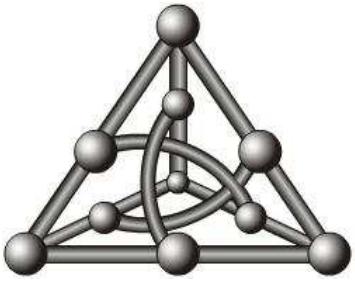
# Exploração do Espaço de Projetos de Sistemas Heterogêneos Aplicada ao Problema de Alocação de Recursos em Nuvem

**FACOM**
**Faculdade de Computação**
**Universidade Federal de Mato Grosso do Sul**



Danillo Christi Almeida Arigoni

Orientador: Ricardo Ribeiro dos Santos

Coorientadora: Liana Dessandre Duenha Garanhani

1 de agosto de 2022

# Exploração do Espaço de Projetos de Sistemas Heterogêneos Aplicada ao Problema de Alocação de Recursos em Nuvem[1]

1 de agosto de 2022

Banca Examinadora:

- Prof. Dr. Ricardo Ribeiro dos Santos (FACOM/UFMS) - orientador
- Profa. Dra. Liana Dessandre Duenha Garanhani (FACOM/UFMS) - coorientadora
- Prof. Dr. César Augusto Fonticielha De Rose (PUCRS) - membro externo
- Prof. Dr. Renato Porfirio Ishii (FACOM/UFMS) - membro interno

[1]Trabalho Realizado com Auxílio da FUNDECT Proc. No: 71/700.151/2020

# Resumo

Os serviços de computação em nuvem oferecem uma gama de recursos computacionais disponíveis sob demanda. Contudo, encontrar a melhor configuração de recursos que reduza custos e atenda as exigências do usuário tornou-se um grande desafio. Este desafio compartilha características essenciais com um problema da área de arquitetura de computadores, a exploração de espaço de projetos - *Design Space Exploration* (*DSE*). Em *DSE*, o foco é escolher, dentre uma grande quantidade de soluções arquiteturais, qual a mais indicada para uma determinada demanda, buscando atender objetivos e cumprindo as restrições de projeto. Diante disso, este trabalho propõe a aplicação de técnicas de exploração de espaço de projeto como potencial solução para o problema de alocação de recursos em nuvem. Este trabalho projetou e desenvolveu a ferramenta MultiExplorer-VM, uma extensão da ferramenta de *DSE* MultiExplorer. Essa extensão possui um fluxo de execução adaptado para a resolução do problema de alocação de recursos em nuvem, utilizando e adaptando algoritmos e técnicas de exploração de espaço de projeto. Experimentos foram realizados visando a validação estatística e comparação com outro trabalho existente na literatura da área que utiliza a técnica *Paramount Interation* (*PI*). Os resultados mostram que MultiExplorer-VM possibilita alcançar configurações de recursos com melhorias significativas em relação a técnica de *PI*. Seguindo as restrições de um modelo de otimização, para algumas aplicações, as soluções obtidas por MultiExplorer-VM são similares às configurações ótimas obtidas por uma ferramenta de busca exaustiva de soluções aplicada ao mesmo problema.

# Abstract

Cloud computing services offer on demand computing resources to applications. Finding the best cloud resource allocation that fits application budget, meets performance, and follows the applications constraints is still a research challenge. Our studies have shown that such challenge is quite akin to the *Design Space Exploration* (*DSE*) problem once that they both have to find suitable hardware configurations in a large design space, having incompatible objectives subject to several constraints. In that scenario, the goal of this work is to solve the cloud resource allocation problem by adopting techniques that originally are applied to solve the design space exploration problem. This work designed and develop a software extension named MultiExplorer-VM from a *DSE* tool called MultiExplorer. This new extension has a workflow to work with the cloud resource allocation problem by applying techniques originally targeted to solve the *DSE* problem. A comprehensive set of experiments has been performed to statistical validation and to compare solutions from our proposal to other existing research work, focused on the cloud resource allocation problem, based on the *Paramount Interation* (*PI*) technique. The results show that the proposed extension achieves significant (better) results compared to the *PI* technique. According to the model objectives and constraints, for some applications, solutions from MultiExplorer-VM are similar to the optimal cloud configuration.

# Agradecimentos

Aos meus pais Rita e Waldomiro por priorizar e incentivar meus estudos além de oferecer todo suporte e apoio para que isto fosse possível.

Ao Professor Ricardo que me honrou com a oportunidade ser seu orientando, compartilhando comigo parte de seu vasto saber sobre o ensino de Ciência da Computação e me auxiliando no desenvolvimento deste trabalho.

À Faculdade de Computação (FACOM) da UFMS, pela estrutura e pelo enorme acesso ao conhecimento, me ofertado por meio dos seus excelentes professores.

À Professora Liana pelas contribuições de extrema relevância feitas ao projeto.

Aos colegas do LSCAD, em especial aos do grupo de pesquisa Multi-Explorer (Rhayssa, Daniela, Erick, Wilton, Maurício e Ernesto), por toda troca de experiências e colaboração no projeto.

À Comissão de Aperfeiçoamento de Pessoal do Nível Superior (CAPES) pelo apoio financeiro concedido às pesquisas realizadas no LSCAD/-FACOM/UFMS.

À Fundect-MS, pelo financiamento do projeto Proc. No: 71/700.151/2020.

# Sumário

# Lista de Algoritmos

# Lista de Figuras

# Lista de Tabelas

# Siglas

***AWS*** *Amazon Web Services*. 15, 32, 34, 36, 41, 72

***Amazon EC2*** *Amazon Elastic Compute Cloud*. 33

***CPU*** *Central Processing Unit*. 14, 15, 35, 44

***DSE*** *Design Space Exploration*. 2, 3, 15, 20–23, 27, 54–56, 59

***GA*** *Genetic Algorithms*. 23

***GPU*** *Graphics Processing Unit*. 14

***GP*** *Gaussian Process*. 32

***I/O*** *Input/Output*. 14, 15, 35

***MAPE*** *Mean Absolute Percentage Error*. 52–55, 72

***MIPS*** *Millions of Instructions Per Second*. 49, 50, 53, 54, 58

***MOOP*** *Multiobjective Optimization Problem*. 18, 21–23

***McPAT*** *Multicore Power, Area and Timing*. 39

***NPB*** *Numerical Aerodynamic Simulation Parallel Benchmarks*. 32, 72

***NSGA-II*** *Non-dominated Sorting Genetic Algorithm II*. 24, 26, 40, 41

***NSGA*** *Non-dominated Sorting Genetic Algorithm*. 24

***PARIS*** *Performance Aware Resource Inference System*. 28, 30, 34–36

***PI*** *Paramount Interation*. 2, 3, 16, 32, 33, 35, 36, 56, 72

***RF*** *Random Forest*. 32, 35, 53, 55

***SMBO*** *Sequential model-based optimization*. 19, 31, 33

***SQL*** *Standard Query Language*. 33

***SVR*** *Support Vector Regression Machines*. 51, 53

***VM*** *Virtual Machine*. 9, 11, 14, 18, 19, 30, 32, 34–36, 41, 42, 44–50, 53, 54, 56–61, 72, 73

**LSCAD** Laboratório de Sistemas Computacionais de Alto Desempenho. 38

**OB** Otimização Bayesiana. 36

# Capítulo 1

# Introdução

O aumento da densidade da carga de trabalho das aplicações para execução em nuvem torna essencial a otimização dos recursos e serviços que serão disponibilizados nesse ambiente. Adicionalmente, a heterogeneidade das aplicações gera demandas por recursos computacionais também heterogêneos; por exemplo, cargas de trabalho podem necessitar uso intenso de *Central Processing Unit* (*CPU*) (são definidas como *CPU-intensive*), enquanto outras podem demandar uso intenso de entrada e saída (*I/O-intensive*), ou ainda necessitar de *Graphics Processing Unit* (*GPU*) ou outros aceleradores [44]. Uma vez conhecida a aplicação, faz-se necessário alocar uma combinação de recursos de hardware e software que melhor atenda a sua necessidade. Para muitas aplicações, a alocação de recursos computacionais na nuvem deve procurar atender mais de um objetivo simultaneamente.

Em suma, busca-se uma (ou um conjunto de) máquina(s) virtual(is) que atenda(m) a objetivos como: maximizar o desempenho da aplicação, minimizar custo, minimizar tempo de execução, minimizar tempo de comunicação, entre outros [16, 18]. No contexto de aplicações em nuvem computacional, a máquina virtual - *Virtual Machine* (*VM*) - é um recurso implementado como uma combinação de um hardware real e software de virtualização [66]. O software de virtualização possibilita que, na perspectiva do usuário, a *VM* apresente recursos necessários de hardware (processador, memória, dispositivos de E/S, armazenamento) e software (sistema operacional) de maneira individualizada para a aplicação. Entretanto, o hardware original (*host*), pode ser compartilhado entre diferentes aplicações ou usuários.

A indústria de serviços em nuvem computacional tem adotado fer-

ramentas para indicar configurações de infraestruturas adequadas para aplicações dos usuários. Como exemplo, o provedor de serviços e infraestrutura de nuvem *Amazon Web Services* (*AWS*) oferece a ferramenta *AWS Compute Optimizer* que utiliza técnicas de *machine learning* para analisar o histórico de métricas de utilização - incluindo utilização de *I/O*, armazenamento, *CPU* e rede. Baseado nessa análise, *Compute Optimizer* cria recomendações que ajudam a reduzir custos e otimizar poder computacional e desempenho da aplicação. Contudo, essa ferramenta somente gera recomendações para algumas instâncias *AWS* e pode levar até 12 horas para o serviço finalizar a análise e as recomendações de recursos serem entregues ao usuário [1].

Na área de sistemas de computação, mais especificamente de arquitetura de computadores, a demanda por sistemas automáticos que indiquem combinações de recursos físicos adequados para cargas de trabalho ou aplicações não é novidade. O termo exploração do espaço de projeto, *Design Space Exploration* (*DSE*), consiste na utilização de algoritmos e técnicas de otimização para escolher, dentro de um conjunto de parâmetros arquiteturais (o espaço de projeto), soluções (ou configurações) computacionais para atendimento das demandas definidas pelo usuário. As técnicas de *DSE*, de forma geral, visam atender múltiplos objetivos (como maximizar desempenho ou minimizar consumo energético) e obedecendo a diversas restrições de projeto (custo, área, dissipação de potência, entre outros).

Observa-se que a alocação de recursos em nuvem e exploração do espaço de projetos parecem compartilhar características essenciais. Enquanto um demanda configurações adequadas, o outro oferece técnicas para explorar alternativas de configurações para atender essas demandas. Diante do exposto, este trabalho, em nível de mestrado, tem como objetivo desenvolver uma solução para o problema alocação de recursos em nuvem adotando técnicas de exploração do espaço de projetos. A motivação para combinar problemas de alocação com técnicas para solução de exploração do espaço de projetos é aproveitar a capacidade dessas técnicas para expandir e tratar questões como heterogeneidade das aplicações e dos recursos disponíveis.

Neste trabalho foi adotada uma infraestrutura de exploração de espaço de projetos de sistemas computacionais denominada MultiExplorer [27]. Essa infraestrutura foi utilizada como base para o projeto e desenvolvimento de uma extensão (MultiExplorer-VM) cujo objetivo é oferecer soluções alternativas de máquinas virtuais para o atendimento das aplicações que demandam recursos na nuvem computacional. O trabalho tem como diferencial em

relação aos demais trabalhos da literatura o uso de simulação, com a inclusão do simulador CloudSim [18] no fluxo de execução do MultiExplorer-VM. Os resultados mostram que o MultiExplorer-VM alcança resultados significativos (melhores) do que a técnica *PI*. Os resultados de custo trazidos pelo MultiExplorer-VM foram até 8,8 vezes menores comparado com a técnica *PI*. Os experimentos também revelam que, para a maioria das aplicações, o MultiExplorer-VM alcançou a configuração de nuvem ideal.

O texto dessa desta dissertação está estruturado da seguinte forma:

- A fundamentação teórica com os conceitos sobre problemas de alocação de recursos em nuvem e de exploração de espaço de projeto é apresentada no Capítulo 2.
- no Capítulo 3 apresenta-se uma revisão detalhada da literatura e discussão dos trabalhos relacionados ao tema deste projeto.
- O Capítulo 4 detalha o desenvolvimento da extensão MultiExplorer-VM e o desenvolvimento e avaliação dos preditores de tempo e custo para máquinas virtuais em nuvem utilizados na extensão.
- O Capítulo 5 apresenta todo o procedimento experimental, de validação e discussão dos resultados obtidos, assim como comparações com outras técnicas disponíveis na literatura da área.
- O Capítulo 6 apresenta as conclusões finais e proposições de trabalhos futuros.

# Capítulo 2

# Referencial Teórico

Neste capítulo serão abordados os referenciais teóricos que contribuem para melhor compreensão dos assuntos envolvidos no trabalho. Conceitos acerca do problema de alocação de recursos em nuvem e de exploração de espaço de projeto, bem como técnicas e abordagens que propõem soluções para esses problemas.

## 2.1 Problema de Alocação de Recursos em Nuvem

Os provedores de serviços e infraestruturas de computação em nuvem permitem o acesso a poderosos recursos físicos computacionais sem ter a necessidade de adquiri-los ou gerenciá-los localmente. Essa disponibilidade de recursos e serviços oferece flexibilidade e baixo custo para usuários, empresas e, principalmente, para a comunidade científica. Aplicações de *Big Data* e de computação de alto de desempenho que demandam grande quantidade de recursos computacionais beneficiam-se da oferta desses recursos em ambiente de nuvem.

A mudança de aquisição e manutenção de recursos locais para utilização de recursos em nuvem gera questionamentos quanto ao desempenho e flexibilidade de gerenciamento desses recursos. Diversos trabalhos acadêmicos abordam, especificamente, a questão da alocação dos recursos necessários para as aplicações. Esse problema é conhecimento como alocação de recur-

sos en nuvem [6].

O problema de alocação de recursos em nuvem computacional pode ser formalizado como um problema geral de otimização multiobjetivo, *Multiobjective Optimization Problem* (*MOOP*). Para uma carga de trabalho qualquer, o *MOOP* inclui um conjunto de $n$ variáveis de decisão (conjunto de $VM$s), $k$ funções objetivo (tempo de execução e custo) e $m$ restrições (tempo máximo de execução e custo máximo disponível). Funções objetivo e restrições são funções de variáveis de decisão. Isso pode ser expresso como

$$\begin{aligned} \text{Otimizar} \quad & y = f(x) = (f_1(x), f_2(x), \ldots, f_k(x)) && (2.1) \\ \text{Sujeito a} \quad & e(x) = (e_1(x), e_2(x), \ldots, e_m(x)) \leq 0 && (2.2) \\ \text{Onde} \quad & x = (x_1, x_2, \ldots, x_n) \in X \\ & y = (y_1, y_2, \ldots, y_k) \in Y \end{aligned}$$

em que $x$ é o vetor de decisão, $y$ é o vetor objetivo, $X$ é denotado como o espaço de decisão e $Y$ é chamado de espaço de objetivo. As restrições $e(x) \leq 0$ determinam o conjunto de soluções viáveis. Em geral, não existe uma única "melhor" solução, mas um conjunto de soluções, nenhuma das quais pode ser considerada melhor do que as outras quando todos os $k$ objetivos são considerados ao mesmo tempo. Isso decorre do fato de que pode haver objetivos conflitantes [6].

Em se tratando de modelos de otimização para o problema de alocação de recursos em nuvem, um objetivo comum é minimizar a diferença de desempenho entre a *VM* disponível e a *VM* ótima. Do mesmo modo, é importante a minimização do custo da solução, que está relacionada com a quantidade de recursos requeridos para alcançar uma configuração ótima ou próxima a ótima. Igualmente importante, na perspectiva da definição da técnica para resolver esse problema, é que a solução seja obtida em tempo viável. Dessa forma, busca-se métodos eficientes no tempo que ofereçam soluções para o problema de alocar recursos computacionais no ambiente de nuvem de forma a maximizar o desempenho da aplicação com o menor custo possível dispendido para adquirir esses recursos.

## 2.2 Abordagem para o Problema de Alocação de Recursos em Nuvem: Modelo Sequencial

Algoritmos de otimização baseada em modelo sequencial (*Sequential model-based optimization* (*SMBO*)) têm sido usados em muitas aplicações onde a avaliação da função objetivo é cara [40, 41]. Algoritmos baseados em modelo sequencial adotam um modelo substituto que é mais barato de avaliar [11]. A otimização baseada em *SMBO* encontra iterativamente soluções (tipos de $VM$s) para otimizar para um objetivo (tempo de execução ou custo) [42].

Um algoritmo *SMBO* típico para o problema de alocação de recursos em nuvem é apresentado no Algoritmo 1. O algoritmo *SMBO* requer 4 entradas, uma nuvem configurada para executar uma carga de trabalho ($w$), lista de características de $VM$s ($vm \in VM$) ou espaço de instância, uma função de aquisição ($S$), e uma escolha de modelo substituto ($M$). Na linha 1 tem-se uma amostra inicial de $VM$s (escolhidas aleatoriamente) que são então avaliadas na carga de trabalho $w$. Na linha 3, o *SMBO* cria um modelo substituto ou um modelo de aprendizado de máquina para estimar e prever o desempenho da carga de trabalho. O modelo é construído usando características da $VM$ e o desempenho medido. Na linha 4 uma $VM$ é selecionada com base no modelo substituto junto com uma função de aquisição predefinida. Na linha 5, a $VM$ selecionada ($vm_k$) é então mensurada na carga de trabalho. A $VM$ ($vm_k$) junto com o desempenho ($y_k$) são adicionados ao conjunto de $VM$s já medidas ($D = (vm_1, y_1), (vm_2, y_2), \ldots, (vm_n, y_n)$) na linha 6. Entre as linhas 7-9, esse processo termina depois que um critério de parada é satisfeito [37].

**Algoritmo 1:** Otimização baseada em modelo sequencial.

**Entrada:** *w, VM, S, M*
**Saída :** $D$ (Configurações próximas à ótima)

```
1  D := amostragem inicial (w, VM)
2  for k em |VM ∉ D| do
3  |  p(D) := ajusta um modelo substituto (M, D)
4  |  vm_k := argmax_{vm∈VM} S(p(D))
5  |  y_k := w(vm_k)
6  |  D := D ⋃ (vm_k, y_k)
7  |  if critério de parada é verdadeiro then
8  |  |  break
9  |  end
10 end
```

Os processos gaussianos têm sido utilizados como método para gerar modelos substitutos [51]. Ao escolher o Processo Gaussiano, assume-se que a função aproximada é uma amostra do Processo Gaussiano. Como o Processo Gaussiano é não paramétrico, ele é flexível o suficiente para aproximar a função real, dadas amostras de dados suficientes. Quanto mais próximo o Processo Gaussiano estiver de uma função real, menos amostras de dados e buscas serão necessárias [37].

## 2.3 Problema da Exploração de Espaço de Projeto

A exploração do espaço de projeto - *Design Space Exploration* (*DSE*) - refere-se à atividade de explorar soluções alternativas para o projeto de um sistema antes da sua implementação. O fato de trabalhar no espaço de soluções potenciais torna o *DSE* útil para uma variedade de projetos visando alcançar [43]:

**Prototipagem rápida:** utilizada para gerar um conjunto de protótipos antes da implementação de fato;

**Otimização:** o *DSE* é aplicado na execução da otimização, empregando métricas de comparação entre os projetos, possibilitando eliminar projetos inferiores;

**Integração de sistemas:** integrar um sistema exige a montagem e configuração de diversos componentes. A exploração do espaço de projeto pode ser usada para conjuntos válidos e configurações que atendam as restrições globais impostas ao projeto.

Conforme [8], a exploração de espaço de projeto pode ser representada por um fluxograma como na Figura 2.1. Começando com uma configuração inicial, o processo de exploração é composto pelas fases de avaliação e ajuste de parâmetros. A fase de avaliação pode ser resumida a uma simulação em nível de sistema. A fase de ajustes utiliza os resultados da fase de avaliação para modificar os parâmetros de configuração do sistema, com a intenção de otimizar os índices de desempenho. O ciclo se encerra quando é obtida uma configuração de sistema que cumpra as restrições de projeto.

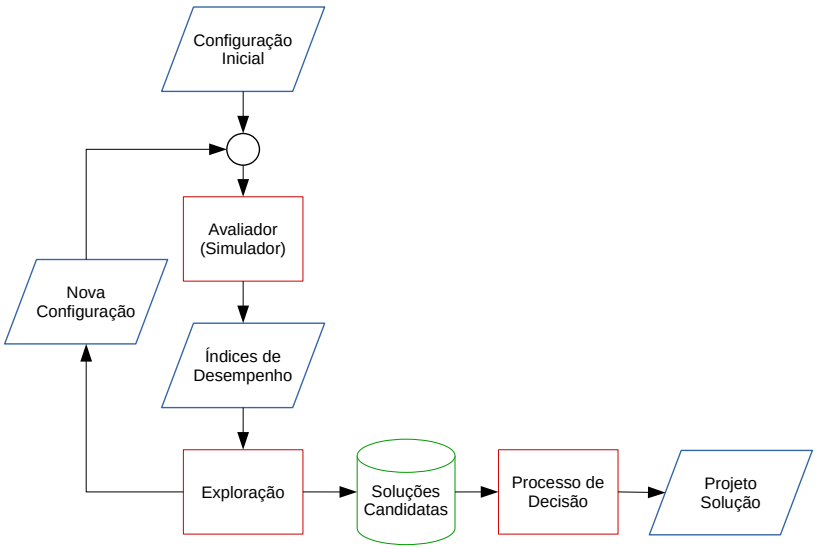



Figura 2.1: Fluxograma da exploração de espaço de projeto [8].

Assim como o problema de alocação de recursos em nuvem, o problema de exploração de espaço de projeto também pode ser formalizado como um *MOOP*. Dado um conjunto de $n$ variáveis de decisão, que são os graus de liberdade (por exemplo, parâmetros como o número e tipo de processadores, mapeamento de aplicativos, etc.) que são explorados durante o *DSE*, uma função de aptidão deve otimizar as $k$ funções objetivo respeitando $m$ restrições. A função de aptidão $f_i$ traduz um ponto no espaço de solução $X$ no

$i$-ésimo valor objetivo (onde $1 \leq i \leq k$). Por exemplo, uma função de aptidão particular $f_i$ poderia avaliar o desempenho ou a eficiência energética de uma determinada solução $x$ (representando uma instância de projeto específica). A função de aptidão combinada $f(x)$ posteriormente traduz um ponto no espaço de solução para o espaço objetivo $Y$. Formalmente, o *DSE* pode ser formulado como um problema de otimização multiobjetivo (*MOOP*) que tenta identificar uma solução $x$ para as $n$ variáveis de decisão que minimiza os $k$ valores objetivos usando funções objetivo $f_i$ com $1 \leq i \leq k$ e respeitando o conjunto de restrições $e(x)$

$$\begin{aligned} \text{Otimizar} \quad & y = f(x) = (f_1(x), f_2(x), \ldots, f_k(x)) && (2.3) \\ \text{Sujeito a} \quad & e(x) = (e_1(x), e_2(x), \ldots, e_m(x)) \leq 0 && (2.4) \\ \text{Onde} \quad & x = (x_1, x_2, \ldots, x_n) \in X \\ & y = (y_1, y_2, \ldots, y_k) \in Y. \end{aligned}$$

As restrições $e(x) \leq 0$ determinam o conjunto de soluções viáveis [59].

Observando as formalizações dos problemas de alocação de recursos em nuvem e de exploração de espaco de projeto pode-se notar que as equações 2.3 (função-objetivo) e 2.4 (restrições) são similares as equações 2.1 e 2.2 do problema de alocação de recursos em nuvem. Entende-se que, a partir de eventuais adequações de ajustes dos modelos, técnicas e metodologias utilizadas para encontrar soluções para o problema de exploração de espaço de projeto podem ser aplicadas para o problema de alocação de recursos em nuvem.

## 2.4 Abordagem Evolutiva para Exploração do Espaço de Projeto

A exploração de espaço de projeto aplicada em situações onde há necessidade de definir uma configuração de máquina (hardware) ideal, adota técnicas em que um modelo arquitetural é refinado passo-a-passo. Esse modelo possui características multiobjetivo pois, comumente, compreende a otimização de área, desempenho, consumo energético, custo de aquisição, entre outros. Assim, otimizar refere-se a encontrar uma solução que forneça os valores de todas as funções objetivo aceitáveis para o projeto [73].

Não se pode melhorar o valor de uma determinada função objetivo sem piorar o valor de outra, pois geralmente as funções objetivo aplicadas aos *MOOP*s são conflitantes entre si. A dominância de Pareto, comumente encontrada na literatura, é empregada para comparar as soluções possíveis para um determinado problema de forma a eleger as soluções preferíveis. O conceito de dominância de Pareto diz que $f$ domina $g$ quando a solução de $f$ é superior a $g$ em, pelo menos, uma função objetivo e quando a solução de $f$ é, pelo menos, igual $g$ em todas as funções objetivo. Uma solução é preferível se não existir solução que melhore pelo menos uma das funções objetivos sem piorar outra. Assim, a fronteira de Pareto é formada pela coleção de soluções não dominadas. Um dos problemas da dominância de Pareto é o número elevado de soluções dominantes para problemas com mais de três objetivos [65]. A Figura 2.2 apresenta um conjunto de soluções viáveis $a, b, c$. O ponto $b$ domina o ponto $c$ uma vez que $f_1(b)$ e $f_2(b)$ são menores que $f_1(c)$ e $f_2(c)$. Neste caso, portanto, têm-se que a fronteira de Pareto aproximada para o exemplo são os pontos $a, b$.

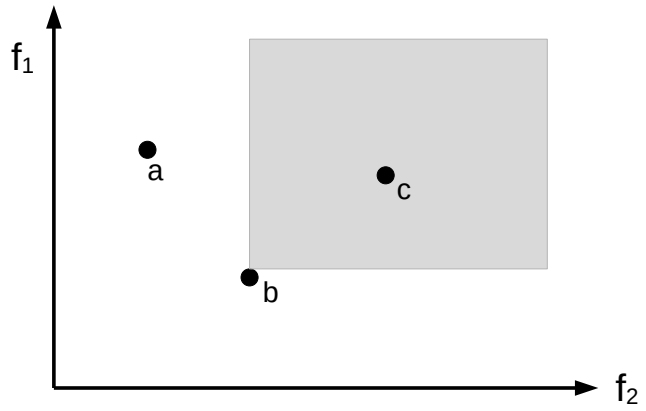



Figura 2.2: O conceito de dominância de Pareto para a, b, c, em que o ponto c é dominado pelo ponto b [65].

A abordagem evolutiva, via algoritmos evolutivos, é uma técnica amplamente adotada em problemas multiobjetivo de exploração de espaço de projeto pois lidam simultaneamente com um conjunto de soluções possíveis, o que permite encontrar vários elementos de Pareto definidos em uma única execução do algoritmo [22]. Especificamente, algoritmos genéticos - *Genetic Algorithms* (*GA*) - têm sido aplicados em vários tipos de situações que envolvem *DSE*. Algoritmos genéticos requerem a definição de apenas uma re-

presentação da configuração, os operadores e as funções genéticas objetivas para serem otimizadas. Além disso, não necessita de um conhecimento detalhado do sistema, por exemplo, a arquitetura ou parâmetro de dependência interna [8].

O algoritmo genético com ordenação não dominada II (*Non-dominated Sorting Genetic Algorithm II* (*NSGA-II*)) [26], é uma variação do algoritmo *NSGA* [67] e é baseado em uma ordenação elitista por dominância (Pareto *ranking*), em que combina a população atual com a população anterior para preservar os melhores indivíduos. O algoritmo é dividido em três componentes: seleção rápida não dominada (*fast nondominated sorting*), distância de agrupamento (*crowding distance*) e o laço principal.

O algoritmo utiliza uma abordagem de seleção rápida não dominada. Para cada solução é calculada a contagem de domínio, o número de soluções que a dominam, e o conjunto de soluções que são dominadas. Todas as soluções da primeira fronteira não dominada terão zero como contagem de dominação. O processo é realizado para cada solução e, a cada iteração, são retiradas as soluções que não são dominadas, diminuindo-se do contador das soluções dominadas. A cada repetição uma nova fronteira é criada com as soluções retiradas do conjunto. Os indivíduos localizados nas primeiras fronteiras, próximos da fronteira de Pareto, são os que apresentam as melhores soluções para a geração [64]. Utilizando o critério de dominância, o algoritmo une o conceito de elitismo que classifica a população em diferentes níveis de qualidade, permitindo a priorização dos indivíduos melhores classificados.

Após a criação das fronteiras, um cálculo de distância entre as soluções da mesma fronteira é realizado, de modo a assegurar uma seleção melhor para as soluções localizadas em áreas menos povoadas. Soluções com valores mais longínquos recebem infinito para a distância e as outras soluções têm sua distância normalizada com as duas soluções mais próximas, formando um cuboide em relação ao ponto central, cujos vértices são seus vizinhos mais próximos [64]. A ideia é distribuir os resultados ao longo da fronteira de Pareto, conforme ilustrado na Figura 2.3.

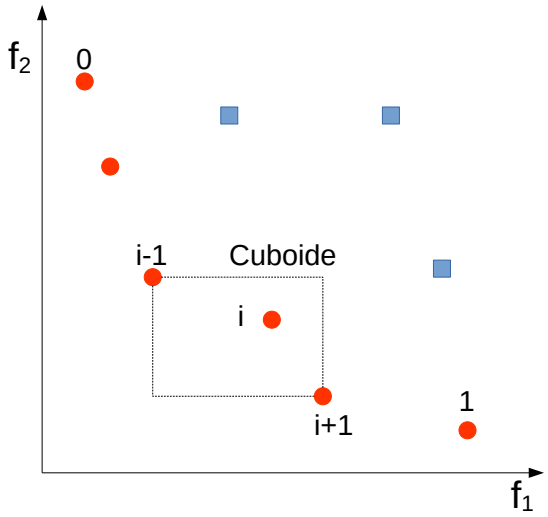



Figura 2.3: Cálculo do *crowding distance*. Círculos representam soluções não dominadas da mesma fronteira [26].

Depois da formação de uma população aleatória $P_0$, ordenação com base na não dominação e classificação de acordo com seu grau de dominação, o algoritmo utiliza um processo de seleção por torneio, cruzamento e mutação para obter a população filha $Q_0$, ambas de tamanho $n$. As populações $P_0$ e $Q_0$ são unidas para gerar uma população $R_0$.

Posterior ao procedimento inicial, um novo processo para as próximas gerações é considerado para a criação da *t-ésima* geração, $R_t$ (Figura 2.4). Agora, as soluções pertencentes ao melhor conjunto não dominado $F_1$, devem ser enfatizadas com relação a todas as outras soluções combinadas da população. Se o tamanho de $F_1$ for menor do que $n$, soluções da fronteira seguinte são escolhidos sucessivamente até que a população esteja completa (tamanho $n$). A nova população $P_t + 1$ de tamanho $n$ é agora usada para seleção, cruzamento e mutação para a criação da população $Q_t + 1$, e assim sucessivamente. Para escolher uma solução em detrimento de outra será comparado se a mesma possui um *ranking* menor, melhor nível de não dominância. Quando ambas possuírem o mesmo *ranking*, como critério de desempate prevalecerá a que possuir o maior *crowding distance*.

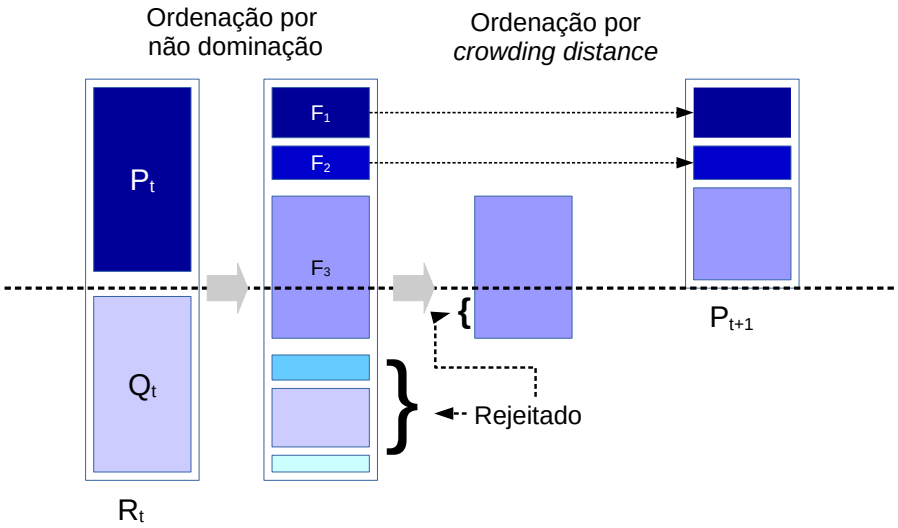



Figura 2.4: Procedimento para geração de novas populações no NSGA-II [26].

A complexidade de pior caso para esse procedimento é $O(mn^2)$, onde $m$ é o número de objetivos e $n$ o tamanho da população [26]. A principal vantagem do algoritmo *NSGA-II* está na manutenção da diversidade entre as soluções não dominadas, no método de comparação por multidão e na dispensabilidade da criação de nichos. Suas desvantagens são para uma fronteira $F_1$ maior que $n$, pois soluções são perdidas e o algoritmo entra num ciclo de soluções Pareto-ótimas e soluções não Pareto-ótimas até confluir para um conjunto de soluções Pareto-ótimas [25]. O pseudocódigo do *NSGA-II* é apresentado no Algoritmo 2 [24].

**Algoritmo 2:** Algoritmo NSGA-II.

**Entrada:** Parametros relevantes ao NSGA-II
**Saída :** Soluções nas populações

1 Gerar a população aleatória inicial $P_0$ de tamanho $n$;
2 Atribui $t = 0$ (iteração atual);
3 Ordenar $P_0$ conforme a não dominância;
4 Gerar população $Q_0$, de tamanho $n$, a partir de operadores de cruzamento e mutação em $P_0$;
5 **while** *critério de parada não é satisfeito* **do**
6 $R_t = P_t \cup Q_t$;
7 $F =$ ordenar $R_t$ por não dominância;
8 $F = (F1, F2, F3, \ldots)$, todos os $fronts$ de $R_t$;
9 **while** $|P_t + 1| < n$ **do**
10 Atribuir *crowding distance* para cada solução em $F_t$;
11 $P_{t+1} = P_{t+1} \cup F_t$;
12 **end**
13 Ordenar $P_{t+1}$ com base em não dominância e *crowding distance*;
14 $P_{t+1} = P_{t+1}[0 : n]$;
15 População $Q_{t+1}$ de tamanho $n$ é obtida a partir da aplicação dos operadores de seleção (baseado no *crowding distance*), cruzamento e mutação sobre a população $P_{t+1}$;
16 $t = t + 1$ e retorna para a linha 5;
17 **end**
18 Retorna população $P_t$;

## 2.5 Considerações Finais

Nesse capítulo foram apresentados os principais conceitos utilizados ao longo do desenvolvimento deste trabalho de mestrado. Os problemas de alocação de recursos em nuvem e de exploração de espaço de projeto constituem motivação e proposição de solução neste trabalho. Algumas técnicas utilizadas para resolver o problema de alocação e o problema de exploração de espaço de projeto também foram apresentadas e discutidas. Especificamente, o algoritmo NSGA-II que é a técnica base para resolução do *DSE* no âmbito deste trabalho. O próximo capítulo apresenta trabalhos científicos que abordam o problema de alocação de recursos em nuvem e traz um panorama das técnicas utilizadas na literatura da área para resolução desse problema.

# Capítulo 3

# Trabalhos Relacionados

Este capítulo apresenta trabalhos científicos com soluções para problemas relacionados ao proposto neste projeto. Os trabalhos a seguir propõem, em sua maioria, sistemas de software que utilizam preditores baseados em aprendizagem de máquina e outras técnicas de otimização como a otimização Bayesiana como potenciais soluções para alocação adequada de recursos no ambiente de nuvem computacional.

## 3.1 Sistemas para Predição de Desempenho em Ambiente de Nuvem Computacional

Técnicas de aprendizado de máquina têm sido amplamente utilizadas para construir modelos de previsão de desempenho para alocação de recursos na nuvem computacional. Ernest [70] e *PARIS* [72] são trabalhos que utilizam essas técnicas. A acurácia da predição depende muito da seleção de recursos, seleção de modelo e ajuste de parâmetros e da qualidade dos dados [38].

### 3.1.1 Ernest

O trabalho de [70] propõe um preditor de desempenho baseado em técnicas de aprendizado de máquina, cuja a ideia principal é executar um conjunto de instâncias do trabalho como amostras de entrada, e utilizar os dados proveni-

entes dessas execuções de treinamento para criar um modelo de desempenho. Essa abordagem apresenta menos de 5% de sobrecarga, pois em geral leva-se muito menos tempo para executar o conjunto de treinamento do que a carga completa de trabalho.

O modelo construído possui duas características principais, o número de linhas ou fração de dados usados, aqui chamados de escala ($e$), e o número de máquinas usadas para execução ($m$). O modelo linear é composto por quatro termos: (a) um valor fixo que representa o tempo gasto em computação serial, (b) a interação entre a escala e inverso do número de máquinas para capturar o tempo de computação paralela para algoritmos cuja computação escala linearmente com os dados, (c) um termo $log(m)$ para modelar padrões de comunicação como árvore de agregação, (d) um termo linear $O(m)$ que captura o padrão de comunicação todos-para-um e *overheads* fixos como escalonamento de tarefas.

Então, o modelo geral tenta aprender os valores para $\theta_0, \theta_1, \theta_2, \theta_3$ na equação 3.1, onde $y_t$ é tempo.

$$y_t = \theta_0 + \theta_1 \times (e \times \frac{1}{m}) + \theta_2 \times log(m) + \theta_3 \times m \tag{3.1}$$

A acurácia da predição do modelo foi avaliada por um conjunto de cargas de trabalho incluindo nove algoritmos de aprendizado de máquina que são parte do Spark MLlib [5], consultas do GenBase [68] e transformações *I/O intensive* usando ADAM [54] em genoma completo, e um *pipeline* de reconhecimento de fala que alcança resultados de estado da arte [39]. Um dos principais resultados obtidos neste trabalho é que o modelo preditivo atinge um erro menor que 20% nas medições de acurácia da predição para o conjunto de aplicações de *benchmark*.

A Figura 3.1 mostra a predição de tempo tomado por interação através de um amplo número de máquinas para a cargas de trabalho TIMIT *pipeline* [19]. Outro resultado deste trabalho é a melhoria de até 4 vezes no preço com a escolha do número ótimo de instâncias. Por exemplo, considere o caso de um usuário que tem o tempo limite de uma hora (3600s) para executar 40 iterações do TIMIT *pipeline* com um limite de instâncias até 64 máquinas. Usando a Figura 3.1 e levando em conta a margem de erro (20%), o modelo mostra que 16 instâncias são suficientes para cumprir o prazo ($73,73 \times 40 = 2949,2s < 3600s$). Dado que o custo de uma instância *r3.xlarge* é \$0,35/h, uma estratégia simples que utiliza todas as

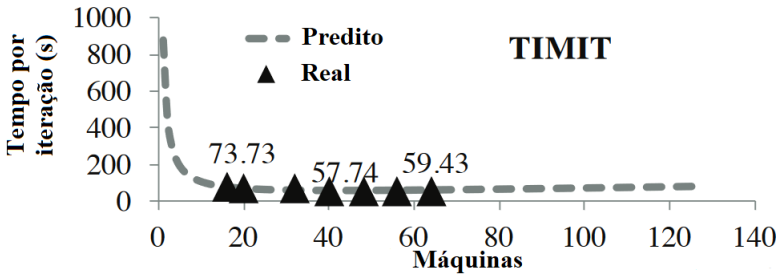



Figura 3.1: Tempo por iteração variando o número de instâncias para TIMIT pipeline [70].

64 máquinas custaria $22,4, enquanto usando 16 máquinas o custo seria de somente $5,6, uma diferença de 4 vezes. Similarmente, se o usuário tem um orçamento de $15 então é possível inferir que usando 40 máquinas seria mais rápido do que usar 64 máquinas, já que o tempo total com 40 máquinas é $57,74 \times 40 = 2309,6s$ e com 64 máquinas é $59,43 \times 40 = 2377,2s$

### 3.1.2 PARIS

O sistema *Performance Aware Resource Inference System* (*PARIS*) é proposto no trabalho [72]. Esse sistema estima o *trade-off* desempenho-custo para todos os tipos de *VM* aproveitando a técnica *Random Forest* [13], permitindo que os usuários equilibrem os ganhos de desempenho com as reduções de custo. *PARIS* apresenta uma nova coleção híbrida de dados *offline* e *online* e *framework* de modelagem que fornece estimativas precisas de desempenho com o uso mínimo de dados.

A Figura 3.2 exibe a arquitetura do sistema *PARIS* separada em dois estágios, *offline* e *online*, que são combinados para construir um modelo de aprendizado de máquina. No estágio *offline*, *PARIS* usa o Criador de Perfil para executar um conjunto de *benchmarks* para cada tipo de VM, e coletar métricas detalhadas de desempenho de sistema. No estágio *online*, o sistema caracteriza cada nova consulta de carga de trabalho (Gerador de impressão digital) executando uma tarefa indicada pelo usuário representativa da carga de trabalho, em um par de VMs de referência e coletando as mesmas estatísticas de perfil como no estágio *offline*.

*PARIS* reduz em 4 vezes o erro na predição de tempo de execução comparado a um modelo de interpolação linear. Entretanto, o sistema apresenta

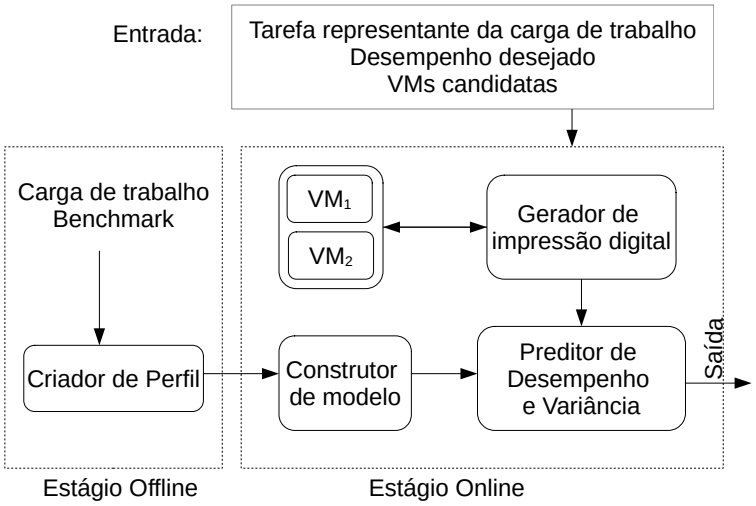



Figura 3.2: Arquitetura PARIS [72].

algumas fraquezas como a necessidade de o usuário fornecer uma tarefa representativa de uma carga de trabalho e não estimar o comportamento em escala.

## 3.2 Otimização Baseada em Modelo Sequencial

Para atingir o problema de alocação de recursos em nuvem outros autores estudaram técnicas de *SMBO* com destaque para Otimização Bayesiana. Técnicas *random-search* ou *grid-search* que buscam extensivamente o espaço de configurações poderiam ser usadas, mas para diminuir o número de configurações a serem observadas dinamicamente alguns trabalhos propuseram o uso de métodos estatísticos como Otimização Bayesiana usando Random Forrest (RF) ou Processo Gaussiano (GP). Essas abordagens reduzem o custo de pesquisa observando menos configurações [50]. Otimização Bayesiana com *Paramount Interation* [50], CherryPick [7]] e SCOUT [38] são trabalhos que utilizam técnicas *SMBO*.

### 3.2.1 Otimização Bayesiana com *Paramount Interation*

Para encontrar a melhor configuração que reduza custos e desperdício de recursos ao mesmo tempo que atinge um desempenho aceitável, o trabalho de [50] combina duas técnicas para realizar menos observações sobre o espaço de busca de alternativas. A primeira técnica é a Otimização Bayesiana que reduz o custo de busca ao observar a menor quantidade possível de configurações. A abordagem bayesiana fornece uma distribuição posterior da função caixa preta e estima a incerteza que ajuda a decidir onde encontrar um máximo (ou mínimo) da função objetivo. Os modelos mais frequentes usados são *Gaussian Process* ($GP$) e *Random Forest* ($RF$). A segunda técnica é a de *Paramount Interation* ($PI$) que reduz o custo de uma observação, extraindo o desempenho de diferentes tipos de máquinas virtuais com pouco tempo de execução, apenas coletando informações sobre iterações do ciclo de execução principal do aplicativo, ou *Paramount Interation*, conforme o trabalho [14].

O *benchmark* utilizado no trabalho foi o *Numerical Aerodynamic Simulation Parallel Benchmarks* ($NPB$) [9] em máquinas virtuais *AWS*. *NPB* é um conjunto de *benchmarks* de desempenho de computação paralela que possui cinco *kernels*. Três classes de dados de entrada foram definidas, totalizando 15 cargas de trabalho. O espaço de pesquisa de configurações de nuvem explorados nos experimentos foi composto por todos os pares $(vm, n)$ onde $vm$ indica uma das 32 máquinas virtuais ($VM$s) e $n \in \{1, 2, 4, 8, 16, 32\}$, indica o número de instâncias usadas na configuração. Portanto, um total de 192 configurações ($32 \times 6$=192) possíveis por programa. Para encontrar a configuração de nuvem que minimize o custo da execução de uma carga de trabalho $w$, o custo $c$ é calculado pelo tempo ($t(w)$) para executar o programa $w$, o preço ($p(vm) \times\ n$) de cada $vm$ pelo número de instâncias $n$, e um ruído aleatório $\epsilon$ que vem do contexto do provedor de nuvem e o mecanismo de medição.

$$c(w_{vm,n}) = t(w) \times p(vm) \times n \times \epsilon \tag{3.2}$$

Para cada uma das 2880 execuções ($15 \times 192$), obteve-se o tempo de execução das primeiras quatro *Paramount Interation*. Então, o custo das configurações é calculado usando a Equação 3.2 para formar o espaço de custo de nuvem onde foram aplicadas as buscas. Foram usados dois tipos de busca nos experimentos: busca aleatória e Otimização Bayesiana.

O trabalho concluiu que as abordagens de Otimização baseada em modelo sequencial são muito mais eficientes do que a busca aleatória com resultados 1,68 vezes melhores. Outra conclusão é que a técnica de *Paramount Interation* usada para comparar o desempenho das cargas de trabalho enquanto interrompe o aplicativo antecipadamente pode ser usada para reduzir o custo de pesquisa sem afetar seus resultados. Comparado a busca aleatória, a utilização de *SMBO* com *PI* alcançou uma redução de custo de busca de 6 vezes.

### 3.2.2 CherryPick

O trabalho de [7] apresenta o sistema CherryPick que obtém configurações de nuvem ideais ou quase ideais que minimizam o custo de uso, garantem o desempenho do aplicativo e limitam a sobrecarga de pesquisa para trabalhos analíticos de *Big Data* recorrentes.

CherryPick também utiliza modelos de Otimização Bayesiana, com intuito de julgar qual configuração de nuvem deve ser amostrada em seguida para reduzir a incerteza na modelagem e se aproximar do ideal. O CherryPick usa o intervalo de confiança para decidir quando interromper a pesquisa.

O sistema foi avaliado utilizando cinco cargas de trabalho com 66 configurações no *Amazon Elastic Compute Cloud* (*Amazon EC2*). Os resultados mostraram que CherryPick possui entre 45%-90% de possibilidade de escolher a configuração ideal e, de outra forma, pode-se encontrar uma solução quase ideal (a uma distância de 5% na mediana). Soluções alternativas, como busca aleatória, podem levar até 75% mais tempo de execução e 45% mais custo de pesquisa (despesa para executar todas as configurações amostradas). Comparando os sistemas CherryPick e Ernest [70], as configurações retornadas pelo CherryPick podem melhorar o tempo de pesquisa em 90% e o custo de pesquisa em 75% para consultas *SQL*.

### 3.2.3 SCOUT

Com a ideia central de reduzir significativamente o custo da busca pela melhor configuração de nuvem para uma carga de trabalho usando dados históricos, o trabalho [38] apresenta o sistema SCOUT.

A estratégia de busca combina duas técnicas, a primeira é a utilização de uma técnica de modelagem em pares usando ExtraTrees [31]. O modelo de desempenho utiliza a função de mapeamento $y(Y(vm(w_i)), Y(vm(w_j)), l(vm(w_i)) = C_{w_{ij}}$ onde $vm(w_i)$ é uma configuração de nuvem para uma carga de trabalho $w_i$, $Y$ é a função que representa as características de uma configuração de nuvem, $l(vm(w_i))$ são as métricas de baixo nível coletadas durante a execução de uma carga de trabalho em uma configuração de nuvem $vm(w_i)$ e $C_{w_{ij}}$ é a previsão da classe ("boa", "regular" ou "ruim") à qual $vm(w_j)$ pertence.

A segunda é uma estratégia que escolhe a configuração com a maior probabilidade de predição. Essa estratégia possui um baixo custo de pesquisa, aumentando a velocidade de convergência. O desempenho e o custo da busca do SCOUT foi examinado utilizando 107 cargas de trabalho. O Scout encontra as configurações quase ideais (com uma diferença de 10%) para 87% das cargas de trabalho. SCOUT apresenta melhores resultados do que CherryPick em todas as medidas (tempo de execução, custo de implantação e custo de pesquisa). A variação no desempenho (em termos de tempo de execução e custo de implantação) do SCOUT é menor do que os outros métodos (busca aleatória, descida coordenada e CherryPick).

## 3.3 Análise e Discussão dos Trabalhos Relacionados

Provedores de nuvem como *Amazon Web Services* [1], *Google Cloud* [2], *Microsoft Azure* [4] fornecem a seus usuários uma diversidade de tipos de máquinas virtuais e configurações de nuvem para serem instanciadas. Existem diferentes custos e desempenhos dependendo da instância e provedor escolhidos. Cada provedor de nuvem busca ajudar o usuário na escolha entre os tipos de *VM*s, e nesse aspecto, vários trabalhos na literatura da área buscam encontrar a melhor configuração de *VM*.

Há propostas que buscam resolver o problema de alocação de recursos na nuvem extraindo informações de múltiplas execuções para modelar esses recursos e o desempenho das aplicações [72], [70]. Essas informações coletadas são utilizadas para treinar algoritmos de aprendizagem de máquina visando recomendar configurações de nuvem. Ernest e *PARIS* são exemplos de trabalhos que seguiram por esse caminho.

Ernest [70] explora a estrutura interna de uma carga de trabalho para predizer o tempo de execução. A predição precisa apenas de uma pequena quantidade de dados como entrada, reduzindo significativamente o custo de medição. Entretanto, Ernest possui limitações devido ao modelo de predição ser específico ao tipo da *VM*, não conseguindo prever a alta variação de desempenho da nuvem [30], causada principalmente pelo uso simultâneo dos recursos físicos de diferentes *VM*s e usuários[50].

*PARIS* [72] constrói um modelo *Random Forest* baseado em medições para realizar predições de tempo de execução ou custo de operação de cargas de trabalho. Essa abordagem possui fraquezas como requerer uma quantidade significativa de dados para construir o modelo e não considerar que o desempenho do ambiente em nuvem é dinâmico [46], devido ao compartilhamento dos recursos físicos por diferentes *VM*s e usuários. Então, as medições coletadas após a execução de uma carga de trabalho podem não refletir o desempenho real do ambiente em nuvem.

Percebendo as dificuldades encontradas pelas abordagem baseadas em aprendizagem de máquina, alguns autores voltaram-se para a exploração do espaço de busca das configurações de instâncias. Técnicas como busca aleatória e busca em rede, alcançaram desempenho inferior quando comparadas com métodos estatísticos como Otimização Bayesiana com *Random Forest* [37], [50] ou Processo Gausiano [7]. Essas aproximações têm o benefício de busca de baixo custo por observar poucas configurações. Em [50] o custo da busca é ainda menor com o uso de *Paramount Interation* que diminui o custo de cada observação.

A Tabela 3.1 mostra a relação entre trabalhos, suas principais características e técnicas usadas para a busca da melhor configuração em nuvem. Entre as características e técnicas estudadas são destacadas:

**Otimização Bayesiana:** técnica utilizada para reduzir o custo da busca pela melhor configuração em nuvem;

**Métricas de baixo nível:** são conhecimentos essenciais de sistemas, como utilização de *CPU*, quantidade de memória efetivamente usada e tempo de espera de *I/O*, que podem ser manipulados para auxiliar na seleção de recursos em nuvem;

**Dados históricos:** são dados de desempenho de *VM* originados de execuções anteriores, usados na predição de desempenho;

**Provedor de nuvem:** utilização de provedores de nuvem conhecidos (Amazon[1], Google[2] e Azure[4]) para ter acesso às instâncias e executar os experimentos;

**Simulação:** utilizada para representar o ambiente de nuvem e ajudar a entender, através dos dados gerados, o impacto da escolha da configuração de nuvem na execução da aplicação;

**Heterogeneidade:** possibilidade de combinar dois ou mais tipos de instância em uma configuração de nuvem;

**Elasticidade:** mais de uma *VM* pode ser alocada para a configuração em nuvem desejada.

Tabela 3.1: Síntese dos trabalhos relacionados.

| Trabalhos | Otimização Bayesiana | Métricas de baixo nível | Dados históricos | Provedor de nuvem | Simulação | Hetero-geneidade | Elasti-cidade |
|---|---|---|---|---|---|---|---|
| Ernest [70] | Não | Não | Sim | Sim | Não | Não | Sim |
| PARIS [72] | Não | Sim | Sim | Não | Não | Não | Sim |
| CherryPick [7] | Sim | Não | Não | Sim | Não | Não | Sim |
| SCOUT [38] | Sim | Sim | Sim | Não | Não | Não | Sim |
| OB com PI [50] | Sim | Não | Não | Sim | Não | Não | Sim |
| **Proposta** | **Não** | **Não** | **Sim** | **Não** | **Sim** | **Sim** | **Sim** |

Conforme pode ser observado na Tabela 3.1, a proposta apresentada neste trabalho se difere dos demais por não utilizar o provedor de nuvem em nenhuma etapa. Essa decisão leva a uma economia de custos e ao uso de simulação para, utilizando a ferramenta CloudSim [32], obter os dados de desempenho que alimentarão o preditor. Outros trabalhos que não utilizam provedores de nuvem usam *clusters* próprios para realizar seus experimentos, são os casos do *PARIS* [72] e SCOUT [38]. Ernest [70] usa provedor de nuvem para obter os dados históricos de execuções passadas, enquanto OB com *PI*[50] lança várias instâncias na nuvem *AWS* [1] e extrai o desempenho das cargas de trabalho com uma técnica de *early stop* que executa apenas uma pequena porção de execução total. Já, CherryPick [7] realiza um monitoramento executando a cada poucas horas uma pequena quantidade de instâncias com objetivo de medir o ruído da nuvem, evitando o aumento de custo de pesquisa que uma superistimação do ruído causaria.

Outro ponto de destaque da proposta deste trabalho é permitir elasticidade e configuração de nuvem heterogênea. Entende-se que tais características podem levar a melhores relações de custo $\times$ tempo de execução para determinar a configuração de instâncias de *VM*s adequadas para a carga

de trabalho. Observa-se que a heterogeneidade na busca por instâncias não é uma característica suportada pelos trabalhos estudados no levantamento bibliográfico, que implementam apenas a elasticidade.

## 3.4 Considerações Finais

Neste capítulo foram apresentados trabalhos da literatura e técnicas utilizadas para enfrentamento do problema de alocação de recursos em nuvem. Inicialmente apresentou-se os trabalhos que foram pioneiros ao debruçarem-se sobre o tema. Tais trabalhos usavam técnicas de aprendizado de máquina. Entretanto, essas técnicas entregavam resultados inferiores comparados com trabalhos que utilizam otimização Bayesiana, sendo esta, atualmente, a técnica presente em trabalhos com melhores resultados quanto à questão da alocação de recursos em nuvem. Uma nova abordagem para o problema de alocação de recursos computacionais no ambiente de nuvem, objeto de estudo desta proposta, será apresentada com maior nível de detalhes no Capítulo 4.

# Capítulo 4

# Projeto e Desenvolvimento da Extensão MultiExplorer-VM

## 4.1 Considerações Iniciais

O MultiExplorer [27][1] é uma ferramenta para exploração de espaço de projeto desenvolvida no Laboratório de Sistemas Computacionais de Alto Desempenho (LSCAD) da Universidade Federal de Mato Grosso do Sul. Esta ferramenta possui como principal característica um fluxo automatizado de simulação, estimativas físicas e exploração do espaço de projetos de sistemas multiprocessados, que considera restrições e objetivos arquiteturais informados pelo usuário.

Considerando a característica multiobjetivo do problema de Alocação de Recursos na Nuvem Computacional, este trabalho atuou na extensão das técnicas e mecanismos disponíveis no MultiExplorer para resolução desse problema. Para tanto, uma extensão do MultiExplorer (denominada MultiExplorer-VM) foi projetada e desenvolvida contemplando simulação de aplicações e a determinação de configurações de máquinas virtuais de acordo com restrições e objetivos dos usuários.

[1]Disponível em https://github.com/lscad-facom-ufms/multiexplorer

## 4.2 MultiExplorer: Exploração de Projetos em Sistemas Computacionais

A ferramenta MultiExplorer [27], Figura 4.1, tem como foco a automatização e experimentação de diversas plataformas computacionais. O usuário do MultiExplorer pode definir, em alto nível, parâmetros arquiteturais e simular uma proposta de arquitetura, obter estimativas físicas ou mesmo explorar alternativas arquiteturais considerando restrições e objetivos em termos de área, desempenho e potência. Para isso, foi desenvolvido um ambiente integrado para simulação e projeto de sistemas multicore heterogêneos utilizando simuladores de desempenho (MPSoCBench [28], Multi2Sim [69] e Sniper [20]), ferramentas de estimativas de parâmetros físicos para sistemas computacionais (*Multicore Power, Area and Timing* (*McPAT*) [47, 48]) e algoritmos para exploração do espaço de projeto ciente de *dark silicon* [63][64]. Uma visão geral dos recursos disponíveis e fluxograma de execução do MultiExplorer é apresentado na Figura 4.1.

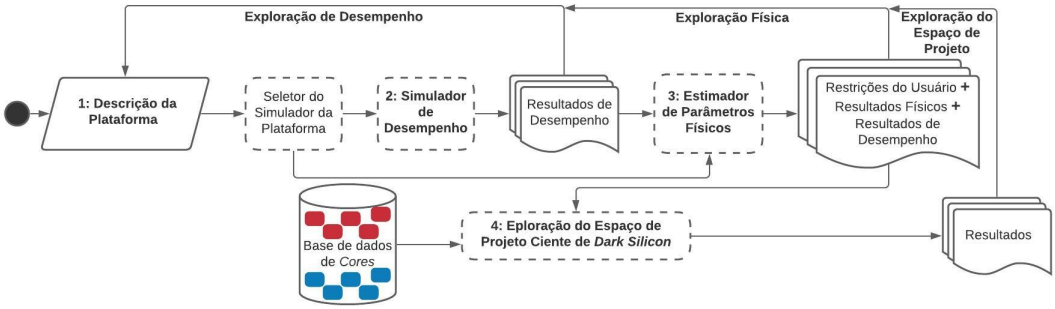



Figura 4.1: Fluxo de execução do MultiExplorer [64].

Na Figura 4.1 observa-se a presença de quatro etapas principais no fluxo de execução do MultiExplorer:

1. O usuário especifica parâmetros iniciais em um arquivo contendo as descrições da plataforma alvo. O Multiexplorer possui um módulo seletor para definir qual a ferramenta de simulação mais adequada diante das aplicações do usuário;

2. Os parâmetros definidos pelo usuário, assim como as configurações padrões da ferramenta, são passados para o simulador que deve simular a execução de aplicações sobre a especificação arquitetural;

3. Parte dos resultados da simulação são utilizados por ferramenta de estimativas físicas visando caracterizar esse projeto de plataforma computacional;

4. O módulo de exploração do espaço de projeto apresenta alternativas arquiteturais (configurações de recursos físicos computacionais) que atendem requisitos e demandas como área do chip, densidade de potência e desempenho da aplicação. Esse módulo utiliza-se de um banco de dados de núcleos de exploração que foi previamente definido e é a fonte para a busca dessas alternativas arquiteturais. Há dois algoritmos de exploração implementados no módulo: um algoritmo de busca exaustiva (força bruta) que retorna todas as possíveis configurações de acordo com os objetivos do projeto; e o algoritmo *NSGA-II*, baseado em computação genética, para explorar a combinação entre núcleos visando a determinação de novas configurações arquiteturais.

Em se tratando do problema de alocação de recursos na nuvem computacional, observa-se constantemente a necessidade de determinar uma ou mais máquinas virtuais que atendam a múltiplos objetivos, por exemplo: maximizar o desempenho da aplicação, minimizar custo, minimizar tempo de execução, minimizar a quantidade de máquinas do cluster, minimizar tempo de comunicação entre máquinas do cluster [16, 18]. Diante desse contexto, uma solução para o problema de alocação de recursos em nuvem pode ser obtido a partir de técnicas de exploração do espaço de projetos [62, 61]. Essa associação é possível uma vez que a exploração pode contemplar tanto a possibilidade de utilizar várias instâncias de uma mesma máquinas virtual (elasticidade) quanto a possibilidade de compor plataformas heterogêneas para o atendimento das demandas das aplicações.

## 4.3 Desenvolvimento da Extensão MultiExplorer-VM

A extensão MultiExplorer-VM[2] consiste na adição de um novo fluxo de execução junto à ferramenta MultiExplorer. Nesse fluxo, uma nova interface para o usuário é disponibilizada para que seja possível simular aplicações

[2]Disponível em https://github.com/lscad-facom-ufms/MultiExplorerVM

em máquinas virtuais, além de oferecer alternativas arquiteturais, via exploração do espaço de projeto, de acordo com as demandas dos usuários. A Figura 4.2 ilustra o fluxo de execução MultiExplorer-VM. De maneira geral, as principais alterações no fluxo original da ferramenta MultiExplorer foram:

- Na etapa de simulação de desempenho, aplicações foram selecionadas para obter o desempenho de máquinas virtuais. Nesse sentido, foi desenvolvida uma interface para que, a partir da entrada fornecida pelo usuário, o simulador de nuvem CloudSim [32] seja executado visando determinar o desempenho da aplicação(ões) sobre a(s) máquina(s) virtual(is) escolhida(s) como solução inicial[3].

- A etapa de estimar parâmetros físicos presente no MultiExplorer não é preponderante para resolução do problema de alocação de recursos na nuvem. O custo de processamento é o dado associado mais diretamente às características físicas de cada *VM*. Entretanto, o custo já é informado pelo provedor dos serviços de nuvem computacional assim que não é necessário estabelece-lo a partir de alguma ferramenta de estimativas.

- Na etapa de exploração do espaço de projeto, a solução proposta caracterizou um conjunto de máquinas virtuais e inseriu-as como elementos de alocação de recursos em um banco de dados (Figura 4.1). Utilizou-se, como referência de máquinas virtuais, instâncias *AWS*. Uma vez no banco de dados, essas instâncias serão utilizadas como núcleos de exploração de espaço de projetos (ajustando o algoritmo genético *NSGA-II* utilizado no módulo de exploração), para atender os restrições e objetivos de custo, tempo e desempenho definidos pelo usuário. Nesse sentido, a cada configuração de *VM* sugerida, preditores de tempo e custo devem ser utilizados para fornecer estimativas de performance e custo de acordo com as características dessa configuração. Ressalta-se que o algoritmo *NSGA-II* foi customizado visando um menor tempo de execução sem deixar de prezar pela qualidade dos resultados obtidos. A partir de experimentos preliminares, observou-se que configurações do *NSGA-II* com 50 gerações e 20 indivíduos foi a que apresentou a melhor relação entre as estimativas de tempo e custo.

[3]No contexto do MultiExplorer, essa solução arquitetural inicial é denominada solução original.

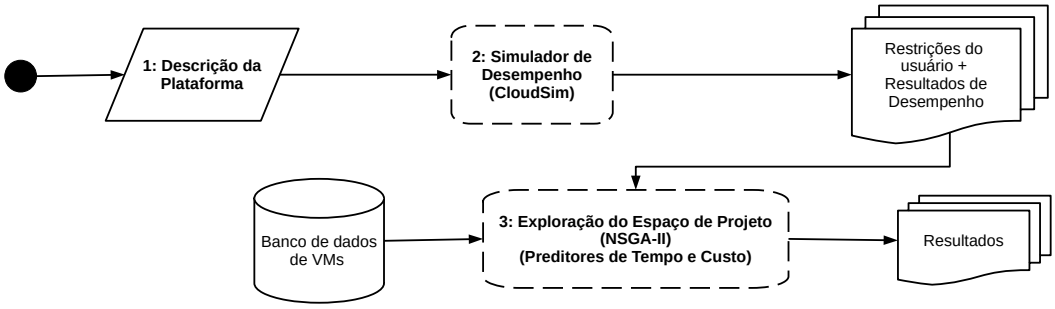



Figura 4.2: Fluxo de execução MultiExplorer-VM.

## 4.4 CloudSim

Com o intuito de oferecer um ambiente que possa ser usado diretamente pela comunidade de computação em nuvem foi proposto o CloudSim [18]. Um *framework* que permite modelagem, simulação e experimentação ininterruptas de infraestruturas e serviços de aplicativos de computação em nuvem.

A Figura 4.3 ilustra a implementação em camadas da estrutura do software CloudSim. Na camada mais baixa está o mecanismo de simulação de eventos discretos SimJava [36] que implementa as funcionalidades centrais necessárias para estruturas de simulação de nível superior. Em seguida estão as bibliotecas que implementam o kit de ferramentas GridSim [15] que suportam componentes de software de alto nível para modelar múltiplas infraestruturas Grid, incluindo redes e perfis de tráfego associados.

O CloudSim é implementado de forma a estender as principais funcionalidades expostas pela camada GridSim. CloudSim fornece suporte para modelagem e simulação de ambientes virtualizados baseados em nuvem, como interfaces de gerenciamento dedicadas para *VM*s, memória, armazenamento e largura de banda. A camada CloudSim gerencia a instanciação e execução de entidades principais (VMs, *hosts, data centers,* aplicativos) durante o período de simulação. Essa camada é capaz de instanciar e gerenciar de forma transparente uma infraestrutura de nuvem em larga escala que consiste em milhares de componentes do sistema. As questões fundamentais como provisionamento de hosts para *VM*s com base em solicitações de usuários, gerenciamento de execução de aplicativos e monitoramento dinâmico são tratados

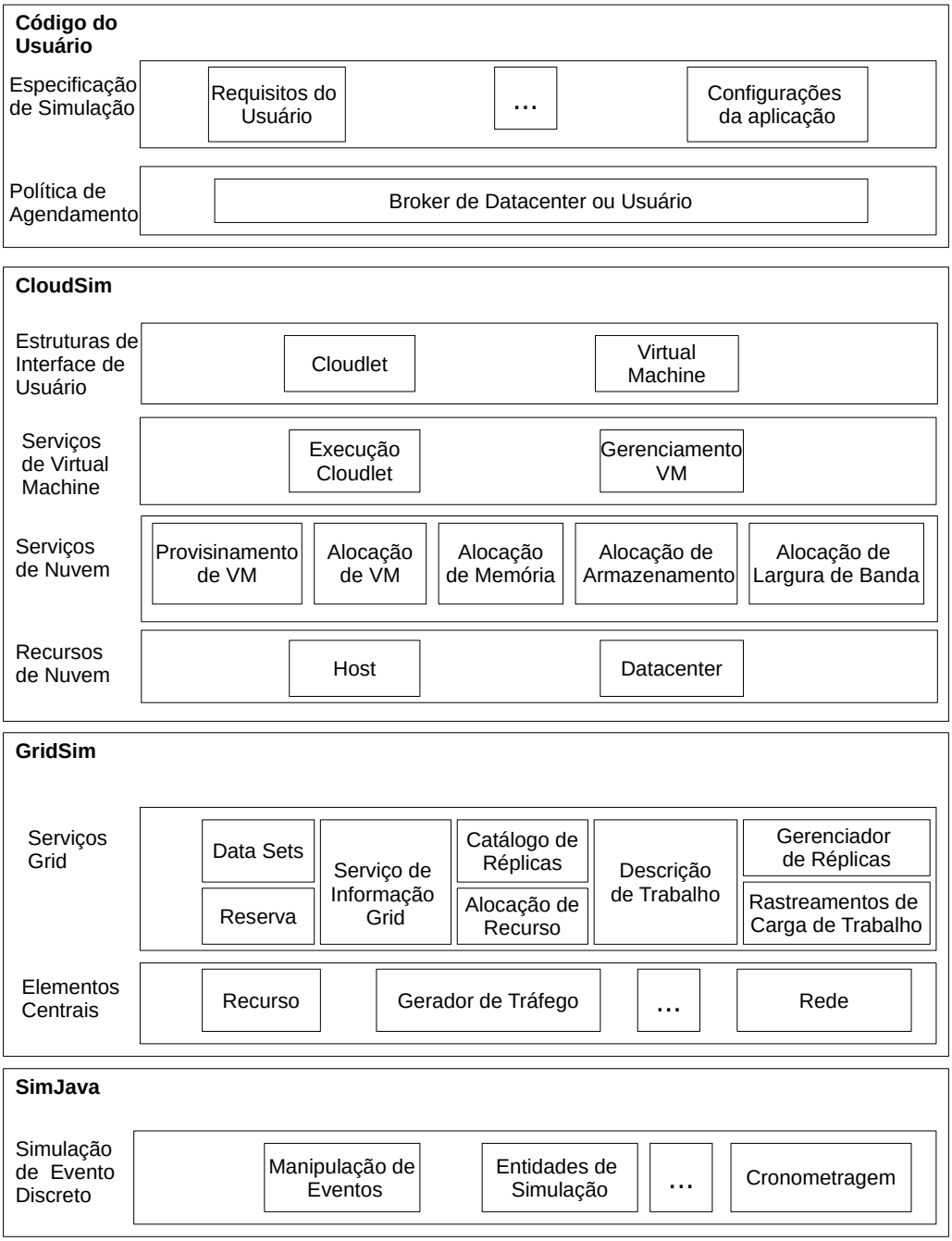



Figura 4.3: Arquitetura CloudSim em camadas. Baseado em [18].

por essa camada [18].

A camada superior na pilha de simulação é a Código do Usuário que expõe as funcionalidades relacionadas à configuração para hosts, aplicativos (número de tarefas e seus requisitos), $VM$s, número de usuários e seus tipos de aplicações e políticas de agendamento [18].

A *Cloudlet* é a classe que modela aplicações e serviços baseados em nuvem que são comumente empregados em *data centers*. CloudSim representa uma aplicação por meio dos requerimentos computacionais exigidos. Cada componente do aplicativo tem um comprimento de instrução pré-atribuído e a quantidade de transferência de dados que precisa ser realizada para hospedar o aplicativo com sucesso [18].

O CloudSim suporta provisionamento de $VM$ em dois níveis: primeiro, no nível do host e segundo, no nível da $VM$. No nível do host é possível especificar quanto do poder de processamento geral de cada *core* será atribuído a cada $VM$. No nível da $VM$, cada $VM$ atribui uma quantidade fixa de poder de processamento disponível aos serviços de aplicativos individuais (*cloudlets*) hospedados em seu mecanismo de execução [18]. Em cada nível, o CloudSim implementa as políticas de provisionamento por tempo compartilhado e espaço compartilhado. A diferença entre as políticas e seus impactos nos desempenhos das aplicações é representado na Figura 4.4. Nessa figura, um *host* com dois *CPU cores* recebe uma requisição para hospedar duas $VM$s. Cada uma requer dois cores e executa quatro cloudlets. As cloudlets $t1, t2, t3$ e $t4$ são executadas na $vm1$, enquanto $t5, t6, t7$ e $t8$ são executadas na $vm2$.

A Figura 4.4(a) apresenta um cenário de provisionamento onde a política de espaço compartilhado é aplicada para $VM$s e cloudlets. Como cada $VM$ requer dois *cores*, no modo espaço compartilhado somente uma $VM$ pode executar em um dado instante de tempo. Portanto, $vm2$ pode somente ser designada aos *cores* uma vez que $vm1$ encerrou a execução das cloudlets. O mesmo acontece para tarefas hospedadas na $VM$ [18]. Usando uma política de espaço compartilhado, o tempo final estimado ($y_{tfe}$) de uma cloudlet $t$ gerenciada por uma $vmi$ é dado pela Equação 4.1. Onde $y_{tie}$ é o tempo de início estimado da cloudlet, *instructions* é o total de instruções que são executadas no processador e $cores(t)$ é o número de *cores* requisitados pela cloudlet $t$. O tempo de início estimado depende da posição da cloudlet na fila de execução. Na política de espaço compartilhado, a capacidade total (*capacity*) de um *host* que tem $np$ elementos processadores é dada

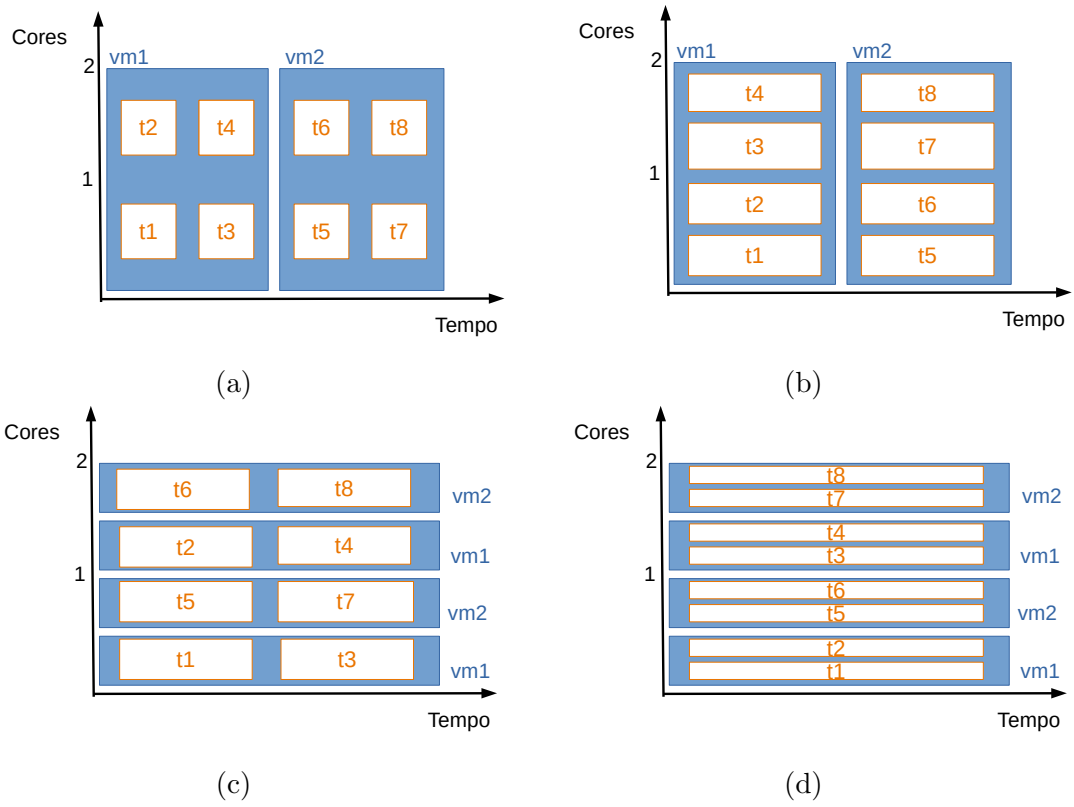



Figura 4.4: Políticas de provisionamento na execução de cloudlets: (a) Provisionamento de espaço compartilhado para *VM*s e cloudlets; (b) Provisionamento de espaço compartilhado para *VM*s e de tempo compartilhado para cloudlets; (c) Provisionamento de tempo compartilhado para *VM*s e de espaço compartilhado para cloudlets; (d) Provisionamento de tempo compartilhado para *VM*s e cloudlets. Baseado em [18].

pela Equação 4.2, onde $cap(i)$ é a capacidade de processamento de elementos individuais.

$$y_{tfe}(t) = y_{tie}(t) + \frac{instructions}{capacity \times cores(t)} \tag{4.1}$$

$$capacity = \sum_{i=1}^{np} \frac{cap(i)}{np} \tag{4.2}$$

Na Figura 4.4(b), a política de espaço compartilhado é usada para alocar $VM$s, mas uma política de tempo compartilhado é usada para alocar cloudlets dentro da $VM$. Então, durante o tempo de vida da $VM$, todas as cloudlets atribuídas a ela mudam de contexto dinamicamente até a sua conclusão. Essa política de alocação permite que as cloudlets sejam escaladas em um momento anterior, mas afetando significativamente o tempo de conclusão das unidades de tarefa que encabeçam a fila [18]. Usando uma política de tempo compartilhado, o tempo final estimado ($y_{tfe}$) de uma cloudlet $t$ gerenciada por uma $vmi$ é dado pela Equação 4.3, onde $y_{tas}$ é o tempo atual da simulação e $cores(t)$ é o número de *cores* requisitados pela cloudlet $t$. No modo tempo compartilhado, multiplas cloudlets podem realizar várias tarefas simultaneamente em uma $VM$. Neste caso, a capacidade total de processamento do *host* em nuvem é dado como na Equação 4.4

$$y_{tfe}(t) = y_{tas}(t) + \frac{instructions}{capacity \times cores(t)} \tag{4.3}$$

$$capacity = \frac{\sum_{i=1}^{np} cap(i)}{\max(\sum_{j=1}^{cloudlets} cores(j), np)} \tag{4.4}$$

Na Figura 4.4(c), uma alocação de tempo compartilhado é usada para $VM$s e uma de espaço compartilhado é usado para cloudlets. Nesse caso, cada $VM$ recebe uma fatia de tempo de cada *core* de processamento e, em seguida, as fatias são distribuídas para unidades de tarefa em espaço compartilhado. Como o *core* é compartilhado, a quantidade de poder de processamento disponível para a $VM$ é comparativamente menor do que nos cenários já mencionados. Como a atribuição da cloudlet é espaço compartilhado, apenas uma cloudlet pode ser alocada para cada *core*, enquanto outras são enfileiradas para consideração futura [18].

Finalmente, na Figura 4.4(d), uma alocação de tempo compartilhado é aplicada para *VM*s e cloulets. Consequentemente, o poder de processamento é simultaneamente compartilhado pelas *VM*s e os compartilhamentos de cada *VM* são divididos simultaneamente entre as cloudlets atribuídas a cada *VM*. Nesse caso, não há filas para máquinas virtuais ou para unidades de tarefa [18].

## 4.5 Configurações de Máquinas Virtuais e *Dataset* Utilizados no Trabalho

As configurações de *VM*s adotadas neste trabalho como referências de desempenho e custo são nove famílias (disponível no AWS[1]) {*c3, c4, c5, m3, m4, m5, r3, r4* e *r5*} e três tamanhos {*large, xlarge* e *2xlarge*}. O tamanho da *VM* representa o total de núcleos de processamento disponível. Por exemplo, *c4.large* possui dois núcleos, *c4.xlarge* possui quatro núcleos e *c4.2xlarge* possui oito núcleos. A Tabela 4.1 apresenta um resumo das configurações das *VM*s e o preço por hora. A Tabela 4.2 apresenta as especificações dos processadores que compõem as famílias de instâncias AWS. Na Tabela 4.2, AWS é o tipo de instância da AWS, PIX é processador Intel Xeon, L é a litografia, #N é o número de núcleos, #T é o número de threads, FB é a frequência base, FTM é a frequência turbo max, Cache é o tamanho da cache L3, VB é a velocidade do barramento e TDP é a potência de *design* térmico que representa o consumo médio de energia pelo processador quando o mesmo funciona em uma frequência base.

O conjunto de aplicações utilizado neste trabalho é composto por cargas de trabalho do conjunto *Numerical Aerodynamic Simulation Parallel Benchmarks* (NPB) [9] [4]. O NPB é um conjunto de *benchmarks* de desempenho de computação paralela que tem cinco *kernels* e três aplicativos de dinâmica de fluídos. Cada benchmark possui um conjunto de entradas que são divididas em classes: S, W, A, B, C, D, E e F. A classe S consiste de pequenas cargas de trabalho para testes rápidos; classe W é projetada para ser executada em estações de trabalho; as classes A, B e C contêm cargas de trabalho padrão; e as classes D, E e F são compostas por grandes cargas de trabalho. Neste trabalho foi utilizada a versão 3.4.2 do NPB (NPB com suporte MPI). Os cinco *kernels* disponíveis foram utilizados com as classes de entrada S, W,

[4]https://www.nas.nasa.gov/publications/npb.html

Tabela 4.1: Parâmetros e preço de *VM*s.

| Tipo de *VM* | vCPU | Memória (GB) | Preço (USD/h) | Tipo de *VM* | vCPU | Memória (GB) | Preço (USD/h) |
|---|---|---|---|---|---|---|---|
| c3.large | 2 | 4 | 0,163 | r4.large | 2 | 16 | 0,133 |
| c3.xlarge | 4 | 8 | 0,325 | r4.xlarge | 4 | 32 | 0,266 |
| c3.2xlarge | 8 | 16 | 0,650 | r4.2xlarge | 8 | 64 | 0,532 |
| m3.large | 2 | 8 | 0,190 | c5.large | 2 | 4 | 0,131 |
| m3.xlarge | 4 | 16 | 0,381 | c5.xlarge | 4 | 8 | 0,262 |
| m3.2xlarge | 8 | 32 | 0,761 | c5.2xlarge | 8 | 16 | 0,524 |
| r3.large | 2 | 16 | 0,350 | m5.large | 2 | 8 | 0,153 |
| r3.xlarge | 4 | 32 | 0,700 | m5.xlarge | 4 | 16 | 0,306 |
| r3.2xlarge | 8 | 64 | 1,399 | m5.2xlarge | 8 | 32 | 0,612 |
| c4.large | 2 | 4 | 0,100 | r5.large | 2 | 16 | 0,201 |
| c4.xlarge | 4 | 8 | 0,199 | r5.xlarge | 4 | 32 | 0,402 |
| c4.2xlarge | 8 | 16 | 0,398 | r5.2xlarge | 8 | 64 | 0,804 |
| m4.large | 2 | 8 | 0,100 | | | | |
| m4.xlarge | 4 | 16 | 0,200 | | | | |
| m4.2xlarge | 8 | 32 | 0,400 | | | | |

Tabela 4.2: Especificações dos tipos de instâncias [3].

| AWS | PIX | L (nm) | #N | #T | FB (GHz) | FTM (GHz) | Cache (MB) | VB (GT/s) | TDP (W) |
|---|---|---|---|---|---|---|---|---|---|
| c3 | E5-2680 v2 | 22 | 10 | 20 | 2.8 | 3.6 | 25 | 8 | 115 |
| m3 | E5-2670 | 32 | 8 | 16 | 2.6 | 3.3 | 20 | 8 | 115 |
| r3 | E5-2670 v2 | 22 | 10 | 20 | 2.5 | 3.3 | 25 | 8 | 115 |
| c4 | E5-2666 v3 | 22 | 10 | 20 | 2.6 | 3.3 | 25 | 9.6 | 105 |
| m4 | E5-2686 v4 | 14 | 18 | 36 | 2.3 | 3.0 | 45 | 9.6 | 145 |
| | E5-2676 v3 | 22 | 12 | 24 | 2.4 | 3.0 | 30 | 5 | 120 |
| r4 | E5-2686 v4 | 14 | 18 | 36 | 2.3 | 3.0 | 45 | 9.6 | 145 |
| c5 | Platinum 8124M | 14 | 18 | 36 | 3 | 3.5 | 25 | 8 | 150 |
| m5 | Platinum 8175M | 14 | 24 | 48 | 2.5 | 3.5 | 33 | - | - |
| r5 | Platinum 8175 | 14 | 24 | 48 | 2.5 | 3.1 | 33 | 10.4 | 165 |

A, B, C, D, E. As classes de entrada D e E tiveram êxito somente para o *kernel* EP quando executadas na máquina local, ocorrendo erro de alocação de espaço nas demais classes de entrada. A máquina utilizada neste trabalho como referência para as estimativas de tempo de resposta dos preditores contém processador Intel Core i7-5500U CPU 2,40GHz × 4, memória RAM de 12 GB, sistema operacional Ubuntu 20.04.2 LTS 64 bits. Cada um dos *kernels* usados são detalhados na Tabela 4.3.

Tabela 4.3: Descrição dos kernels NPB utilizados neste trabalho.

| Benchmark | Descrição | Linguagem |
|---|---|---|
| EP | Embaraçosamente Paralelo | Fortran |
| FT | Transformada Fourier rápida 3D discreta | Fortran |
| MG | Solucionador tri-diagonal de blocos | Fortran |
| IS | Ordenação de inteiro, acesso aleatório à memória | C |
| CG | Gradiente Conjugado | Fortran |

O *dataset* de configurações de *VM*s adotado neste trabalho possui 27 configurações de *VM*s (Tipos de *VM* na Tabela 4.1) e 27 configurações de cloudlets (5 aplicações × 5 entradas + 1 aplicação × 2 entradas). A informação do total de instruções, em cada configuração de cloudlet, foi obtida após a utilização da ferramenta de *profiling* Pin [49]. Com isso, foi possível preencher a propriedade de comprimento da cloudlet, que possui como unidade de medida o número de instruções (I). Outro aspecto das configurações das cloudlets são as demandas de cores das cloudlets que foram variadas em um intervalo de 5 configurações (2, 4, 8, 16 e 32 cores).

Aplicando as configurações de *VM* e de cloudlet no simulador CloudSim obteve-se os tempos de simulação de cada cloudlet para cada configuração de *VM*. Então, com resultados de tempo e com a informação de custo da Tabela 4.1, foram calculados os custos de execução. O *dataset* também contempla sistemas de *VM*s heterogêneas, compostas por *VM*s com recursos arquiteturais distintos. Considerando que a ferramenta Cloudsim permite a especificação de apenas uma *VM*, houve necessidade de especificar "virtualmente" uma *VM* heterogênea para representar sistemas heterogêneos. Nesse caso, o *MIPS* foi obtido da média ponderada pelo número de *cores* de cada *VM* e o restante das características como número de *cores*, memória RAM e preço foram obtidos a partir da soma dos valores das respectivas características presentes nas *VM*s do sistema heterogêneo. Portanto, o *dataset*

possui um total de 82862 saídas de simulação. As variáveis independentes (fatores) e dependentes (variáveis de resposta) que compõem o *dataset* são:

- Quantidade de *MIPS* de um core da *VM* (*MIPS*);
- Número de *cores* da *VM* (Cores *VM*);
- Quantidade de memória RAM da *VM* (RAM Memory);
- Preço por hora de cada *VM* (Price);
- Comprimento da cloudlet (Instructions);
- Quantidade de cores demandados pela cloudlet (Cores Cloudlet);
- Se a configuração é heterogênea ou não (Heterogeneous).
- Variável de resposta: Tempo de simulação (Time);
- Variável de resposta: Custo (Cost).

## 4.6 Desenvolvimento dos Preditores de Tempo e Custo

Diante da demanda por estimar dados de tempo e custo que representam as configurações de *VM*s indicadas pelo módulo de exploração do espaço de projeto, há necessidade do desenvolvimento de softwares preditores para esses dados. Tais preditores, a partir do *dataset* gerado neste trabalho, devem ser capazes de fornecer estimativas acuradas de tempo e custo, a cada vez que uma nova configuração de *VM* seja sugerida.

O desenvolvimento dos preditores utilizou implementações (em linguagem Python) de algoritmos de aprendizado de máquina disponíveis na biblioteca scikit-learn (versão 1.0.2) [58]. Considerando que os preditores abordados neste trabalho podem considerar o mesmo conjunto de variáveis independentes, um conjunto de algoritmos de aprendizado de máquina, com enfoque destacado em problemas de regressão, foram selecionados para avaliação:

- **Regressão Polinomial** é um dos algoritmos de aprendizado de máquina mais simples e uma das primeiras formas de análise regressiva

usadas em aplicações práticas. É uma generalização da regressão linear que prevê um valor numérico resultante de uma função linear dos recursos de entrada. A regressão polinomial de grau $n$ vem da aplicação de regressão linear em combinações de até $n$ recursos de entrada [34].

- **$k$-Nearest Neighbours** consiste em usar os $k$ vizinhos mais próximos ($k$ é um parâmetro) de um determinado exemplo para prever este exemplo. Em problemas de regressão, o algoritmo coleta os valores associados aos $k$ vizinhos mais próximos do exemplo dado e os combina adequadamente para gerar sua previsão [56, 17].

- **Decision Tree** (DT) corresponde a um conjunto de nós de forma que cada nó execute um teste em algum recurso de entrada. Cada ramo descendente corresponde a um valor possível do recurso testado e os nós folha correspondem às saídas previstas. Cada caminho da raiz para um nó folha é uma regra de predição [52].

- **Random Forest** é um método de conjunto baseado em árvores de decisão e *bagging* [13].

- **Support Vector Machines (SVM)** é utilizado para analisar dados e reconhecer padrões usados para classificação ou análise de regressão. O SVM adaptado para regressão mantém todas as principais características do algoritmo original e é chamado de *Support Vector Regression Machines* (*SVR*). Inclui o conceito de margem, representando a distância mais curta entre um separador e um ponto no espaço. Para alguns problemas de regressão, a dependência entre variáveis de entrada e saída não pode ser representada por uma função linear para que os recursos de entrada sejam projetados eficientemente em um espaço de dimensão superior usando uma função do kernel [57]. As funções do kernel [29] usadas neste trabalho são RBF, linear, polinomial (grau= 3 e 5) e sigmoid.

### 4.6.1 Experimentação

Após a definição do *dataset* e escolhidos os algoritmos de aprendizado de máquina a serem avaliados, o próximo passo foi definir a estratégia de experimentação, validação e avaliação que auxiliará na escolha do modelo para o desenvolvimento dos preditores. O conjunto de dados foi dividido em 20% para o conjunto de treino e 80% para o conjunto de teste. O procedimento de

ajuste (*fitting*) de cada modelo consistiu em separar o conjunto de treino em 10 rodadas sendo que em cada rodada 90% desse conjunto foi dedicado ao treino e 10% foi dedicado a validação do modelo. Para avaliação dos modelos foram adotadas as métricas do *score* de teste (Coeficiente de determinação $R^2$ [53]), *score* de treino, o tempo de resposta do *score* de teste e o erro percentual absoluto médio (*Mean Absolute Percentage Error* (*MAPE*)). A Figura 4.5 ilustra a estratégia avaliativa empregada neste trabalho.

O *MAPE* (equação 4.5) indica a porcentagem média da diferença entre a métrica de desempenho real $Y^i$ e o valor predito $\hat{Y}^i$ para $n$ amostras ($i = 1, \ldots, n$) [71, 21, 12].

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} \frac{|Y^i - \hat{Y}^i|}{Y^i} \tag{4.5}$$

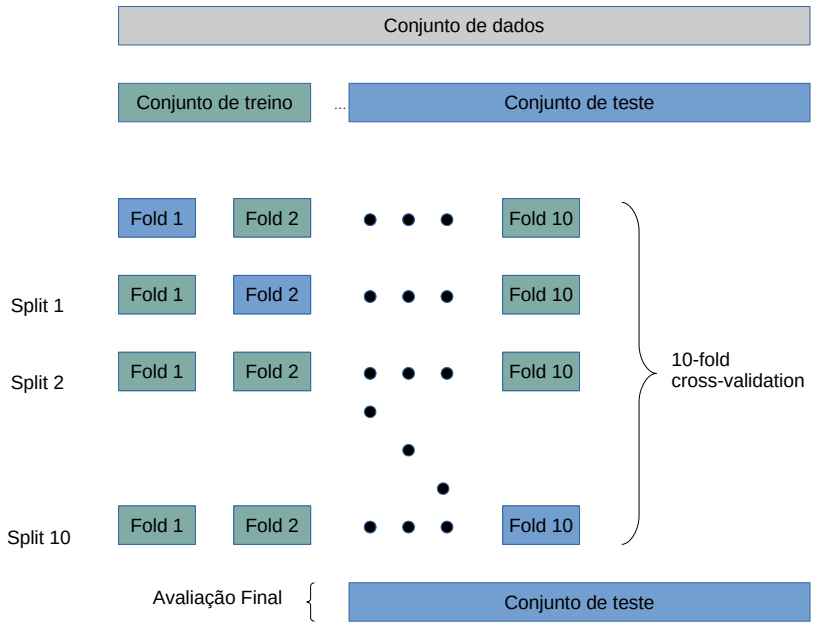



Figura 4.5: Estratégia avaliativa utilizada no treino e teste para ajuste e avaliação dos preditores de tempo e custo.

### 4.6.2 Resultados do Desenvolvimento de Preditor de Tempo

A Figura 4.6 apresenta um *heat map* gerado da correlação de Pearson [10] considerando apenas a variável de resposta *Time*. Observa-se que as variáveis *MIPS*, *Instructions*, *Cores Cloudlet* e *Heterogeneous* apresentam baixa correlação entre as variáveis estudadas e por isso foram utilizadas como ponto de partida no desenvolvimento do preditor de tempo. A partir da combinação inicial de variáveis foram adicionadas outras variáveis independentes e analizado o impacto dessa mudança. O desenvolvimento do preditor de tempo utilizou a combinação de variáveis que apresentou os melhores resultados (de acordo com as métricas *MAPE* e score de teste) foi baseada nas variáveis *MIPS*, *Instructions*, *Cores Cloudlet*, *Cores VM* e *Heterogeneous*.

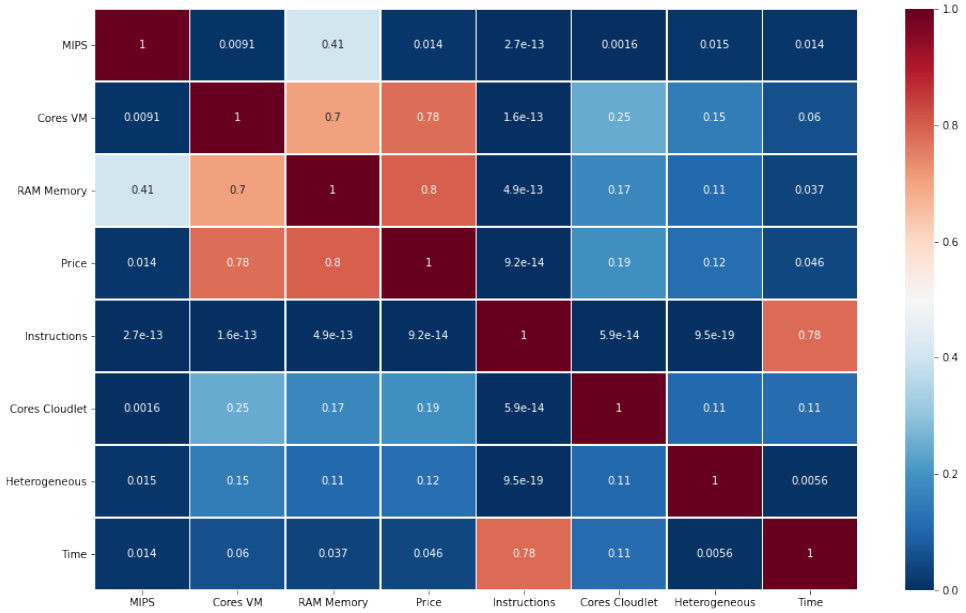



Figura 4.6: Correlação de Pearson entre os atributos do conjunto de dados.

Os modelos avaliados foram Regressão Polinomial (com grau $k = 1, \ldots, 5$), kNN, *Decision Tree*, *Random Forest* e *SVR*. Para avaliação e ajuste dos parâmetros dos modelos de preditores foi aplicada a técnica de *grid search*. Uma *grid search* gera exaustivamente candidatos a preditores a partir de uma grade de valores de parâmetro especificados [45]. Os parâmetros e técnicas candidatas passaram por uma validação cruzada (10-*fold cross-validation*) e foram avaliados quanto a $R^2$, *MAPE* e tempo de resposta.

A Tabela 4.4 apresenta os preditores, os parâmetros de configuração que alcançaram os melhores resultados e os resultados de acordo com as métricas avaliadas. Para a predição de tempo, o modelo baseado em *Decision Tree* obteve os melhores resultados em relação aos demais modelos, tendo o menor tempo de resposta, o menor *MAPE* e o maior $R^2$ tanto para treino como para teste. Portanto, o modelo baseado em *Decision Tree* foi escolhido para predição de tempo no módulo *DSE* da extensão MultiExplorer-VM.

Tabela 4.4: Características dos modelos de preditor de tempo com melhores scores.

| **Modelo** | **Parâmetros** | $R^2$ **Treino (MAPE)** (%) | $R^2$ **Teste (MAPE)** (%) | **Tempo de Respsota** (s) |
|---|---|---|---|---|
| Decision Tree | max_depth: None, min_samples_split: 2 | 99,86 (0,009) | 99,89 (0,008) | 0,020 |
| Random Forest | max_depth: 20, min_samples_split: 2, n_estimators: 50 | 99,85 (0,013) | 99,90 (0,012) | 0,613 |
| Polynomial | degree: 5 | 99,50 (50,77) | 99,89 (46,72) | 0,599 |
| KNN | n_neighbors: 3 | 97,27 (1,46) | 97,76 (1,74) | 1,127 |
| SVR | kernel: poly, degree: 5 | 93,85 (113,41) | 94,65 (84,94) | 19,973 |

### 4.6.3 Resultados do Desenvolvimento de Preditor de Custo

A Figura 4.7 apresenta um *heat map* gerado da correlação de Pearson [10] considerando apenas a variável de resposta *Cost*. Observa-se que as variáveis *MIPS*, *Instructions*, *Cores Cloudlet* e e *Heterogeneous* apresentam baixa correlação entre as variáveis estudadas e por isso foram utilizadas como ponto de partida no desenvolvimento do preditor de custo. A partir da combinação inicial de variáveis foram adicionadas outras variáveis independentes e analizado o impacto dessa mudança. A combinação de variáveis que apresentou os melhores resultados de *MAPE* e score de teste foi utilizando *MIPS*, *Instructions*, *Cores Cloudlet*, *Cores VM*, *Price* e *Heterogeneous*.

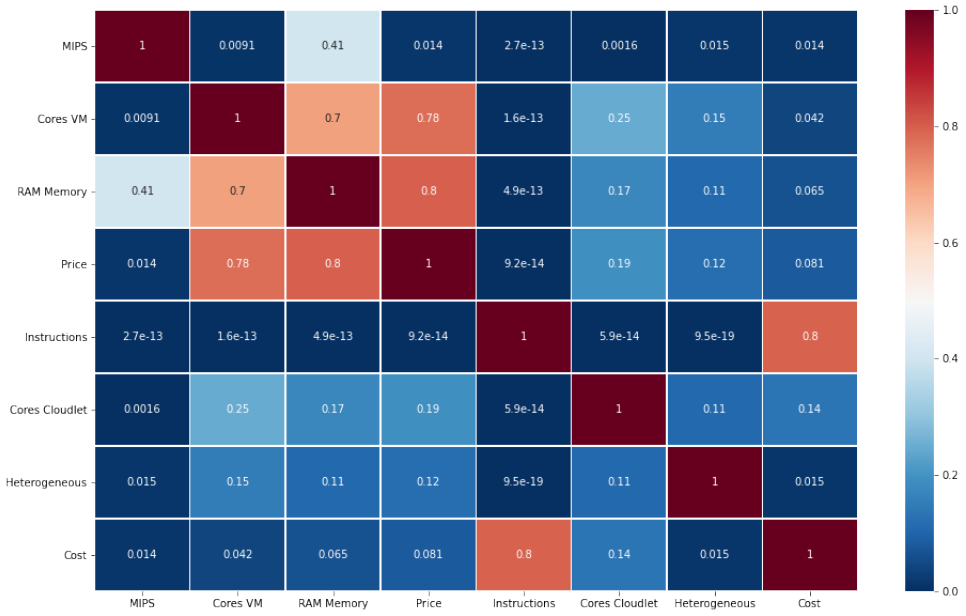



Figura 4.7: Correlação de Pearson entre os atributos do conjunto de dados.

Ressalta-se que os mesmos métodos para escolha dos hiperparâmetros e modelos de preditores utilizados para o preditor de tempo foram também adotados na escolha do preditor de custo. A Tabela 4.5 apresenta os preditores, os parâmetros de configuração que alcançaram os melhores resultados e os resultados de acordo com as métricas avaliadas. Para a predição de custo, os modelos baseados em regressão polinomial (grau = 5), *Decision Tree* e *Random Forest* atingiram os maiores *scores* de $R^2$, respectivamente. *Random Forest* e *Decision Tree* conseguiram os menores *MAPE*s, porém o tempo de resposta que o modelo *Decision Tree* alcançou é 30 vezes menor que o segundo melhor resultado. Portanto, o modelo baseado em *Decision Tree* foi utilizado para predição de custo no módulo *DSE* da extensão MuliExplorer-VM.

Tabela 4.5: Características dos modelos de preditor de custo com melhores scores.

| **Modelo** | **Parâmetros** | $R^2$ **Treino (MAPE)** (%) | $R^2$ **Teste (MAPE)** (%) | **Tempo de Resposta** (s) |
|---|---|---|---|---|
| Decision Tree | max_depth: 25, min_samples_split: 2 | 96,85 (0,025) | 97,64 (0,025) | 0,017 |
| Random Forest | max_depth: 20, min_samples_split: 2, n_estimators: 50 | 98,27 (0,024) | 98,27 (0,023) | 0,525 |
| Polynomial | degree: 5 | 99,89 (0,134) | 99,88 (0,135) | 0,751 |
| KNN | n_neighbors: 2 | 95,88 (0,107) | 96,55 (0,105) | 0,777 |
| SVR | kernel: poly, degree: 5 | 93,02 (0,413) | 93,12 (0,536) | 20,851 |

## 4.7 Considerações Finais

Este capítulo apresentou o desenvolvimento da extensão MultiExplorer-VM que possibilita explorar configurações de máquinas virtuais atendendo aos objetivos e restrições de tempo e custo de um usuário. A efetividade e acurácia das configurações de *VM*s sugeridas pelo algoritmo de exploração de espaço de projeto dependem fortemente de preditores de tempo e custo. Dessa forma, parte significativa do trabalho foi dedicada ao projeto, desenvolvimento, validação e avaliação desses preditores.

O próximo capítulo apresenta resultados e discussões da extensão MultiExplorer-VM considerando a resolução do problema de alocação de recursos em nuvem sob a perspectiva de solução de técnicas de *DSE*. Adicionalmente, realiza-se também a comparação entre a qualidade dos resultados de MultiExplorer-VM com a técnica *Paramount Interation*.

# Capítulo 5

# Experimentos e Resultados

Este capítulo apresenta experimentos e resultados com a ferramenta MultiExplorer-VM para exploração de projetos de *VM*s. O Capítulo também apresenta comparações entre os resultados de MultiExplorer-VM e a solução baseada na técnica *Paramount Interaction* (PI).

## 5.1 Infraestrutura para Experimentos de Exploração do Espaço de Projeto

A experimentação envolvendo a busca por configurações de *VM*s viáveis de acordo com objetivos e restrições do usuário utilizou a extensão MultiExplorer-VM. No fluxo dessa ferramenta, o módulo de exploração de espaço de projeto possui algoritmos de busca exaustiva (Força Bruta) de soluções alternativas e um algoritmo genético, denominado NSGA-II.

O algoritmo de Força Bruta possui a função de calcular o tempo e o custo de todas as configurações viáveis (que atendem as restrições e objetivos do usuário) de *VM*s, a partir de configurações básicas disponíveis no banco de dados de *VM*s. Com isso, configurações ótimas devem também ser encontradas. A desvantagem mais evidente dessa abordagem consiste no tempo (demasiado) de busca de soluções.

O algoritmo NSGA-II disponível em MultiExplorer-VM é baseado em ordenação elitista por dominância (Pareto *ranking*). O algoritmo utiliza

seleção rápida não dominada (*fast nondominated sorting*) e distância de agrupamento (*crowding distance*) para selecionar configurações (indivíduos na nomenclatura de computação genética) que estão mais próximos da fronteira de Pareto.

## 5.2 Modelo de Otimização MultiObjetivo

A fim de modelar o problema multiobjetivo para alocação de recursos em nuvem, o usuário do ambiente MultiExplorer-VM pode informar restrições para guiar o processo de exploração de espaço do projeto. Além disso, os dados utilizados para elaboração do modelo de otimização consistem em saídas do módulo de simulação do MultiExplorer-VM. Para padronizar os termos utilizados neste trabalho, as VMs do projeto inicial definido pelo usuário serão indicados como "*VM*s originais". O termo "*VM*s suplementares" diz respeito às VMs (existente no banco de dados do MultiExplorer-VM), adicionadas como alternativas para exploração do projeto. As informações oriundas do fluxo do MultiExplorer-VM para o módulo de exploração de espaço de projeto são:

- Quantidade de *VM* ($n$): representa o número de *VM*s do projeto original;
- Identidade da *VM* original ($I_o$): refere-e a identificação da *VM* original;
- *MIPS* da VM original: retrata a quantidade de *MIPS* de um core da *VM* original;
- *Cores* da *VM* original: representa o número de *cores* da *VM* original;
- Preço da *VM* original: Custo por hora da *VM* original;
- Memória RAM da *VM* original: Quantidade de memória RAM da *VM* original;

Salienta-se que as informações sobre *MIPS*, *Cores*, Preço (custo/h) e Memória RAM também são utilizadas na caracterização das *VM*s suplementares. Adicionalmente, existe uma variável denominada "Tipo da *VM*" que é utilizada para identificar cada *VM* do banco de dados do MultiExplorer-VM.

A solução do problema de alocação de recursos, aqui considerada, pode envolver a indicação de *VM*s suplementares cujos *cores* são diferentes da *VM* original, formando, nesse caso, uma solução de configuração de *VM* heterogênea. Dessa forma, o usuário tem a possibilidade de especificar, como parte da definição do problema, as seguintes variáveis:

- Quantidade de *VM*s originais ($n_o$);
- Tipo de *VM* suplementar ($T_s$);
- Quantidade de *VM*s suplementares ($n_s$)

Para confrontar os resultados obtidos a partir da aplicação de *DSE* e comparar com os resultados do projeto original (a partir da simulação no CloudSim) é necessário obter o tempo de execução ($t_{total}$) e o custo ($c_{total}$) total das *VM*s. Ambos valores são obtidos a partir dos preditores de tempo e custo já apresentados no Capítulo 4.

A partir da definição do tempo e custo total da solução, tem-se bem definido os dois objetivos para o problema de exploração de espaço de projeto aplicado ao problema de alocação de recursos em nuvem: a minimização do tempo ($t_{total}$) e a minimização do custo ($c_{total}$). Considerando a delimitação do espaço de projeto como restrição, tem-se então um modelo geral de otimização para o problema de alocação de *VM*s na nuvem computacional no contexto da extensão MultiExplorer-VM:

$$
\begin{aligned}
\text{Minimizar} \quad & t_{total}, c_{total} \\
\text{Sujeito a:} \quad & \\
& t_{total}^{inf} \leq t_{total} \leq t_{total}^{sup} \\
& c_{total}^{inf} \leq c_{total} \leq c_{total}^{sup} \\
& 1 \leq n_o \leq n-1 \\
& 1 \leq n_s \leq n_s^{max} \\
& t_{total}, c_{total} \in R+ \\
& n_o, n_s \in N \\
& I_o, T_s \in B
\end{aligned}
$$

Em que $t_{total}^{inf}$, $t_{total}^{sup}$, $c_{total}^{inf}$, $c_{total}^{sup}$ representam, respectivamente, o limite inferior e superior da restrição de tempo de execução total e o limite inferior

e superior da restrição de custo total do projeto. Esses limites podem ser definidos pelo usuário. A solução resultante deve possuir pelo menos uma *VM* original e está restrita por um limite $n$ definido pelo usuário. A quantidade de *VM*s suplementares da plataforma resultante deve ser de pelo menos um e está limitada pela quantidade total de *VM*s suplementares definida pelo usuário ($n_s^{max}$). $B$ representa o conjunto das *VM*s disponíveis no banco de dados do MultiExplorer-VM, assim que as *VM*s original ($I_o$) e suplementar ($T_s$) escolhidas devem pertencer a esse banco de dados.

## 5.3 Metodologia de Validação e Avaliação dos Resultados

Os resultados da exploração do espaço de projeto de *VM*s passaram por etapadas de validação estatística dos dados como a estimativa do poder de teste, teste de normalidade das amostras (teste de Shapiro-Wilk [60]), teste de homogeneidade de variâncias (testes de Bartlett [33] e Fligner-Killeen [23]). Após a validação dos dados amostrais, foram aplicados testes t [55] e de Wilcoxon [35] para avaliar hipóteses de igualdade ou de diferença estatística significativa entre as melhores (seguindo critérios de tempo e custo) configurações de *VM*s geradas por MultiExplorer-VM, Força Bruta e *Paramount Interaction* [50]. Assim, as hipóteses nula ($H_0$) e alternativa ($H_1$) adotadas nos testes foram:

$$\begin{aligned} H_0: &\quad Custo(VM_{ME}) = Custo(VM_{FB,PI}) \\ H_0: &\quad Tempo(VM_{ME}) = Tempo(VM_{FB,PI}) \\ H_1: &\quad Custo(VM_{ME}) \neq Custo(VM_{FB,PI}) \\ H_1: &\quad Tempo(VM_{ME}) \neq Tempo(VM_{FB,PI}) \end{aligned} \tag{5.1}$$

As hipóteses $H_0$ indicam que não há diferença de custo ou tempo entre as soluções de melhor custo obtida pelo algoritmo de MultiExplorer-VM (ME) e as soluções de menor custo obtidas por força bruta (FB) e *Paramount Interaction* (PI). As hipóteses alternativas ($H_1$) indicam que há diferença estatisticamente significativa entre as soluções seguindo essas mesmas métricas de custo e tempo. O nível de significância escolhido para os testes é de $\alpha = 0,05$. Para um poder de teste de 80% (redução de ocorrência de falso negativo ou erro do tipo II) foram geradas 21 amostras de resultados da extensão MultiExplorer-VM, para cada aplicação. As restrições impostas para

a exploração do espaço de projeto foram: tempo máximo de 1 hora e custo de 2 USD/h para cada uma das 27 aplicações do *benchmark* utilizado neste trabalho.

## 5.4 Resultados e Discussão

Em cada execução da estratégia de exploração do espaço de projeto, a configuração com menor tempo e a configuração com o menor custo foram selecionadas. As soluções (configurações de *VM*s) para a aplicação EP-E não atenderam as restrições de tempo e por isso seus resultados não foram considerados. As Figuras 5.1 e 5.3 apresentam gráficos *boxplots* de todas as aplicações considerando tempo e custo. As Figuras 5.2 e 5.4 apresentam gráficos *boxplots* das aplicações em comum com o trabalho [50] que utilizou a técnica PI. Nas Figuras 5.1 e 5.2, o eixo $x$ representa as aplicações e o eixo $y$ o tempo. Nas Figuras 5.3 e 5.4, o eixo $x$ representa as aplicações e o eixo $y$ o custo.

A configuração com o menor tempo e a configuração com menor custo encontradas para cada aplicação, entre as 21 execuções realizadas sobre MultiExplorer-VM, são apresentadas nas Tabelas 5.1 e 5.2. A coluna *VM original* apresenta o nome da *VM* indicada como solução inicial (entrada) de MultiExplorer-VM, a coluna *VM suplementar* apresenta a *VM* encontrada pelo MultiExplorer-VM para compor a configuração de nuvem e as colunas *Num VM original* e *Num VM suplementar* indicam as quantidades de *VM*s original e *VM* suplementar, respectivamente, indicadas pela solução de MultiExplorer-VM. Essas configurações foram comparadas com as configurações retornadas pelos algoritmos de Força Bruta e do trabalho [50]. A coluna erro absoluto diz respeito a diferença absoluta entre a predição de tempo (ou custo) de uma configuração da solução FB com uma configuração de ME (ou PI).

Os valores de predição de tempo e predição de custo das configurações obtidas com Multiexplorer-VM, Força Bruta e PI [50] são apresentados nos gráficos das Figuras 5.5 e 5.6.

Os resultados apresentados nas Tabelas 5.1 e 5.2 e representados nas Figuras 5.5 e 5.6 indicam que há proximidade entre as soluções mais eficientes obtidas por MultiExplorer-VM e o algoritmo FB. O MultiExplorer-VM encontrou a configuração ótima de tempo em 17 aplicações e para custo

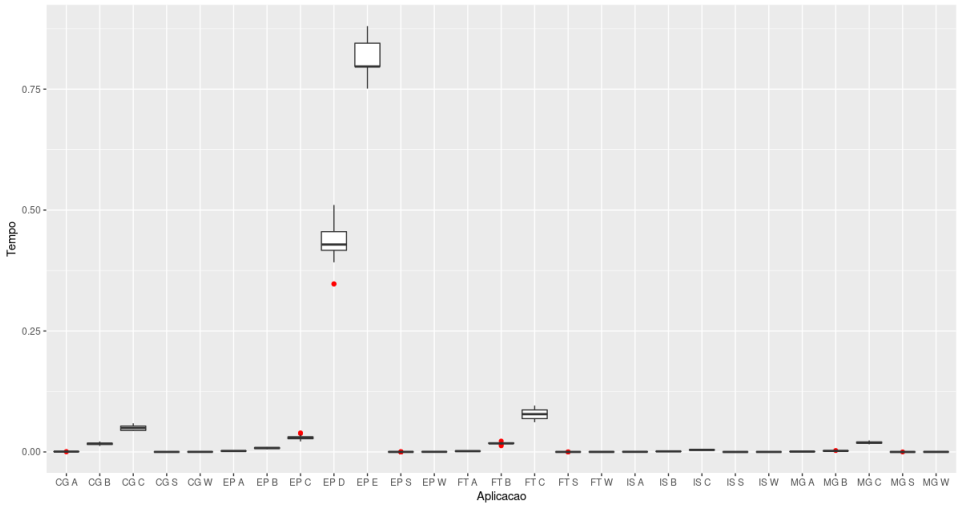



Figura 5.1: Boxplot de tempo para todas as aplicações.

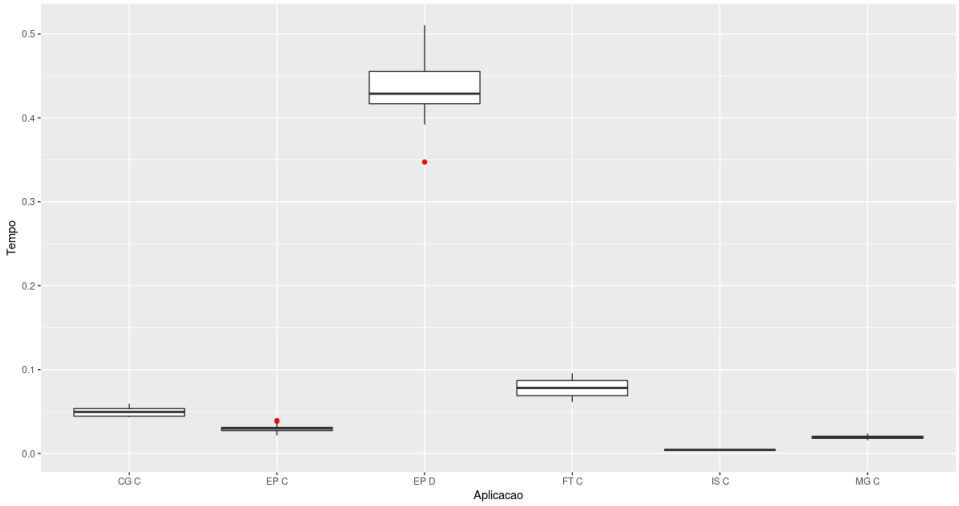



Figura 5.2: Boxplot de tempo para aplicações em comum com o trabalho [50].

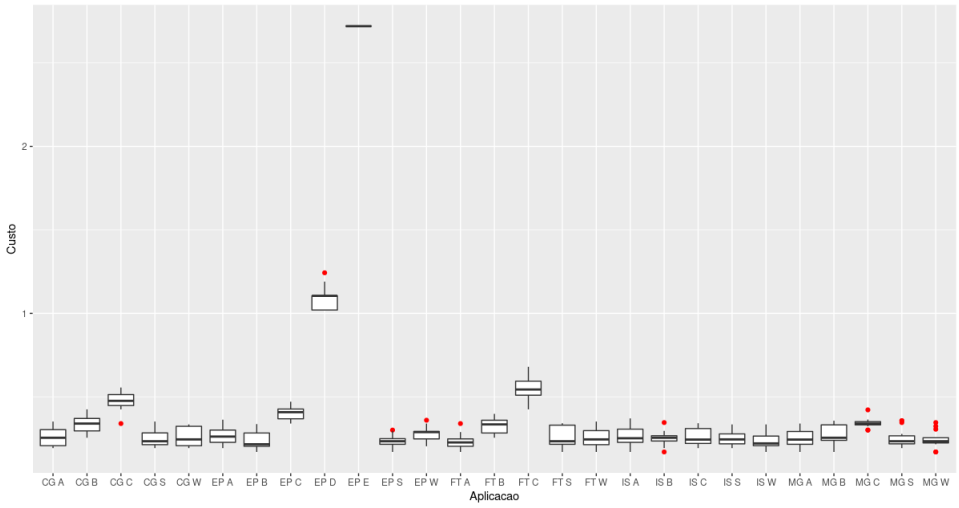



Figura 5.3: Boxplot de custo para todas as aplicações.

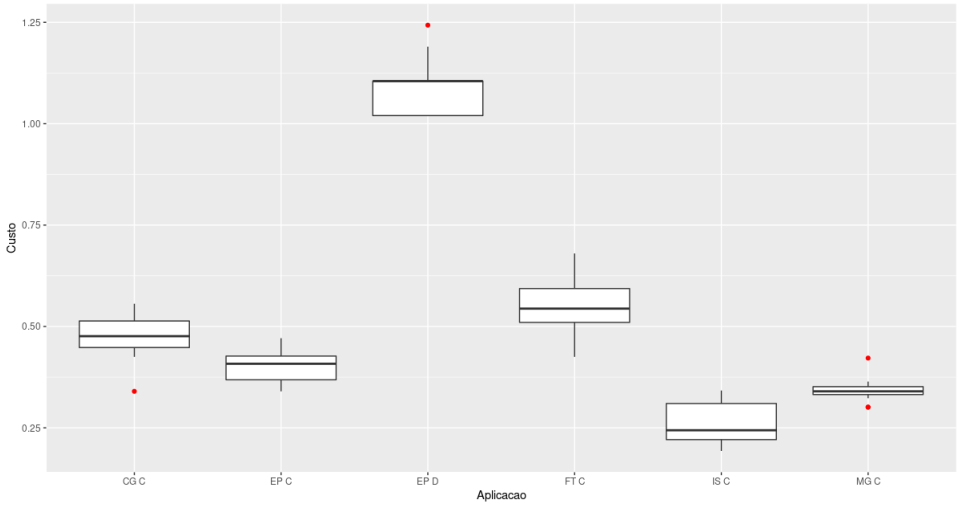



Figura 5.4: Boxplot de custo para aplicações em comum com o trabalho [50].

Tabela 5.1: Configurações de VMs e erro absoluto de tempo dos resultados das técnicas MultiExplorer-VM (ME) e Paramount Interaction (PI[50]) em relação ao Força Bruta (FB).

| **Técnica** | **Aplicação** | **Erro Absoluto Tempo** (s) | **Num VM original** | **VM original** | **Num VM suplementar** | **VM suplementar** |
|---|---|---|---|---|---|---|
| ME | EP S | 0,0622 | 10 | c5.large | 10 | c5n.large |
| ME | EP W | 0,1296 | 10 | c5.large | 10 | c5n.large |
| ME | EP A | 0 | 10 | c5.large | 10 | c5.large |
| ME | EP B | 0 | 10 | c5.large | 10 | c5.large |
| ME | EP C | 0 | 10 | c5.large | 10 | c5.large |
| PI[50] | | 2190,24 | 4 | m5n.large | | |
| ME | EP D | 0 | 10 | c5.large | 10 | c5.large |
| PI[50] | | 148799 | 2 | c5n.large | | |
| ME | FT S | 0 | 10 | c5.large | 10 | c5.large |
| ME | FT W | 0,0551 | 9 | c5.large | 10 | m5n.large |
| ME | FT A | 0 | 10 | c5.large | 10 | c5.large |
| ME | FT B | 0 | 10 | c5.large | 10 | c5.large |
| ME | FT C | 0 | 9 | c5.large | 10 | c5.large |
| PI[50] | | 4674,6 | 4 | c5.large | | |
| ME | MG S | 0 | 10 | c5.large | 10 | c5.large |
| ME | MG W | 0,0511 | 10 | c5n.large | 10 | c5.large |
| ME | MG A | 0,3744 | 10 | c5.large | 10 | c5n.large |
| ME | MG B | 0 | 10 | c5.large | 10 | c5.large |
| ME | MG C | 9,36 | 10 | c5.large | 10 | c5n.large |
| PI[50] | | 1140,84 | 4 | c5.large | | |
| ME | IS S | 0,0019 | 10 | c5n.large | 10 | c5.large |
| ME | IS W | 0 | 10 | c5.large | 10 | c5.large |
| ME | IS A | 0 | 10 | c5.large | 10 | c5.large |
| ME | IS B | 0,5868 | 10 | c5.large | 10 | c5n.large |
| ME | IS C | 0 | 10 | c5.large | 10 | c5.large |
| PI[50] | | 5108,54 | 1 | m5n.large | | |
| ME | CG S | 0 | 9 | c5.large | 10 | c5.large |
| ME | CG W | 0 | 10 | c5.large | 10 | c5.large |
| ME | CG A | 0 | 10 | c5.large | 10 | c5.large |
| ME | CG B | 0 | 10 | c5.large | 10 | c5.large |
| ME | CG C | 25,92 | 10 | c5n.large | 10 | c5.large |
| PI[50] | | 1494 | 2 | c5.4xlarge | | |

Tabela 5.2: Configurações de VMs e erro absoluto de custo dos resultados das técnicas MultiExplorer-VM (ME) e Paramount Interaction (PI[50]) em relação ao Força Bruta (FB).

| **Técnica** | **Aplicação** | **Erro Absoluto Custo** (USD) | **Num VM original** | **VM original** | **Num VM suplementar** | **VM suplementar** |
|---|---|---|---|---|---|---|
| ME | EP S | 0 | 1 | c5.large | 1 | c5.large |
| ME | EP W | 0,034 | 1 | c5.large | 1 | m5n.large |
| ME | EP A | 0,023 | 1 | c5.large | 1 | c5n.large |
| ME | EP B | 0 | 1 | c5.large | 1 | c5.large |
| ME | EP C | 0 | 1 | c5.large | 3 | c5.large |
| PI[50] | | 0,136 | 4 | m5n.large | | |
| ME | EP D | 0 | 2 | c5.large | 10 | c5.large |
| PI[50] | | 8,052 | 2 | c5n.large | | |
| ME | FT S | 0 | 1 | c5.large | 1 | c5.large |
| ME | FT W | 0 | 1 | c5.large | 1 | c5.large |
| ME | FT A | 0 | 1 | c5.large | 1 | c5.large |
| ME | FT B | 0 | 1 | c5.large | 2 | c5.large |
| ME | FT C | 0 | 1 | c5.large | 4 | c5.large |
| PI[50] | | 0,255 | 4 | c5.large | | |
| ME | MG S | 0 | 1 | c5.large | 1 | c5.large |
| ME | MG W | 0 | 1 | c5.large | 1 | c5.large |
| ME | MG A | 0 | 1 | c5.large | 1 | c5.large |
| ME | MG B | 0 | 1 | c5.large | 1 | c5.large |
| ME | MG C | 0,023 | 2 | c5.large | 1 | c5n.large |
| PI[50] | | 0,085 | 4 | c5.large | | |
| ME | IS S | 0,023 | 1 | c5.large | 1 | c5n.large |
| ME | IS W | 0 | 1 | c5.large | 1 | c5.large |
| ME | IS A | 0 | 1 | c5.large | 1 | c5.large |
| ME | IS B | 0 | 1 | c5.large | 1 | c5.large |
| ME | IS C | 0,023 | 1 | c5.large | 1 | c5n.large |
| PI[50] | | 0,068 | 1 | m5n.large | | |
| ME | CG S | 0,023 | 1 | c5.large | 1 | c5n.large |
| ME | CG W | 0,023 | 1 | c5.large | 1 | c5n.large |
| ME | CG A | 0,023 | 1 | c5.large | 1 | c5n.large |
| ME | CG B | 0 | 1 | c5.large | 2 | c5.large |
| ME | CG C | 0 | 1 | c5.large | 3 | c5.large |
| PI[50] | | 1,02 | 2 | c5.4xlarge | | |

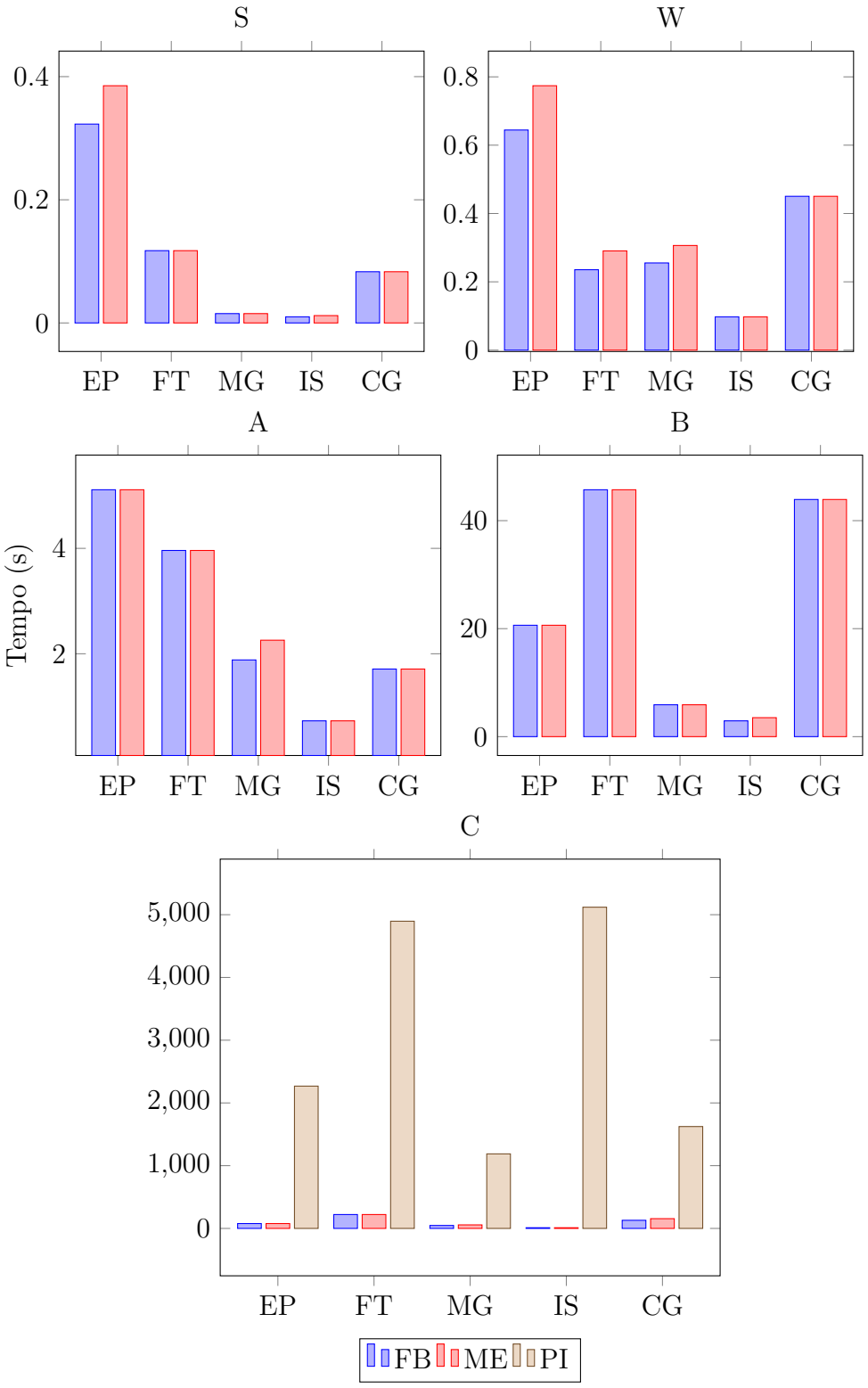



Figura 5.5: Menores tempos das técnicas Força Bruta (FB), MultiExplorer-VM (ME) e *Paramount Interaction* (PI [50]).

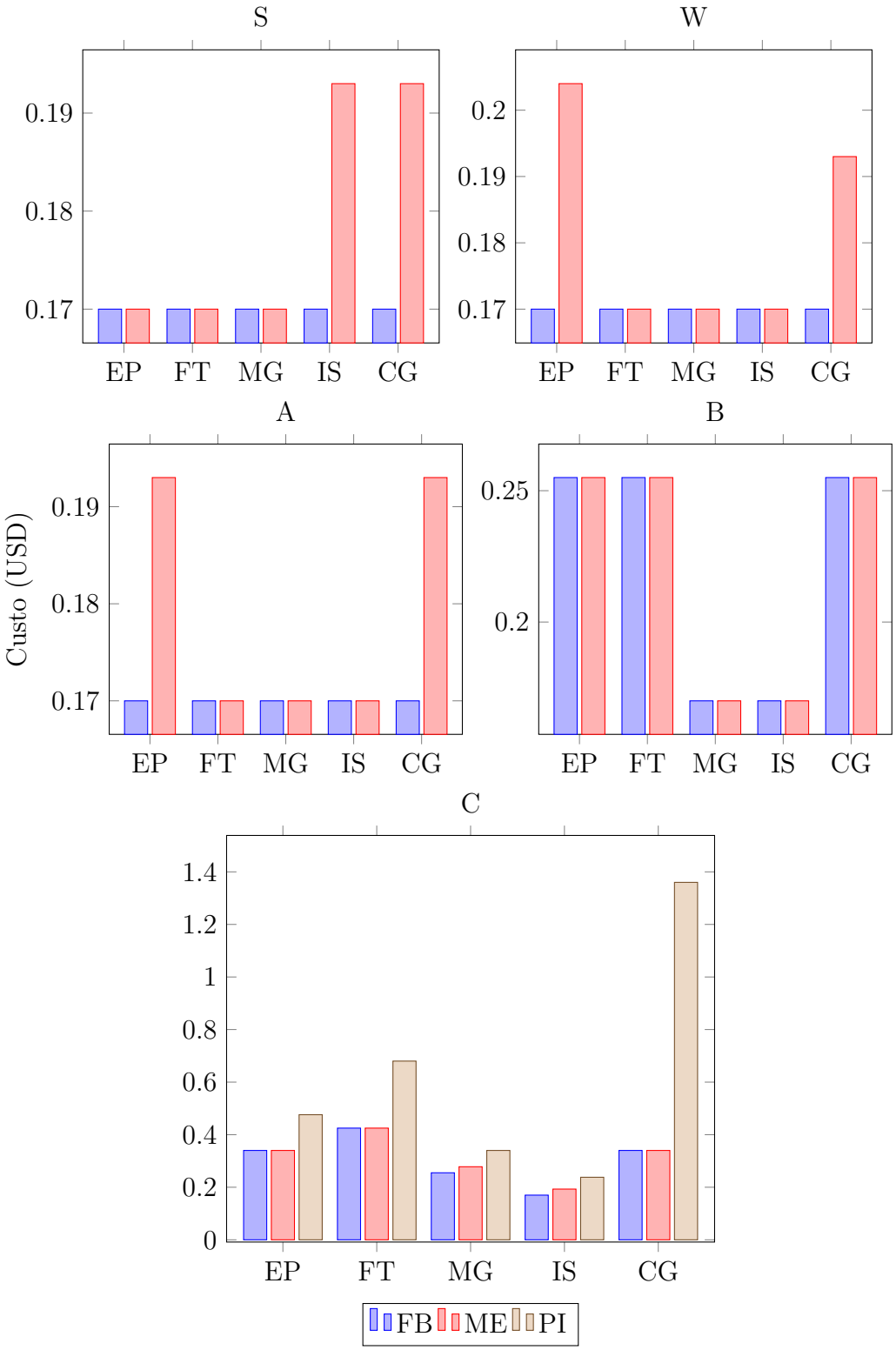



Figura 5.6: Menores custos das técnicas Força Bruta (FB), MultiExplorer-VM (ME) e *Paramount Interaction* (PI [50]).

foi encontrada a configuração ótima em 18 aplicações de um total de 26 aplicações. Da mesma forma, há indicação de que as configurações obtidas por MultiExplorer-VM obtém melhores resultados do que aquelas geradas pelo algoritmo PI [50]. As soluções apresentadas pelo MultiExplorer-VM obtiveram menores tempo de excução das aplicações e menores custos de VM que os da técnica PI [50] chegando a alcançar resultados de custo de VM 8,8 vezes menores como mostrado na Tabela 5.2 para aplicação EP D. Essas indicações podem ser esclarecidas pelos resultados dos testes aplicados e apresentados nas Tabelas 5.3 e 5.4.

As Tabelas 5.3 e 5.4 apresentam os p-valores retornados pelos testes de Shapiro-Wilk [60], teste t [55] e o teste de Wilcoxon [35] considerando as menores predições de tempo e custo, respectivamente, obtidas pelas configurações sugeridas por FB, ME e PI para aplicações do *benchmark*.

Durante a avaliação da amostras de predição de tempo, não pode ser constatado a homogeneidade de variância das aplicações através dos testes de Bartlett [33] e Fligner-Killeen [23]. Em seguida, foi verificada a normalidade das aplicações por meio do teste de Shapiro-Wilk[60], onde pode-se aceitar a hipótese de normalidade dos dados para 24 aplicações. A fim de avaliar a existência de diferenças significativas entre os resultados de exploração obtidos por MultExplorer-VM e o algoritmo de Força Bruta, o teste t [55] foi aplicado para as 24 aplicações que atingiram a normalidade e o teste de Wilcoxon [35], para 2 aplicações onde não pode-se assuimir normalidade (p-valor $< 0,05$). Os resultados dos testes demonstraram que há diferença estatística significativa entre os resultados apresentados nessas duas técnicas. Para as 6 aplicações em comum com o trabalho [50], apenas 4 pode-se assumir a normalidade. Então foi verificado se existe diferença significativa entre os tempos apresentados pelo MultiExplorer-VM e o resultado apresentado pelo trabalho [50]. Para o todas as seis aplicações houve diferença significativa representando assim a vantagem da técnica MultiExplorer-VM.

Durante a avaliação da amostras de predição de custo, a homogeneidade de variância do custo das aplicações foi avaliada através dos testes de Bartlett [33] e Fligner-Killeen [23]. Em seguida foi verificada a normalidade das aplicações por meio do teste de Shapiro-Wilk [60]. O teste t [55] foi aplicado sobre as 20 aplicações que atingiram a normalidade e o teste de Wilcoxon [35] para as 6 aplicações onde a normalidade dos dados foi rejeitada. Em todas as aplicações houve diferença significativa entre os resultados de ME e FB revelando que a estratégia MultiExplorer-VM não alcança os resultados ótimos obtidos pelo algoritmo de força bruta. Também foi avaliado se existe

Tabela 5.3: Comparação dos resultados de tempo do MultiExplorer-VM (ME) em relação aos algoritmos de Força Bruta (FB) e *Paramount Interaction* (PI [50]).

| **Comparação** | **Aplicação** | **Shapiro p-value** | **t test p-value** |
|---|---|---|---|
| ME,FB | EP S | 0,034 | $5{,}28\times10^{-07}$ |
| ME,FB | EP W | 0,033 | $1{,}21\times10^{-05}$ |
| ME,FB | EP A | 0,692 | $2{,}24\times10^{-09}$ |
| ME,FB | EP B | 0,498 | $1{,}74\times10^{-08}$ |
| ME,FB | EP C | 0,266 | $1{,}05\times10^{-08}$ |
| ME,PI | | 0,266 | $1{,}78\times10^{-18}$ |
| ME,FB | EP D | 0,168 | $5{,}34\times10^{-08}$ |
| ME,PI | | 0,168 | $4{,}56\times10^{-21}$ |
| ME,FB | FT S | 0,165 | $5{,}71\times10^{-10}$ |
| ME,FB | FT W | 0,352 | $5{,}38\times10^{-09}$ |
| ME,FB | FT A | 0,451 | $2{,}80\times10^{-08}$ |
| ME,FB | FT B | 0,302 | $3{,}49\times10^{-10}$ |
| ME,FB | FT C | 0,415 | $1{,}98\times10^{-07}$ |
| ME,PI | | 0,415 | $4{,}63\times10^{-50}$ |
| ME,FB | MG S | 0,091 | $4{,}74\times10^{-10}$ |
| ME,FB | MG W | 0,030 | $1{,}61\times10^{-07}$ |
| ME,FB | MG A | 0,214 | $1{,}23\times10^{-06}$ |
| ME,FB | MG B | 0,644 | $5{,}95\times10^{-08}$ |
| ME,FB | MG C | 0,337 | $1{,}20\times10^{-06}$ |
| ME,PI | | 0,337 | $5{,}63\times10^{-47}$ |
| ME,FB | IS S | 0,025 | $3{,}16\times10^{-09}*$ |
| ME,FB | IS W | 0,223 | $7{,}41\times10^{-11}$ |
| ME,FB | IS A | 0,444 | $7{,}42\times10^{-08}$ |
| ME,FB | IS B | 0,102 | $5{,}07\times10^{-06}$ |
| ME,FB | IS C | 0,668 | $3{,}93\times10^{-09}$ |
| ME,PI | | 0,668 | $2{,}69\times10^{-64}$ |
| ME,FB | CG S | 0,172 | $1{,}20\times10^{-07}$ |
| ME,FB | CG W | 0,727 | $8{,}56\times10^{-09}$ |
| ME,FB | CG A | 0,011 | $1{,}68\times10^{-06}*$ |
| ME,FB | CG B | 0,988 | $9{,}41\times10^{-09}$ |
| ME,FB | CG C | 0,035 | $2{,}00\times10^{-05}$ |
| ME,PI | | 0,035 | $4{,}71\times10^{-25}$ |

$*$ p-valor obtido pelo teste de Wilcoxon [35]

Tabela 5.4: Comparação dos resultados de custo do MultiExplorer-VM (ME) em relação aos algoritmos de Força Bruta (FB) e *Paramount Interaction* (PI [50]).

| **Comparação** | **Aplicação** | **Shapiro p-value** | **t test p-value** |
|---|---|---|---|
| ME,FB | EP S | 0,878 | $1{,}08\times10^{-08}$ |
| ME,FB | EP W | 0,159 | $5{,}25\times10^{-06}$ |
| ME,FB | EP A | 0,113 | $4{,}73\times10^{-08}$ |
| ME,FB | EP B | 0,102 | $1{,}19\times10^{-06}$ |
| ME,FB | EP C | 0,344 | $2{,}05\times10^{-06}$ |
| ME,PI | | 0,344 | $1{,}75\times10^{-06}$ |
| ME,FB | EP D | 0,001 | $1{,}037\times10^{-05}*$ |
| ME,PI | | 0,001 | $4{,}02\times10^{-25}$ |
| ME,FB | FT S | 0,012 | $1{,}204\times10^{-08}*$ |
| ME,FB | FT W | 0,843 | $8{,}46\times10^{-08}$ |
| ME,FB | FT A | 0,110 | $8{,}02\times10^{-07}$ |
| ME,FB | FT B | 0,317 | $1{,}31\times10^{-06}$ |
| ME,FB | FT C | 0,662 | $2{,}49\times10^{-07}$ |
| ME,PI | | 0,662 | $3{,}95\times10^{-18}$ |
| ME,FB | MG S | 0,003 | $1{,}18\times10^{-08}*$ |
| ME,FB | MG W | 0,059 | $2{,}65\times10^{-07}$ |
| ME,FB | MG A | 0,472 | $1{,}71\times10^{-07}$ |
| ME,FB | MG B | 0,219 | $1{,}81\times10^{-07}$ |
| ME,FB | MG C | 0,053 | $1{,}87\times10^{-09}$ |
| ME,PI | | 0,053 | $5{,}59\times10^{-01}$ |
| ME,FB | IS S | 0,110 | $5{,}68\times10^{-06}$ |
| ME,FB | IS W | 0,026 | $1{,}208\times10^{-08}*$ |
| ME,FB | IS A | 0,066 | $6{,}02\times10^{-07}$ |
| ME,FB | IS B | 0,471 | $6{,}84\times10^{-08}$ |
| ME,FB | IS C | 0,008 | $3{,}235\times10^{-09}*$ |
| ME,PI | | 0,008 | $1{,}58\times10^{-11}$ |
| ME,FB | CG S | 0,083 | $1{,}45\times10^{-05}$ |
| ME,FB | CG W | 0,027 | $3{,}235\times10^{-09}*$ |
| ME,FB | CG A | 0,399 | $2{,}25\times10^{-06}$ |
| ME,FB | CG B | 0,410 | $4{,}91\times10^{-07}$ |
| ME,FB | CG C | 0,325 | $5{,}31\times10^{-11}$ |
| ME,PI | | 0,325 | $1{,}07\times10^{-25}$ |

$*$ p-valor obtido pelo teste de Wilcoxon [35]

diferença significativa entre os custos apresentados pelo MultiExplorer-VM e o resultado apresentado pelo trabalho [50]. Para uma aplicação (MG C) não houve diferença significativa para os resultados e para o restante das aplicações houve diferença, demonstrando assim melhores resultados obtidos por MultiExplorer-VM.

Uma das razões para a obtenção de melhores configurações nos resultados de MultiExplorer-VM reside na exploração da heterogeneidade das configurações. Por outro lado, as restrições de utilização (4 *paramount interactions*) também podem ter afetado a geração de soluções pouco eficientes por parte da técnica PI. Adicionalmente, ressalta-se que o trabalho [50] tem como objetivo desenvolver uma solução de busca de configurações de nuvem visando somente a minimização de custo.

# Capítulo 6

# Conclusões e Trabalhos Futuros

Este trabalho apresentou uma solução para o problema de alocação de recursos em nuvem computacional a partir de uma estratégia que utiliza técnicas de exploração de espaço de projeto. Utilizando como base uma infraestrutura de exploração de espaço de projetos de sistemas computacionais, denominada MultiExplorer [27], uma extensão MultiExplorer-VM foi projetada e desenvolvida provendo soluções alternativas de máquinas virtuais para o atendimento das aplicações que demandam recursos na nuvem computacional.

No processo de desenvolvimento de MultiExplorer-VM houve a necessidade de projetar e desenvolver sistemas preditores de tempo de execução e custo de utilização de configurações de *VM*s. Para tanto, foram utilizados dados de *VM*s do provedor *AWS* e aplicações do *benchmark NPB* simuladas no simulador CloudSim. Dentre as técnicas e modelos de *machine learning* avaliados, optou-se pela utilização de modelos baseados na técnica *Decision Tree* devido ao baixo erro de predição (*MAPE*), alto *score* de teste e baixo tempo de resposta em relação aos demais modelos.

Os resultados da extensão MultiExplorer-VM foram validados estatisticamente e comparados com os resultados de um algoritmo de busca exaustiva (força bruta), além de resultados do trabalho [50] para cada aplicação do *benchmark*. Os resultados obtidos demonstraram a viabilidade da utilização de técnicas de exploração de espaço de projeto como alternativas para solução do problema de alocação de recursos em nuvem. Para a maioria das aplicações avaliadas, os resultados obtidos por MultiExplorer-VM foram melhores que aqueles obtidos pela técnica *PI* [50]. A justificativa para esses melhores resul-

tados deve-se à geração de configurações heterogêneas de *VM*s que alcançam relacionamentos viáveis entre desempenho e custo, além de maior cobertura sobre o espaço de busca de alternativas arquiteturais. Os resultados dos testes estatísticos que compararam as soluções de MultiExplorer-VM e PI evidenciaram a presença de diferença significativa entre as soluções geradas por essas duas técnicas, demonstrando assim a viabilidade de MultiExplorer-VM uma vez que obteve soluções com menor tempo e custo.

Adicionalmente, ao comparar ambas as técnicas com o algoritmo de força-bruta, MultiExplorer-VM apresentou um erro absoluto de custo de USD 0,034 na aplicação EP W, enquanto a técnica PI obteve como pior resultado um erro absoluto de custo de USD 8,052 na aplicação EP D. Para tempo, o MultiExplorer-VM apresentou um erro absoluto de quase 26s, enquanto que PI obteve erro absoluto de aproximadamente 148799s. Os resultados do MultiExplorer-VM foram melhores (menor tempo de execução das aplicações e menor custo de VM) que os da técnica PI alcançando resultados de custo de VM até 8,8 vezes menores. É importante destacar que MultiExplorer-VM conseguiu, no melhor caso, encontrar a configuração ótima de tempo em 17 aplicações e para custo foi encontrada a configuração ótima em 18 aplicações.

Ressalta-se que mesmo realizando frequentes pesquisas bibliográficas, não foram encontrados outros trabalhos na literatura que ofereça soluções para o problema de alocação de recursos em nuvem a partir de algoritmos ou técnicas de exploração do espaço de projeto. Também não foram encontrados trabalhos que apresentem configurações heterogêneas como potenciais soluções viáveis para recursos de *VM*s.

Como trabalhos futuros sugere-se avaliar as configurações heterogêneas sugeridas em provedores de infraestrutura de nuvem computacional. Além disso, entende-se que avaliações de performance das aplicações diretamente sobre as configurações de *VM*s do banco de dados podem ter um impacto mais acurado sobre o real desempenho dessas configurações sob cada aplicação e, como consequência, aumentar a acurácia real de preditores de tempo e custo.

# Referências Bibliográficas


[1] Amazon Web Services. `https://aws.amazon.com/`. Acessado: 08-02-2020.

[2] Google VM rightsizing service. `https://cloud.google.com/compute/docs/instances/apply-sizing-recommendations-for-instances`. Acessado: 08-02-2020.

[3] Intel Product Specifications. `https://ark.intel.com/content/www/us/en/ark.html#@Processors`. Acessado: 12-03-2021.

[4] Microsoft Azure. `https://azure.microsoft.com`. Acessado: 17-03-2021.

[5] Spark mllib. `https://spark.apache.org/mllib/`. Acessado: 09-05-2020.

[6] Amol C Adamuthe, Rupali M Pandharpatte, and Gopakumaran T Thampi. Multiobjective virtual machine placement in cloud environment. In *2013 International Conference on Cloud & Ubiquitous Computing & Emerging Technologies*, pages 8–13. IEEE, 2013.

[7] Omid Alipourfard, Hongqiang Harry Liu, Jianshu Chen, Shivaram Venkataraman, Minlan Yu, and Ming Zhang. Cherrypick: Adaptively unearthing the best cloud configurations for big data analytics. In *14th Symposium on Networked Systems Design and Implementation*, pages 469–482, 2017.

[8] Giuseppe Ascia, Vincenzo Catania, Alessandro G Di Nuovo, Maurizio Palesi, and Davide Patti. Efficient design space exploration for application specific systems-on-a-chip. *Journal of Systems Architecture*, 53(10):733–750, 2007.

[9] David Bailey, Tim Harris, William Saphir, Rob Van Der Wijngaart, Alex Woo, and Maurice Yarrow. The NAS parallel benchmarks 2.0. Technical report, Technical Report NAS-95-020, NASA Ames Research Center, 1995.

[10] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[11] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *25th Annual Conference on Neural Information Processing Systems*, volume 24. Neural Information Processing Systems Foundation, 2011.

[12] Bruce L Bowerman, Richard T O'Connell, and Anne B Koehler. *Forecasting, time series, and regression: an applied approach*, volume 4. South-Western Pub, 2005.

[13] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[14] Jeferson R Brunetta and Edson Borin. Selecting efficient cloud resources for hpc workloads. In *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*, pages 155–164, 2019.

[15] Rajkumar Buyya and Manzur Murshed. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and computation: practice and experience*, 14(13-15):1175–1220, 2002.

[16] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N Calheiros. Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. In *2009 International Conference on High Performance Computing & Simulation*, pages 1–11. IEEE, 2009.

[17] Krisztian Buza, Alexandros Nanopoulos, and Gábor Nagy. Nearest neighbor regression in the presence of bad hubs. *Knowledge-Based Systems*, 86:250–260, 2015.

[18] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1):23–50, 2011.

[19] Joseph P Campbell and Douglas A Reynolds. Corpora for the evaluation of speaker recognition systems. In *Proceedings of the 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 829–832. IEEE, 1999.

[20] Trevor E. Carlson, Wim Heirman, Stijn Eyerman, Ibrahim Hur, and Lieven Eeckhout. An evaluation of high-level mechanistic core models. *ACM Transactions on Architecture and Code Optimization (TACO)*, 2014.

[21] Tianfeng Chai and Roland R Draxler. Root mean square error (RMSE) or mean absolute error (MAE)– Arguments against avoiding RMSE in the literature. *Geoscientific Model Development*, 7(3):1247–1250, 2014.

[22] Carlos A Coello Coello and Col San Pedro Zacatenco. Twenty years of evolutionary multi-objective optimization: A historical view of the field. *CINVESTAV-IPN Evolutionary Computing Group*, 2005.

[23] William J Conover, Mark E Johnson, and Myrle M Johnson. A comparative study of tests for homogeneity of variances, with applications to the outer continental shelf bidding data. *Technometrics*, 23(4):351–361, 1981.

[24] FR Cordeiro and AG e Silva-Filho. NSGAII applied to unified second level cache memory hierarchy tuning aiming energy and performance optimization. In *2010 11th Symposium on Computing Systems*, pages 64–71. IEEE, 2010.

[25] Kalyanmoy Deb. Multi-objective optimization using evolutionary algorithms: an introduction. *KanGAL Report*, 2011003, 2011.

[26] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. A. M. T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[27] Rodrigo Devigo, Liana Duenha, Rodolfo Azevedo, and Ricardo Santos. Multiexplorer: A tool set for multicore system-on-chip design exploration. In *26th International Conference on Application-specific Systems, Architectures and Processors*, pages 160–161. IEEE, 2015.

[28] Liana Duenha, Marcelo Guedes, Henrique Almeida, Matheus Boy, and Rodolfo Azevedo. Mpsocbench: A toolset for mpsoc system level evaluation. In *International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation*, pages 164–171. IEEE, 2014.

[29] Katti Faceli, Ana Carolina Lorena, João Gama, and André Carlos Ponce de Leon Ferreira de Carvalho. *Inteligência artificial: uma abordagem de aprendizado de máquina*. Grupo Gen - LTC, 2011.

[30] Andrew D Ferguson, Peter Bodik, Srikanth Kandula, Eric Boutin, and Rodrigo Fonseca. Jockey: guaranteed job latency in data parallel clusters. In *Proceedings of the 7th ACM European Conference on Computer Systems*, pages 99–112, 2012.

[31] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[32] Tarun Goyal, Ajit Singh, and Aakanksha Agrawal. Cloudsim: simulator for cloud computing infrastructure and modeling. *Procedia Engineering*, 38:3566–3572, 2012.

[33] Marc Hallin. Bartlett test. *Encyclopedia of Environmetrics*, 2006.

[34] Peter Harrington. Machine learning in action (vol. 5). *Greenwich, CT: Manning*, 2012.

[35] Myles Hollander, Douglas A Wolfe, and Eric Chicken. *Nonparametric statistical methods*, volume 751. John Wiley & Sons, 2013.

[36] Fred Howell and Ross McNab. Simjava: A discrete event simulation library for java. *Simulation Series*, 30:51–56, 1998.

[37] Chin-Jung Hsu, Vivek Nair, Vincent W Freeh, and Tim Menzies. Arrow: Low-level augmented bayesian optimization for finding the best cloud vm. In *2018 IEEE 38th International Conference on Distributed Computing Systems*, pages 660–670. IEEE, 2018.

[38] Chin-Jung Hsu, Vivek Nair, Tim Menzies, and Vincent W Freeh. Scout: An experienced guide to find the best cloud configuration. *arXiv preprint arXiv:1803.01296*, 2018.

[39] Po-Sen Huang, Haim Avron, Tara N Sainath, Vikas Sindhwani, and Bhuvana Ramabhadran. Kernel methods match deep neural networks on timit. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 205–209. IEEE, 2014.

[40] Frank Hutter. *Automated configuration of algorithms for solving hard computational problems*. PhD thesis, University of British Columbia, 2009.

[41] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, pages 507–523. Springer, 2011.

[42] M. McCourt I. Dewancker and S. Clark. Bayesian optimization primer. `https://static.sigopt.com/b/20a144d208ef255d3b981ce419667ec25d8412e2/static/pdf/SigOpt_Bayesian_Optimization_Primer.pdf`, 2015. Acessado: 14-02-2020.

[43] Eunsuk Kang, Ethan Jackson, and Wolfram Schulte. An approach for effective design space exploration. In *Monterey Workshop*, pages 33–54. Springer, 2010.

[44] Gunho Lee and Randy H Katz. Heterogeneity-aware resource allocation and scheduling in the cloud. *HotCloud*, 11:4–8, 2011.

[45] PM Lerman. Fitting segmented regression models by grid search. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 29(1):77–84, 1980.

[46] Ang Li, Xiaowei Yang, Srikanth Kandula, and Ming Zhang. Cloudcmp: comparing public cloud providers. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 1–14, 2010.

[47] Sheng Li, Jung Ho Ahn, Richard D Strong, Jay B Brockman, Dean M Tullsen, and Norman P Jouppi. Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 469–480. IEEE, 2009.

[48] Sheng Li, Jung Ho Ahn, Richard D Strong, Jay B Brockman, Dean M Tullsen, and Norman P Jouppi. The McPAT framework for multicore and manycore architectures: Simultaneously modeling power, area, and timing. *ACM Transactions on Architecture and Code Optimization*, 10(1):1–29, 2013.

[49] Chi-Keung Luk, Robert Cohn, Robert Muth, Harish Patil, Artur Klauser, Geoff Lowney, Steven Wallace, Vijay Janapa Reddi, and Kim Hazelwood. Pin: building customized program analysis tools with dynamic instrumentation. *ACM sigplan notices*, 40(6):190–200, 2005.

[50] Vanderson Martins Do Rosario, Thais A Silva Camacho, Otávio O Napoli, and Edson Borin. Fast and low-cost search for efficient cloud configurations for HPC workloads. *arXiv e-prints*, 2020.

[51] Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of bayesian methods for seeking the extremum. *Towards global optimization*, 2(117-129):2, 1978.

[52] Anthony J Myles, Robert N Feudale, Yang Liu, Nathaniel A Woody, and Steven D Brown. An introduction to decision tree modeling. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 18(6):275–285, 2004.

[53] Nico J. D. Nagelkerke et al. A note on a general definition of the coefficient of determination. *Biometrika*, 78(3):691–692, 1991.

[54] Frank Austin Nothaft, Matt Massie, Timothy Danford, Zhao Zhang, Uri Laserson, Carl Yeksigian, Jey Kottalam, Arun Ahuja, Jeff Hammerbacher, Michael Linderman, et al. Rethinking data-intensive science using scalable analytics systems. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 631–646, 2015.

[55] R Lyman Ott and Micheal T Longnecker. *An introduction to statistical methods and data analysis*. Cengage Learning, 2015.

[56] Mahmoud Parsian. *Data algorithms: recipes for scaling up with Hadoop and Spark*. O'Reilly Media, Inc., 2015.

[57] Arti Patle and Deepak Singh Chouhan. SVM kernel functions for classification. In *2013 International Conference on Advances in Technology and Engineering (ICATE)*, pages 1–9. 2013 International Conference on Advances in Technology and Engineering (ICATE), 2013.

[58] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[59] Andy D Pimentel. Exploring exploration: A tutorial introduction to embedded systems design space exploration. *IEEE Design & Test*, 34(1):77–90, 2016.

[60] J Patrick Royston. An extension of Shapiro and Wilk's W test for normality to large samples. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 31(2):115–124, 1982.

[61] Ricardo Santos, Liana Duenha, Ana Caroline Silva, Matheus Sousa, Luiz Augusto Tedesco, Joao Carlos Melgarejo, Tony Santos, Rodolfo Azevedo, and Edward Moreno. Dark-silicon aware design space exploration. *Journal of Parallel and Distributed Computing*, 120:295–306, 2018.

[62] Tony Santos, Ana Silva, Liana Duenha, Ricardo Santos, Edward Moreno, and Rodolfo Azevedo. On the dark silicon automatic evaluation on multicore processors. In *28th International Symposium on Computer Architecture and High Performance Computing*, pages 166–173. IEEE, 2016.

[63] Ana Silva, Tony Bignardi, Edilson de Palma, Rafael Alves, Clara Hayashi, and Ricardo Santos. Identificação automatica de dark silicon em processadores multicore. In *Anais do Simposio de Sistemas Computacionais de Alto Desempenho-WSCAD*, 2015.

[64] Ana Caroline dos Santos Silva. Exploração do espaço de projetos de sistemas multiprocessadores guiado por dark silicon. Master's thesis, Federal University of Mato Grosso do Sul, 2017.

[65] Cristina Silvano, William Fornaciari, Gianluca Palermo, Vittorio Zaccaria, Fabrizio Castro, Marcos Martinez, Sara Bocchio, Roberto Zafalon, Prabhat Avasare, Geert Vanmeerbeeck, et al. Multicube: Multi-objective design space exploration of multi-core architectures. In *VLSI 2010 Annual Symposium*, pages 47–63. Springer, 2011.

[66] Jim Smith and Ravi Nair. *Virtual machines: versatile platforms for systems and processes*. Elsevier, 2005.

[67] Nidamarthi Srinivas and Kalyanmoy Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248, 1994.

[68] Rebecca Taft, Manasi Vartak, Nadathur Rajagopalan Satish, Narayanan Sundaram, Samuel Madden, and Michael Stonebraker. Genbase: A complex analytics genomics benchmark. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, pages 177–188, 2014.

[69] Rafael Ubal, Julio Sahuquillo, Salvador Petit, and Pedro Lopez. Multi2sim: A simulation framework to evaluate multicore-multithreaded processors. In *19th International Symposium on Computer Architecture and High Performance Computing*, pages 62–68, 2007.

[70] Shivaram Venkataraman, Zongheng Yang, Michael Franklin, Benjamin Recht, and Ion Stoica. Ernest: Efficient performance prediction for large-scale advanced analytics. In *13th Symposium on Networked Systems Design and Implementation*, pages 363–378, 2016.

[71] Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate research*, 30(1):79–82, 2005.

[72] Neeraja J Yadwadkar, Bharath Hariharan, Joseph E Gonzalez, Burton Smith, and Randy H Katz. Selecting the best vm across multiple public clouds: A data-driven performance modeling approach. In *Proceedings of the 2017 Symposium on Cloud Computing*, pages 452–465, 2017.

[73] Vittorio Zaccaria, Gianluca Palermo, Fabrizio Castro, Cristina Silvano, and Giovanni Mariani. Multicube explorer: An open source framework for design space exploration of chip multi-processors. In *23th International Conference on Architecture of Computing Systems 2010*, pages 1–7. VDE, 2010.