UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE
UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

JOCKSAM GONÇALVES DE MATOS

META-HEURÍSTICA E ALGORITMO ESTÁTICO PARA O ESCALONAMENTO DE TAREFAS EM COMPUTAÇÃO NAS NUVENS

MOSSORÓ - RN
2015

JOCKSAM GONÇALVES DE MATOS

META-HEURÍSTICA E ALGORITMO ESTÁTICO PARA O ESCALONAMENTO DE TAREFAS EM COMPUTAÇÃO NAS NUVENS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação - associação ampla entre a Universidade do Estado do Rio Grande do Norte e a Universidade Federal Rural do Semi-Árido, para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Profº Carla Katarina de Monteiro Marques, Dra.
Coorientador: Profº Carlos Heitor Pereira Liberalino, Dr.

MOSSORÓ - RN
2015

Catalogação da Publicação na Fonte.
Universidade do Estado do Rio Grande do Norte.

Matos, Jocksam Gonçalves de
Meta-heurística e algoritmo estático para o escalonamento de tarefas em computação nas nuvens. / Jocksam Gonçalves de Matos. – Mossoró, RN, 2015.

60 f.
Orientador: Profª. Drª. Carla Katarina de Monteiro Marques.

Dissertação (Mestrado em Ciência da Computação) Universidade do Estado do Rio Grande do Norte. Programa de Pós-Graduação em Ciência da Computação.

1. Ciencia da Computação. 2. Computação nas Nuvens. 3. Escalonamento de Tarefas. 4. Meta-Heurística. 5.Algoritmo Estático. 6.CloudSim . I. Marques, Carla Katarina de Monteiro. II.Universidade do Estado do Rio Grande do Norte. III. Universidade Federal Rural do Semi-Árido. IV. Título.

UERN/BC CDD 004

Bibliotecária: Elaine Paiva de Assunção CRB 15 / 492

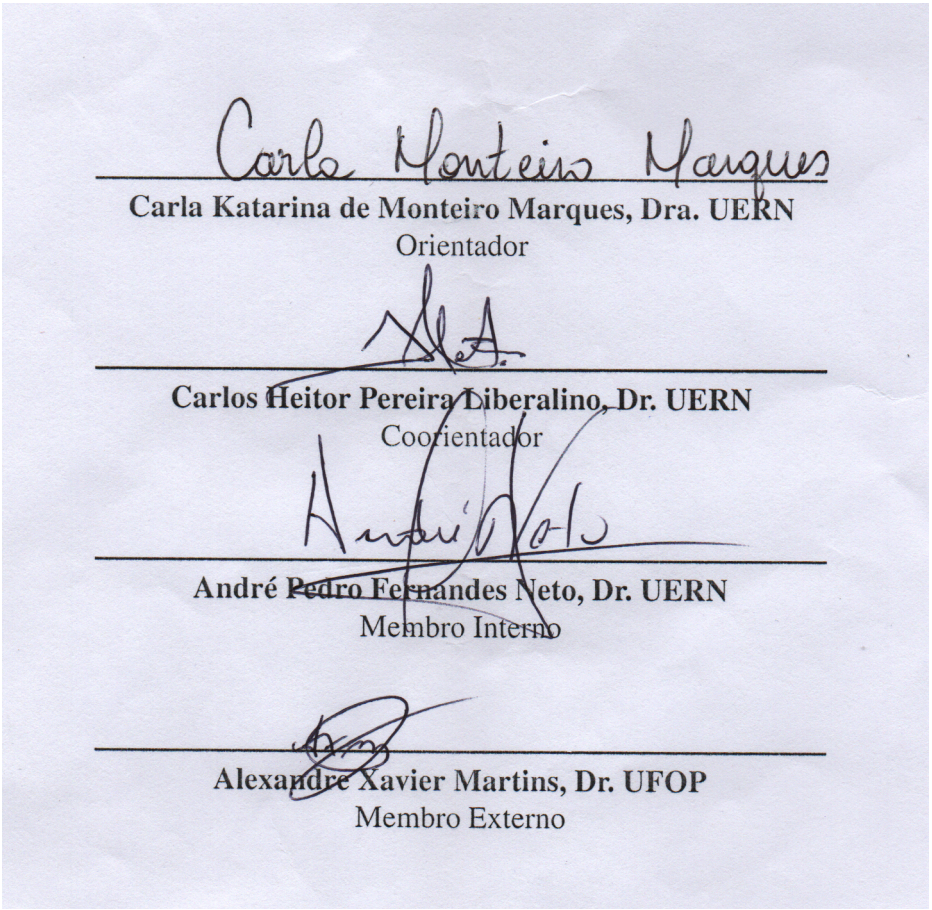
JOCKSAM GONÇALVES DE MATOS

META-HEURÍSTICA E ALGORITMO ESTÁTICO PARA O ESCALONAMENTO DE TAREFAS EM COMPUTAÇÃO NAS NUVENS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do título de Mestre em Ciência da Computação.

APROVADA EM: ____ de _____ de _____.

BANCA EXAMINADORA



Carla Katarina de Monteiro Marques, Dra. UERN
Orientador

Carlos Heitor Pereira Liberalino, Dr. UERN
Coorientador

André Pedro Fernandes Neto, Dr. UERN
Membro Interno

Alexandre Xavier Martins, Dr. UFOP
Membro Externo

Dedico este trabalho à minha família, pois nas incertezas e dificuldades da vida, serão sempre os que estarão ao meu lado.

AGRADECIMENTOS

Agradeço a Deus pela oportunidade e por iluminar meu caminho durante esta caminhada.

A minha mãe Urânia, que durante todo o tempo pensou em fazer o melhor por seus filhos, tornando-se ainda mais essencial na minha vida.

Agradeço aos meus avós, Dagoberto e Silvandira que são exemplos de seres humanos, pois sempre lutaram e torceram pelo meu sucesso.

As minhas tias, Mônica e Cleuma, por sempre me ajudarem. Em especial agradeço a minha tia Maria Dulce, pela sua dedicação em sempre me ajudar. A meus tios, em especial a meu tio Arnaldo pelo grande incentivo. A minha irmã Graziella por sempre torcer pelo meu sucesso. Com certeza sem vocês eu não estaria aqui.

Agradeço a minha namorada Jaiene Santos, que sempre me deu forças para que eu pudesse concluir esse mestrado, principalmente nas dificuldades e incertezas que encontrei na minha vinda a Mossoró. Muito obrigado por sempre estar ao meu lado.

Um muito obrigado a minha orientadora Dra. Carla Katarina de Monteiro Marques e meu Co-Orientador Dr. Carlos Heitor Pereira Liberalino, pelo aprendizado e dedicação durante o tempo que trabalhamos juntos, e também pela compreensão e profissionalismo.

Aos amigos pelo incentivo e companheirismo durante o tempo que convivemos juntos cursando o mestrado em Ciência da Computação e principalmente ao meu grupo de pesquisa.

E finalmente, agradeço a todos que me ajudaram direto ou indiretamente nesta caminhada. Um muito obrigado a todos vocês!

“A informática está interligada ao mundo sobre as reações intergalaxias!”
Bill Gates

RESUMO

O avanço tecnológico tem exigido cada vez mais recursos computacionais. Nesse contexto a computação nas nuvens surge como um novo paradigma para atender essa demanda, no entanto seus recursos são fisicamente limitados em função do crescente tráfego de dados que esse sistema pode estar sujeito. O escalonamento de tarefas tem como objetivo distribuir as tarefas de modo a torná-las mais eficientes no consumo de recursos computacionais. Desta forma, este trabalho tem como finalidade propor uma solução para o problema de escalonamento de tarefas em computação nas nuvens, de forma que reduza o tempo de processamento das tarefas e o número de máquinas virtuais. Tal algoritmo foi projetado a partir de solução heurística com auxílio de um algoritmo estático. O algoritmo proposto foi inspirado principalmente no problema de particionamento de conjuntos que tem como objetivo reduzir o número de máquinas virtuais. A meta-heurística algoritmo genético foi utilizada na primeira fase do algoritmo, como forma de reduzir o tempo de processamento das tarefas (*makespan*). O algoritmo estático foi projetado para resolver o particionamento de conjuntos. Seu desempenho foi comparado com dois algoritmos, clássico e heurístico. O CloudSim, um simulador de computação nas nuvens que possui características e atributos de uma nuvem real foi utilizado como forma de avaliar o algoritmo proposto, juntamente com cargas de trabalho realistas em experimentos que evidenciaram o comportamento do algoritmo em diferentes condições de uso.

Palavras-chave: Computação nas Nuvens. Escalonamento de Tarefas. Meta-Heurística. Algoritmo Estático. CloudSim.

ABSTRACT

Technological advancement has required ever more computing resources. In this context the cloud computing emerges as a new paradigm to meet this demand, though its resources are physically limited due to the growing data traffic that the system may be subject. The task scheduling aims to distribute tasks in order to make them more efficient in the use of computing resources. Thus, this paper aims to propose a solution to the task scheduling problem in cloud computing in order to reduce the processing time of the tasks and the number of virtual machines. This algorithm was designed from heuristic solution with the aid of a static algorithm. The proposed algorithm was mainly inspired by the partitioning problem sets that aims to reduce the number of virtual machines. The meta-heuristic genetic algorithm was used in the first stage of the algorithm, in order to reduce the processing time of the tasks. The static algorithm is designed to solve the partitioning sets. Their performance was compared with two algorithms, classic and heuristic. The CloudSim, a computer simulator in the cloud that has characteristics and attributes of a real cloud was used as a way to evaluate the proposed algorithm, along with realistic workloads in experiments that showed the algorithm's behavior under different conditions of use.

Key-words: Cloud Computing. Scheduling Tasks. Meta-heuristics. Static algorithm. CloudSim.

LISTA DE FIGURAS

Figura 1 – Modelo de resolução proposto. Fonte: Autoria própria 15
Figura 2 – Modelo de serviço na nuvem. (SOUSA; MOREIRA; MACHADO, 2009). . 18
Figura 3 – Visão geral de uma nuvem. (CLOUD, 2015) 19
Figura 4 – Funcionamento geral de um escalonador de tarefas nas nuvens. Fonte: autoria própria 20
Figura 5 – Taxonomia para o problema de escalonamento proposta por Casavant e Kuhl (1988). Adaptado de (DANTAS, 2005) 21
Figura 6 – Visão geral do funcionamento do AG (COLE, 1998). 26
Figura 7 – Função aptidão. Fonte: Autoria própria. 28
Figura 8 – Arquitetura CloudSim. (CALHEIROS et al., 2011) 31
Figura 9 – Funcionamento do modelo matemático no ambiente de computação nas nuvens. Fonte: Adaptada do modelo matemático 36
Figura 10 – Representação de um indivíduo. Fonte: Autoria própria. 38
Figura 11 – Operador *crossover*. Fonte: Autoria própria. 39
Figura 12 – Novos indivíduos por *crossover*. Fonte: Autoria própria. 39
Figura 13 – Fluxograma Algoritmo Genético. Fonte: Autoria própria. 40
Figura 14 – Fluxograma do escalonador de tarefas EAGRVMs em sua primeira fase. Fonte: autoria própria. 41
Figura 15 – Fluxograma de funcionamento do algoritmo estático *Maximum Resource*. Fonte: Autoria própria 42
Figura 16 – Processo de escalonamento realizados através do *Maximum Resource*. Fonte: autoria própria. 43
Figura 17 – Processo geral do escalonamento realizado através do EAGRVMs. Fonte: autoria própria 43
Figura 18 – Gráfico de valores do tempo total da simulação do CloudSim comparando o AG aos algoritmos em *Timespan* em segundos - LPC-EGEE 47
Figura 19 – Gráfico de valores do tempo total da simulação do CloudSim comparando o AG aos algoritmos em *Timespan* em segundos - UNILU GAIA 48
Figura 20 – Gráfico de valores do tempo total da simulação do CloudSim comparando o AG aos algoritmos em *Timespan* em segundos - CEA CURIE 49
Figura 21 – Gráfico de valores do tempo total da simulação do CloudSim comparando o *Maximum Resource* aos algoritmos em *Timespan* em segundos - LPC-EGEE 50
Figura 22 – Gráfico de valores do tempo total da simulação do CloudSim comparando o *Maximum Resource* aos algoritmos em *Timespan* em segundos - UNILU GAIA 51
Figura 23 – Gráfico de valores do tempo total da simulação do CloudSim comparando o *Maximum Resource* aos algoritmos em *Timespan* em segundos - CEA CURIE 52

LISTA DE TABELAS

Tabela 1 – Definições de termos dos algoritmos genéticos 38
Tabela 2 – Configuração do ambiente simulado 46
Tabela 3 – Comparação de algoritmos segundo valores do tempo total da simulação do CloudSim - LPC-EGEE. 47
Tabela 4 – Comparação de algoritmos segundo valores do tempo total da simulação do CloudSim - UNILU GAIA. 48
Tabela 5 – Comparação de algoritmos segundo valores do tempo total da simulação do CloudSim - CEA CURIE. 49

LISTA DE ABREVIATURAS E SIGLAS

CEA organismo de investigação tecnológica francesa financiada pelo governo, página 46

CPU Unidade central de processamento, página 38

DAG *Directed acyclic graph*, página 38

EAGRVMs Escalonador de tarefas baseado em Algoritmo Genético para Redução do uso de Máquinas Virtuais, página 40

ECT *Estimated Completion Time*, página 38

FIFO *First In First Out*, página 37

HEFT *Heterogeneous Earliest Finish Time*, página 38

LOIA Laboratório de Otimização e Inteligência Artificial, página 31

LPC Laboratório de Física Corpuscular, página 46

MR Algoritmo estático proposto, página 42

PSO Algoritmo por enxame de partícula, página 47

SPP Problema de Particionamento de Conjuntos, página 14

SWF formato padrão de carga de trabalho, página 46

TI Tecnologia da Informação, página 13

UERN Universidade Do Estado do Rio grande do Norte, página 31

Unilu Universidade do Luxemburgo, página 46

VM Máquina Virtual, página 14

WQ *Workqueue*, página 39

WQR *Workqueue with Replication*, página 39

SUMÁRIO

1 INTRODUÇÃO 12
1.1 CONTEXTUALIZAÇÃO 12
1.2 PROBLEMÁTICA 14
1.3 OBJETIVO 15
1.4 METODOLOGIA 15
1.5 ORGANIZAÇÃO DA DISSERTAÇÃO 16
2 ASPECTOS CONCEITUAIS 17
2.1 COMPUTAÇÃO NAS NUVENS 17
2.2 ESCALONAMENTO DE TAREFAS 20
2.2.1 TAXONOMIA DOS ALGORITMOS DE ESCALONAMENTO . . 21
2.2.2 ALGORITMOS DE ESCALONAMENTO NA LITERATURA . . . 22
2.3 PARTICIONAMENTO DE CONJUNTOS 24
2.4 ALGORITMOS GENÉTICOS 25
2.4.1 TERMINOLOGIA DOS ALGORITMOS GENÉTICOS 27
2.4.2 ESTRUTURA DE FUNCIONAMENTO DO ALGORITMO GENÉTICO 28
2.5 CLOUDSIM 30
2.5.1 ARQUITETURA 31
3 TRABALHOS RELACIONADOS 32
4 ALGORITMO DE ESCALONAMENTO DE TAREFAS PARA COMPUTAÇÃO NAS NUVENS 34
4.1 MODELAGEM MATEMÁTICA APLICADA AO PROBLEMA 34
4.2 POLÍTICA DE ESCALONAMENTO BASEADA EM ALGORITMO GENÉTICO 37
4.3 ALGORITMO ESTÁTICO PROPOSTO 41
5 VALIDAÇÃO E RESULTADOS 45
5.1 FLUXOS DE TRABALHO UTILIZADOS 45
5.2 AMBIENTE DE TESTE 45
5.3 RESULTADOS EXPERIMENTAIS 46
5.3.1 RESULTADOS OBTIDOS ATRAVÉS DO ALGORITMO GENÉTICO 46
5.3.2 RESULTADOS OBTIDOS ATRAVÉS DO ALGORITMO ESTÁTICO 50
5.4 ANÁLISE DO DESEMPENHO 52
6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS 54
6.1 TRABALHOS FUTUROS 54
REFERÊNCIAS 56

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

À medida que a tecnologia avança, surge uma demanda cada vez maior por poder computacional. Um dos fatores que contribui significativamente para o aumento do uso de recursos computacionais são as redes de computadores, que nos tempos atuais tem sido algo essencial nas atividades cotidianas. Dentro desse contexto, a computação nas nuvens *(Cloud Computing)* surge como um novo modelo computacional que representa a computação distribuída (GROSSMAN, 2009).

A computação nas nuvens pode ser considerada como um sistema distribuído que oferece serviços de computação através da Internet. Os usuários não precisam se preocupar com a compra de hardwares e softwares, os quais requerem instalação, configuração e manutenção. Computação nas nuvens oferece infra-estrutura, plataforma e software (aplicativo) como serviços e os usuários pagam apenas pelos recursos consumidos (MURUGESAN, 2011), (SADIKU; MUSA; MOMOH, 2014).

Com isso, a computação nas nuvens tem transferido a entrega dos serviços de TI (Tecnologia da Informação) aos seus usuários, para um novo patamar, como acontece com os tradicionais utilitários tais como água e eletricidade, ou seja, o usuário paga apenas o que foi consumido. As vantagens dessa plataforma, tais como a relação custo eficácia, escalabilidade e facilidade de gerenciamento, incentivam cada vez mais empresas e prestadores de serviços a adotar a plataforma em nuvens, e a oferecer suas soluções através de modelos nas Nuvens (DASTJERDI; BUYYA, 2012).

A computação nas nuvens é baseada em serviços por demanda (*on-demand*), isso significa que somente é pago os recursos computacionais consumidos. Esse modelo oferece várias vantagens, em primeiro lugar, isso inclui a redução de despesas de capital, já que não seria necessário, grandes investimentos em infraestruturas de TI. Em segundo lugar, os serviços nas nuvens desfrutam as mesmas economias em escala que centros de dados fornecem com a prestação de serviços, é possível fornecer operações, continuidade de negócios e nível de segurança (GROSSMAN, 2009).

O custo unitário para serviços baseados em computação nas nuvens é muitas vezes inferior ao custo de serviços que foram prestados diretamente pelas organização em si. Finalmente, a arquitetura da computação nas nuvens prova ser escalável, por exemplo, serviços de armazenamento baseados em nuvens podem gerenciar facilmente um petabyte de dados, considerando este mesmo volume de dados em um banco de dados tradicional tem-se uma grande problemática. (GROSSMAN, 2009)

Como discutido anteriormente, a computação nas nuvens traz inúmeras vantagens. Em

vista disso, há uma demanda cada vez maior por poder computacional, já que existe cada vez mais a vinda de usuários e consequentemente o aumento do fluxo de dados. No entanto, a computação nas nuvens também levanta importantes desafios em muitos aspectos da computação científica, incluindo seu desempenho, que é o foco deste trabalho. Torna-se evidente a importância do escalonamento de tarefas eficiente nesse ambiente.

Considerando a migração do grande número de usuários para a computação nas nuvens, na tentativa de se obter mais recursos computacionais para que no futuro não haja problemas de escassez, o objetivo desse trabalho é otimizar de forma mais eficiente possível, o problema de escalonamento de tarefas em computação nas nuvens. Deste modo, fazer com quer as máquinas virtuais (*virtual machine* - VM) , executem as tarefas o mais rápido possível. Para isso, esse trabalho analisa o tempo de envio, processamento e reposta das tarefas.

Tendo em vista toda essa concepção, o presente trabalho propõe a criação de um algoritmo que reduza o tempo de processamento de tarefas nas nuvens, tal algoritmo é inspirado no problema de particionamento de conjuntos (*set partitioning problem* - SPP) que tem como objetivo reduzir o número de máquinas virtuais e, resolvido através do uso de Meta-Heurística (Algoritmo genético) com o auxílio da técnica presente no algoritmo estático aqui proposto para reduzir o número de máquinas virtuais, para que dessa forma seja possível resolver o problema de escalonamento de tarefas em computação nas nuvens.

Esse trabalho também propõem o uso do simulador CloudSim como forma de avaliar o algoritmo proposto e comparar sua eficiência com outros algoritmos presentes na literatura, isso servirá de base para que outros pesquisadores também possam usar o resultado dessa pesquisa como parâmetros para novas propostas. Outro ponto importante nessa pesquisa é o uso de um modelo matemático projetado especificamente parar tratar o problema de escalonamento de tarefas em sistemas distribuídos. Esse modelo matemático, tem como objetivo avaliar o tempo de execução das tarefas, desde o seu envio até o seu processamento e resposta.

Além de reduzir o tempo de processamento das tarefas executadas nas VMs (chamado *makespan*) essa pesquisa também tenta reduzir em paralelo o número de VMs para a execução das tarefas. Para isso, será introduzida uma técnica definida como sendo um algoritmo estático que soluciona o problema de particionamento de conjuntos.

Para avaliar o desempenho do algoritmo proposto, o mesmo foi comparado com outros dois algoritmos, um clássico que são considerados algoritmos de fácil implementação e possuem um modelo matemático relativamente simples e um heurístico. Além disso, os algoritmos fizeram uso de cargas de trabalhos reais em experimentos que evidenciem o comportamento do algoritmo em diferentes condições de uso.

Por fim, esse trabalho faz uso da meta-heurística algoritmo genético para resolver o possível problema eminente de escassez de recursos em sistemas distribuídos, mas especificamente na computação nas nuvens, diminuindo assim, o tempo de processamento das tarefas executadas nas nuvens.

1.2 PROBLEMÁTICA

A computação nas nuvens oferece seus recursos de forma ilimitada, porém esses recursos são fisicamente limitados em função do crescente aumento do trafego de dados, discutido na seção anterior. Com isso há uma preocupação crescente de que o desempenho dessa arquitetura deve ser avaliado, de modo a tornar esse modelo computacional mais reativo e sensível a novos serviços, já que como é esperado, esse paradigma computacional irá enfrentar mais intensidade de tráfego em um futuro cada vez mais próximo.

Desde a década de 1990, houve um aumento crescente pelo uso de recursos computacionais necessários para execução de aplicações. A computação nas nuvens surge como forma de atender essa demanda. Por isso há uma grande preocupação com a escassez de recurso computacional nesse ambiente (BUYYA; ABRAMSON; VENUGOPAL, 2005).

Devido ao aumento por uso de recursos computacionais um bom escalonador de tarefas pode melhorar ainda mais a questão do uso eficiente dos recursos em uma nuvem. Segundo Johnson e Garey (1979) o escalonamento de tarefas é um problema NP-Completo, dessa forma o uso de heurísticas tem sido recomendado para tratá-lo de forma mais eficiente possível Barroso et al. (1997). Esse trabalho faz uso da meta-heurística algoritmo genético, que prova ser eficiente para o problema de escalonamento de tarefas.

Um problema de escalonamento consiste em alocar um conjunto de n tarefas $T = \{ t0, t1,.....,Tn\text{-}1 \}$para serem executadas em um conjunto de m VMs $M = \{ m0, m1,.....,Mm\text{-}1 \}$. Para cada tarefa é designada uma VM para processá-la. Esse trabalho leva em consideração o tempo de envio, processamento e resposta das tarefas.

A escassez bibliográfica no que concerne em um único ambiente, o (i) escalonamento de tarefas em computação nas nuvens, (ii) algoritmo genético, (iii) algoritmo estático, (iv) problema de particionamento de conjuntos e (v) redução de máquinas virtuais, abrindo assim um vasto espaço para pesquisa. Entretanto, no Capítulo 3 são apontados alguns trabalhos relacionados com propostas semelhantes que influenciaram o desenvolvimento do trabalho proposto.

Tendo em vista toda essa problemática, este trabalho trata do problema de escalonamento de tarefas, onde as principais preocupações são a melhoria na utilização dos recursos e a escassez de recursos computacional em um futuro cada vez mais próximo. Além disso, esse trabalho tenta reduzir o custo com recursos computacionais tanto para provedores nas nuvens como para clientes nas nuvens. Na literatura em geral, os trabalhos tratam principalmente o problema do aproveitamento energético, assim não se preocupam com o tempo de execução das tarefas, consequentemente tem-se o custo elevado de recurso computacional. Há também pesquisas que tratam apenas o tempo de execução das tarefas (*makespan*). Apesar do foco dessa pesquisa estar voltada principalmente para o tempo de execução das tarefas, há também uma preocupação em tentar reduzir o número de máquinas virtuais necessárias.

1.3 OBJETIVO

Um dos problemas mais desafiadores da computação paralela e distribuída é conhecido como problema de escalonamento de tarefas. O objetivo do escalonamento de tarefas é determinar uma atribuição de tarefas a elementos de processamento com o intuito de otimizar certos índices de desempenho, como tempo processamento (EL-REWINI; ABD-EL-BARR, 2005).

O presente trabalho tem como objetivo geral executar o escalonamento de tarefas em computação nas nuvens e a redução do uso de máquinas virtuais inspirado no problema de particionamento de conjuntos, que consiste em dividir um problema em *n* subconjuntos à serem tratados.

O objetivo específico é tentar diminuir o tempo de processamento das tarefas com o uso do algoritmo genético, a partir daí, tendo como ponto de referência o resultado encontrado, tentar diminuir o número de máquinas virtuais necessárias para executar as tarefas em um limite de tempo aceitável fazendo uso do algoritmo estático para resolver o problema de particionamento de conjuntos.

O problema do particionamento de conjuntos tem como objetivo reduzir o número de máquinas virtuais necessárias para a execução das tarefas, por isso o uso do algoritmo estático. É importante entender, que o algoritmo estático se baseia no resultado encontrado no algoritmo genético. Ele tenta reduzir o número de máquinas virtuais ao ponto em que o limite de tempo de execução seja aceitável baseado no resultado encontrado no algoritmo genético.

1.4 METODOLOGIA

A Figura 1 mostra como se dá o processo de resolução da pesquisa.

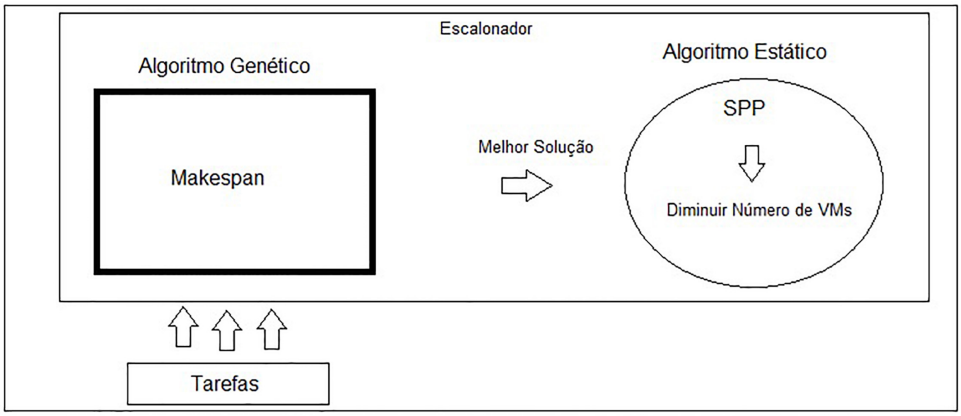


Figura 1 – Modelo de resolução proposto. Fonte: Autoria própria

A Figura 1, demonstra as tarefas sendo recebidas e escalonadas pelo algoritmo genético, em seguida o AG analisa a melhor solução, ou seja, o tempo com menor *makespan*. Logo

após, com base no tempo de execução encontrado no AG, o algoritmo estático tenta resolver o problema do SPP para tentar reduzir o número de VMs necessárias para executar as mesmas tarefas escalonadas no AG. No entanto, o algoritmo estático tem como base encontrar um valor aproximado ou melhor daquele encontrado no AG. Dessa forma, como será discutido na seção 4.3, dependendo da taxa que o algoritmo estático estiver trabalhando o restante das tarefas vão estar otimizadas por conta do AG.

1.5 ORGANIZAÇÃO DA DISSERTAÇÃO

O restante desta dissertação está organizado da seguinte forma. O capítulo 2 está dividido em cinco seções que apresentam, respectivamente, os conceitos de Computação nas Nuvens, Escalonamento de Tarefas, Problema de Particionamento de Conjuntos, Algoritmos Genéticos e o Simulador CloudSim. O capítulo 3 apresenta alguns Trabalhos Relacionados. No capítulo 4 o Escalonador Baseado em Meta-Heurística e Algoritmo Estático. O Capítulo 5 trata da validação e dos resultados obtidos com o trabalho. E, por fim, o Capítulo 6 apresenta as considerações finais e os trabalhos futuros.

2 ASPECTOS CONCEITUAIS

Este capítulo apresenta o referencial teórico abordado no presente trabalho. A Seção 2.1 discute assuntos relevantes e conceitos sobre computação nas nuvens. A seção 2.2 explica o problema de escalonamento de tarefas. Na Seção 2.3 encontram-se informações sobre o problema de particionamento de conjuntos. Enquanto a Seção 2.4 contempla informações que definem o algoritmo genético. Por fim, a Seção 2.5 fala sobre o simulador cloudsim utilizado no trabalho.

Este capítulo auxiliará na absorção de definições a cerca das técnicas e recursos utilizados na abordagem aqui apresentada e, levanta questões sobre o estado da arte.

2.1 COMPUTAÇÃO NAS NUVENS

Apesar de muitas pessoas apresentarem a computação nas nuvens como uma grande novidade, o conceito é quase tão antigo quanto o próprio computador. A principal funcionalidade da Computação nas nuvens é a utilização de recursos computacionais por meio da internet, e essa ideia não é tão recente como muitos pensam. Por incrível que pareça, a ideia já existia em 1960, com Joseph Carl Robnett Licklider que foi um dos desenvolvedores da ARPANET (*Advanced Research Projects Agency Network*), o antecessor direto da internet, que tinha como objetivo de interligar as bases militares e os departamentos de pesquisa do governo americano (ALMEIDA; CASTRO; FILHO, 2013).

O termo computação nas nuvens foi introduzido em 2006, quando o então CEO (*Chief Executive Officer* - Diretor Executivo) da Google, Eric Schmidt, utilizou o termo para descrever os serviços da própria empresa, e, pouco tempo depois, quando a Amazon utilizou o mesmo termo para lançar seu serviço EC2 (*Elastic Compute Cloud*). Foi, na verdade popularizado através de um artigo na edição de Outubro da Wired, através do artigo de George Gilder entitulado *"The Information Factories"* (GILDER, 2006), (ALMEIDA; CASTRO; FILHO, 2013).

A computação nas nuvens denota a infra-estrutura como uma *"nuvem"*, no qual as empresas e os usuários são capazes de acessar aplicativos de qualquer lugar do mundo sob demanda. Computação nas Nuvens oferece infra-estrutura, plataforma e software (aplicativo) como serviços, que são disponíveis como serviços baseado em um modelo *"pré-pago"* para os consumidores. Estes serviços na indústria são referidos respectivamente como infra-estrutura como um serviço *(infrastructure as a service - Iaas)* , a plataforma como serviço *(platform as a service - PaaS)* e Software como serviço *(Software as a service - SaaS)* (BUYYA et al., 2009).

A Figura 2 mostra uma visão geral de modelo de serviço em uma nuvem.

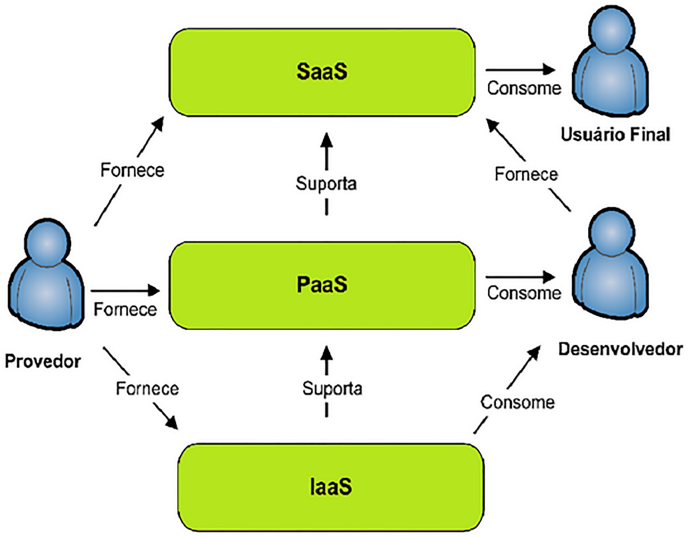


Figura 2 – Modelo de serviço na nuvem. (SOUSA; MOREIRA; MACHADO, 2009).

Os serviços de computação nas nuvens são classificados quanto à restrição de acesso aos usuários, nas seguintes categorias (VERAS; TOZER, 2012):

Nuvens Públicas: a ideia central da nuvem pública é permitir que as organizações executem boa parte dos serviços que hoje são executados em *Data Centers* corporativos em *Data Centers* na rede, providos por terceiros.

Nuvens Privadas: este modelo de nuvem é voltado exclusivamente a apenas um usuário que tem o total controle sobre a mesma, que gerencia as aplicações implementadas na nuvem, implementa políticas e níveis de acesso, garantindo a integridade dos dados. Em geral uma nuvem privada pode ser definida sobre a ideia que os dados e aplicações fazem parte de uma única organização.

Nuvens Híbridas: Formada por nuvens públicas e privadas. Elas ampliam os recursos das nuvens privadas permitindo a utilização de recursos em caso de necessidade. Operações que tem picos de utilização podem fazer o uso das nuvens híbridas sem a necessidade de investimentos na capacidade total.

Segundo Taurion (2009) a computação nas nuvens é um termo para descrever um ambiente de computação baseado em uma rede de servidores, sejam virtuais ou físicos. Uma definição simples pode ser "*um conjunto de recursos como capacidade de processamento, armazenamento, conectividade, plataformas, aplicações e serviços disponibilizados na internet*". O resultado é que a computação nas nuvens pode ser vista como o estágio mais evoluído do conceito de virtualização, a virtualização do próprio data center.

Suas principais características são: disponibilidade de recursos infinitos, acessíveis sob demanda, elimina a necessidade de adquirir e aprovisionar recursos antecipadamente e oferece elasticidade, permitindo que usem os recursos na quantidade que forem necessários, de forma dinâmica.

A computação nas nuvens possui um grande nível de abstração, os usuários podem

adquirir unilateralmente recurso computacional, como tempo de processamento no servidor ou armazenamento na rede na medida em que necessite e sem precisar de interação humana com os provedores de cada serviço. O hardware e o software dentro de uma nuvem podem ser automaticamente reconfigurados, orquestrados e estas modificações são apresentadas de forma transparente para os usuários, que possuem perfis diferentes e assim podem personalizar os seus ambientes computacionais, por exemplo, instalação de software e configuração de rede para a definição de determinados privilégios.

A Figura 3 mostra uma visão geral de uma nuvem.

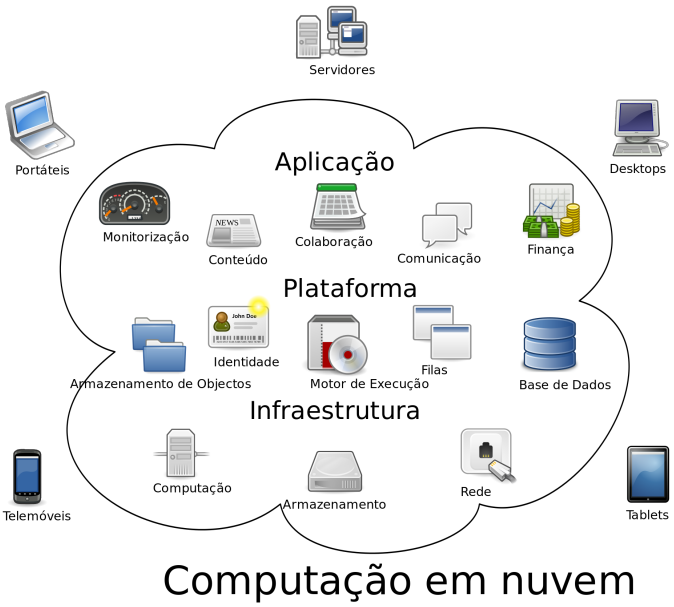


Figura 3 – Visão geral de uma nuvem. (CLOUD, 2015)

A Figura 3 exemplifica de modo geral a grande interatividade e diversidade que ocorre dentro da computação nas nuvens em diferentes ambientes.

Na computação nas nuvens, usuários podem acessar aplicações e dados de qualquer lugar e a qualquer hora, diminuir gastos como por exemplo: com infraestrutura, equipamentos, licenças de aplicativos proprietários, conta de luz e manutenção de equipamentos, capacidade do ambiente computacional da nuvem aumentar ou diminuir de forma automática os recursos computacionais demandados e provisionados para cada usuário (ALMEIDA; CASTRO; FILHO, 2013).

Como já visto, há muita vantagens no uso de computação nas nuvens e consequentemente um número cada vez maior de usuários que adotam esse sistema. Por isso a preocupação crescente de que esse sistema possa sofrer escassez de recurso computacional, motivo esse que influenciou a realização desse trabalho. Por esse motivo o escalonador de tarefa proposto tem como finalidade reduzir o tempo de processamento das tarefas e o número de máquinas virtuais, com isso, vai haver uma redução no consumo de recurso computacional e consequentemente a nuvem vai executar mais tarefas com os mesmos recursos computacionais utilizados antes da otimização

do escalonador. Dessa forma, o escalonador de tarefas deve melhorar a utilização de recursos computacionais reduzindo a manutenção, custos com hardware e os custos de energia.

2.2 ESCALONAMENTO DE TAREFAS

O problema do escalonamento de tarefas, considerando o contexto de computação nas nuvens, pode ser dividido segundo dois pontos de vista: do usuário da nuvem e do provedor da nuvem. Do ponto de vista do usuário, o escalonamento de tarefas tem como objetivo a redução do tempo de execução das tarefas. Dessa forma, o usuário terá um custo menor, pois o uso de recursos computacionais será inferior devido ao fato que, na computação nas nuvens o cliente paga pelo que consumiu Bittencourt e Madeira (2011). Por outro lado, do ponto de vista do provedor, um escalonador de tarefas deve melhorar a utilização de recursos computacionais reduzindo a manutenção e os custos de energia Dasgupta et al. (2011). Esse trabalho, considera os dois pontos de vista, já que, com a redução no tempo de processamento das tarefas, ambos os lados são beneficiados como justificado acima.

A Figura 4 exemplifica o funcionamento do escalonador dentro da nuvem.

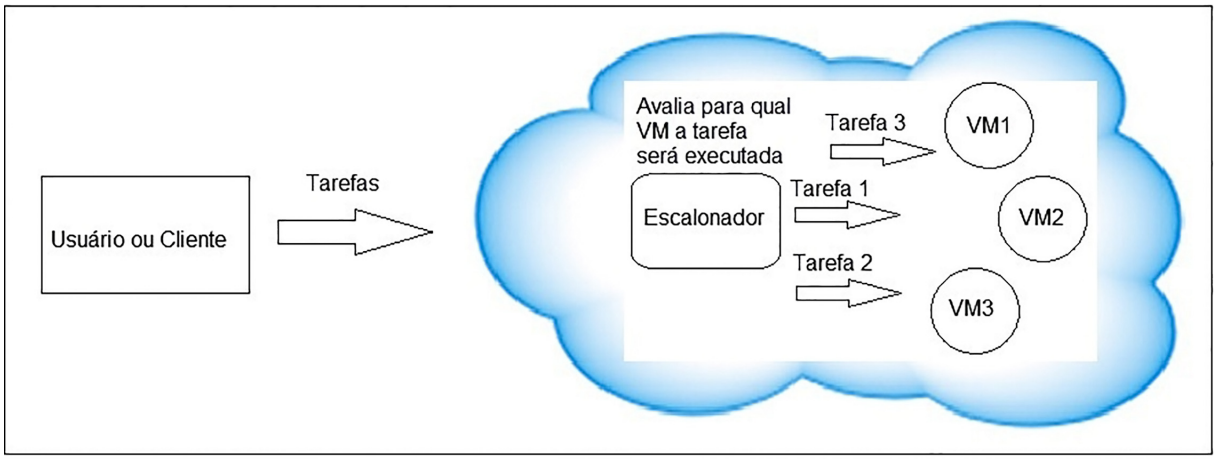


Figura 4 – Funcionamento geral de um escalonador de tarefas nas nuvens. Fonte: autoria própria

A Figura 4 mostra o usuário ou cliente enviado as tarefas para serem processadas por um servidor na nuvem, em seguida o escalonador avalia qual VM vai processar uma determinada tarefa. Essa avaliação faz parte do modelo matemático.

O escalonamento de tarefas, tem forte influência no desempenho de um ambiente nas nuvens e em outros sistemas distribuídos, por ser o responsável por determinar o momento em que cada tarefa será executada e quais recursos serão usados para isso. Esse tipo de técnica visa, em geral, otimizar objetivos como o de minimizar o tempo total necessário para a conclusão das tarefas. Para esse trabalho, como citado em capítulos anteriores, a principal preocupação é com a redução do tempo de processamento das tarefas, consequentemente, haverá uma diminuição no

tempo de uso de recursos computacionais. Dessa forma, os mesmos recursos estarão livres mais rápidos para executar outras tarefas.

2.2.1 TAXONOMIA DOS ALGORITMOS DE ESCALONAMENTO

Casavant e Kuhl (1988) propuseram uma classificação hierárquica dos algoritmos de escalonamento em sistemas computacionais distribuídos a Figura 5 ilustra esse modelo. De acordo com Borro (2014), nessa taxonomia, os algoritmos são primeiramente divididos como locais e globais. O escalonamento local envolve o mapeamento das fatias de tempo de uso de um único processador às tarefas residentes localmente. Por outro lado, o escalonamento global se refere ao problema de decisão de quais dos diversos recursos disponíveis as tarefas serão executadas. O problema de escalonamento em computação nas nuvens, obviamente, pertence ao ramo global, que pode ser subdividido em mais duas classes de escalonamento: estático e dinâmico.

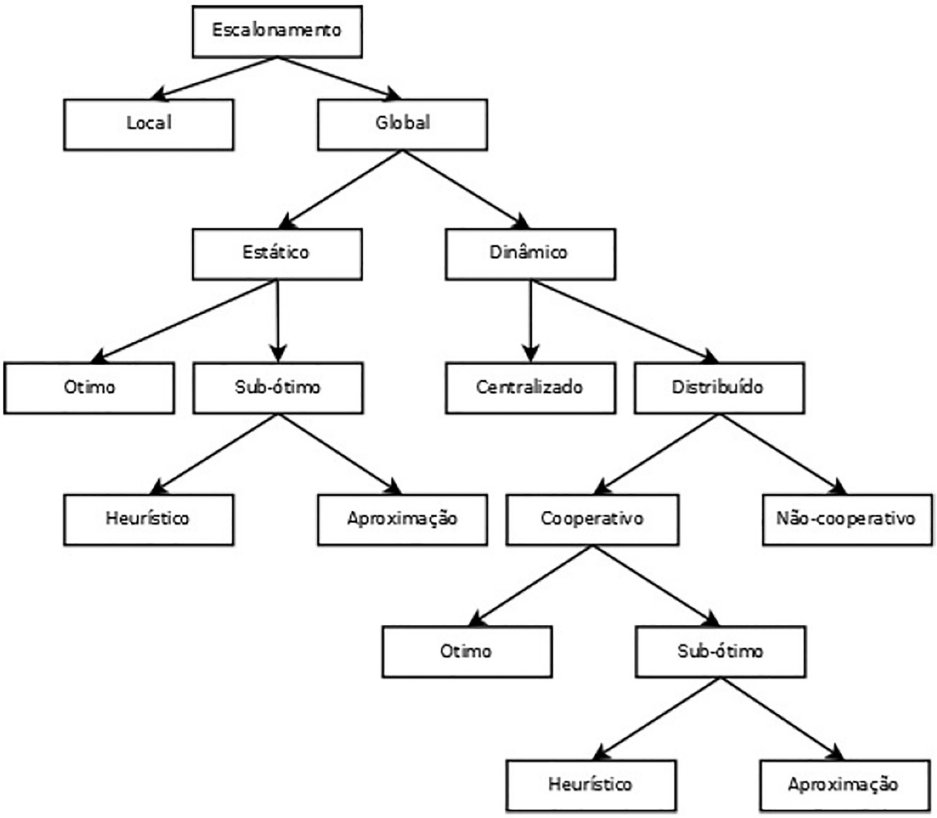


Figura 5 – Taxonomia para o problema de escalonamento proposta por Casavant e Kuhl (1988). Adaptado de (DANTAS, 2005)

No escalonamento estático, o mapeamento das tarefas é realizado antes da execução da aplicação, sendo, portanto, necessário o conhecimento antecipado de informações acerca dos recursos. Mesmo em caso de falha de algum recurso de grade, o mapeamento obtido não pode ser alterado em tempo de execução. Por outro lado, no escalonamento dinâmico, assume-se que

muito pouco se sabe sobres os recursos que compõem a nuvem, sendo o mapeamento das tarefas realizado durante o tempo de vida da aplicação. Devido à natureza dinâmica das nuvens, em geral, faz-se também necessária a redistribuição das tarefas entre os recursos computacionais (CASAVANT; KUHL, 1988), (BORRO, 2014).

Os algoritmos de escalonamento também podem ser classificados como ótimos e sub-ótimos. Um escalonamento ótimo pode apenas ser realizado quando se há conhecimento prévio de todas as informações relativas aos recursos e à aplicação. No entanto, a complexidade do problema de escalonamento em sistemas distribuídos juntamente com a dinamicidade inerente à computação nas nuvens torna praticamente impossível um mapeamento ótimo das tarefas Akl e Dong (2006). Por esse motivo, os algoritmos de escalonamento geralmente visam, por meio de aproximação ou heurísticas, obter resultados satisfatórios (BORRO, 2014).

Os algoritmos sub-ótimos são divididos em duas classes: aproximados e heurísticos. Em um algoritmo aproximado, reduz-se o espaço de busca do problema de escalonamento. Busca-se portanto, um mapeamento de tarefas suficientemente bom em relação a estimativa previamente definida. Já um algoritmo heurístico utiliza regras empíricas para orientar a busca por um mapeamento de boa qualidade (BORRO, 2014).

Quando implementado em uma abordagem dinâmica, os algoritmos de escalonamento podem ser empregados em um cenário centralizado, como também podem ser executados por múltiplos escalonadores distribuídos. A vantagem do cenário centralizado esta na facilidade de implementação. No entanto, em função de problemas de escalabilidade, o escalonamento dinâmico centralizado não é indicado para nuvens de grande escala Abraham e Xhafa (2010). Além disso, em caso de uma falha do escalonador, todo sistema é prejudicado (AKL; DONG, 2006), (ABRAHAM; XHAFA, 2010).

Para algoritmos de escalonamento dinâmicos e distribuídos, deve-se considerar mais um importante aspecto: a interação entre os diferentes escalonadores, que pode ser cooperativa ou independente (não cooperativa). No modo não-cooperativo, o escalonamento é feito localmente, não se preocupando com o desempenho da nuvem como um todo. Por outro lado, no modo cooperativo, as decisões locais de escalonamento são tomadas de forma coordenada, objetivando uma meta global (BORRO, 2014).

2.2.2 ALGORITMOS DE ESCALONAMENTO NA LITERATURA

Abaixo são citados alguns algoritmos presentes na literatura utilizados para o escalonamento de tarefas em sistemas distribuídos, como em computação nas nuvens e grades computacionais.

Essa seção é importante para que se entenda melhor os conceitos de algoritmos estáticos e dinâmicos, assim como suas diferenças. Pois esse trabalho faz uso de um algoritmo estático.

- **FIFO** (*Algoritmo dinâmico*)

O primeiro a entrar é o primeiro a sair, o FIFO é conhecido popularmente por algoritmo de fila simples. Nesse algoritmo, as tarefas são colocadas numa fila organizada por ordem de chegada. Na sua vez, cada tarefa recebe o uso da unidade central de processamento (*Central Processing Unit* - CPU) até que seja completado, ou seja, a tarefa permanece em execução até que seja finalizada, de forma que as demais tarefas na fila fiquem esperando por sua oportunidade de processamento assim sendo, o escalonamento FIFO é um algoritmo não preemptivo (WOODHULL; TANENBAUM, 1997).

O FIFO é considerado um algoritmo clássico, no entanto, como ele não fornece uma política justa de escalonamento, ele pode levar um grande tempo de resposta quando tem-se uma grande quantidade de tarefas a ser processadas.

Outro ponto é que tarefas importantes podem ficar à espera devido à execução de outras tarefas menos importantes dado que o escalonamento FIFO não concebe qualquer mecanismo de distinção entre as tarefas.

- **Workqueue** (*Algoritmo dinâmico*)

  Workqueue - WQ é um algoritmo de escalonamento que não usa qualquer informação sobre recursos para agendamento de tarefas. A primeira tarefa à espera de ser agendada é escolhida e um recurso livre é atribuído arbitrariamente para executá-lo. Este procedimento é repetido até que todas as tarefas programadas sejam executadas. A ideia por trás do WQ é que mais tarefas serão atribuídas aos recursos mais rápidos (SILVA; CIRNE; BRASILEIRO, 2003).

- **Workqueue With Replication** (*Algoritmo dinâmico*)

  Semelhante ao algoritmo WQ, O *Workqueue* com Replicação (*Workqueue with Replication* - WQR) , também não usa qualquer informação, entretanto, quando não há mais tarefas a serem executadas e ainda existem recursos ociosos, as tarefas que ainda estão em execução são replicadas nestes recursos ociosos, ou seja, eles também são executados nos outros recursos. Quando uma tarefa replicada termina, todas as outras réplicas serão paradas (SILVA; CIRNE; BRASILEIRO, 2003).

  A ideia desse algoritmo é a de que, quando uma tarefa é replicada, há uma possibilidade de que uma réplica seja atribuída a um nó mais rápido, aumentando assim a probabilidade de uma conclusão de forma mais rápida da tarefa. Este algoritmo apresenta um bom desempenho quando há recursos suficientes para a replicação das tarefas.

- **HEFT** (*Algoritmo estático*)

  Tempo heterogêneo mais cedo de conclusão, o algoritmo *Heterogeneous Earliest Finish Time* – HEFT é um algoritmo de escalonamento de fluxos de trabalho modelados utilizando um dígrafo acíclico (*Directed acyclic graph* – DAG) para um número limitado de computadores heterogêneos (KIM; HARIRI, 2001).

O HEFT é um algoritmo baseado em listas de tarefas. Ele faz uso de dados históricos para prever o desempenho das tarefas nos recursos computacionais e, a partir do desempenho previsto, realizar o escalonamento. Ele associa tarefas a prioridades atribuindo um peso a cada vértice e arco no DAG, ordenando-as em uma lista, de modo que as tarefas de maior prioridade ficam no início da lista e são escalonadas primeiro.

Devido à sua simplicidade e baixa complexidade, é considerado um dos algoritmos mais populares para escalonamento. Em uma pesquisa sobre HEFT em sites de busca durante o desenvolvimento desse trabalho, foi obtido mais de 58.000 resultados, por conta disso o HEFT é frequentemente utilizado como referência na literatura para comparar o desempenho de novas propostas, como por exemplo em (SHETTI; FAHMY; BRETSCHNEIDER, 2013), (NASONOV et al., 2014).

- **Min-Min** (*Algoritmo estático*)

  O *Min-Min* é uma heurística que se baseia na estimativa de Tempo de Conclusão (*Estimated Completion Time* - ECT) . Para cada tarefa disponível, é encontrado o menor tempo de conclusão ECT, ou seja, o algoritmo calcula o tempo de conclusão que cada tarefa gastaria para ser concluída em cada máquina. Em seguida, o algoritmo irá buscar o menor tempo de execução da tarefa e sua respectiva máquina para execução. Este processo é repetido até que todos as tarefas tenham sido concluídas (BLYTHE et al., 2005).

- **Min-Max** (Algoritmo estático)

  O *Max-Min* ao contrário do *Min-Min* é uma heurística que tem como objetivo alocar tarefas de maior carga para recursos que possuam maior capacidade de processamento. O algoritmo realiza uma busca procurando o conjunto de menor ECT de cada tarefa, ou seja, em qual máquina determinada tarefa pode ser executada em menor tempo como visto no *Min-Min*. Deste conjunto, é selecionado o maior ECT ou a tarefa com a maior carga. Finalmente, as tarefas vão sendo mapeadas uma a uma até que todas sejam executadas (BLYTHE et al., 2005).

2.3 PARTICIONAMENTO DE CONJUNTOS

Este trabalho trata especificamente do problema de escalonamento de tarefas em computação nas nuvens, no qual foi inspirado principalmente por diversas pesquisas que utiliza o problema de particionamento de conjuntos (*set partitioning problem* - SPP) no agendamento de tripulação como observados em vários trabalhos na literatura, como em (MINGOZZI et al., 1999).

O problema de particionamento de conjunto tem sido aplicado com sucesso para muitos problemas de escalonamento. Muitos pesquisadores observaram que é uma técnica muito

poderosa para resolver uma ampla gama de problemas para soluções próximas ou soluções aproximadas. Em Borovskiy et al. (2011), faz uso do problema de particionamento de conjuntos para diminuir o número de servidores necessários para executar grandes cargas de trabalhos na computação nas nuvens.

Aplicado ao contexto de computação nas nuvens, a ideia do SPP é utilizar o número menor possível de máquinas virtuais - VMs, para executar o maior número de tarefas. Vale ressaltar, que o SPP é um modelo de problema é não uma técnica. Esse problema será resolvido com o uso do algoritmo estático, proposto exclusivamente para resolvê-lo.

Sendo o objetivo dessa pesquisa diminuir o uso de recursos computacionais, o SPP servirá como base para tentar diminuir o número maior possível de máquinas virtuais necessárias para executar um conjunto de tarefas nas nuvens. Dessa forma, se temos um tempo X de execução para um conjunto de VMs, o objetivo do SPP é reduzir ou manter esse tempo X com um número de VMs reduzido.

É importante deixar claro a atuação do SPP nessa pesquisa, veja que o SPP atua como um objetivo secundário, ou seja, antes dele tentar reduzir o número de VMs, o algoritmo genético - AG fornece um resultado e, com base nesse resultado ele tenta diminuir o número de VMs, o resultado que o AG fornece, será um solução viável para o SPP, a partir daí ele vai atuar para encontrar um resultado aproximado ou até melhor se for o caso. A Figura 1 explica como se dá esse processo.

2.4 ALGORITMOS GENÉTICOS

Algoritmos Genéticos, *AGs*, são métodos de otimização e busca inspirados nos mecanismos de evolução de populações de seres vivos. Foram introduzidos por Holland (1975) e popularizados por um dos seus alunos Goldberg (1989). Estes algoritmos seguem o princípio da seleção natural e sobrevivência do mais apto, declarado em 1859 pelo naturalista e fisiologista inglês Charles Darwin em seu livro "A Origem das Espécies". De acordo com Charles Darwin, *"Quanto melhor um indivíduo se adaptar ao seu meio ambiente, maior será sua chance de sobreviver e gerar descendentes"* (LACERDA; CARVALHO, 1999).

A escolha desse algoritmo se deu, principalmente, pela sua vasta utilização em problemas de agendamento de tripulação como estudado em Kotecha, Sanghani e Gambhava (2014), Liu, Haghani e Toobaie (2010), além de ser utilizado exaustivamente em problemas de otimização.

A teoria sobre algoritmos genéticos é vastamente encontrada em muitos livros e artigos. Linden (2008) traz um bom referencial teórico sobre algoritmos genéticos. Os algoritmos genéticos resolvem problemas computacionais através da modelagem do processo de evolução natural, realizando processos de reprodução e mutação, até encontrar soluções adequadas para o problema em questão. A função fitness fica responsável por verificar quais soluções são viáveis, dentre as soluções geradas pelo AG.

De acordo com Santos (2008) os algoritmos genéticos imitam a ideia de seleção natural e evolução que se acredita ocorrerem na natureza. Num algoritmo genético existe uma população de indivíduos, cada um representando uma solução em potencial. As características dos indivíduos são codificadas na forma semelhante aos genes de um cromossomo. Assim, cada cromossomo representa um indivíduo da população, nesse caso uma possível solução do problema em questão. Como cada solução possui um valor, os cromossomos também podem ser avaliados para produzir um valor, para que sejam comparados uns com os outros qualitativamente. Esses valores representam a adaptação *(fitness)* do indivíduo codificado nesse cromossomo, e indivíduos de melhor adaptação têm mais chance de participar das etapas de reprodução e mutação para gerar novos indivíduos com características semelhantes. A população de indivíduos evolui através de sucessivas gerações (iterações do método), uma geração sendo produzida por recombinação das características dos indivíduos da geração anterior.

Os indivíduos de melhor adaptação são selecionados e combinados entre si por operadores de cruzamento. Geralmente dois indivíduos são selecionados como *"pais"*, e produzem um ou dois *"filhos"* combinando-se os genes de seus cromossomos. Os indivíduos podem passar ainda por um operador de mutação, que modifica alguns genes de seus cromossomos, introduzindo novas características na população. É um método probabilístico em que, idealmente, uma população inicial produz sucessivas populações, cada vez mais próximas da solução ótima. Isso acontece porque os melhores indivíduos são selecionados para se reproduzir e gerar novos indivíduos semelhantes para a população seguinte.

De forma mais simples, um AG é um processo iterativo, as quais são aplicados uma série de operadores genéticos no processo de seleção, tais como *crossover* e mutação (COLE, 1998). A Figura 6 mostra o funcionamento geral de um algoritmo genético.

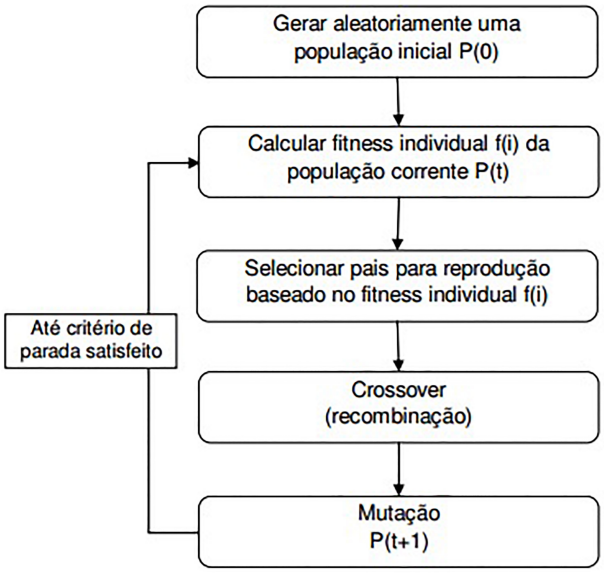


Figura 6 – Visão geral do funcionamento do AG (COLE, 1998).

O fluxograma presente na Figura 6 gera inicialmente e aleatoriamente uma população, em seguida é avaliado o desempenho de cada solução pela função *fitness*, logo depois são selecionados os pais para a reprodução, a escolha é baseada no cálculo da função *fitness*. Por último são utilizados os operadores genéticos *crossover* e mutação para gerar a nova população.

2.4.1 TERMINOLOGIA DOS ALGORITMOS GENÉTICOS

Anteriormente foi visto que os AGs são inspirados na evolução natural das espécies. No entanto, é importante entender sua terminologia, para não confundir com os termos utilizados na biologia, devido a sua natureza análoga.

1. Indivíduo

   Um simples membro da população. Nos AGs, um indivíduo é formado pelo cromossomo e sua aptidão (LACERDA; CARVALHO, 1999).

2. População

   Uma população é formada por um conjunto de indivíduos, que representam possíveis soluções do problema a ser resolvido (LACERDA; CARVALHO, 1999).

3. Cromossomos

   Um cromossomo é uma estrutura de dados, geralmente vetor ou cadeia de bits (cadeia de bits é a estrutura mais tradicional, porém nem sempre é a melhor), que representa uma possível solução do problema a ser otimizado Lacerda e Carvalho (1999). Nos sistemas naturais um ou mais cromossomos se combinam para formar as características genéticas básicas do indivíduo em questão. Na área dos AGs, os termos cromossomos e individuo são intercambiáveis (LINDEN, 2008).

4. Operadores Genéticos

   Os operadores genéticos têm como objetivo transformar a população através de sucessivas gerações, estendendo a busca até chegar a um resultado satisfatório. Os operadores genéticos são necessários para que a população se diversifique e mantenha características de adaptação adquiridas pelas gerações anteriores, em AG os operadores genéticos utilizados assim como nos termos da biologia são *crossover* e *mutação*.

   Os operadores de crossover e a mutação são os principais mecanismos de busca dos AGs para explorar regiões desconhecidas do espaço de busca. O operador crossover é aplicado a um par de cromossomos retirados de uma população, gerando dois cromossomos filhos. Cada um dos cromossomos pais tem sua cadeia de bits cortada em uma posição aleatória, produzindo duas cabeças e duas caudas. As caudas são trocadas, gerando dois novos cromossomos. A pós a operação de crossover, o operador de mutação é aplicado, com dada probabilidade, em cada bit dos dois filhos. O operador de mutação inverte os valores

de bits, ou seja, muda o valor de um dado bit de 1 para 0 ou de 0 para 1 (LACERDA; CARVALHO, 1999).

5. Função de Aptidão

   A função *fitness* é usada como medida de acordo com aptidão de cada indivíduo, é utilizada como base para selecionar cromossomas para a reprodução. Os cromossomos com melhor aptidão têm mais chances de ser selecionados, e assim, passar o seu material genético (recombinados via cruzamentos) para a próxima geração. De forma clara, o sucesso do AG para o problema, é dependente da escolha da função *fitness*, a escolha adequada da função *fitness* irá aumentar a probabilidade de retenção de material genético associado com as melhores soluções ótimas ou aproximadas (COLE, 1998).

   O modelo matemático adotado neste trabalho discutido no capítulo **??**, é representado pela função *fitness*, dessa forma, a eficiência do algoritmo está fortemente atrelado a formulação da função *fitness*.

   A Figura 7 exemplifica esse processo:

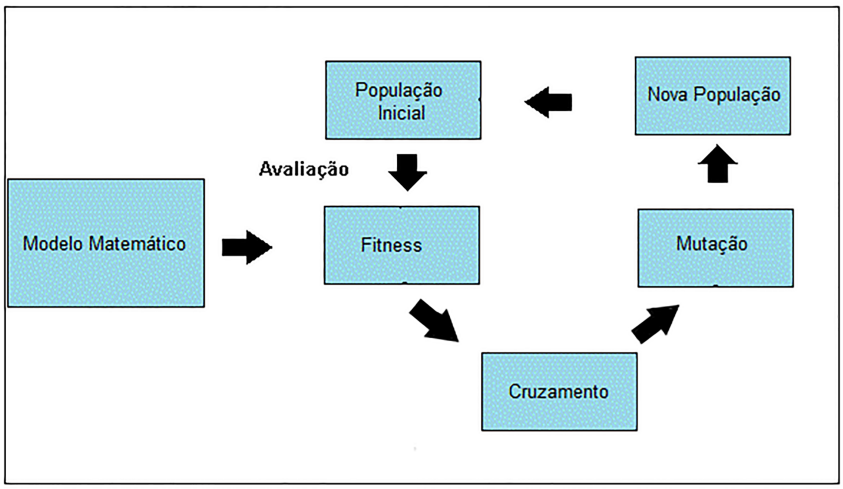


Figura 7 – Função aptidão. Fonte: Autoria própria.

A Figura 7 explica o processo de interação interno do algoritmo genético, e como o modelo matemático está atrelado no AG. A cada iteração o modelo matemático calcula o tempo de processamento de cada tarefa. Para este trabalho o algoritmo genético leva em consideração o menor tempo de execução das tarefas.

2.4.2 ESTRUTURA DE FUNCIONAMENTO DO ALGORITMO GENÉTICO

Vistos os conceitos anteriormente, o AG pode ser descrito segundo Michalewicz (1996) e organizado de acordo com Alckmin e Varejão (1996) da seguinte maneira:

- durante a iteração t, um AG mantém uma população de soluções potenciais (cromossomos ou indivíduos) *P(t)* = $\{x_1^t,...,x_n^t\}$;
- cada solução x_1^t é avaliada e produz uma medida de sua adaptação (fitness);
- uma nova população (iteração $t + 1$) é então formada privilegiando a participação dos indivíduos mais adaptados;
- alguns membros da nova geração passam por modificações, através de *crossover* e mutação, para formar novas soluções;
- este processo se repete até que um certo número de iterações seja atingido, ou até que um nível de adaptação esperado seja alcançado.

O Algoritmo 1 apresenta o procedimento geral de um AG:

Algoritmo 1 Procedimento geral do Algoritmo Genético

Entrada:

tamanho da população

taxa de *crossover*

taxa de mutação

Saída: melhor solução encontrada s

1: $t \leftarrow 0$
2: *Gere a populacao inicial* $P(t)$
3: *Avalie* $P(t)$
4: **while** *Criterio de parada não for satisfeito* **do**
5: $t \leftarrow t + 1$
6: *Selecione* $P(t)$ *a partir de* $P(t-1)$
7: *Aplique crossover* $P(t)$
8: *Aplique mutação* $P(t)$
9: *Avalie* $P(t)$
10: **end while**
11: **return** *s*

No algoritmo 1 é definido o tamanho para a população a ser trabalhada e definida também uma taxa de *crossover* e mutação para o cruzamento e geração de uma nova população. Na linha 2 uma população inicial é gerada e avaliada na linha 3. O critério de parada é definido pelo número de iterações, enquanto esse critério não for satisfeito os operadores genéticos *crossover* e mutação agem gerando novos indivíduos. Assim o algoritmo genético retorna a melhor solução.

2.5 CLOUDSIM

A escolha do simulador, se deu por ser uma ferramenta de simulação que permite a perfeita modelagem, simulação e experimentação de uma infra-estrtura em computação nas nuvens, além de serviços de aplicativos. Usando CloudSim, pesquisadores e desenvolvedores podem testar o desempenho de um serviço em um ambiente controlado e de fácil instalação. Com os resultados das avaliação relatadas podem avaliar o desempenho de seus serviços. As principais vantagens de se usar um simulador para teste de desempenho inicial incluem: *(i)* eficácia de tempo: exige muito menos esforço e tempo para implementar aplicativos *(ii)* flexibilidade e aplicabilidade: os desenvolvedores podem modelar e testar o desempenho de seus serviços de aplicativo em ambientes heterogêneos com pouco esforço de programação e implantação (CALHEIROS et al., 2011). O cloudSim torna-se mais útil principalmente quando pensa-se em abordagens alternativas para testes e desenvolvimento que alavanquem novas tecnologias em computação nas nuvens (CALHEIROS et al., 2009).

O CloudSim foi desenvolvido pelo grupo de pesquisa e desenvolvimento de software do Departamento de Ciência da Computação e Engenharia de Software na Universidade de Melbourne, Austrália. Até o desenvolvimento deste trabalho o projeto podia ser acessado em: (CLOUDSIM, 2015).

Como visto anteriormente, a computação nas nuvens tem sofrido uma demanda cada vez maior por parte de seus usuários, consequentemente, há uma preocupação eminente com a qualidade de serviço nesse sistema, por conta disso, tecnologias que ofereçam um ambiente de simulação adequado é muito importante, já que, seria necessário um custo financeiro muito elevado caso precisasse pagar para que pudesse fazer testes com algoritmos em ambientes reais.

Uma alternativa viável é o uso de ferramentas de simulação que abre a possibilidade de avaliar as aplicações em um ambiente controlado, onde se pode facilmente reproduzir resultados. Abordagens baseadas em simulação oferecem benefícios significativos para as empresas, ou quem quiser oferecer seus serviços através das nuvens, permitindo-lhes: *(i)* testar seus serviços em ambiente controlável e replicável *(ii)* sintonizar os gargalos do sistema antes mesmo de sua implantação nas nuvens *(iii)* fazer experiência em cenários com um mix de recurso e carga de trabalho diferentes (QUIROZ et al., 2009).

Em uma rápida busca sobre o simulador CloudSim em sites de buscas durante o desenvolvimento desse trabalho, teve-se mais de 79.000 resultados. O ClousSim é um dos simuladores mais utilizados em computação nas nuvens, já que, outros simuladores conhecidos fazem parte de sua composição, como no caso do GridSim, que é um dos mais utilizados em Grades Computacionais.

De acordo com Calheiros et al. (2011), nenhumas das ofertas atuais em sistemas distribuídos, podem ser usadas diretamente para a modelagem em ambientes de Computação nas Nuvens. O CloudSim é uma alternativa bastante viável, já que possui um quadro de simulação generalizada e extensível que permite a perfeita modelagem, simulação e experimentação de

infraestruturas e serviços de aplicativos em Computação nas Nuvens.

2.5.1 ARQUITETURA

A arquitetura do CloudSim pode ser dividida em 3 camadas principais (CALHEIROS et al., 2011), como mostra a Figura 8.

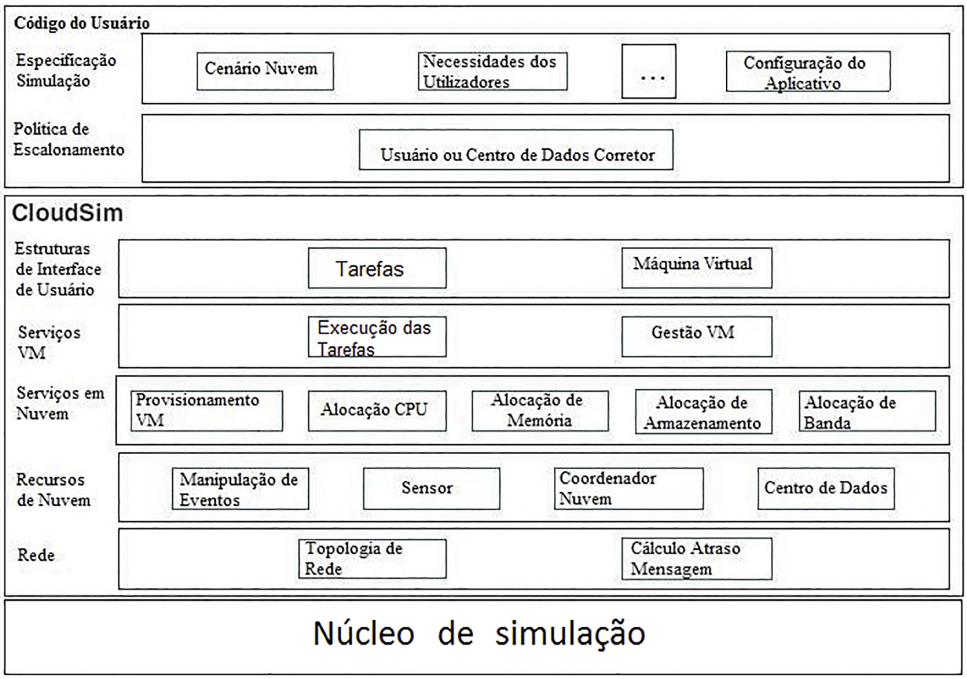


Figura 8 – Arquitetura CloudSim. (CALHEIROS et al., 2011)

.

A Figura 8 mostra as camadas presentes no CLoudSim, que são elas:

- Camada do usuário: é a camada onde é definido o ambiente o qual vai ser simulado, bem como algumas configurações. Também aqui é definida algumas políticas relacionadas ao usuário.
- Camada Principal do CloudSim: onde fica todo o controle da simulação em se tratando da nuvem, também é onde são definidos os objetos os quais vão ser simulados, como rede, recursos da nuvem, serviços da nuvem.
- Camada do núcleo da simulação: onde é feito o controle da simulação, faz a gerência da execução dos eventos que acontecem na simulação e faz o controle da troca de mensagens entre as entidades presentes na simulação.

3 TRABALHOS RELACIONADOS

Nesta seção serão abordados os trabalhos relacionados com a problemática e objetivo visto anteriormente. Problema de Particionamento de Conjuntos *(SPP)*, Algoritmos Genéticos, Simulador CloudSim e Computação nas Nuvens. Além disso, esse capítulo é de grande importância, pois discute o estado da arte sobre assuntos dessa pesquisa.

Shimpy e Sidhu (2014), mostram a diferença e eficiência de vários algoritmos de escalonamento em diferentes ambientes de computação nas nuvens, aqui os autores concluem que as técnicas mais apropriadas para o escalonamento de tarefas são as que estão presentes as heurísticas. Dessa forma, essa pesquisa faz o uso da meta-heurística algoritmo genético como forma de encontrar soluções com o *makespan* mínimo.

Assim como o trabalho de Shimpy e Sidhu (2014), em Deshmane e Pandhare (2014) estuda os algoritmos de escalonamentos presentes na literatura, porém aqui, ressalta a importância do escalonamento em computação nas nuvens, isso torna cada vez mais evidente a necessidade de algoritmos de escalonamento eficientes.

Em Tsai et al. (2014), o algoritmo proposto é baseado em hiper-heurística, um algoritmo que integra vários algoritmos heurísticos. A ideia básica do algoritmo proposto é alavancar os pontos fortes dos algoritmos heurísticos, tais como "recozimento simulado", algoritmo genético, a otimização por enxame de partículas e por colônia de formigas, integrando-os em um algoritmo único. No entanto, um escalonador não pode demorar muito tempo para escalonar as tarefas e o uso de várias heurísticas pode deixa o processo de escalonamento um pouco mais complexo. Por conta disto, essa pesquisa trabalha apenas com uma heurística. Isso torna o processo de escalonamento mais simples, como foi observado durante o desenvolvimento do algoritmo enquanto se realizava as simulações.

No trabalho discutido em Dong, Liu e Rojas-Cessa (2015), o alvo é a redução do gasto energético dos servidores em data centers na computação nas nuvens. Aqui os autores utilizaram uma técnica gulosa para escalonar as tarefas entre os servidores. Porém, não faz uso de nenhuma heurística como o algoritmo genético presente neste trabalho, além disso, o objetivo do trabalho está voltado para a computação verde que tem como objetivo o aproveitamento energético.

Como visto em capítulos anteriores, outro grande paradigma de sistemas distribuídos, são as grades computacionais. Os escalonadores propostos para grades computacionais, principalmente quando esses são algoritmos estáticos, podem facilmente ser adaptados em outros sistemas distribuídos, como no caso da computação nas nuvens. Em Reda et al. (2014) é proposto um algoritmo estático, no entanto, como visto em Shimpy e Sidhu (2014) e observado durante os experimentos realizados nesta pesquisa, que os algoritmos estáticos não são tão eficientes como as heurísticas para o problema de escalonamento. Uma solução possível para aproveitar o que tem de melhor em técnicas estáticas é mesclar heurísticas e algoritmo estático, como nesse trabalho que utiliza as duas técnicas.

O *SPP* é um problema de otimização combinatória que pode ser utilizado em várias

situações. O mesmo já é utilizado em outros trabalhos no contexto de computação nas nuvens como em Borovskiy et al. (2011). Porém nesse mesmo trabalho, não se utiliza nenhuma meta-heurística.

Diversos trabalhos na literatura tratam do problema de particionamento de conjuntos, porém a maioria deles trata do problema de alocação de tripulação, principalmente no escalonamento de aeronaves, como em Chen, Chen e Zhang (2010), Ozdemir e Mohan (1999) e Mingozzi et al. (1999). Apesar de usarem meta-heurísticas nenhum deles estão relacionados com a computação nas nuvens. Tais pesquisas, influenciaram no desenvolvimento desse trabalho.

Como visto anteriormente, a computação nas nuvens já é uma realidade eminente nos tempos atuais. Por isso, muitos pesquisadores já estudam estratégias que melhorem a qualidade de serviço nesse sistema, em trabalhos como em Arfeen, Pawlikowski e Willig (2011), Maguluri, Srikant e Ying (2012), Mohan e Raj (2012), Dai et al. (2012), Tomita e Kuribayashi (2011), Mochizuki e Kuribayashi (2011), Nair e Vaidehi (2011), Bessai et al. (2012) estudam estratégias de alocação de recursos. Muitos trabalhos tratam o problema de escalonamento, no trabalho Müller e Gómez (2006) estuda estratégias de escalonamento, porém as técnicas aqui estudadas são outras.

Em geral as principais diferenças deste trabalho em relação a outros vistos no início desse Capítulo são: a ideia de um escalonador que não seja tão complexo no processo de escalonamento das tarefas, por isso o uso de apenas uma heurística e não de várias heurísticas vistas em trabalhos anteriores. Essa pesquisa também faz a junção de um algoritmo estático e uma heurística como formar de reduzir o número de máquinas virtuais - VMs, isso não foi encontrado em trabalhos anteriores, ou seja, há uma preocupação em reduzir o número de VMs e não apenas em encontrar o *makespan* mínimo. Além disso, outro ponto importante nessa pesquisa foi trabalhar com o problema de particionamento de conjuntos, apesar dele já ser utilizado em outros trabalhos voltados para a computação nas nuvens, porém nesse trabalho o SPP trabalha com a ideia de reduzir o número de VMs, isso não foi encontrado em trabalhos na literatura.

Os trabalhos encontrados na literatura, não possuem todos os requisitos aqui abordados em um único trabalho, tais como: Problema de Particionamento de Conjuntos *(SPP)*, Algoritmos Genéticos, Simulador CloudSim e Computação nas Nuvens.

4 ALGORITMO DE ESCALONAMENTO DE TAREFAS PARA COMPUTAÇÃO NAS NUVENS

Esse capítulo apresenta o escalonador de tarefas baseado em Algoritmo Genético para Redução do uso de Máquinas Virtuais (EAGRVMs) proposto. O algoritmo proposto é dividido em duas fases, a primeira fase faz uso do algoritmo genético discutido na seção 2.4. A segunda fase é continuidade da primeira e faz uso da técnica presente no algoritmo estático discutido neste capítulo. Essa técnica presente no algoritmo estático foi projetada exclusivamente para resolver a questão do SPP discutido na seção 2.3.

Como discutido em capítulos anteriores, os algoritmos de escalonamento levam em consideração diversos aspectos, como por exemplo: a estimativa de Tempo de Conclusão (*Estimated Conclusion Time* - ECT). Para o algoritmo proposto, os aspectos levados em consideração são discutidos no modelo matemático descrito a seguir.

4.1 MODELAGEM MATEMÁTICA APLICADA AO PROBLEMA

Vale ressaltar que o modelo matemático utilizado nesse projeto, foi desenvolvido por integrantes do grupo de pesquisa do *Laboratório de Otimização e Inteligência Artificial* - LOIA da *Universidade Do Estado do Rio grande do Norte* - UERN . O modelo matemático proposto foi adaptado para que pudesse se encaixar nesta pesquisa. Apesar do foco desse trabalho tratar de computação nas nuvens, o mesmo modelo matemático aqui utilizado, pode ser inserido em outros contextos de sistemas distribuídos como em computação em grade *(grid computing)*, para que isso seja possível basta adaptar os parâmetros necessários ao modelo em questão. O modelo matemático aqui estudado, leva em consideração o tempo de envio, processamento e resposta de uma tarefa.

O modelo matemático aqui utilizado se deu por ele ser um modelo mais preciso, já que ele trabalha com um número de variáveis maior e seu processo é constituído por três etapas.

Considerando o ambiente de computação nas nuvens, o cliente envia as tarefas para ser processadas nas nuvens, o tempo de envio vai depender da largura de banda da conexão entre o cliente e o provedor da nuvem, ou seja, quantos bits podem ser enviados por segundo. Porém, normalmente, a largura de banda do provedor é superior à dos clientes, pode-se dizer que, na prática, esse tempo pode ser somente o tamanho em bits da tarefa *fileSize* dividido pela largura de banda *lBeC*.

$$T_{env} = \frac{fileSize}{lBeC} \tag{4.1}$$

Onde:
T_{env} = Tempo de envio da tarefa
fileSize = Tamanho em bits da entrada da tarefa
lBeC = Largura de banda de envio do cliente

Em seguida, a tarefa será processada no provedor da nuvem. O tempo de processamento é impreciso, pois varia muito devido as configurações das VMs e as tarefas que estão em execução no momento. O tempo de processamento da tarefa será definido como o tempo de processamento estimado da tarefa *pesVmT* dividido pelo poder de processamento da VM *pesVm*, que nesse caso, será o número de unidades de processamento.

$$T_{proc} = \frac{pesVmT}{pesVm} \tag{4.2}$$

Onde:
T_{proc} = Tempo de processamento da tarefa
pesVmT = Número de unidades de processamento para execução da tarefa na VM
pesVm = Número de unidades de processamento na VM

E, por fim, uma resposta é enviada ao cliente. Essa seguirá a mesma lógica do envio, porém olhando pelo lado do recebimento do cliente. Assim, o tempo será calculado pelo tamanho da resposta *outputSize* e a largura de banda de recebimento do cliente *lBrC*.

$$T_{resp} = \frac{outputSize}{lBrC} \tag{4.3}$$

Onde:
T_{resp} = Tempo de resposta da tarefa
outputSize = Tamanho em bits da saída da tarefa
lBrC = largura de banda de recebimento do cliente

Em geral o modelo matemático é representado como na fórmula a seguir:

$$T_{total} = \frac{\sum_{i=1}^{n} fileSize_i}{lBeC} + \frac{\sum_{i=1}^{n} pesVmT_i}{pesVm} + \frac{\sum_{i=1}^{n} outputSize_i}{lBrC} \tag{4.4}$$

Onde:

fileSize = Tamanho em bits da entrada da tarefa
lBeC = Largura de banda de envio do cliente
pesVmT = Número de unidades de processamento para execução da tarefa na VM
pesVm = Número de unidades de processamento na VM
outputSize = Tamanho em bits da saída da tarefa
lBrC = largura de banda de recebimento do cliente
T_{total} = Tempo total de processamento das N tarefas

A Figura 9 exemplifica o processo de transição das tarefas no modelo matemático.

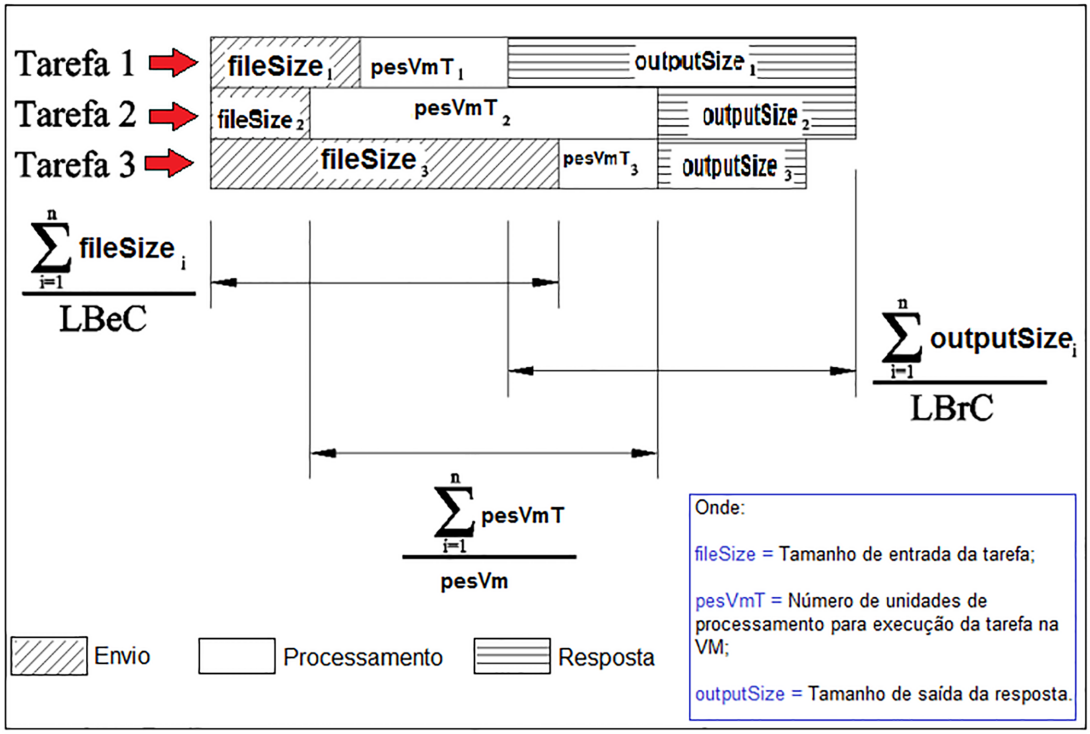


Figura 9 – Funcionamento do modelo matemático no ambiente de computação nas nuvens. Fonte: Adaptada do modelo matemático

A Figura 9 mostra as tarefas sendo enviadas por vários usuários ou clientes da nuvem, os usuários podem enviar mais de uma tarefa. Essas tarefas serão processadas na nuvem e, em seguida irão retornar como resposta para o usuário. Assim que o envio da tarefa estiver completo no servidor, ele vai começar a processá-la e a responder ao cliente assim que acabar o processamento, de modo que enquanto uma tarefa maior ainda estiver sendo transferida, uma menor pode já estar em processamento.

Dessa forma, o tempo necessário para executar mais de uma tarefa seria simplesmente o somatório dos tempos de cada tarefa, como descrito na fórmula 4.4.

A maneira para saber quantas tarefas estão na etapa de envio em um determinado intervalo de tempo é:

$$nTe_j = \sum_{i=1}^{n} X_{ij}, \forall j \tag{4.5}$$

Onde:
nTe_j = Número de tarefas sendo enviadas no intervalo j
X_{ij} = variável binária que diz se a tarefa i está sendo enviada para a VM no intervalo de tempo j

O número de tarefas em processamento e envio da resposta seguem a mesma lógica:

$$nTp_j = \sum_{i=1}^{n} Y_{ij}, \forall, j \tag{4.6}$$

Onde:
nTp_j = Número de tarefas sendo enviadas no intervalo j
Y_{ij} = variável binária que diz se a tarefa está sendo processada no intervalo de tempo j

$$nTr_j = \sum_{i=1}^{n} Z_{ij}, \forall, j \tag{4.7}$$

Onde:
nTr_j = Número de respostas sendo enviadas ao cliente no intervalo j
Z_{ij} = variável binária que diz se a tarefa está em resposta no intervalo de tempo j

É importante deixar evidente que os aspectos citados anteriormente, fazem parte do modelo matemático, que avalia o quanto é eficiente uma solução. O modelo matemático é usado apenas para avaliar as soluções. De certo modo, o modelo matemático age de forma externa ao algoritmo genético. Isso acontece pois em sua essência, o algoritmo genético pode funcionar com qualquer método de avaliação para as soluções.

4.2 POLÍTICA DE ESCALONAMENTO BASEADA EM ALGORITMO GENÉTICO

O objetivo específico dessa pesquisa é diminuir o tempo que as tarefas levam para serem executadas nas nuvens. A primeira fase do escalonador proposto, faz uso do algoritmo genético, como forma de encontrar soluções com o *makespan* mínimo possível. Dessa forma, o escalonador se comporta seguindo o mesmo fluxo do algoritmo genético.

É necessário entender a terminologia dos AGs para que se possa entender o algoritmo proposto, pois há uma analogia muito forte entre os termos da biologia e os termos usados no campo dos AGs. A Tabela 1 abaixo mostra essas diferenças.

Tabela 1 – Definições de termos dos algoritmos genéticos

Linguagem Natural	**Algoritmo Genético**
Cromossomo	Indivíduo
População	Conjunto de Soluções
Crossover e Mutação	Operadores Genéticos
Fitness	Qualidade da solução

O indivíduo ou solução é representado através de um vetor, seu tamanho é definido pelo número de tarefas. Cada tarefa só pode ser atendida por uma única VM, ou seja, não pode haver repetições, onde a posição representa a tarefa e o valor representa a VM. Uma VM pode atender mais de uma tarefa, isso é definido aleatoriamente. A Figura 10 exemplifica esse processo, onde uma mesma VM pode atender mais de uma tarefa, enquanto que uma tarefa só é atendida exclusivamente por uma única VM.

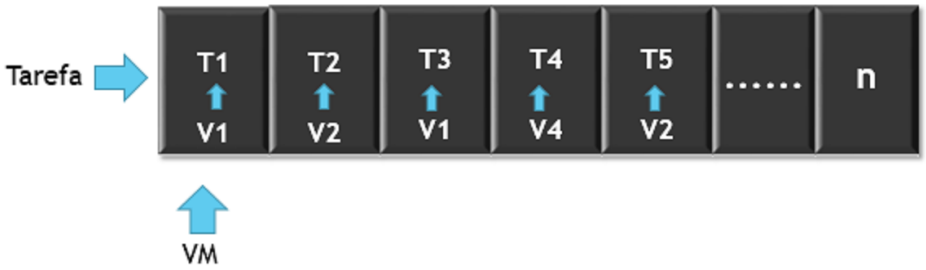


Figura 10 – Representação de um indivíduo. Fonte: Autoria própria.

- **A seleção dos pares para a reprodução dos indivíduos**

  A seleção do indivíduo *A* e feita através do método da roleta, o mesmo faz parte da população elite, sendo assim, está entre os 30% melhores (População Elite) ou seja, os mais aptos. O indivíduo *B* é escolhido entre todos os indivíduos também através do método roleta.

- ***Crossover***

  A taxa definida para o *crossover*, seleciona no mínimo 10% e no máximo 89% de cada par, para gerar os próximos dois indivíduos, na tentativa de melhorar o processo de fitness.

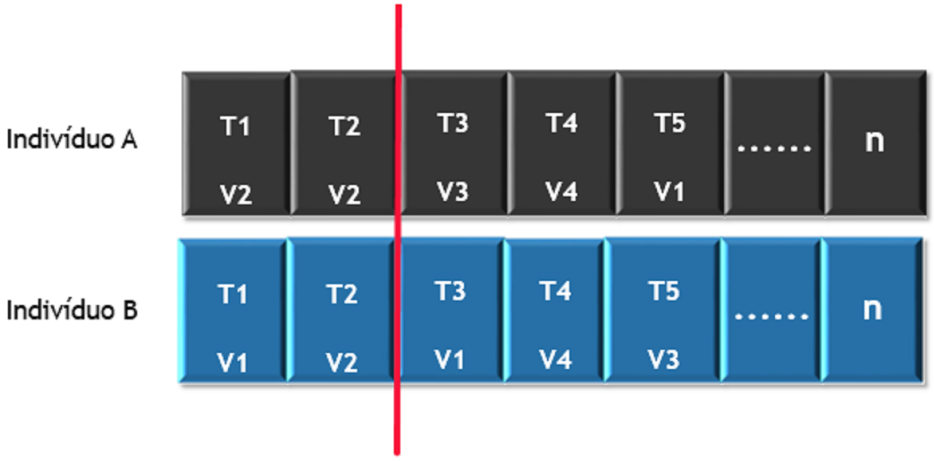


Figura 11 – Operador *crossover*. Fonte: Autoria própria.

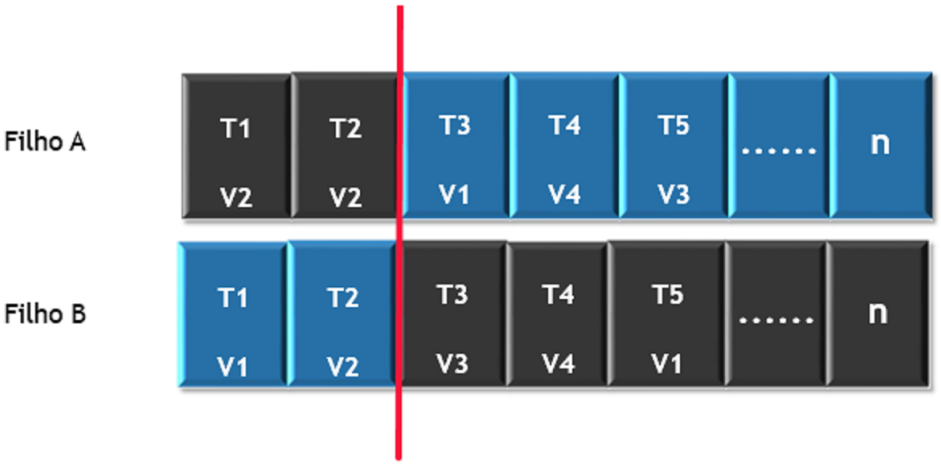


Figura 12 – Novos indivíduos por *crossover*. Fonte: Autoria própria.

A Figura 11 é aplicada a taxa de crossover nos indivíduos pais, gerando novos indivíduos com características diferentes na Figura 12. Esse processo é utilizado para melhorar a qualidade da solução.

- **Mutação**

  Em cada nova população indivíduos mutantes são gerados, seu total é igual a 10% da população inteira. O operador de mutação inverte os valores das VMs modificando assim a estrutura genética da solução.

As variáveis presente na Tabela 1 podem ser visualizadas no fluxograma apresentado na Figura 13. Essas variáveis fazem parte do processo de execução do algoritmo genético.

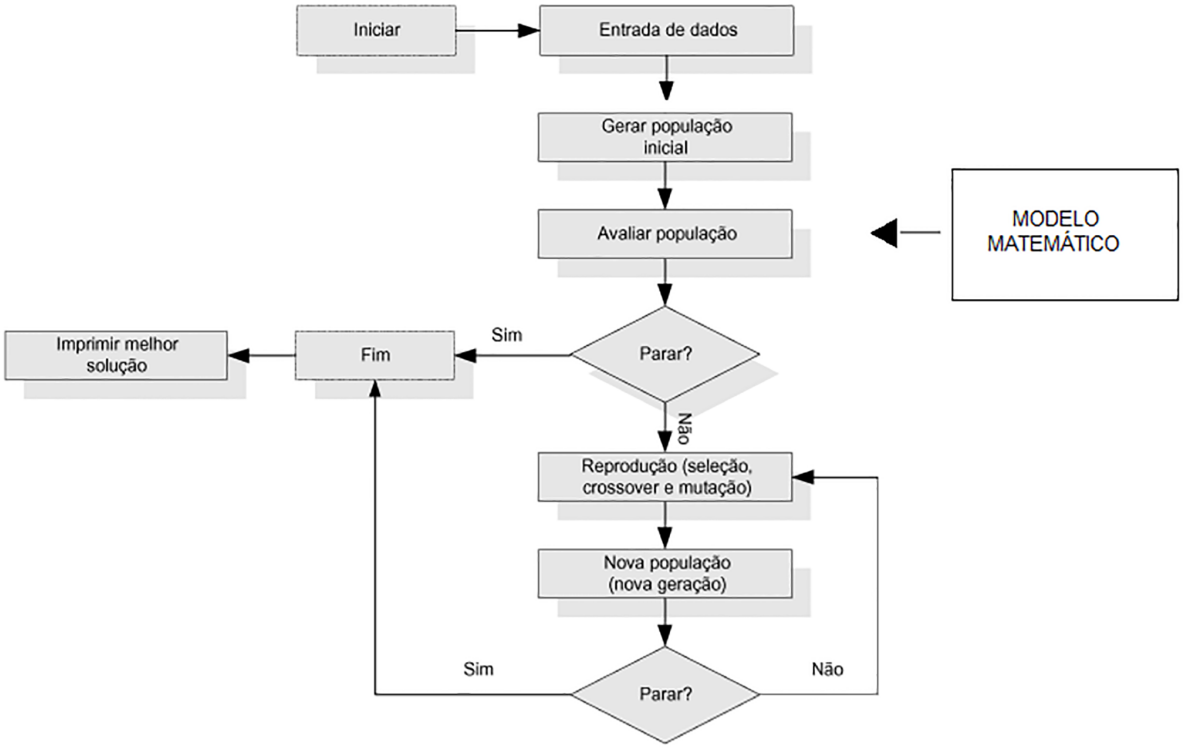


Figura 13 – Fluxograma Algoritmo Genético. Fonte: Autoria própria.

A Figura 13 mostra o processo de seleção das tarefas dentro do algoritmo genético, esse processo tem como objetivo encontrar o *makespan* mínimo. Esse fluxograma é importante para que se entenda o processo do algoritmo genético no contexto de computação nas nuvens. Seguindo a sequência do fluxograma, o AG gera uma população inicial (conjunto de solução) dentro desse conjunto de soluções estão as tarefas. Estas tarefas são representadas por indivíduos que representa uma solução já escalonada. Em seguida, cada solução é avaliada (*fitness*), essa avaliação é representada pelo modelo matemático. Por fim, o operador genético (*crossover*) faz a troca de material genético entre os indivíduos e logo depois é feito o processo de mutação e, por fim, o algoritmo genético retorna uma solução. Esta solução nada mais é que as próprias tarefas já escalonadas. Esse processo é repetido até que o critério de parada seja satisfeito. O critério de parada é definido pelo número de interações. Já as interações são definidas com base na qualidade da resposta e tempo gasto para que as tarefas sejam escalonadas.

A Figura 14 exemplifica o processo de execução da primeira fase do algoritmo proposto no simulador CloudSim.

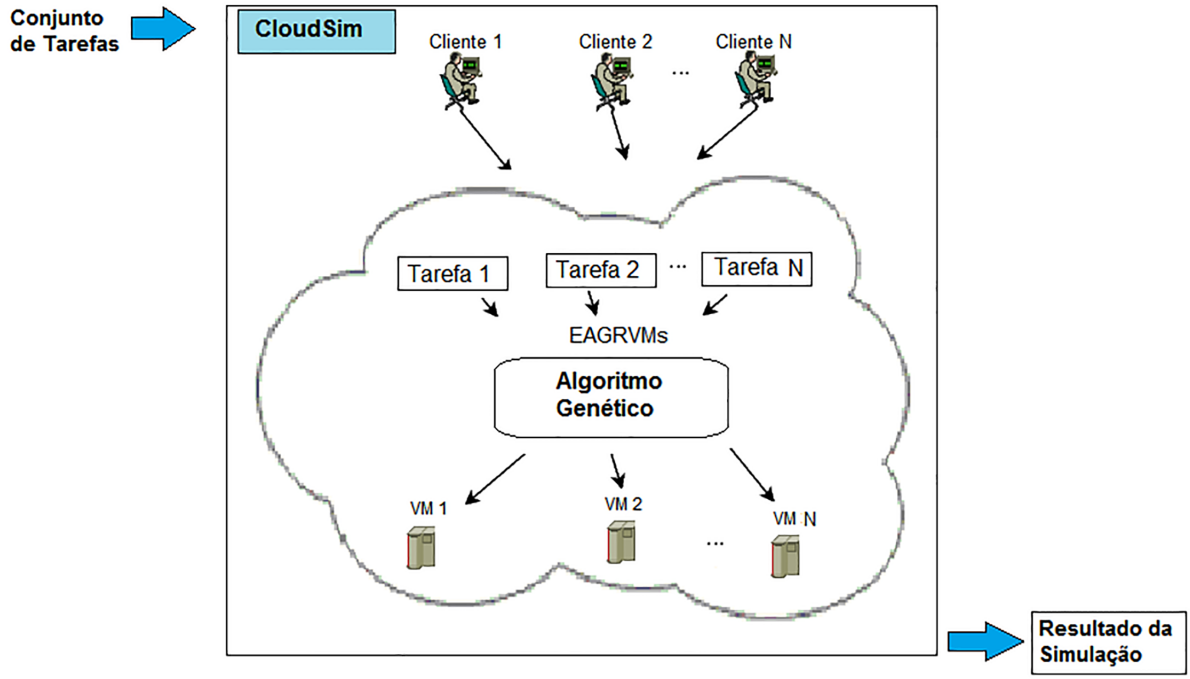


Figura 14 – Fluxograma do escalonador de tarefas EAGRVMs em sua primeira fase. Fonte: autoria própria.

O escalonador EAGRVMs executa o algoritmo genético dentro do simulador CloudSim. O cliente da nuvem envia um conjunto de tarefas para serem processadas, o simulador CloudSim recebe como entrada esse conjunto de tarefas e aplica o algoritmo genético para encontrar o *makespan* com menor tempo.

4.3 ALGORITMO ESTÁTICO PROPOSTO

Um diferencial deste trabalho é tentar reduzir o número de máquinas virtuais – VMs, diferentemente de outras propostas já discutidas no Capitulo 3 que se preocupa apenas em reduzir o tempo de execução das tarefas – *makespan*. Esse trabalho visa os dois objetivos reduzindo assim o tempo de processamento das tarefas e o número de VMs.

Para tentar diminuir o número de máquinas virtuais necessárias para executar um conjunto de tarefas já escalonadas pelo algoritmo genético com um tempo de execução que no mínimo seja aproximado ou até mesmo melhor do encontrado na primeira fase do escalonador proposto (algoritmo genético), foi necessário projetar um algoritmo estático que aqui é chamado de Recurso Máximo (*Maximum Resource* - MR) que tem como fundamento o conhecimento necessário acerca dos recursos disponíveis. O fundamento de um algoritmo estático foi discutido na seção 2.2. Como já visto, o algoritmo estático está presente nesse trabalho para resolver o problema do particionamento de conjuntos, que aqui é representado de modo a reduzir o número de VMs.

A Figura 15 mostra o fluxograma de funcionamento do MR.

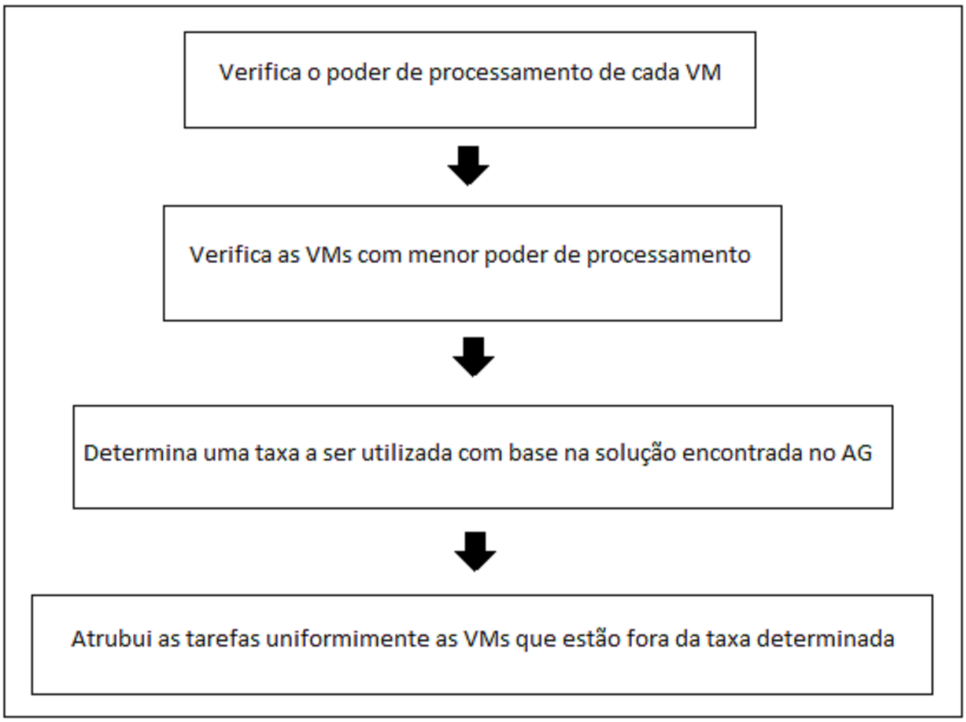


Figura 15 – Fluxograma de funcionamento do algoritmo estático *Maximum Resource*. Fonte: Autoria própria

A Figura 15 exemplifica o processo de escalonamento feito pelo MR, com base na solução já encontrada no algoritmo genético, o primeiro passo do MR é verificar o poder de processamento das máquinas virtuais - VMs. Esse poder de processamento é definido pela quantidade de unidades de processamento presente em cada VM. Com essa informação, o MR determina uma taxa para ser trabalhada com as VMs com menor poder de processamento. Por exemplo: se a taxa escolhida for 20% e, tivesse um cenário com 50 VMs, seriam utilizados apenas 10 máquinas virtuais. A tarefas que foram alocadas nessas 10 VMs pelo algoritmo genético, irão ser distribuídas uniformemente para as outras 40 máquinas virtuais. Dessa forma, o MR reduz o número de VMs enquanto que o restante das tarefas já está otimizado pelo algoritmo genético.

A seguir uma descrição mais detalhada do algoritmo MR:

- Verifica o poder de processamento de cada VM. O poder de processamento é calculado pelo número de unidades de processamento alocada para a máquina virtual.
- Verifica quais as máquinas virtuais têm o menor poder de processamento.
- Uma taxa é definida para ser trabalhada, por exemplo: se a taxa escolhida for 20% e, tivéssemos uma cenário com 50 máquinas virtuais, iria-se utilizar apenas 10 VMs.
- A tarefas que foram alocadas nessas 10 máquinas virtuais pelo algoritmo genético, irão ser distribuídas uniformemente para as outras 40 VMs. Dessa forma, o MR reduz o número de máquinas virtuais enquanto que o restante das tarefas já está otimizado pelo algoritmo genético.

A Figura 16 ilustra o processo de execução da segunda fase do algoritmo proposto no simulador CloudSim.

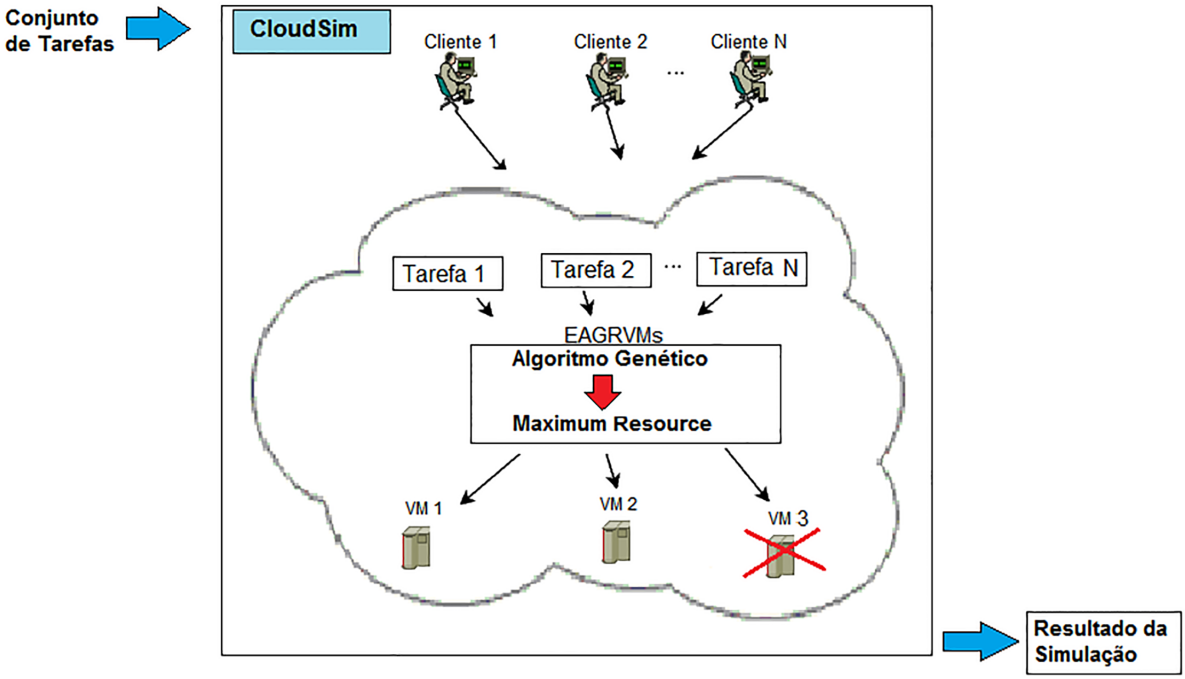


Figura 16 – Processo de escalonamento realizados através do *Maximum Resource*. Fonte: autoria própria.

A Figura 16 demonstra o funcionamento do MR em conjunto com o algoritmo genético. O algoritmo genético encontra a melhor solução e em seguida o MR começa o processo final de escalonamento.

Para finalizar, a Figura 17 mostra o processo geral presente no algoritmo proposto neste trabalho.

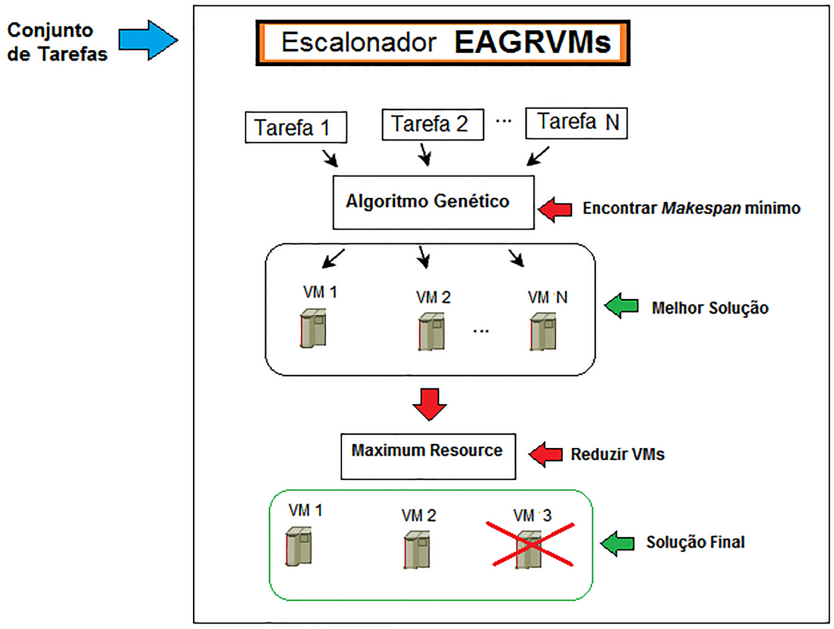


Figura 17 – Processo geral do escalonamento realizado através do EAGRVMs. Fonte: autoria própria

Na Figura 17, um conjunto de tarefas é enviada. Como estudado, o primeiro passo do escalonador EAGRVMs é o algoritmo genético, este encontra a melhor solução com base no *makespan* mínimo. Essa solução servirá como ponto de referência para novas soluções que o Maximum Resource encontrar, ou seja, com a redução das máquinas virtuais pelo *Maximum Resource*, o novo valor do *makespan* tem que ser aproximado do valor encontrado no algoritmo genético ou mesmo melhor. Um detalhe importante é que as tarefas que não forem escalonadas pelo *Maximum Resource* já estarão otimizadas pelo algoritmo genético. E por fim, é gerada uma solução final com um número de VMs menor do que o presente na solução encontrada pelo o algoritmo genético.

5 VALIDAÇÃO E RESULTADOS

Neste capítulo, são detalhados os experimentos que foram executados para avaliar o algoritmo proposto. Nos experimentos são analisados o tempo de conclusão de cada aplicação - *makespan* e o número de máquinas virtuais necessárias para executar as aplicações. Como já comentado na secção 2.5 o simulador CloudSim é utilizado nos experimentos.

5.1 FLUXOS DE TRABALHO UTILIZADOS

O fluxo de trabalho utilizado foi o formato padrão de carga de trabalho (*standard workload format* - swf) proposto por Chapin et al. (1999). Os resultados extraídos das simulações foram feitos com os três fluxos de trabalho descritos abaixo. Esses fluxos são representados por dados reais de aplicações monitoradas por meses, que evidenciam o comportamento do algoritmo em diferentes condições de uso. A cargas aqui utilizadas, foram retiradas de: (SWF, 2015).

- **CEA Curie**

  Essa aplicação contém um log de 20 meses no valor de dados do supercomputador Curie operado pela CEA um organismo de investigação tecnológica francesa financiada pelo governo . Os dados provêm de um total de 11.808 processadores Intel (93.312 núcleos) e um adicional de 288 Nvidia PGUs.

- **UniLu Gaia**

  Este registro dispõe uma base de dados de 3 (três) meses de uso do cluster Gaia na Universidade do Luxemburgo . É usado principalmente por biólogos que trabalham com grandes problemas de dados e pessoas de engenharia que trabalham com simulações físicas.

- **LPC EGEE**

  LPC significa *Laboratoire de Physique Corpusculaire* (Laboratório de Física Corpuscular) da Universidade Blaise-Pascal, de Clermont-Ferrand, França. Um dos objetivos deste projeto é desenvolver uma infra-estrutura para tratar e analisar os esperados por ano 15 petabytes de dados a ser gerado pelo acelerador de partículas (*Large Hadron Collider* - LHC)

5.2 AMBIENTE DE TESTE

Para o desenvolvimento do ambiente de teste, foi levado em conta o número de máquinas virtuais, a configuração de cada máquina virtual, o número de tarefas, número de usuários

e configuração dos *hosts* (servidores). As mesmas configurações foram utilizadas em cada algoritmo testado. As configurações utilizadas estão na Tabela 2.

Tabela 2 – Configuração do ambiente simulado

Configuração	**Valor**
Número de máquinas físicas (*hosts*)	2
Número de núcleos por host	8
RAM por host	32GB
Espaço em disco por host	10TB
Frequência de cada núcleo	8000 MIPS
Numero de usuários	100
Número de Máquinas Virtuais	40

Para a comparação com o algoritmo proposto, foi utilizado o escalonador FIFO discutido no Capítulo 2.2. O FIFO é considerado um algoritmo clássico para o problema de escalonamento e ainda bastante utilizado em grandes projetos, como em Apache (2015). Por isso é frequentemente utilizado como referência na literatura para comparar o desempenho de novas propostas.

Também foi utilizado o algoritmo heurístico baseado em otimização por enxame de partículas (*particle swarm optimization* - PSO) estudado nos trabalhos Ramezani, Lu e Hussain (2013), Pandey et al. (2010). Como já discutido em capítulos anteriores, em geral os escalonadores baseados em heurísticas apresentam resultados mais satisfatórios. Por isso é justo comparar a proposta deste trabalho que faz uso de heurística com outro que também utilize heurística.

5.3 RESULTADOS EXPERIMENTAIS

Para as tarefas, em cada aplicação foi testado um conjunto de 1000, 2500 e 5000 tarefas. Os resultados dos experimentos são ilustrados em duas partes: A primeira é utilizado apenas o algoritmo genético e, a segunda faz uso do algoritmo estático (*Maximum Resource*), abaixo os resultados das simulações.

5.3.1 RESULTADOS OBTIDOS ATRAVÉS DO ALGORITMO GENÉTICO

A seleção dos pares para a reprodução dos indivíduos é feita de maneira aleatória. É definida uma população elite onde estão presentes os 30% melhores indivíduos, ou seja, os mais aptos. Dessa população elite é escolhido aleatoriamente um indivíduo que vai cruzar com outro escolhido entre todos os indivíduos também aleatoriamente.

A taxa de *crossove*r seleciona no mínimo 10% e no máximo 89% de cada par para gerar os novos indivíduos na tentativa de melhorar a solução *fitness*. Essa taxa foi definida para garantir

que haja a variabilidade e continuação genética nos novos indivíduos gerados. Para a mutação foi aplicado uma taxa de 3% em 10% da população. Os parâmetros escolhidos para a taxa de *crossover* e de mutação se mostram satisfatórios para a resolução do problema. Pois os valores definidos para a taxa de *crossover* e mutação conseguiram alcançar a convergência e gerar boas soluções.

Os resultados mostrados na Tabela 3, são os valores obtidos com a variabilidade da quantidade de tarefas para aplicação LPC-EGEE.

Tabela 3 – Comparação de algoritmos segundo valores do tempo total da simulação do CloudSim - LPC-EGEE.

Número de tarefas x algoritmos – *Timespan* em segundos			
	1000	***2500***	***5000***
PSO	467336,69	1292866,63	2665785,58
FIFO	1324065,07	3511411,15	7238186,17
AG	294656,75	788752,58	2295063,72

A Tabela 3, demostra um ganho significativo em média de 74 % com a política de escalonamento com o algoritmo genético e um ganho de 64% do PSO em relação ao FIFO. Em comparação com o PSO, o AG teve um ganho satisfatório em média de 29%. A Figura 18 apresenta os resultados da simulação de forma gráfica.

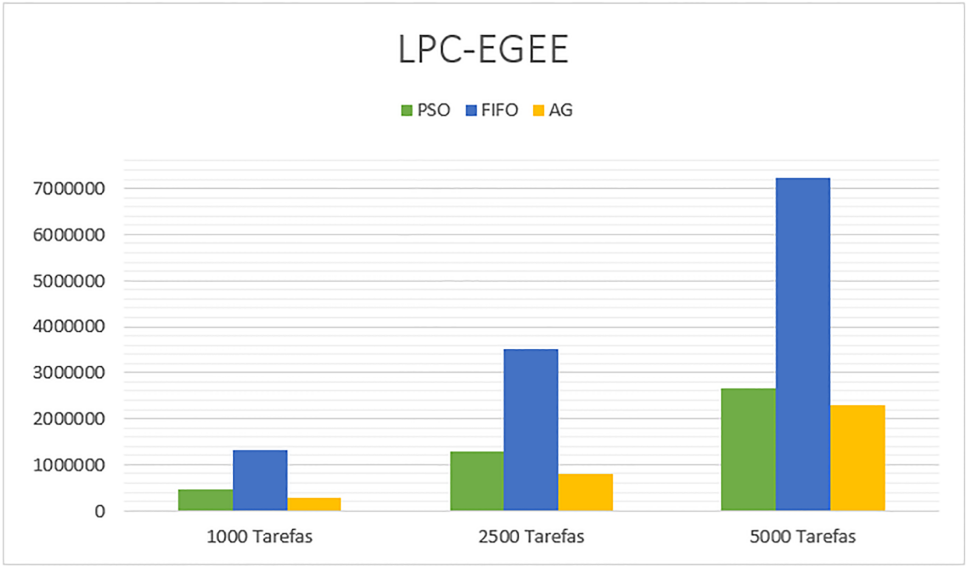


Figura 18 – Gráfico de valores do tempo total da simulação do CloudSim comparando o AG aos algoritmos em *Timespan* em segundos - LPC-EGEE

Como visto na Figura 18, os resultados encontrados no AG para aplicação LPC-EGEE em termos de objetivo da pesquisa foram satisfatórios, pois reduziu o *makespan* nos 3 (três) conjuntos de tarefas testados, obtendo um bom desempenho sobre o FIFO e um desempenho satisfatório em relação ao PSO.

Os resultados mostrados na Tabela 4, são os valores obtidos com a variabilidade da quantidade de tarefas para aplicação UNILU GAIA.

Tabela 4 – Comparação de algoritmos segundo valores do tempo total da simulação do CloudSim - UNILU GAIA.

Número de tarefas x algoritmos – *Timespan* em segundos			
	1000	***2500***	***5000***
PSO	534024,91	1372611,53	2714870,17
FIFO	2025248,22	5290996,2	11748661,73
AG	437587,94	1102413,46	2393124,29

Na Tabela 4, para a aplicação UNILU GAIA obteve-se um *Timespan* com valores mais altos, aqui o AG e PSO obtiveram valores mais aproximados com um leve ganho para o AG em média de 16%. Em relação ao FIFO a diferença permaneceu comparando com a aplicação LPC-EGEE. A Figura 19 apresenta os resultados da simulação de forma gráfica.

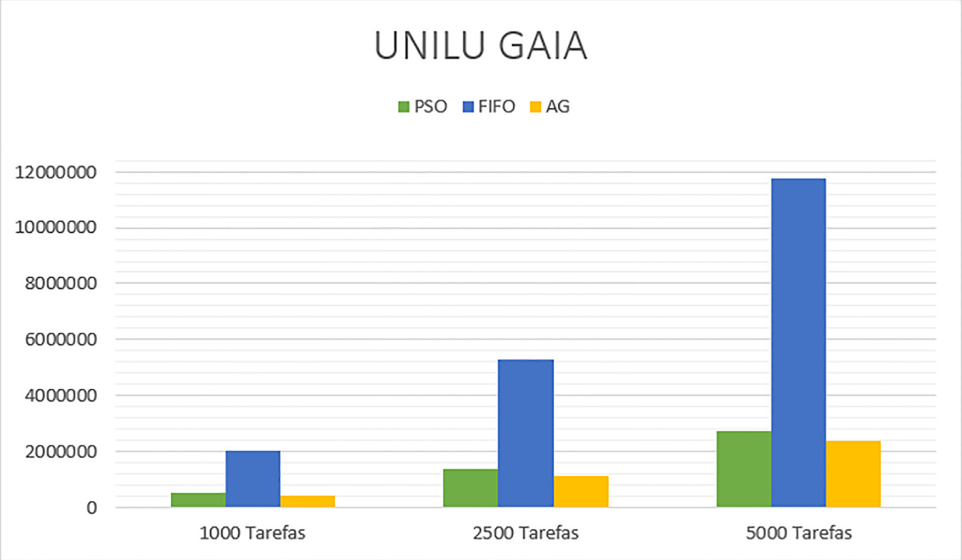


Figura 19 – Gráfico de valores do tempo total da simulação do CloudSim comparando o AG aos algoritmos em *Timespan* em segundos - UNILU GAIA

Para a aplicação UNILU GAIA, os resultados obtidos na Figura 19 mostra que apesar da redução na diferença entre o AG e PSO, em termos de objetivo o AG ainda entregou resultados satisfatórios.

Os resultados mostrados na Tabela 5, são os valores obtidos com a variabilidade da quantidade de tarefas para aplicação CEA CURIE.

Tabela 5 – Comparação de algoritmos segundo valores do tempo total da simulação do CloudSim - CEA CURIE.

Número de tarefas x algoritmos – *Timespan* em segundos			
	1000	***2500***	***5000***
PSO	533895,78	1335871,27	2698518,91
FIFO	691515,7	276606,47	138303,4
AG	149687,59	394904,82	853346,91

A Tabela 5 com a aplicação CEA CURIE foi onde se obteve maior variação dos resultados nos testes aplicado a 1000 tarefas o AG teve um resultado melhor em relação aos outros dois algoritmos, com um ganho de 71% sobre o PSO e um ganho de mais de 100% em relação ao FIFO.

No teste feito com 2500 tarefas o FIFO obteve o melhor resultado. O AG teve um desempenho ruim em relação ao FIFO. Já o PSO teve um péssimo desempenho.

Com 5000 tarefas o FIFO teve um ganho significante em relação aos dois algoritmos, no entanto o AG teve um ganho considerável em relação ao PSO. A Figura 20 apresenta os resultados da simulação de forma gráfica.

Os resultados não satisfatórios encontrados no AG e PSO em relação ao FIFO para a aplicação CEA CURIE, é devido a variação de valores encontrados nessa aplicação em relação as outras duas aplicações.

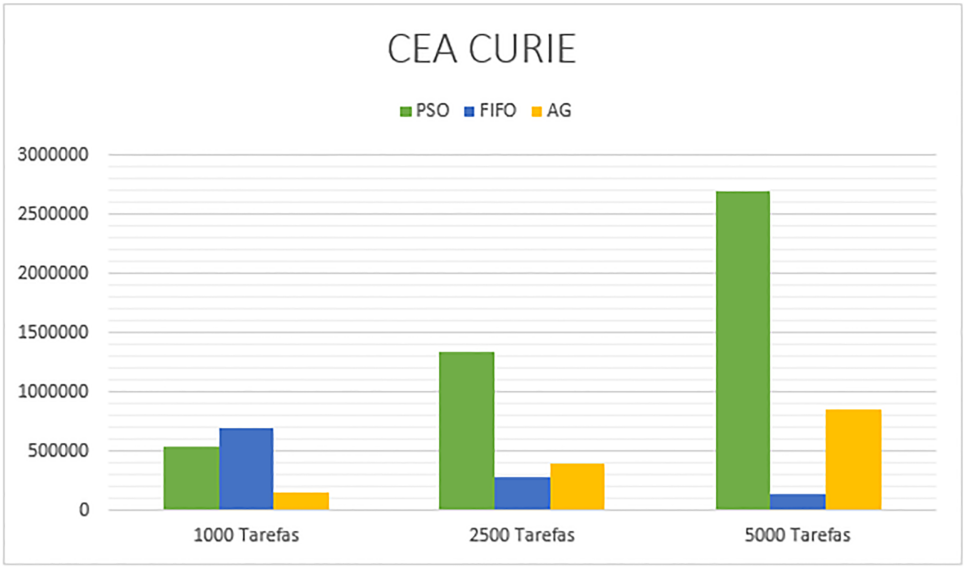


Figura 20 – Gráfico de valores do tempo total da simulação do CloudSim comparando o AG aos algoritmos em *Timespan* em segundos - CEA CURIE

A aplicação CEA CURIE com seus valores ilustrados na Figura 20, mostra que o AG obteve seu pior resultado nos conjuntos de instâncias com 2500 e 5000 tarefas, no entanto,

em questão de objetivo de pesquisa o AG se manteve em uma posição intermediaria obtendo resultados melhores que o PSO.

5.3.2 RESULTADOS OBTIDOS ATRAVÉS DO ALGORITMO ESTÁTICO

A segunda parte dos experimentos foi o algoritmo estático (*Maximum Resource* - MR). Durante as simulações observou-se que para as 3 aplicações utilizadas nos experimentos, a taxa de máquinas virtuais - VM definida para o MR foi de 20%. Essa taxa foi definida com base nos resultados através dos experimentos, pois com valores mais elevados os resultados obtidos pelo algoritmo proposto não foram satisfatórios. Por isso os resultados aqui mostrados já se sabe que a taxa utilizada foi de 20%.

Os resultados apresentados abaixo para os algoritmos PSO e FIFO são os mesmos encontrados na primeira parte desse capítulo. No entanto, agora tem-se a inserção do algoritmo estático (*Maximum Resource*) e seus novos valores. As Figuras 21, 22 e 23 apresenta os resultados da simulação de forma gráfica.

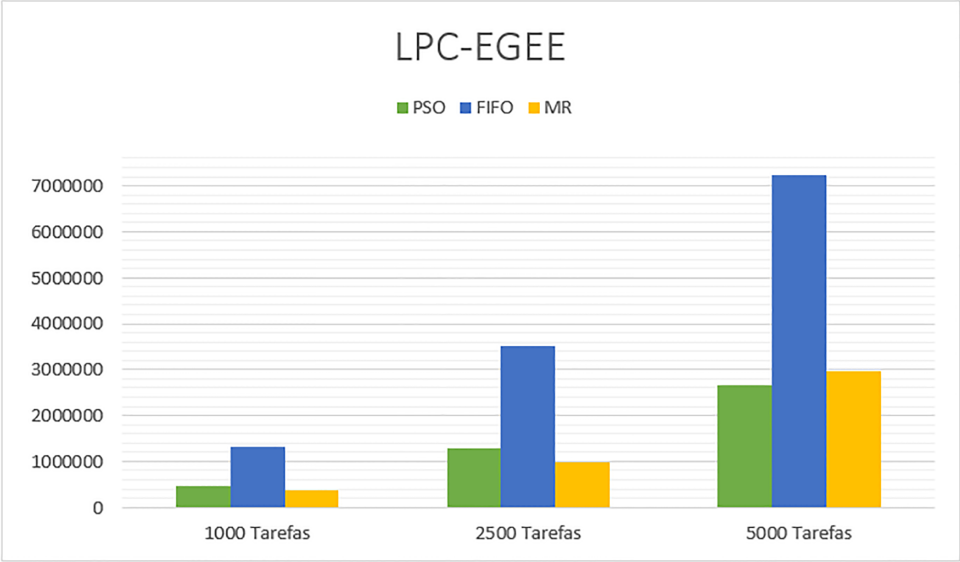


Figura 21 – Gráfico de valores do tempo total da simulação do CloudSim comparando o *Maximum Resource* aos algoritmos em *Timespan* em segundos - LPC-EGEE

A Figura 21 mostra que o MR teve um ganho satisfatório em média de 17% em relação ao PSO e um ganho muito bom com média de 71% em relação ao FIFO utilizando 1000 e 2500 tarefas. Com 5000 tarefas o MR teve um ganho de 58% em relação ao FIFO e uma perda aceitável de 11% em relação ao PSO.

Para o objetivo dessa pesquisa, os resultados apresentados na Figura 21 sobre a aplicação LPC-EGEE, mostraram que o MR superou um algoritmo clássico, pois conseguiu reduzir o número de máquinas virtuais e ao mesmo tempo que se manteve à frente do FIFO em tempo de execução das tarefas.

Em comparação ao PSO, trabalhando com um número de tarefa maior, o MR apesentou uma leve perda, porém vale ressaltar que ele reduziu o número de máquinas virtuais em 20%. Essa perda já era esperada, pois o PSO por se tratar de um algoritmo heurístico e se levarmos em consideração os resultados obtidos com o algoritmo genético que foram bem próximos dos encontrados no PSO, para que houvesse uma redução de VMs o MR teria que sacrificar o tempo de processamento das tarefas.

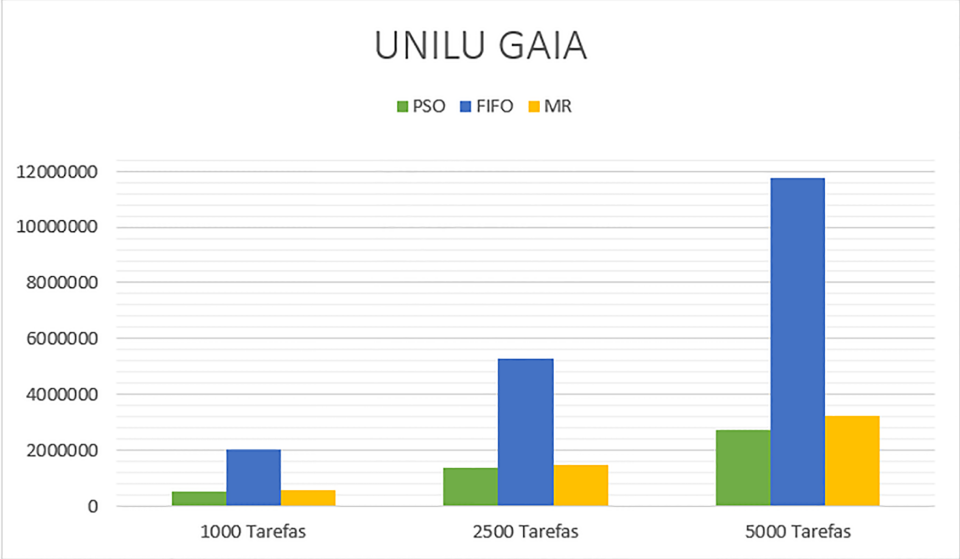


Figura 22 – Gráfico de valores do tempo total da simulação do CloudSim comparando o *Maximum Resource* aos algoritmos em *Timespan* em segundos - UNILU GAIA

Para a o resultado presente na Figura 22, o MR teve uma perda aceitável em média de 7% utilizado 1000 e 2500 tarefas sobre o PSO e teve um ótimo desempenho com média de mais de 100% em relação ao FIFO. E também manteve sua leve perda de 15% em relação ao PSO para as 5000 tarefas.

A nível de objetivo para a pesquisa em relação a aplicação UNILUGAIA, o MR manteve seu ótimo desempenho sobre o algoritmo clássico, isso prova que os resultados obtidos foram satisfatórios para a pesquisa, pois reduziu o número de máquinas virtuais e manteve o bom tempo de execução das tarefas. No entanto, o MR não foi tão eficiente quanto o PSO em tempo de execução das tarefas, porém conseguiu reduzir o número de máquinas virtuais se mantendo em um nível intermediário.

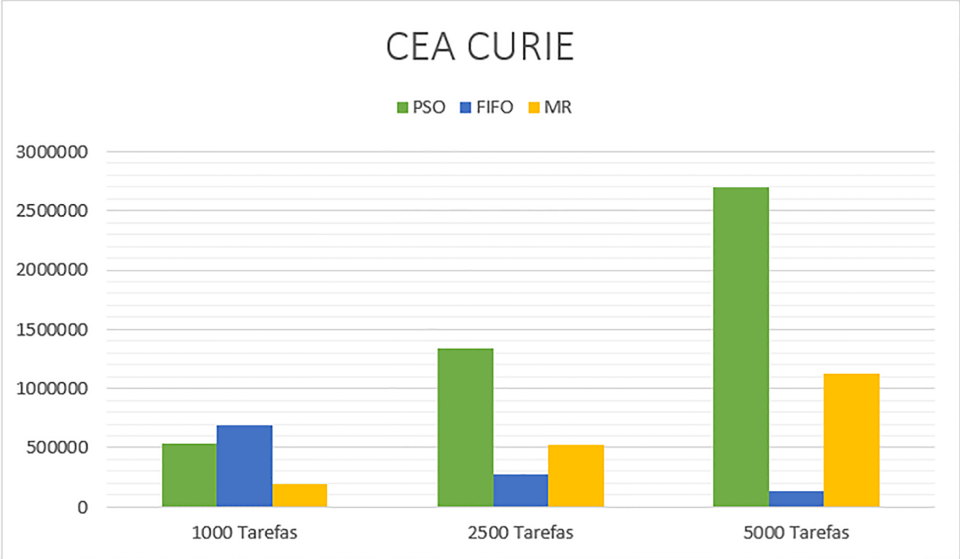


Figura 23 – Gráfico de valores do tempo total da simulação do CloudSim comparando o *Maximum Resource* aos algoritmos em *Timespan* em segundos - CEA CURIE

No último resultado presente na figura 23 o MR ainda manteve seu bom desempenho com 1000 tarefas obtendo 63% sobre o PSO. Porém com 2500 e 5000 tarefas teve seu pior desempenho em relação ao FIFO, pois perdeu com uma média de 67%.

Nessa aplicação o MR continuou provando sua viabilidade mesmo perdendo com o aumento no número de tarefas. Pois teve um bom resultado com um número menor de tarefas e ficou em uma posição intermediaria quando aumentado o número de tarefas. Porém ainda conseguiu reduzir o número de máquinas virtuais em 20%.

5.4 ANÁLISE DO DESEMPENHO

Desempenho e eficiência são duas características utilizadas para avaliar um bom escalonador. Apesar do objetivo dessa pesquisa não ser o tempo gasto pelo algoritmo, é sim o tempo de processamento de sua solução (*makespan*). No entanto, teve-se uma preocupação com o tempo gasto pelo algoritmo, por isso a ideia de desenvolver um escalonador que não fosse tão complexo como já discutido no Capítulo 3.

Vale ressaltar que o tempo gasto pelo algoritmo proposto, foi maior em relação aos outros dois comparados, fato esse que também estar relacionado a inserção do algoritmo estático e a própria questão de otimização de código. Outro fator importante relacionado ao tempo gasto pelo algoritmo é o número de iterações ou critério de parada do algoritmo genético. O critério de parada definido para o AG e PSO foi de acordo com a qualidade da solução, dessa forma, não houve um número fixo definido de iterações em cada algoritmo. Mas para que outros pesquisadores possam se basear por esta pesquisa, é importante ressaltar que de acordo com os testes executados no simulador CloudSim, em todos os 3 (três) conjuntos de tarefas para cada aplicação o EAGRVMs obteve o maior tempo para escalonar as tarefas.

Para finalizar esse capítulo, é importante entender que o MR faz parte da segunda e última etapa do algoritmo proposto, com isso pode-se concluir que os resultados aqui apresentados nessa segunda etapa dos experimentos, representa o resultado final do algoritmo proposto EAGRVMs. Mesmo com alguns resultados negativos nessa segunda etapa, o EAGRVMs provou ser eficiente, pois reduziu o número de máquinas virtuais em 20% em relação aos experimentos e obteve um tempo de execução das tarefas (*makespan*) em níveis satisfatório em torno de 34% para instâncias com 1000 e 2500 tarefas e uma perda aceitável em torno de 30 % em relação ao pior algoritmo para instâncias com 5000 tarefas. Em termos gerais o EAGRVMs cumpriu o que foi proposto, conseguiu reduzir o número de máquinas virtuais e manteve um bom tempo de execução das tarefas.

6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

No ambiente de computação nas nuvens, um fator muito importante é a questão do escalonamento de tarefas. Logo, o gerenciamento ineficaz dos recursos computacionais pode afetar de maneira significativa o consumo de recurso computacional, de modo a prejudicar usuários e até mesmo o próprio provedor da nuvem. Sabe-se que hoje a tecnologia avança constantemente, por isso a grande preocupação com a falta e o custo dos recursos computacionais.

Considerando o contexto do uso eficiente dos recursos computacionais dentro da computação nas nuvens, este trabalho propôs um algoritmo de escalonamento (EAGRVMs) que, além de buscar minimizar o tempo de processamento das tarefas (*makespan*), visa assegurar a redução do número de máquinas virtuais. Tal algoritmo foi projetado a partir do uso de solução heurística para o problema de escalonamento de tarefa. Mais precisamente, o problema de escalonamento foi formulado para resolver o problema do particionamento de conjuntos, que teve como objetivo reduzir o número de máquinas virtuais para a execução das tarefas no ambiente de computação nas nuvens.

A avaliação de desempenho teve dois objetivos: primeiro, minimizar o tempo de processamento das tarefas (*makespan*) através do algoritmo genético. Segundo, tentar reduzir o número de máquinas virtuais com base na solução encontrada pelo algoritmo genético com o uso do algoritmo estático (*Maximum Resource*). A partir dos experimentos executados, pode-se demostrar a viabilidade das políticas adotadas em temos de minimização do tempo de execução das tarefas e redução de máquinas virtuais. Pois reduziu o número de máquinas virtuais em 20% e obteve um tempo de execução das tarefas em níveis satisfatório em torno de 34%.

No entanto, deve-se também ressaltar que o algoritmo EAGRVMs apresentou uma pequena limitação com o aumento do uso das tarefas em torno de 30%, pois apesar de reduzir o número de máquinas virtuais, em todas as aplicações ele teve um desempenho inferior no tempo de processamento das tarefas (makespan) em pelo menos um dos algoritmos que foram comparados.

Dentre as contribuições deste trabalho, destacam-se: a formulação do problema de escalonamento de tarefas para o gasto ciente de recursos computacionais através do algoritmo genético; a redução do número de máquinas virtuais com o uso de um algoritmo estático; avaliação do algoritmo proposto por meio de experimentos que fizeram uso de cargas de trabalhos reais em experimentos que evidenciaram o comportamento do algoritmo em diferentes condições de uso.

6.1 TRABALHOS FUTUROS

Em computação nas nuvens outros fatores influenciam em um bom escalonamento de tarefas, muitos dos quais não puderam ser considerados neste trabalho em função do tempo ou

escopo. A seguir, são apresentadas possíveis propostas para a continuação deste trabalho:

- O escalonamento de máquinas virtuais para servidores, é algo que influência no tempo do escalonamento de tarefas abordado neste trabalho. Portanto, é interessante que ambos sejam otimizados. Por isso, a ideia é que otimize o problema de escalonamento de máquinas virtuais para servidores e agregue a proposta deste trabalho para que ambos trabalhando otimizados entregue uma solução melhor do que a encontrada neste trabalho.

- Esse trabalho comparou sua proposta com apenas um algoritmo heurístico, para o meio acadêmico é muito interessante a comparação com outros algoritmos heurísticos presentes na literatura que tratam o problema de escalonamento de tarefas em computação nas nuvens.

- Apesar do foco desse trabalho ser o escalonamento de tarefas para a redução do tempo de processamento das tarefas e a redução das maquinas virtuais. A redução do número de máquinas virtuais aborda questões como a redução do consumo energético, é interessante analisar esse quesito.

REFERÊNCIAS

ABRAHAM, A.; XHAFA, F. Computational models and heuristic methods for grid scheduling problems. *Future generation computer systems*, Elsevier, v. 26, n. 4, p. 608–621, 2010. Citado na página 22.

AKL, S. G.; DONG, F. *Scheduling algorithms for grid computing: State of the art and open problems*. [S.l.], 2006. Citado na página 22.

ALCKMIN, D. P. de F.; VAREJÃO, F. M. Hybrid genetic algorithm applied to the clustering problem. Revista Investigacion Operacional, 1996. Citado na página 28.

ALMEIDA, W. R. M.; CASTRO, C. A. L.; FILHO, P. M. de A. Computação em nuvem: Uma visão geral. *CONSELHO EDITORIAL*, p. 62, 2013. Citado 2 vezes nas páginas 17 e 19.

APACHE. *Hadoop*. 2015. [Acesso em: 02-07-2015]. Disponível em: <http://hadoop.apache.org>. Citado na página 46.

ARFEEN, M. A.; PAWLIKOWSKI, K.; WILLIG, A. A framework for resource allocation strategies in cloud computing environment. In: IEEE. *Computer Software and Applications Conference Workshops (COMPSACW), 2011 IEEE 35th Annual*. [S.l.], 2011. p. 261–266. Citado na página 33.

BARROSO, A. et al. Metaheurísticas na alocação de tarefas em sistemas distribuídos de tempo real. *Anais do VII Simpósio de Computadores Tolerantes a Falhas, Campina Grande-PB*, 1997. Citado na página 14.

BESSAI, K. et al. Bi-criteria workflow tasks allocation and scheduling in cloud computing environments. In: IEEE. *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*. [S.l.], 2012. p. 638–645. Citado na página 33.

BITTENCOURT, L. F.; MADEIRA, E. R. M. Hcoc: a cost optimization algorithm for workflow scheduling in hybrid clouds. *Journal of Internet Services and Applications*, Springer, v. 2, n. 3, p. 207–227, 2011. Citado na página 20.

BLYTHE, J. et al. Task scheduling strategies for workflow-based applications in grids. In: IEEE. *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on*. [S.l.], 2005. v. 2, p. 759–767. Citado na página 24.

BOROVSKIY, V. et al. A linear programming approach for optimizing workload distribution in a cloud. In: *CLOUD COMPUTING 2011, The Second International Conference on Cloud Computing, GRIDs, and Virtualization*. [S.l.: s.n.], 2011. p. 127–132. Citado 2 vezes nas páginas 25 e 33.

BORRO, L. C. *Escalonamento em grades móveis: uma abordagem ciente do consumo de energia*. Dissertação (Mestrado) — Universidade de São Paulo, 2014. Citado 2 vezes nas páginas 21 e 22.

BUYYA, R.; ABRAMSON, D.; VENUGOPAL, S. The grid economy. *Proceedings of the IEEE*, IEEE, v. 93, n. 3, p. 698–714, 2005. Citado na página 14.

BUYYA, R. et al. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, Elsevier, v. 25, n. 6, p. 599–616, 2009. Citado na página 17.

CALHEIROS, R. N. et al. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, Wiley Online Library, v. 41, n. 1, p. 23–50, 2011. Citado 3 vezes nas páginas 8, 30 e 31.

CALHEIROS, R. N. et al. Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services. *arXiv preprint arXiv:0903.2525*, 2009. Citado na página 30.

CASAVANT, T. L.; KUHL, J. G. A taxonomy of scheduling in general-purpose distributed computing systems. *Software Engineering, IEEE Transactions on*, IEEE, v. 14, n. 2, p. 141–154, 1988. Citado 3 vezes nas páginas 8, 21 e 22.

CHAPIN, S. J. et al. Benchmarks and standards for the evaluation of parallel job schedulers. In: SPRINGER. *Job Scheduling Strategies for Parallel Processing*. [S.l.], 1999. p. 67–90. Citado na página 45.

CHEN, X.; CHEN, X.; ZHANG, X. Crew scheduling models in airline disruption management. In: IEEE. *Industrial Engineering and Engineering Management (IE&EM), 2010 IEEE 17Th International Conference on*. [S.l.], 2010. p. 1032–1037. Citado na página 33.

CLOUD. *Computação em nuvem*. 2015. [Acesso em: 16-06-2015]. Disponível em: <https://pt.wikipedia.org/wiki/Computa%C3%A7%C3%A3o_em_nuvem>. Citado 2 vezes nas páginas 8 e 19.

CLOUDSIM. *Site oficial do CloudSim*. 2015. [Acesso em: 28-05-2015]. Disponível em: <http://www.cloudbus.org/cloudsim/>. Citado na página 30.

COLE, R. M. *Clustering with genetic algorithms*. [S.l.]: Citeseer, 1998. Citado 3 vezes nas páginas 8, 26 e 28.

DAI, J. et al. Research on dynamic resource allocation with cooperation strategy in cloud computing. In: IEEE. *System Science, Engineering Design and Manufacturing Informatization (ICSEM), 2012 3rd International Conference on*. [S.l.], 2012. v. 1, p. 193–196. Citado na página 33.

DANTAS, M. A. *Computação distribuída de alto desempenho: redes, clusters e grids computacionais*. [S.l.]: Axcel books, 2005. Citado 2 vezes nas páginas 8 e 21.

DASGUPTA, G. et al. Workload management for power efficiency in virtualized data centers. *Communications of the ACM*, ACM, v. 54, n. 7, p. 131–141, 2011. Citado na página 20.

DASTJERDI, A. V.; BUYYA, R. An autonomous reliability-aware negotiation strategy for cloud computing environments. In: IEEE. *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*. [S.l.], 2012. p. 284–291. Citado na página 12.

DESHMANE, M. K.; PANDHARE, M. B. Survey on scheduling algorithms in cloud computing. *International Journal of Engineering Research and General Science*, v. 2, n. 6, 2014. Citado na página 32.

DONG, Z.; LIU, N.; ROJAS-CESSA, R. Greedy scheduling of tasks with time constraints for energy-efficient cloud-computing data centers. *Journal of Cloud Computing*, Springer, v. 4, n. 1, p. 1–14, 2015. Citado na página 32.

EL-REWINI, H.; ABD-EL-BARR, M. *Advanced computer architecture and parallel processing*. [S.l.]: John Wiley & Sons, 2005. Citado na página 15.

GILDER, G. The information factories. *Wired magazine*, v. 14, n. 10, p. 1–5, 2006. Citado na página 17.

GOLDBERG, D. E. Genetic algorithm in search, optimization and machine learning, addison. *W esley Publishing Company, R eading, MA*, v. 1, n. 98, p. 9, 1989. Citado na página 25.

GROSSMAN, R. L. The case for cloud computing. *IT professional*, IEEE, v. 11, n. 2, p. 23–27, 2009. Citado na página 12.

HOLLAND, J. H. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* [S.l.]: U Michigan Press, 1975. Citado na página 25.

JOHNSON, D. S.; GAREY, M. R. Computers and intractability: A guide to the theory of np-completeness. *Freeman&Co, San Francisco*, p. 32, 1979. Citado na página 14.

KIM, D.; HARIRI, S. *Virtual Computing: Concept, Design, and Evaluation*. [S.l.]: Springer Science & Business Media, 2001. Citado na página 23.

KOTECHA, K.; SANGHANI, G.; GAMBHAVA, N. Genetic algorithm for airline crew scheduling problem using cost-based uniform crossover. In: *Applied Computing*. [S.l.]: Springer, 2014. p. 84–91. Citado na página 25.

LACERDA, E. G. de; CARVALHO, A. de. Introdução aos algoritmos genéticos. *Sistemas inteligentes: aplicaçoes a recursos hıdricos e ciências ambientais*, v. 1, p. 99–148, 1999. Citado 3 vezes nas páginas 25, 27 e 28.

LINDEN, R. *Algoritmos Geneticos (2a edicao)*. [S.l.]: Brasport, 2008. Citado 2 vezes nas páginas 25 e 27.

LIU, M.; HAGHANI, A.; TOOBAIE, S. Genetic algorithm-based column generation approach to passenger rail crew scheduling. *Transportation Research Record: Journal of the Transportation Research Board*, Trans Res Board, v. 2159, n. 1, p. 36–43, 2010. Citado na página 25.

MAGULURI, S. T.; SRIKANT, R.; YING, L. Heavy traffic optimal resource allocation algorithms for cloud computing clusters. In: INTERNATIONAL TELETRAFFIC CONGRESS. *Proceedings of the 24th international teletraffic congress*. [S.l.], 2012. p. 25. Citado na página 33.

MICHALEWICZ, Z. *Genetic algorithms+ data structures= evolution programs*. [S.l.]: springer, 1996. Citado na página 28.

MINGOZZI, A. et al. A set partitioning approach to the crew scheduling problem. *Operations Research*, INFORMS, v. 47, n. 6, p. 873–888, 1999. Citado 2 vezes nas páginas 24 e 33.

MOCHIZUKI, K.; KURIBAYASHI, S.-i. Evaluation of optimal resource allocation method for cloud computing environments with limited electric power capacity. In: IEEE. *Network-Based Information Systems (NBiS), 2011 14th International Conference on*. [S.l.], 2011. p. 1–5. Citado na página 33.

MOHAN, N.; RAJ, E. B. Resource allocation techniques in cloud computing–research challenges for applications. In: IEEE. *Computational Intelligence and Communication Networks (CICN), 2012 Fourth International Conference on*. [S.l.], 2012. p. 556–560. Citado na página 33.

MÜLLER, G. I.; GÓMEZ, A. T. Estratégias de escalonamento em um ambiente de job-shop. *Anais do XV SEMINCO, p201-208, Blumenau, Brasil*, 2006. Citado na página 33.

MURUGESAN, S. Cloud computing gives emerging markets a lift. *IT Professional*, IEEE, v. 13, n. 6, p. 60–62, 2011. Citado na página 12.

NAIR, T. G.; VAIDEHI, M. Efficient resource arbitration and allocation strategies in cloud computing through virtualization. In: IEEE. *Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on*. [S.l.], 2011. p. 397–401. Citado na página 33.

NASONOV, D. et al. Hybrid evolutionary workflow scheduling algorithm for dynamic heterogeneous distributed computational environment. In: SPRINGER. *International Joint Conference SOCO'14-CISIS'14-ICEUTE'14*. [S.l.], 2014. p. 83–92. Citado na página 24.

OZDEMIR, H. T.; MOHAN, C. K. Graga: a graph based genetic algorithm for airline crew scheduling. In: IEEE COMPUTER SOCIETY. *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*. [S.l.], 1999. p. 27–27. Citado na página 33.

PANDEY, S. et al. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In: IEEE. *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*. [S.l.], 2010. p. 400–407. Citado na página 46.

QUIROZ, A. et al. Towards autonomic workload provisioning for enterprise grids and clouds. In: IEEE. *Grid Computing, 2009 10th IEEE/ACM International Conference on*. [S.l.], 2009. p. 50–57. Citado na página 30.

RAMEZANI, F.; LU, J.; HUSSAIN, F. Task scheduling optimization in cloud computing applying multi-objective particle swarm optimization. In: *Service-Oriented Computing*. [S.l.]: Springer, 2013. p. 237–251. Citado na página 46.

REDA, N. M. et al. Sort-mid tasks scheduling algorithm in grid computing. *Journal of Advanced Research*, Elsevier, 2014. Citado na página 32.

SADIKU, M. N.; MUSA, S. M.; MOMOH, O. D. Cloud computing: Opportunities and challenges. *Potentials, IEEE*, IEEE, v. 33, n. 1, p. 34–36, 2014. Citado na página 12.

SANTOS, A. G. dos. *Método de Geração de Colunas e Meta-heurísticas para Alocação de Tripulação*. Tese (Doutorado) — Universidade Federal de Minas Gerais, 2008. Citado na página 26.

SHETTI, K. R.; FAHMY, S. A.; BRETSCHNEIDER, T. Optimization of the heft algorithm for a cpu-gpu environment. In: IEEE. *Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2013 International Conference on*. [S.l.], 2013. p. 212–218. Citado na página 24.

SHIMPY, E.; SIDHU, M. J. Different scheduling algorithms in different cloud environment. *algorithms*, v. 3, n. 9, 2014. Citado na página 32.

SILVA, D. P. D.; CIRNE, W.; BRASILEIRO, F. V. Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids. In: *Euro-Par 2003 Parallel Processing*. [S.l.]: Springer, 2003. p. 169–180. Citado na página 23.

SOUSA, F. R.; MOREIRA, L. O.; MACHADO, J. C. Computação em nuvem: Conceitos, tecnologias, aplicações e desafios. *Ceará: Universidade Federal do Ceará*, 2009. Citado 2 vezes nas páginas 8 e 18.

SWF. *Arquivo cargas de trabalho*. 2015. [Acesso em: 28-05-2015]. Disponível em: <http://www.cs.huji.ac.il/labs/parallel/workload/logs.html>. Citado na página 45.

TAURION, C. *Cloud Computing-Computação em Nuvem*. [S.l.]: Brasport, 2009. Citado na página 18.

TOMITA, T.; KURIBAYASHI, S.-i. Congestion control method with fair resource allocation for cloud computing environments. In: IEEE. *Communications, Computers and Signal Processing (PacRim), 2011 IEEE Pacific Rim Conference on*. [S.l.], 2011. p. 1–6. Citado na página 33.

TSAI, C. et al. A hyper-heuristic scheduling algorithm for cloud. IEEE, 2014. Citado na página 32.

VERAS, M.; TOZER, R. *Cloud Computing: nova arquitetura da TI*. [S.l.]: Brasport, 2012. Citado na página 18.

WOODHULL, A. S.; TANENBAUM, A. S. *Operating Systems Design and Implementation*. [S.l.]: Prentice-Hall, 1997. Citado na página 23.