E qualquer valor de feromônio entre esses limites é mantido. Geralmente, esses limites são determinados empiricamente (DORIGO; BIRATTARI; STüTZLE,). Ao se utilizar esses limites, o espaço de busca de soluções será maior, diminuindo a possibilidade de ficar apenas em ótimos locais.

Para um servidor i e uma máquina virtual j, tem-se a equação (DORIGO; BIRATTARI; STüTZLE,):

$$\tau_{i,j} = [(1 - \rho) * \tau_{i,j} + \Delta\tau_{i,j}^{best}]_{\tau_{min}}^{\tau_{max}} \tag{4.9}$$

onde:

- $0 \leq \rho \leq 1$ representa a taxa de evaporação do feromônio adotada.
- Seja f(s) o valor da função objetivo do problema para uma solução s. Para o cálculo do depósito de feromônio sobre os elementos pertencentes a solução (s_{melhor}), utiliza-se o valor da função objetivo calculado sobre a melhor solução encontrada. Dessa forma, tem-se os seguintes valores para $\Delta\tau_{i,j}^{best}$:

$$\Delta\tau_{i,j}^{best} = \begin{cases} \frac{1}{f(s_{best})}, \ se \ (i,j) \in s_{melhor} \\ 0, \ caso \ contrário \end{cases} \tag{4.10}$$

4.2.3 Probabilidade de seleção

Para obter as soluções para o problema, utiliza-se um modelo estocástico visando obter uma diversidade nas soluções com o intuito de evitar a busca exaustiva para achar uma boa solução, onde uma boa solução é o mapeamento de uma máquina virtual em um servidor com o menor consumo de energia. A probabilidade de seleção é a mesma utilizada por Clemente (2010), Esnault (2012), Fidanova (2004), e Alaya, Solnon e Guédira (2004), podendo ser vista na equação 4.11.

$$p_{ij} = \frac{(\tau_{ij}^{\alpha}) * (\eta_{ij}^{\beta})}{\Sigma_{\forall y \in Cand}(\tau_{iy}^{\alpha}) * (\eta_{iy}^{\beta})} \tag{4.11}$$

$$0 < p_{ij} \leq 1; \tau_{ij}, \eta_{ij} > 0; \alpha, \beta \geq 0$$

onde:

- Feromônio (τ_{ij}): indica o quão boa é a alocação da máquina virtual j no servidor i baseado no histórico de alocações;
- Informação heurística (η_{ij}): representa o quanto um servidor está consolidado, onde servidores com uma maior quantidade de recursos disponíveis possuem valores de informação heurística maiores. Esse parâmetro é utilizado para prover uma melhor distribuição das máquinas no *data center*;
- α e β são utilizados para determinar quais dos parâmetros utilizados na probabilidade de seleção tem mais influência na geração da solução.
- Lista de máquinas virtuais candidatas (*Cand*): representa uma lista contendo as máquinas virtuais que podem ser alocadas no servidor de acordo com a quantidade de recursos disponíveis no servidor.

4.3 Algoritmo para o problema de alocação usando ACO

Para solucionar o problema de múltiplas mochilas, foi utilizado o algoritmo 30 baseado na metaheurística de otimização por colônia de formigas. No algoritmo, são utilizadas as seguintes variáveis:

- O conjunto de máquinas virtuais e de servidores são identificados por M = $\{1, \cdots, m\}$ e N = $\{1, \cdots, n\}$, respectivamente. Uma máquina virtual j requisita uma quantidade r_j recursos. Já para um servidor i, utilizam-se duas variáveis: uma com o conjunto de recursos que o servidor possui (C_i), e a outra usada para identificar a quantidade de recurso do servidor já utilizados (b_i).
- A variável num_{sol} diz a quantidade de formigas (soluções) que serão geradas em cada ciclo de execução do algoritmo.
- num_{iter} indica o número de iterações a serem realizadas para se encontrar a solução. Essa variável é utilizada como condição de parada para o algoritmo.

- O conjunto de feromônio é guardado como uma matriz Γ_{nxm}, pois o valor depende da máquina virtual e do servidor onde ela foi alocada. Para uma máquina virtual i alocada em um servidor j, tem-se o valor do feromônio $\tau_{i,j}$.
- O conjunto de informações heurísticas é guardado como uma matriz H_{nxm}, pois também depende da máquina virtual e do servidor onde ela foi alocada. Para uma máquina virtual i alocada em um servidor j, tem-se o valor da informação heurística $\eta_{i,j}$.
- s_{melhor} representa a melhor solução do algoritmo, entre as que foram geradas, de acordo com a função objetivo do problema.

Algoritmo 2: Algoritmo ACO para o problema de alocação de máquinas virtuais

Entrada: conjunto de Máquinas Virtuais VMS a serem alocados e vetor de capacidades dos servidores (C)
$\tau_{i,j} = \tau_{max}$
for *iter = 0 até* num_{iter} **do**
    s_{melhor} = NULL
    for *para i = 1,* $\cdots$*,* num_{sol} **do**
        $s_{parcial} \leftarrow \emptyset$
        V = VMS
        u = 0
        while $V \neq \emptyset$ **do**
            Cand = $\{j \in V \mid r_i \leq C_i - b_i\}$
            if $Cand \neq \emptyset$ **then**
                j = elemento retornado a partir da escolha estocástica sobre Cand
                $x_{u,j} = 1$
                $s_{parcial} = s_{parcial} + \{j\}$
                V = V - {j}
                $b_u = b_u + r_j$
            else
                u = u + 1
        if $f(s_{parcial}) < f(s_{melhor})$ *ou* $(s_{melhor} = NULL)$ **then**
            $s_{melhor} \leftarrow s_{parcial}$;
    for $i = 1$ *até* n **do**
        for $j = 1$ *até* m **do**
            $\tau_{i,j} = (1 - \rho) * \tau_{i,j}$
            if $x_{i,j} = 1$, *em* s_{melhor} **then**
                $\tau_{i,j} = \tau_{i,j} + \Delta\tau i, j$
            if $\tau_{i,j} < \tau_{min}$ **then**
                $\tau_{i,j} = \tau_{min}$
            if $\tau_{i,j} > \tau_{max}$ **then**
                $\tau_{i,j} = \tau_{max}$
Saída: A melhor solução até o momento, s_{melhor}

Para o funcionamento correto, o algoritmo 30 necessita que sejam passados como parâmetros o conjunto de máquinas virtuais a serem alocadas e o vetor de capacidades C dos servidores. Espera-se do algoritmo uma saída contendo a melhor alocação das máquinas virtuais considerando o consumo de energia.

O primeiro passo do algoritmo é realizar a inicialização da taxa de feromônio com τ_{max}, para todo (i,j) $\in$ N x M (linha 2). Na linha 4 inicia-se a execução do procedimento da colônia de formigas, sendo repetido num_{iter} vezes, sendo essa variável responsável pela condição de parada do algoritmo. A criação das formigas é realizada na linha 6, onde cada uma executa a sequência de passos para criar uma solução para o problema (linhas 7 a 20).

Assim, dado o conjunto de máquinas virtuais armazenadas em I, analisa-se quais desse conjunto podem ser alocadas no servidor corrente u (dependente das restrições dos recursos físicos que o servidor pode oferecer), armazenando as aptas em Cand (linha 11). A escolha da máquina virtual a ser alocada será feita por um mecanismo de seleção do tipo roleta.

No mecanismo de roleta (SOARES, 1997), o mesmo aplicado nos algoritmos genéticos, cada máquina virtual tem uma probabilidade de seleção. Dada essa probabilidade, soma-se as probabilidades de todos das máquinas virtuais presentes em Cand e, em seguida, calcula-se a porcentagem de cada máquina virtual. Com isso, monta-se uma rotela, similar as utilizadas em casinos, e utiliza-se a geração aleatória de um número entre 0 e 1 para decidir qual máquina virtual será selecionada.

Um exemplo do mecanismo de figura pode ser visto na figura 20. Nele, quatro máquinas virtuais fazem parte do conjunto de máquinas virtuais candidatas Cand, cada uma com uma determinada probabilidade de seleção calculada conforme descrito anteriormente (segunda coluna da tabela). É feita a soma das probabilidades de seleção das máquinas virtuais e calcula-se a porcentagem (terceira coluna da tabela). Com esses dados, monta-se a roleta conforme mostrada.

--	Probabilidade de Seleção	Porcentagem
VM 1	0.2	27 %
VM 2	0.1	13 %
VM 3	0.3	40 %
VM 4	0.15	20 %
TOTAL	0.75	100%

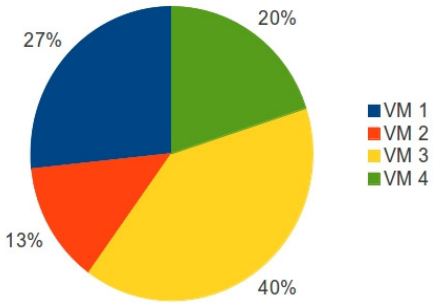


Figura 20: Exemplo de aplicação do mecanismo de roleta

Depois da escolha, a máquina virtual é inserida no conjunto solução (linhas 13 e 14), retira-se do conjunto de itens I (linha 15) e atualiza a taxa de recursos usada pelo servidor (linha 16).

Encontrada a solução, ela é comparada com a melhor anteriormente selecionada (linha 19), e, caso seja maior ou for a primeira formiga, atribui-se o valor da solução $s_{parcial}$ para s_{melhor} (linha 20). Caso contrário, repete todo o procedimento com uma nova formiga, até serem geradas $num_{formigas}$ formigas.

Finalizada a etapa de construção da solução, a etapa de atualização do feromônio é realizada sobre todos os pares (i, j) $\in$ N x M (linhas 21 a 29), onde observa-se:

- Primeiramente, calcula-se a taxa de evaporação do feromônio para todos os pares (i, j), seguindo a equação descrita em 2.13 (linha 23);
- Percorre-se a matriz para determinar quais máquinas virtuais foram selecionadas e em qual servidor elas devem ser alocadas. Sobre os pares (i, j) selecionados, aplica-se o processo de depósito de feromônio (linhas 24 e 25).
- Por fim, é realizada uma análise para verificar se algum dos pares (i, j) violou o limite inferior ou superior das taxas de feromônio. Caso os limites sejam violados por algum par (i, j), será atribuída a taxa de feromônio do valor do limite superior ou do inferior para esses pares, dependendo de qual limite foi violado (linhas 26 a 29).

5 ANÁLISE DE DESEMPENHO

Nesse trabalho será utilizada a ferramenta de modelagem e simulação para um ambiente em computação nas nuvens denominado de CloudSim (CALHEIROS et al.,). Os comportamentos dos diversos elementos integrantes de uma arquitetura em computação nas nuvens, como *data centers*, máquinas virtuais e políticas de provisionamento de recursos, são suportados pelo CloudSim. Por causa da similaridade dos ambientes de computação em grade (*Grid Computing*) e problemas adotados em computação nas nuvens, as ferramentas de simulação de computação em grade serviram de base para a implementação do CloudSim, de modo que ele possui diversos módulos oriundos dessas ferramentas.

A seguir serão descritos os cenários que serão utilizados. Os cenários e as métricas utilizadas na análise dos resultados são as mesmas utilizadas por Lago, Madeira e Bittencourt (2011).

5.1 Simulação

5.1.1 Cloudsim

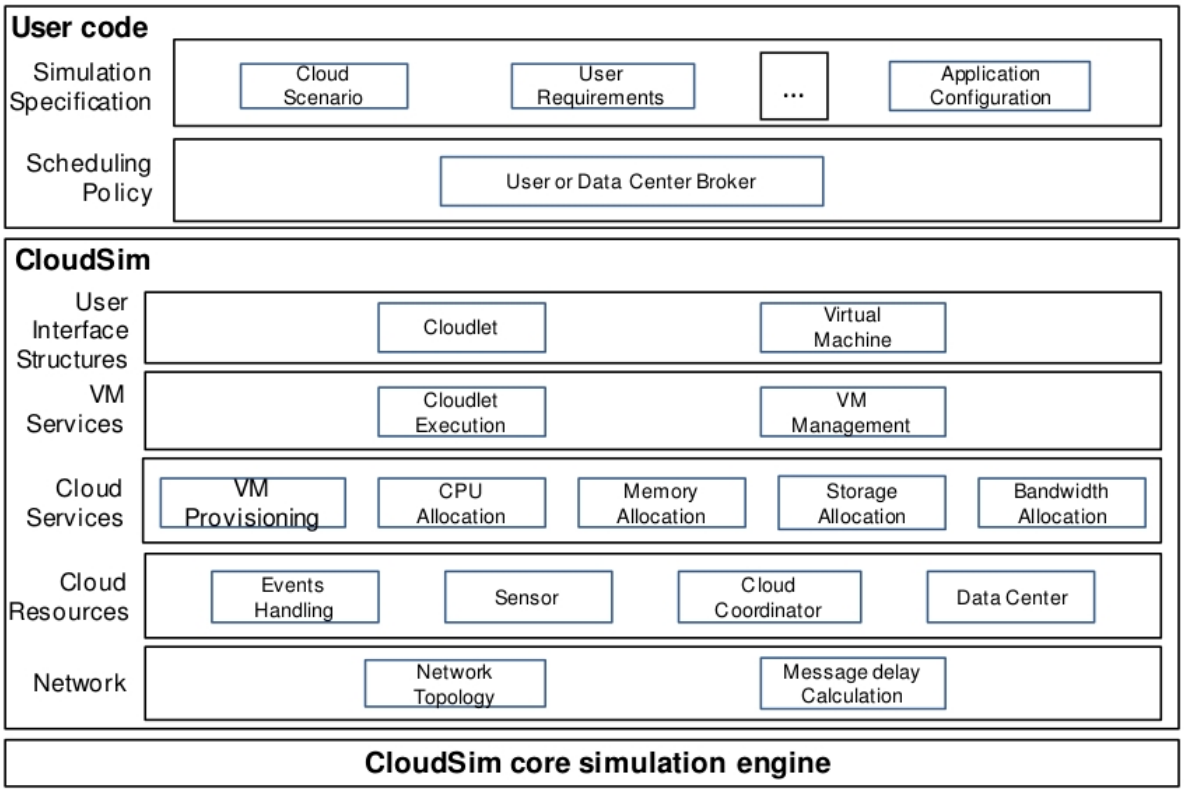
Por ser um ambiente dinâmico, os testes realizados sobre computação nas nuvens em ambientes reais dificilmente podem ser reproduzidos, pois o reteste deveria ser feito sobre a mesma configuração do teste original. Mesmo se não houvesse esse problema, pesquisadores teriam dificuldades para ter acesso a um ambiente de computação nas nuvens real para realizar testes em um ambiente de larga escala. Uma alternativa seria testar a solução em poucas máquinas e generalizar os resultados obtidos para serem aplicados em *data centers* maiores, não sendo uma alternativa viável, pois o comportamento em *data centers* maiores podem sofrer variações que um ambiente com poucas máquinas não sofreria. Optou-se pela simulação pela impossibilidade de testes em ambientes com uma grande quantidade de servidores.

O Cloudsim é um simulador de código aberto, escrito na linguagem Java, onde é possível modelar o comportamento dos componentes do sistema em computação nas nuvens tal como *data centers*, servidores, máquinas virtuais e políticas de alocação de recursos (BELOGLAZOV; ABAWAJY; BUYYA,). Devido as semelhanças entre computação em grade e computação nas nuvens, os simuladores para computação em grade (GridSim, GangSim) eram utilizados para simular a computação nas nuvens. Entretanto, eles não conseguem simular a característica principal de computação nas nuvens: virtualização e gerenciamento dos recursos.

A figura 21 ilustra a arquitetura de Cloudsim. Para adequar-se ao objetivo do teste, alterações podem ser feitas em qualquer componente pertencente ao núcleo do simulador.

A saída do simulador depende do que está sendo testado. No caso deste trabalho, como

Figura 21: Arquitetura do Cloudsim



Fonte: Calheiros et al. (2011).

se trata de um mecanismo de alocação considerando o consumo de energia, a saída mostra dados como o tempo de conclusão de um serviço (*Cloudlets*), a energia para processar todos os serviços, número de migrações de máquinas virtuais etc. Nesse caso, o objetivo principal é verificar o gasto de energia de acordo com a política empregada.

5.2 Cenários

Assim como o algoritmo *Lago Allocator*, a solução proposta funciona tanto para ambientes homogêneos (onde os servidores são iguais, ou seja, possuem os mesmos recursos de *hardware*), como para heterogêneos. Além disso, foi utilizada a configuração de *data centers* não federados; arquitetura onde todos os servidores estão na mesma nuvem.

Para simular testes de maneira similar aos ambientes reais, utilizou-se a simulação de *data centers* de pequeno (10 servidores, 20 máquinas virtuais e 20 *cloudlets*), médio (100 servidores, 200 máquinas virtuais e 200 *cloudlets*) e grande porte (1000 servidores, 2000 máquinas virtuais e 2000 *cloudlets*) (LAGO; MADEIRA; BITTENCOURT,). Foi considerado em cada servidor a utilização dos mecanismos de *Dynamic Voltage and Frequency Scaling*. (DVFS) (SUEUR; HEISER,). Com essa técnica, é possível reduzir a frequência e voltagem do processador de acordo com a carga de trabalho sendo processada, economizando energia em períodos

onde há pouca carga de trabalho para ser processada.

Todos os servidores possuem 24 GB de memória RAM, 1 terabyte de armazenamento secundário e rede de 1 Gbps. Já a capacidade de processamento do servidor é atribuída seguindo uma distribuição circular (*Round Robin*) do conjunto de valores {1000, 1500, 2000, 2500, 3000} MIPS. Desse modo, os servidores h1, h2, h3, h4, h5 e h6 terão as capacidades de processamento 1000, 1500, 2000, 2500, 3000 e 1000 MIPS, respectivamente, seguindo a lógica circular da distribuição utilizada. No relacionado às máquinas virtuais, foi considerado principalmente a capacidade de processamento, atribuída também seguindo uma distribuição circular entre os valores {500, 750, 1000} MIPS. Além da definição da capacidade de processamento para as máquinas virtuais, foi configurada como política de escalonamento, utilizada pelas máquinas virtuais para as tarefas submetidas, a que trabalha somente com a execução de uma tarefa trabalhando como um serviço *online* com carga de trabalho dinâmica.

Para a simulação de uma tarefa no *CloudSim*, utiliza-se a entidade denominada *Cloudlet*. Desse modo, cada tarefa será mapeada em uma *Cloudlet*, de modo a representar a tarefa no simulador com todos as características observadas no ambiente real, incluindo os recursos requisitados para a sua execução. O tamanho de uma tarefa, ou seja, a quantidade de processamento em MI (milhões de instruções) necessário para executar a tarefa, é atribuída seguindo uma distribuição circular selecionando os elementos do conjunto de valores {10.000, 15.000, 20.000, 25.000} MI.

Foram utilizados três tipos de políticas para o *data center*:

- Não considerando energia (*Non Power-Aware* - NPA): nessa política, técnicas de redução de consumo de energia não são utilizadas, ou seja, todos os servidores estarão executando com 100 % da sua capacidade, consumindo, assim, o máximo de energia.
- Considerando energia (*Power Aware* - PA): Realiza otimizações visando reduzir o consumo de energia. Uma dessas otimizações é colocar os servidores sem máquinas virtuais alocadas no modo *sleep*, consumindo uma quantidade menor de energia.
- Considerando energia com utilização da técnica DVFS (*Power Aware with DVFS* PAD): Além das otimizações descritas no item anterior, é aplicada a técnica de DVFS descrita anteriormente.

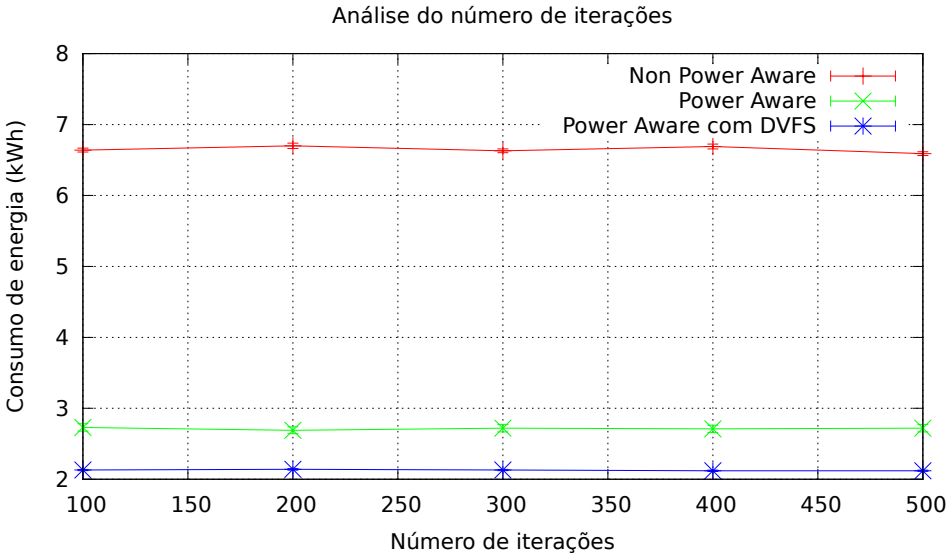
A fim de comparar os resultados obtidos da solução proposta, utilizou-se os algoritmos com políticas de alocação *Round Robin*, *Best Resource Selection*, *Minimum Power Difference* (BUYYA; BELOGLAZOV; ABAWAJY,), e *Lago Allocator* (LAGO; MADEIRA; BITTENCOURT,). No algoritmo *Round Robin*, as máquinas virtuais são atribuídas seguindo uma distribuição circular, onde nenhum outro fator é considerado. Já no *Best Resource Selection*, para uma determinada máquina virtual é selecionado um servidor onde há a maior taxa de utilização,

desde que tenha recursos suficientes para isso. Nos algoritmos *Minimum Power Diff* e *Lago Allocator*, aloca-se a máquina virtual no servidor onde o aumento do consumo de energia seja o menor, havendo diferença em alguns critérios utilizados.

5.2.1 Parâmetros do ACO

Para a execução do algoritmo proposto baseado em colônia de formigas, foram utilizados valores determinados empiricamente. Esses valores foram obtidos através da simulação e avaliação dos parâmetros utilizados, onde procurou-se determinar os parâmetros que gerassem boas soluções independente do cenário ao qual eles fossem submetidos. Como exemplo, o gráfico mostra a avaliação do número de iterações utilizadas, onde percebe-se que a partir de certo ponto, o aumento de iterações não acarreta na melhoria da solução 22.

Figura 22: Influência do número de iterações na solução



Os valores mínimo e máximo de feromônio foram determinados de modo a aumentar a diversidade das soluções obtidas, minimizando os casos onde o algoritmo encontre um ótimo local e não encontre uma solução melhor. Desse modo, o estabelecimento do limite mínimo de feromônio em 0.3 possibilita a escolha de outros nós do grafo não selecionados nas etapas anteriores. Já o limite superior evita o valor máximo do feromônio, e, consequentemente, diversifica a escolha de soluções em iterações futuras.

Esses parâmetros estão especificados na tabela 2. Além disso, por ser uma metaheurística probabilística, o intervalo de confiança de 95 % foi utilizado. Por fim, visando obter dados confiáveis através da normalidade dos mesmos (GRINSTEAD; SNELL, 1997), os algoritmos foram testados 33 vezes para cada configuração de cenário.

A quantidade de formigas na Tabela 2 especifica a quantidade de soluções geradas em

Tabela 2: Resumo dos parâmetros do algoritmo

Parâmetros do algoritmo de colônia de formigas	
Quantidade de iterações	100
Quantidade de formigas	40
Taxa de evaporação de feromônio	0.5
Taxa de atualização de feromônio	$\frac{f(parcial)}{f(melhor)}$
Limite mínimo de feromônio	0.3
Limite máximo de feromônio	0.8

cada iteração do algoritmo, ou seja, a quantidade de mapeamentos diferentes gerados em cada iteração. Já o feromônio é aplicado para que os melhores mapeamentos, baseados no histórico, tenham uma maior probabilidade de seleção.

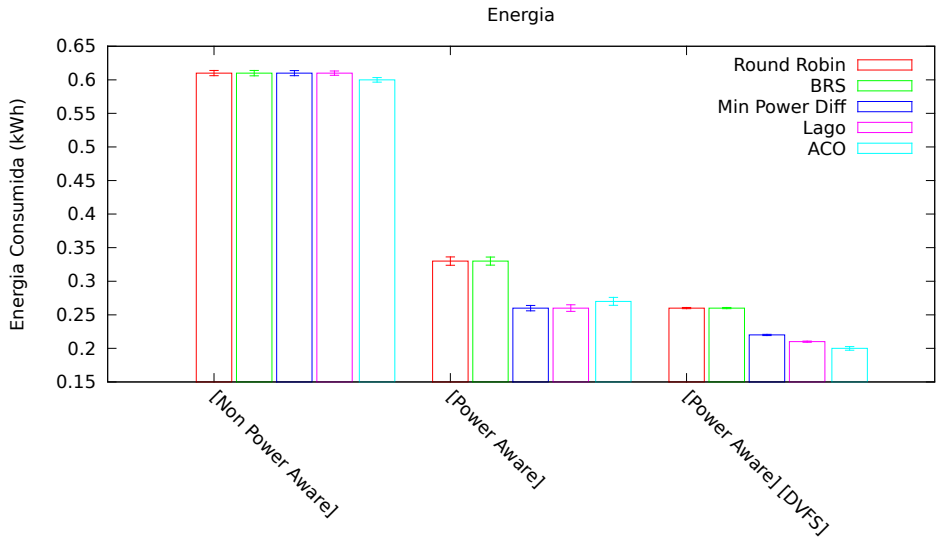
5.3 Métricas

Para medir a eficiência de um mecanismo de alocação considerando a energia, um parâmetro intuitivo é a **quantidade total de energia consumida** para o processamento de uma carga de trabalho (LAGO; MADEIRA; BITTENCOURT, 2011; CALHEIROS et al., 2011; JANSEN; BRENNER, 2011). Outro parâmetro importante relacionado ao consumo é o **tempo de processamento da carga de trabalho** (LAGO; MADEIRA; BITTENCOURT,). Utilizando esse dois parâmetros em conjunto, é possível avaliar se a economia de energia obtida impactou o desempenho. Desse modo, é possível analisar os possíveis efeitos colaterais ocasionados pela alocação de máquinas virtuais considerando energia.

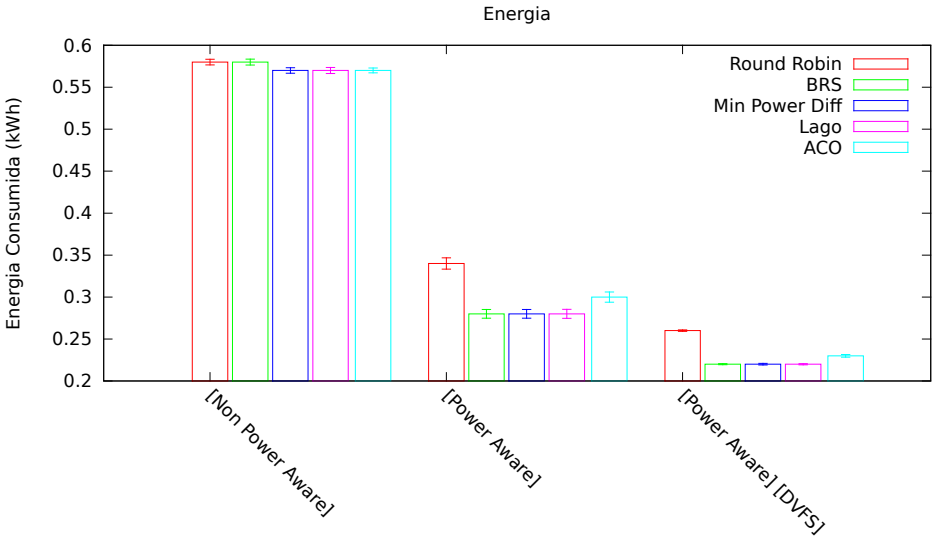
5.4 Resultados

No *data center* de pequeno porte, o desempenho obtido pelo algoritmo proposto (ACO) foi similar ao obtido pelo *Lago Allocator* (LAGO; MADEIRA; BITTENCOURT,) e pelo *Minimum Power Diff* (BUYYA; BELOGLAZOV; ABAWAJY,) no quesito de economia de energia, tanto no *data center* homogêneo (figura 24(b)) como no heterogêneo (figura 24(a)).

Figura 23: Consumo de energia em um *data center* de pequeno porte



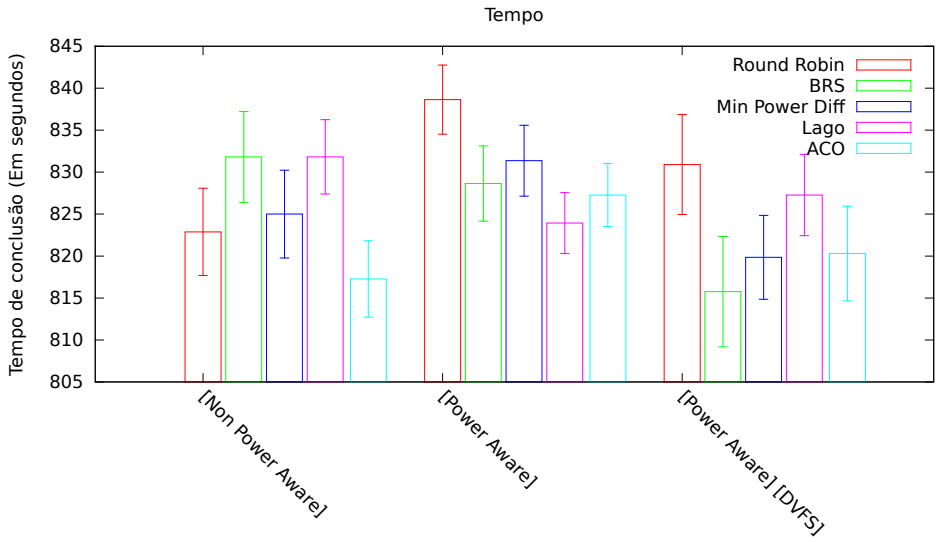
(a) Heterogêneo



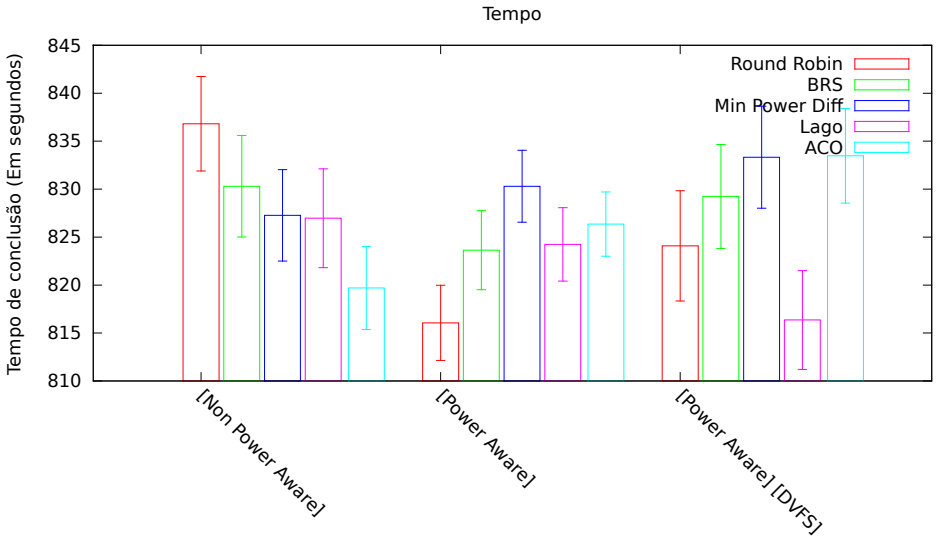
(b) Homogêneo

Já com relação ao tempo de processamento das tarefas, houve um pequeno aumento em alguns dos cenários utilizados (figuras 25(b) e 25(a)). Entretanto, houve cenários onde o tempo obtido foi melhor quando comparado com as outras abordagens. Essa variação é ocasionada pela política de escalonamento configurada na máquina virtual, simulando uma aplicação *online* com carga de trabalho variável no tempo. Como as variações não chegam a 2 %, considera-se os tempos obtidos similares.

Figura 24: Tempo de processamento em um *data center* de pequeno porte



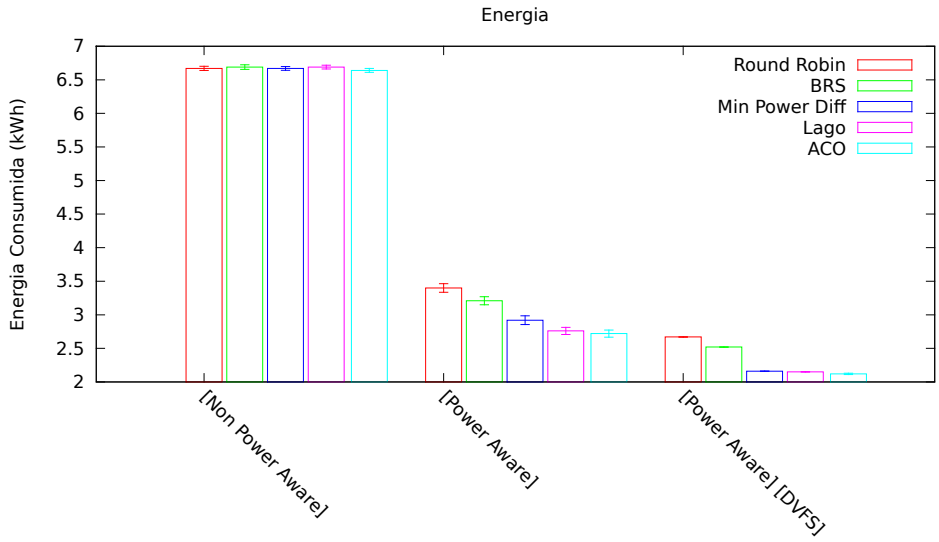
(a) Heterogêneo



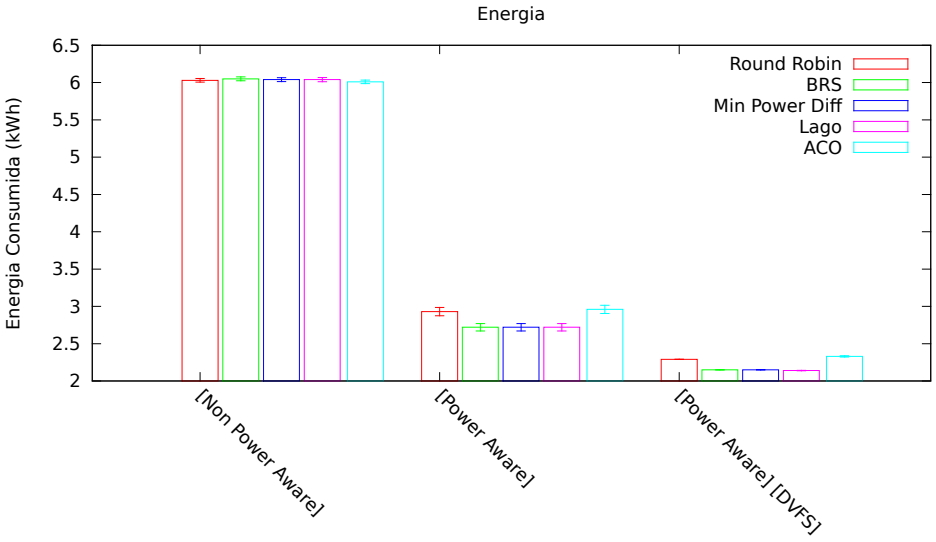
(b) Homogêneo

Já para o *data center* de médio porte heterogêneo, o ACO obteve um desempenho melhor quando comparado com a segunda melhor heurística, o Lago Allocator, no critério de consumo de energia (figura 26(a)). Em um cenário homogêneo, o ACO obteve menor consumo apenas no *data center* NPA. Nos outros tipos de *data center* foram obtidas taxas de consumo, em média, 10 % maiores do obtido pelo *Lago Allocator*, comportamento observado na figura 26(b).

Figura 25: Consumo de energia em um *data center* de médio porte



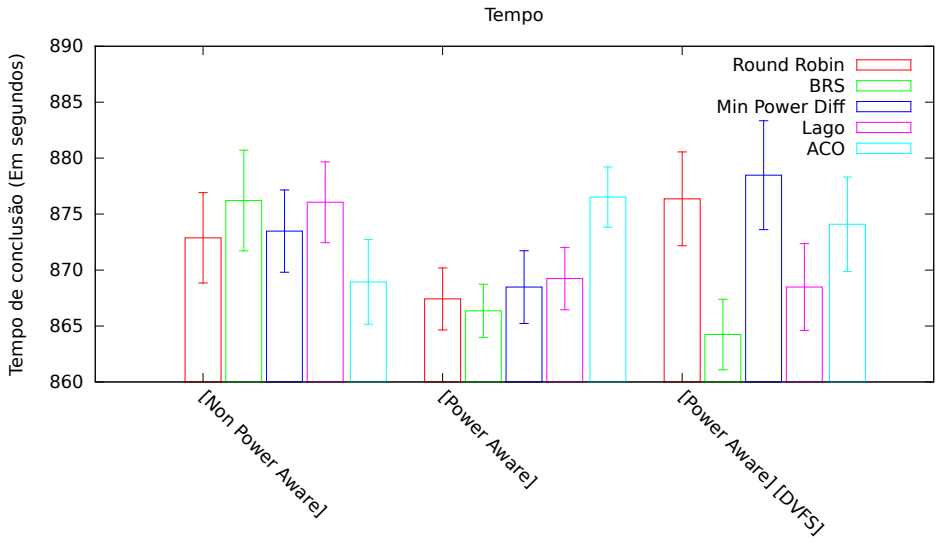
(a) Heterogêneo



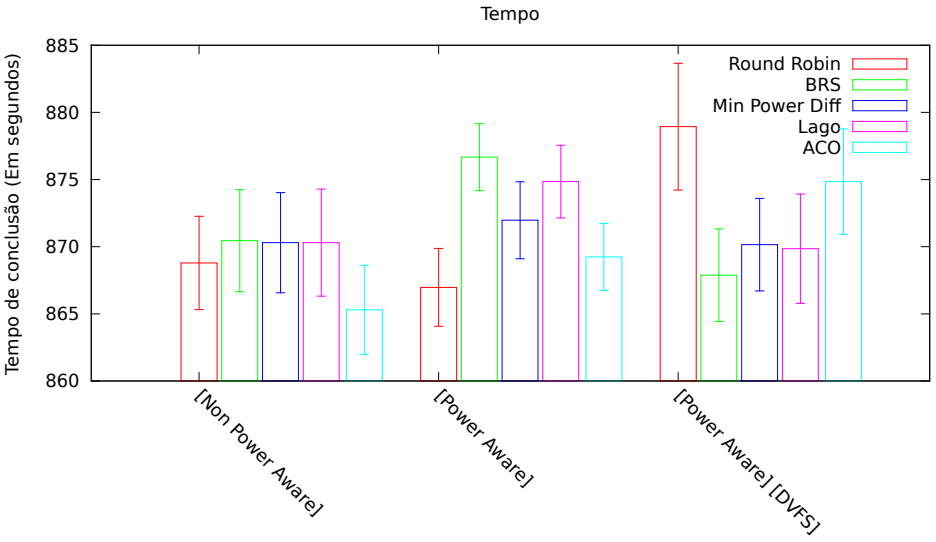
(b) Homogêneo

Com relação ao tempo de processamento, em ambos os cenários, tanto no heterogêneo (figura 27(a)) quanto no homogêneo (figura 27(b)), foi observado o mesmo comportamento descrito para o *data center* de pequeno porte.

Figura 26: Tempo de processamento em um *data center* de médio porte



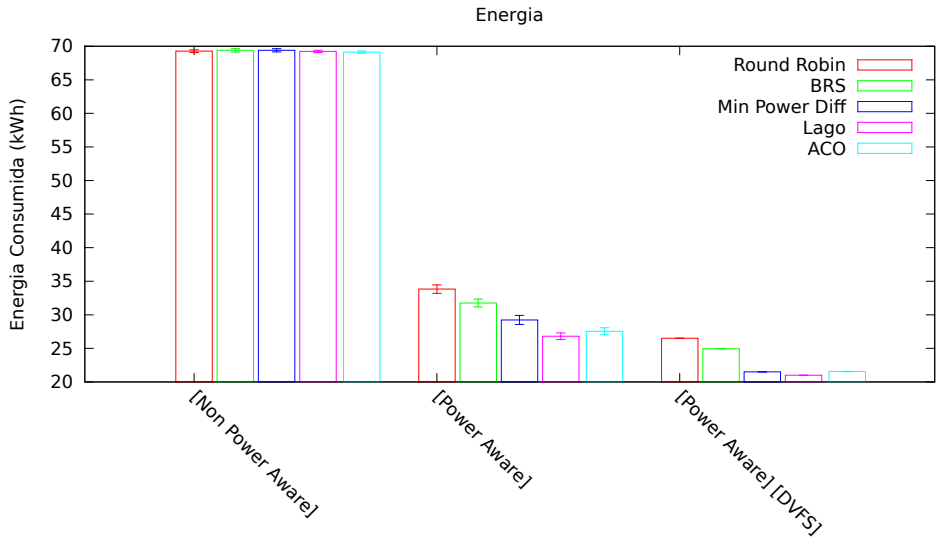
(a) Heterogêneo



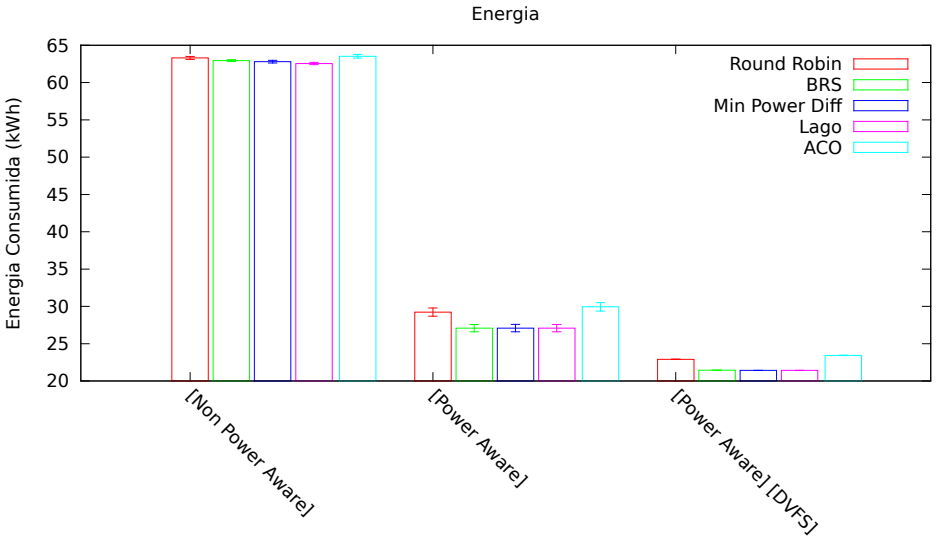
(b) Homogêneo

Por fim, em um *data center* de grande porte heterogêneo, pode-se considerar o desempenho obtido semelhante ao obtido pelo Lago Allocator, no critério de consumo de energia (figura 28(a)). Em um cenário homogêneo, o ACO obteve um comportamento ligeiramente maior do que os obtidos pelas outras abordagens utilizadas, com, no máximo, 2 kwh de consumo adicional (figura 28(b)).

Figura 27: Consumo de energia em um *data center* de grande porte



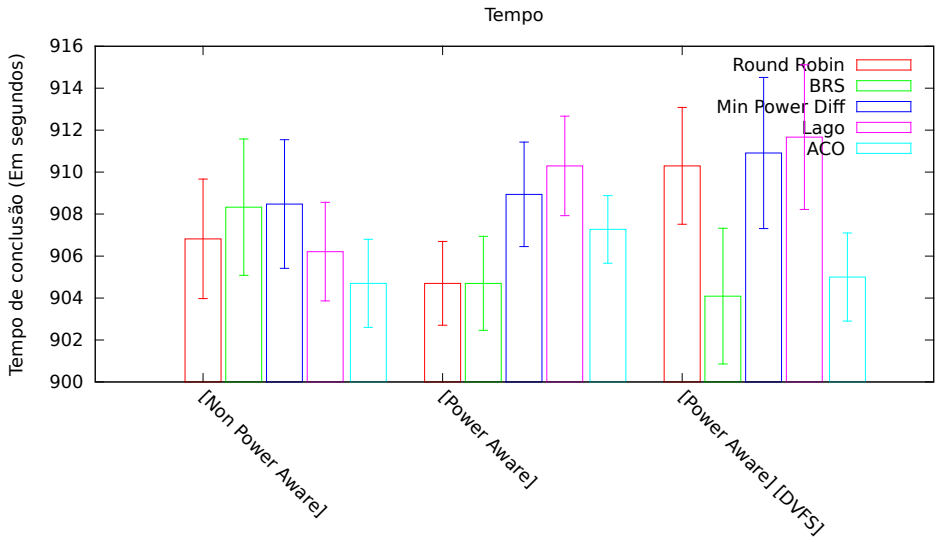
(a) Heterogêneo



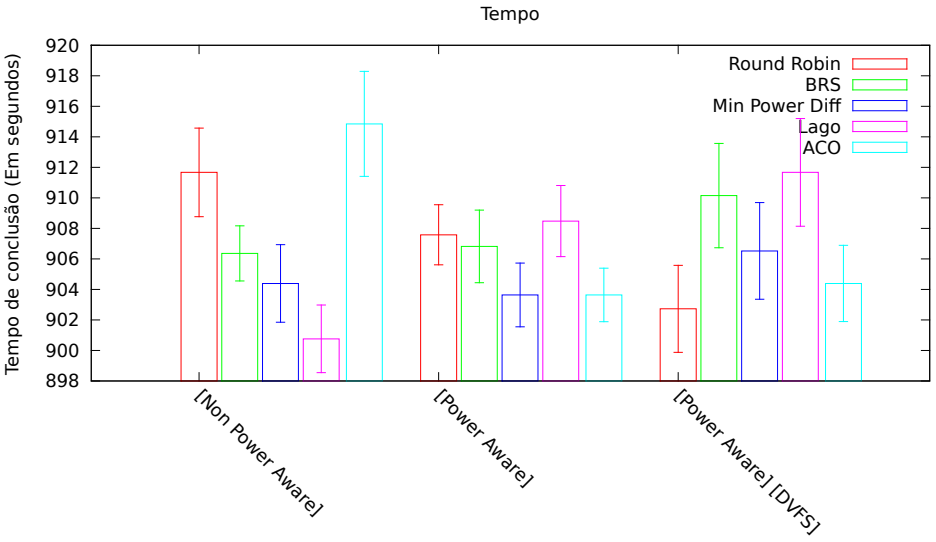
(b) Homogêneo

Já no tempo de processamento, nos dois cenários (figuras 29(a) e 29(b)), foi observado o mesmo comportamento descrito para os *data centers* especificados anteriormente.

Figura 28: Tempo de processamento em um *data center* de grande porte



(a) Heterogêneo



(b) Homogêneo

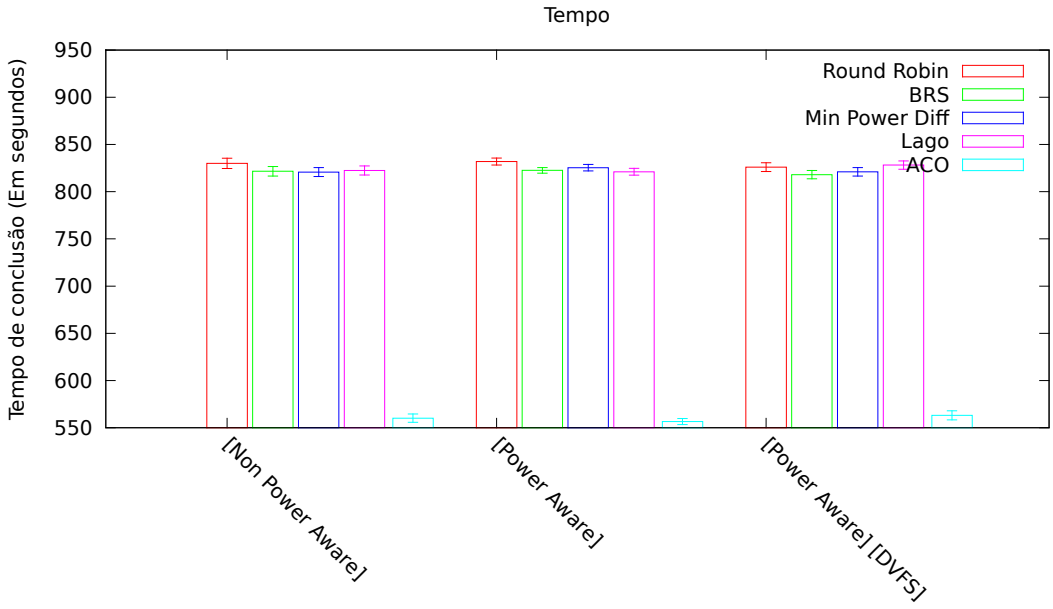
A partir dos resultados obtidos, conclui-se que a solução proposta não se adapta ao cenário homogêneo. Essa limitação é ocasionada pelo método de seleção probabilística dos participantes da solução, característico dos algoritmos baseados na metaheurística de otimização por colônia de formigas. No contexto do problema abordado, trata-se da seleção probabilística do servidor onde a máquina virtual será alocada. Como no cenário homogêneo todos os servidores possuem o mesmo consumo, o algoritmo ACO irá funcionar semelhante a um balanceador de carga devido ao parâmetro de informação heurística utilizado, procurando distribuir as máquinas virtuais entre os servidores disponíveis, tendo como consequência as penalidades de consumo de energia e tempo de processamento apresentados nas figuras 25(b) e 24(b).

Outra conclusão obtida dos resultados foi a identificação do tempo de processamento do conjunto de tarefas como outro fator que contribui com o consumo de energia. Tal contribuição é ocasionada pelo tempo de utilização da infraestrutura para atender a demanda das cargas de trabalho. Desse modo, quanto mais tempo for necessário para processar todo o conjunto de tarefas, maior será o tempo de utilização dos servidores, e, consequentemente, maior o consumo de energia. A fim de analisar o impacto de um mecanismo de alocação de tarefas no consumo de energia, utilizou-se o mecanismo de alocação proposto por Amarante (2013) juntamente com o algoritmo proposto, ambos baseados no algoritmo de otimização por colônia de formigas. Essa implementação foi comparada com outros algoritmos com mecanismo de alocação de tarefa padrão (distribuição circular) para mostrar a melhoria obtida através da utilização desse mecanismo.

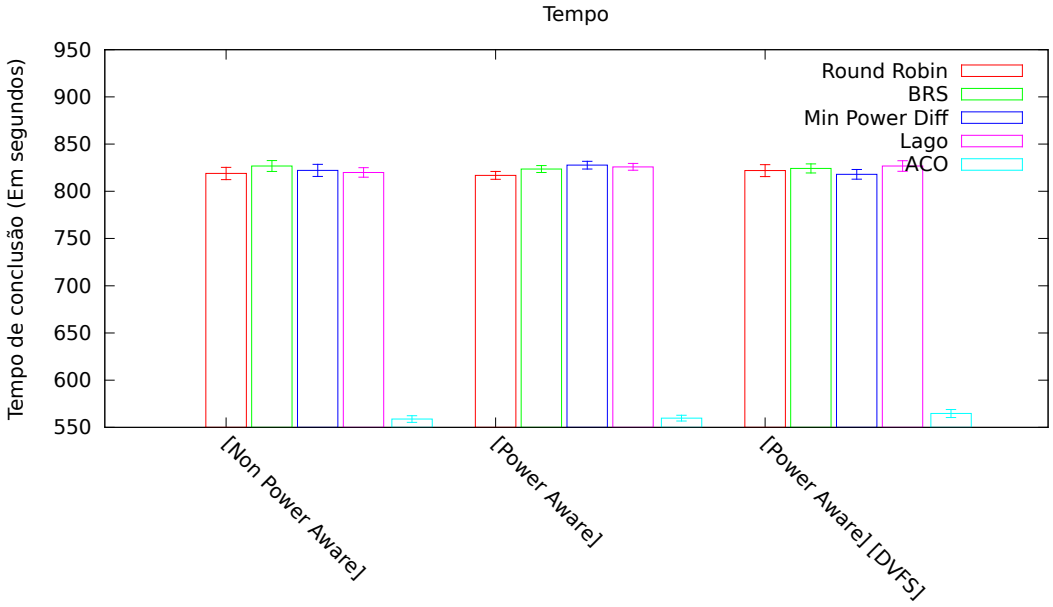
O mecanismo de escalonamento de tarefas influencia, principalmente, na redução do tempo de processamento do conjunto de tarefas submetidas para o *data center* (figuras 30(a) e 30(b)). Já o consumo de energia (figuras 31(a) e 31(b)) é reduzido indiretamente devido a redução no tempo de utilização da infraestrutura computacional. A maior redução no consumo é observada no *data center* que não utiliza mecanismo para economia de energia, pois nesse tipo de *data center* todas as máquinas funcionam no máximo da capacidade todo o tempo. Já nos outros tipos de *data center*, o consumo de energia é pouco influenciado, pois eles já possuem mecanismos para quando o servidor estiver ocioso, realizar ações para reduzir o consumo. Desse modo, o consumo de energia pouco varia nesses *data centers*, tendo resultados similares aos obtidos sem o mecanismo de alocação de tarefa.

Dessa forma, percebe-se que a utilização conjunta de um mecanismo de alocação de máquinas virtuais em *data centers* em conjunto com um mecanismo de alocação de tarefas nas máquinas virtuais ajuda na redução do consumo de energia, principalmente nos *data centers* que não utilizam mecanismos para a economia de energia.

Figura 29: Tempo de processamento em um *data center* de pequeno porte com mecanismo de alocação de tarefas

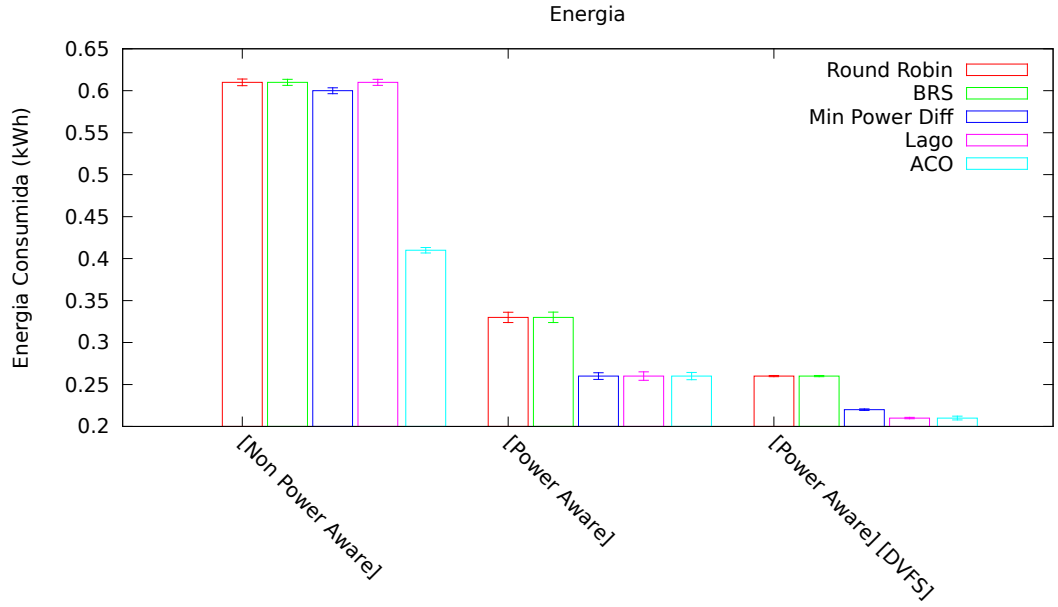


(a) Heterogêneo

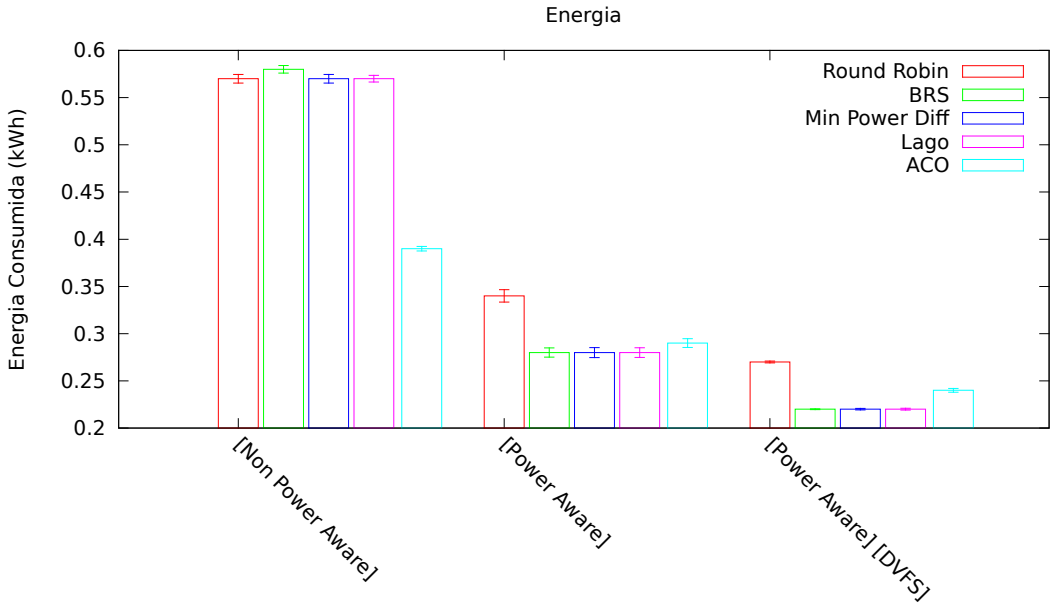


(b) Homogêneo

Figura 30: Consumo de energia em *data center* de pequeno porte com mecanismo de alocação de tarefas



(a) Heterogêneo



(b) Homogêneo

6 CONCLUSÃO

Para abordar o problema de alocação de máquinas virtuais nos servidores em um ambiente de *data center* em computação nas nuvens foi proposta uma modelagem como problema de múltiplas mochilas com o objetivo de minimizar o consumo de energia da infraestrutura, sem ocasionar grandes variações no tempo de processamento das tarefas submetidas às máquinas virtuais. Além disso, foi proposto um algoritmo baseado em colônia de formigas para resolver esse problema. Essas duas propostas compõem as contribuições dadas por esse trabalho.

Através dos resultados obtidos, foi mostrado o desempenho satisfatório do algoritmo proposto quando comparado com outras soluções utilizadas tanto em *data centers* de pequeno como de médio porte, tendo como principal métrica o consumo de energia. Os melhores resultados foram obtidos em *data centers* heterogêneos, onde houve uma redução aproximada de 2 % no consumo de energia quando comparada com a segunda melhor solução, o Lago Allocator. Já no *data center* homogêneo, observa-se que o algoritmo baseado em colônia de formigas não conseguiu adaptar-se ao cenário, apresentando um comportamento ligeiramente pior, obtendo valores para o consumo de energia em média 5 % maior. Tal fato acontece devido ao valor da eficiência energética ($\frac{Capacidade\ total\ de\ processamento}{Consumo\ total\ de\ energia}$) ser igual em todas os servidores, fazendo o algoritmo apresentar comportamento similar ao de um balanceador de carga. Esse fato é amenizado por ser um cenário pouco comum de acontecer nos *data centers* reais, pois os provedores constantemente adicionam novas máquinas à infraestrutura, adquirindo máquinas mais novas com recursos mais avançados, quando comparado com os recursos das máquinas já existentes no *data center*. Desse modo, o desempenho obtido pelo algoritmo baseado em colônia de formigas na solução de um problema modelado como problema de múltiplas mochilas foi satisfatório considerando um *data center* heterogêneo.

Como trabalhos futuros, podem ser implementadas outras metaheurísticas ou mesmo algoritmos bioinspirados a fim de comparar os resultados obtidos pelo algoritmo baseado em colônia de formigas. Também merece atenção os parâmetros do algoritmo de formigas utilizados, onde pode ser realizado um estudo matemático para determinar valores ótimos a fim de gerar boas soluções. Outra melhoria no algoritmo de colônia de formigas seria a utilização de uma fórmula definindo o quão boa é a alocação de uma máquina virtual em um servidor considerando o consumo de energia e o balanceamento de carga nos servidores juntos.

Outra melhoria seria transformar o algoritmo proposto em uma solução *online*, ou seja, a qualquer momento pode ser atribuída uma nova máquina virtual e o algoritmo tratar essa alocação nos servidores, bem como tratar a questão da realocação das máquinas virtuais já alocadas para um estado consumindo uma menor quantidade de energia.

REFERÊNCIAS

ALAYA, Inès; SOLNON, Christine; GHéDIRA, Khaled. Ant algorithm for the multi-dimensional knapsack problem. In: *International Conference on Bioinspired Optimization Methods and their Applications, 2004*. [S.l.: s.n.]. p. 63–72.

AMARANTE, Silvio Roberto Martins. *Utilizando o problema da atribuição generalizada para modelar o problema de alocação de cargas de trabalho em computação nas nuvens*. Monografia (Bacharelado) — Universidade Estadual do Ceará, Fortaleza, Ceará, 2013.

AMAZON. *About AWS*. [S.l.], Janeiro 2013. Disponível em: <http://aws.amazon.com/what-is-aws/>.

ANDERSON, Dave; DYKES, Jim; RIEDEL, Erik. More than an interface—scsi vs. ata. In: *Proceedings of the 2nd USENIX Conference on File and Storage Technologies, 2003*. Berkeley, CA, USA: USENIX Association. p. 245–257. Disponível em: <http://dl.acm.org/citation.cfm?id=1090694.1090724>.

ARMBRUST, Michael et al. *Above the Clouds: A Berkeley View of Cloud Computing, 2009*. [S.l.]. Disponível em: <http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html>.

BECKERS, R.; DENEUBOURG, J. L.; GOSS, S. Trails and u-turns in the selection of a path by the ant lasius niger. *Journal of Theoretical Biology, 1992*, p. 397–415, 1992.

BELOGLAZOV, Anton; ABAWAJY, Jemal; BUYYA, Rajkumar. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems, 2012*, v. 28, n. 5, p. 755 – 768. ISSN 0167-739X. <ce:title>Special Section: Energy efficiency in large-scale distributed systems</ce:title>. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X11000689>.

BELOGLAZOV, Anton; BUYYA, Rajkumar. Energy efficient resource management in virtualized cloud data centers. *IEEE International Symposium on Cluster Computing and the Grid, 2010*, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 826–831.

BOBROFF, N.; KOCHUT, A.; BEATY, K. Dynamic placement of virtual machines for managing sla violations. In: *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*. [S.l.: s.n.], 2007. p. 119–128.

BOLLA, R. et al. Energy efficiency in the future internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures. *IEEE Communications Surveys Tutorials, 2011*, v. 13, n. 2, p. 223–244, 2011. ISSN 1553-877X.

BOZDOGAN, Ali Onder; YILMAZ, Asım Egemen; EFE, Murat. Performance analysis of swarm optimization approaches for the generalized assignment problem in multi-target tracking applications. 2010.

BUYYA, Rajkumar; BELOGLAZOV, Anton; ABAWAJY, Jemal H. Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges. *CoRR, 2010*, abs/1006.0308.

CALHEIROS, Rodrigo N. et al. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exper., 2011*, John Wiley & Sons, Inc., New York, NY, USA, v. 41, n. 1, p. 23–50. ISSN 0038-0644. Disponível em: <http://dx.doi.org/10.1002/spe.995>.

C.GUO et al. Secondnet: A data center network virtualization architecture with bandwidth guarantees. In: *Co-NEXT '10 Proceedings of the 6th International Conference, 2010*. ACM, NY, USA. Disponível em: <http://0-portal.acm.org.millennium.lib.cyut.edu.tw/citation.cfm?id=1921188>.

CLARK, Christopher et al. Live migration of virtual machines. In: *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2, 2005*. Berkeley, CA, USA: USENIX Association. (NSDI'05), p. 273–286. Disponível em: <http://dl.acm.org/citation.cfm?id=1251203.1251223>.

CLEMENTE, Quebo Kenge. *Resolução de problemas da mochila multicritério através de técnicas metaheurísticas*. Dissertação (Mestrado) — Universidade Técnica de Lisboa, 2010.

CLOUDSIGMA. *Why choose cloud servers from CloudSigma?* [S.l.], Janeiro 2013. Disponível em: <http://www.cloudsigma.com/>.

CORREIA, Sérgio Luis Olinda Braga. *Roteamento em redes veiculares utilizando colônias de formigas e predição de mobilidade*. Dissertação (Mestrado) — Universidade Estadual do Ceará, Fortaleza, Ceará, 2011.

D., Christy Sujatha; ABIMANNAN, Satheesh. Energy efficient free cooling system for data centers. In: *Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science, 2011*. Washington, DC, USA: IEEE Computer Society. (CLOUDCOM '11), p. 646–651. ISBN 978-0-7695-4622-3. Disponível em: <http://dx.doi.org/10.1109/CloudCom.2011.100>.

DIFFIE, Whitfield; HELLMAN, Martin E. New Directions in Cryptography. *IEEE Transactions on Information Theory, 1976*, IT-22, n. 6, p. 644–654.

DORIGO, Marco; BIRATTARI, Mauro; STüTZLE, Thomas. Ant colony optimization – artificial ants as a computational intelligence technique. *IEEE COMPUT. INTELL. MAG, 2006*, v. 1, p. 28–39.

DORIGO, Marco; BLUM, Christian. Ant colony optimization theory: A survey. *Theoretical Computer Science, 2005*, v. 344, p. 243 – 278.

DORIGO, M.; GAMBARDELLA, L.M. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation, 1997*, v. 1, n. 1, p. 53 –66.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. Ant system: optimization by a colony of cooperating agents. *Cybernetics, IEEE Transactions on Systems, Man, and Cybernetics, Part B, 1996*, v. 26, n. 1, p. 29 –41.

DROPBOX. *Dropbox*. [S.l.], Janeiro 2013. Disponível em: <https://www.dropbox.com/>.

ESNAULT, Armel. *Energy-Aware Distributed Ant Colony Based Virtual Machine Consolidation in IaaS Clouds*. Dissertação (Mestrado), jun. 2012.

EUCALYPTUS. *Eucalyptus*. [S.l.], Janeiro 2013. Disponível em: <http://www.eucalyptus.com/>.

FAKHIM, B. et al. Cooling solutions in an operational data centre: A case study. *Applied Thermal Engineering, 2011*, v. 31, n. 14–15, p. 2279 – 2291. ISSN 1359-4311. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1359431111001578>.

FANG, Weiwei et al. Vmplanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers. *Computer Networks 2013*, v. 57, n. 1, p. 179 – 196. ISSN 1389-1286. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128612003301>.

FIDANOVA, Stefka. Ant colony optimization for multiple knapsack problem and model bias. In: LI, Zhilin; VULKOV, Lubin G.; WASNIEWSKI, Jerzy (Ed.). *NAA, 2004*. [S.l.]: Springer. (Lecture Notes in Computer Science, v. 3401), p. 280–287.

GOOGLE. *Google App Engine*. [S.l.], Julho 2011. Disponível em: <http://code.google.com/appengine/>.

GOOGLE. *Google Drive*. [S.l.], Janeiro 2013. Disponível em: <http://drive.google.com>.

GRINSTEAD, Charles M.; SNELL, J. Laurie. *Introduction to Probability*. 2. ed. [S.l.]: American Mathematical Society, 1997. ISBN 0821807498.

HÖLLDOBLER, B.; WILSON, E.O. *The Ants*. Springer, 1998. ISBN 9783540520924. Disponível em: <http://books.google.com.br/books?id=o87CQgAACAAJ>.

HU, Zheng; WU, Kaijun; HUANG, Jinsong. An utility-based job scheduling algorithm for current computing cloud considering reliability factor. In: *IEEE 3rd International Conference on Software Engineering and Service Science (ICSESS), 2012*. [S.l.: s.n.]. p. 296–299.

IBM. *Server virtualization with IBM PowerVM*. [S.l.], Janeiro 2013. Disponível em: <http://www-03.ibm.com/systems/power/software/virtualization/index.html>.

JANSEN, R.; BRENNER, P.R. Energy efficient virtual machine allocation in the cloud. In: *International Green Computing Conference and Workshops (IGCC), 2011*. [S.l.: s.n.]. p. 1 –8.

JIN, Hai et al. Live virtual machine migration with adaptive, memory compression, 2009. In: *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*. [S.l.: s.n.]. p. 1 –10.

KELLERER, Hans; PFERSCHY, Ulrich; PISINGER, David. Multiple knapsack problems. In: *Knapsack Problems, 2004*. Springer Berlin Heidelberg. p. 285–316. ISBN 978-3-642-07311-3. Disponível em: <http://dx.doi.org/10.1007/978-3-540-24777-7_10>.

K.MUKHERJEE; G.SAHOO. Article:green cloud: An algorithmic approach. *International Journal of Computer Applications, 2010*, v. 9, n. 9, p. 1–6, November. Published By Foundation of Computer Science.

KUMAR, Saurabh; BUYYA, Rajkumar. *Green Cloud Computing and Environmental Sustainability*. [S.l.: s.n.], 2012. 315–339 p. ISBN 9781118305393.

LAGO, Daniel Guimaraes do; MADEIRA, Edmundo R. M.; BITTENCOURT, Luiz Fernando. Power-aware virtual machine scheduling on clouds using active cooling control and dvfs. In: *Proceedings of the 9th International Workshop on Middleware for Grids, Clouds and e-Science, 2011*. New York, NY, USA: ACM. (MGC '11), p. 2:1–2:6. ISBN 978-1-4503-1068-0. Disponível em: <http://doi.acm.org/10.1145/2089002.2089004>.

LANGLEY, Kent. *Get Your Head in the Clouds*. [S.l.], Fevereiro 2012. Disponível em: <http://www.wavelinks.ie/2009/09/02/cloud-computing-get-your-head-in-the2ns/>.

LAUREANO, Marcos Aurelio Pchek; MAZIERO, Carlos Alberto. Live migration of virtual machines. In: *Simpósio Brasileiro em Sistemas de Informação e de Sistemas Computacionais - SBSEG 2008 - Minicursos, 2008*. [S.l.: s.n.].

LI, Luqun. An optimistic differentiated service job scheduling system for cloud computing service users and providers. In: *Third International Conference on Multimedia and Ubiquitous Engineering, 2009*. [S.l.: s.n.]. p. 295–299.

LI, Xuan; LO, Jine-Chung. Pricing and peak aware scheduling algorithm for cloud computing. In: *Innovative Smart Grid Technologies (ISGT), 2012 IEEE PES*. [S.l.: s.n.], 2012. p. 1–7.

MARTELLO, Silvano; TOTH, Paolo. *Knapsack problems: algorithms and computer implementations*. New York, NY, USA: John Wiley & Sons, Inc., 1990. ISBN 0-471-92420-2.

MEIJER, G. I. Cooling energy-hungry data centers. *Science*, v. 328, n. 5976, p. 318–319, 2010. Disponível em: <http://www.sciencemag.org/content/328/5976/318.short>.

MENG, Xiaoqiao; PAPPAS, Vasileios; ZHANG, Li. Improving the scalability of data center networks with traffic-aware virtual machine placement. In: *Proceedings IEEE INFOCOM, 2010*. [S.l.: s.n.]. p. 1–9. ISSN 0743-166X.

MOORE, Justin et al. Making scheduling "cool": temperature-aware workload placement in data centers. In: *Proceedings of the annual conference on USENIX Annual Technical Conference, 2005*. Berkeley, CA, USA: USENIX Association. (ATEC '05), p. 5–5. Disponível em: <http://dl.acm.org/citation.cfm?id=1247360.1247365>.

OPENNEBULA. *OpenNebula.org project*. [S.l.], Janeiro 2013. Disponível em: <http://opennebula.org/about:about>.

ORACLE. *Virtual Box*. [S.l.], Janeiro 2013. Disponível em: <https://www.virtualbox.org/>.

PARKHILL, D.F. *The challenge of the computer utility*. [S.l.]: Addison-Wesley Pub. Co., 1966. (The Challenge of the Computer Utility, p. 246).

PATEL, C. D. et al. Smart cooling of datacenters. In: *Proceedings of IPACK'03 – The PacificRim/ASME International Electronics Packaging Technical Conference and Exhibition, Kauai, HI, IPACK-35059, 2003*. [S.l.: s.n.].

QEMU. *Qemu*. [S.l.], Janeiro 2013. Disponível em: <http://wiki.qemu.org/>.

REDDIT. *Reddit.com*. [S.l.], Janeiro 2013. Disponível em: <http://http://www.reddit.com/>.

SHARMA, Ratnesh K. et al. Balance of power: Dynamic thermal management for internet data centers. *IEEE Internet Computing, 2005*, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 9, n. 1, p. 42–49. ISSN 1089-7801. Disponível em: <http://dx.doi.org/10.1109/MIC.2005.10>.

SHAW, Paul. *A Constraint for Bin Packing*. [S.l.: s.n.], 2004. 648–662 p.

SHRIBMAN, Aidan; HUDZIA, Benoit. Pre-copy and post-copy vm live migration for memory intensive applications. In: CARAGIANNIS, Ioannis et al. (Ed.). *Euro-Par 2012: Parallel Processing Workshops, 2013*. [S.l.]: Springer Berlin Heidelberg, (Lecture Notes in Computer Science, v. 7640). p. 539–547. ISBN 978-3-642-36948-3.

SINHA, Prabhakant; ZOLTNERS, Andris A. The Multiple-Choice Knapsack Problem. *Operations Research, 1979*, INFORMS, v. 27, n. 3, p. 503–515, maio.

SOARES, Gustavo Luís. *Algoritmos Genéticos: Estudo, Novas Técnicas e Aplicações*. Dissertação (Mestrado) — Universidade Federal de Minas Gerais, Belo Horizonte, Minas Gerais, 1997.

SOARES, Paolo Victor Gonçalves. *Gatekeeper: Controle de tráfego distribuído em datacenters virtualizados*. Dissertação (Mestrado) — Universidade Federal de Minas Gerais, Belo Horizonte, Minas Gerais, 2010.

STüTZLE, Thomas; HOOS, Holger. *Improvements on Ant-System: Introducing MAX-MIN Ant System*. 1996.

SUEUR, Etienne Le; HEISER, Gernot. Dynamic voltage and frequency scaling: the laws of diminishing returns. In: *Proceedings of the 2010 international conference on Power aware computing and systems, 2010*. Berkeley, CA, USA: USENIX Association. p. 1–8. Disponível em: <http://dl.acm.org/citation.cfm?id=1924920.1924921>.

TAI, Jianzhe et al. Ara: Adaptive resource allocation for cloud computing environments under bursty workloads. In: *Performance Computing and Communications Conference (IPCCC), 2011 IEEE 30th International, 2011*. [S.l.: s.n.]. p. 1–8. ISSN 1097-2641.

TELOVATIONS. *Now that you're in the Cloud, what's it like up here?* [S.l.], Fevereiro 2012. Disponível em: <http://telovations.wordpress.com/tag/how-cloud-computing-works/>.

VALENTINI, Giorgio Luigi et al. An overview of energy efficiency techniques in cluster computing systems. *Cluster Computing, 2011*. Disponível em: <http://www.springerlink.com/index/10.1007/s10586-011-0171-x>.

VERDI, Fábio Luciano et al. Novas arquiteturas de data center para cloud computing. In: *SBRC Minicursos, 2010*. [S.l.: s.n.].

VOORSLUYS, William; BROBERG, James; BUYYA, Rajkumar. *Cloud Computing: Principles and Paradigms*. New York, USA: Wiley Press, 2011. ISBN 978-0470887998.

VSPHERE, VMware. *VMware vSphere Hypervisor^{TM} (ESXi)*. [S.l.], Julho 2011. Disponível em: <http://www.vmware.com/products/vsphere-hypervisor/overview.html>.

WANG, Lizhe et al. Towards thermal aware workload scheduling in a data center. In: *Proceedings of the 2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks, 2009*. Washington, DC, USA: IEEE Computer Society. (ISPAN '09), p. 116–122. ISBN 978-0-7695-3908-9. Disponível em: <http://dx.doi.org/10.1109/I-SPAN.2009.22>.

WANG, Qingling; VARELA, C.A. Impact of cloud computing virtualization strategies on workloads' performance. In: *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*. [S.l.: s.n.], 2011. p. 130–137.

WIKIPEDIA. *Data center*. [S.l.], Janeiro 2013. Disponível em: <http://en.wikipedia.org/wiki/Data_center>.

WORKSTATION, VMware. *VMware Workstation*. [S.l.], Janeiro 2013. Disponível em: <http://www.vmware.com/products/workstation/overview.html>.

XEN. *Xen Architecture Overview*. [S.l.], Fevereiro 2008. Disponível em: <http://wiki.xen.org/xenwiki/XenArchitecture>.

XING, Yuping; ZHAN, Yongzhao. Virtualization and cloud computing. In: ZHANG, Ying (Ed.). *Future Wireless Networks and Information Systems, 2012*. [S.l.]: Springer Berlin Heidelberg, (Lecture Notes in Electrical Engineering, v. 143). p. 305–312.

ZECCHIN, Aaron C. et al. Application of two ant colony optimisation algorithms to water distribution system optimisation. *Mathematical and Computer Modelling, 2006*, v. 44, p. 451 – 468.

ZHANG, Hui et al. Intelligent workload factoring for a hybrid cloud computing model. In: *World Conference on Services - I, 2009*. [S.l.: s.n.]. p. 701–708.

ZHANG, Qi; CHENG, Lu; BOUTABA, Raouf. Cloud computing: state-of-the-art and research challenges. 2010.