UNIVERSIDADE FEDERAL DO MARANHÃO

Programa de Pós-Graduação em Ciência da Computação

THIAGO NELSON FARIA DOS REIS

UM RECOMENDADOR DE ALOCAÇÃO DE RECURSOS EM COMPUTAÇÃO EM NUVEM USANDO ALGORITMOS GENÉTICOS E SVR

São Luís
2018

THIAGO NELSON FARIA DOS REIS

Um Recomendador de Alocação de Recursos em Computação em Nuvem usando Algoritmos Genéticos e SVR

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFMA como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Mário Antonio Meireles Teixeira

Coorientador: Prof. Dr. João Dallyson Sousa de Almeida

São Luís-MA

2018

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Reis, Thiago Nelson Faria dos.
Um Recomendador de Alocação de Recursos em Computação em Nuvem usando Algoritmos Genéticos e SVR / Thiago Nelson Faria dos Reis. – 2018.
72 f.

Coorientador(a): João Dallyson Sousa de Almeida.
Orientador(a): Mário Antonio Meireles Teixeira.
Dissertação (Mestrado) – Programa de Pós-graduação em Ciência da Computação/ccet, Universidade Federal do Maranhão, São Luís, 2018.

1. Algoritmo Genético. 2. Computação em Nuvem. 3. Recomendador de configuração. 4. Regressão de Vetores de Suporte. 5. Weka. I. Almeida, João Dallyson Sousa de. II. Teixeira, Mário Antonio Meireles. III. Título.

THIAGO NELSON FARIA DOS REIS

Um Recomendador de Alocação de Recursos em Computação em Nuvem usando Algoritmos Genéticos e SVR

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFMA como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Trabalho aprovado. São Luís-MA, 19 de junho de 2018:

Mário Antonio Meireles Teixeira
Dr. em Ciência da Computação - UFMA
Orientador

João Dallyson Sousa de Almeida
Dr. em Ciência da Computação - UFMA
Coorientador

André Castelo Branco Soares
Dr. em Ciência da Computação - UFPI

Anselmo Cardoso de Paiva
Dr. em Ciência da Computação - UFMA

São Luís-MA
2018

Este trabalho é dedicado à minha Família.

Agradecimentos

Deus em primeiro lugar.

À minha família, em especial a meu pai, pelo constante incentivo à evolução e apoio, aos meus irmãos e irmã pelos incentivos. À minha esposa Apoena, pelo companheirismo, paciência e dedicação, aos meus filhos Bianca e Guilherme pela compreensão pelas horas dedicadas aos estudos e à Rosângela pelo apoio e incentivo, que acreditaram no meu potencial, e sempre estiveram dispostos a me ajudar em tudo que fosse necessário.

Aos professores do curso de Pós-graduação em Ciência da Computação da Universidade Federal do Maranhão, pelo conhecimento que me proporcionaram e por terem me mostrado o que existe fora da caverna.

Ao professor Mário Teixeira, meu orientador, e aos professores Anselmo e João Dallyson, pela paciência, dedicação e conhecimentos passados.

Aos professores Geraldo e Carlos Salles, pelas dicas e orientações em projetos durante o mestrado.

Por fim, agradeço a todos que direta ou indiretamente contribuíram para a realização deste trabalho.

”... Para os acorrentados, as sombras eram cópias de objetos reais e os ecos eram emitidos pelas sombras. Sombras nas paredes e ecos pela caverna era a única realidade verdadeira para aqueles homens ali presentes. Porém, um dos homens consegue se libertar das correntes. Quando ele vai em direção à entrada da caverna, sente inúmeras dificuldades devido a sua não adaptação à luz e aos sons fora da caverna. Após a acomodação fora da caverna e estranhamento, pois até então, tudo aquilo era impossível e irreal, ele compreende o tamanho de sua ignorância. E logo, depois de várias reflexões, percebe que a realidade verdadeira está fora da caverna e é muito mais bela. ... ”

(Platão)

Resumo

A alocação de recursos em Computação em Nuvem tem sido feita de forma reativa, podendo gerar falhas de garantia de serviço e a incidência de cobrança dos recursos ociosos. A fim de minorar esses problemas, este trabalho tem por objetivo apresentar uma solução preditiva de alocação de recursos, na forma de um Recomendador de Configurações, utilizando Regressão de Vetores de Suporte (SVR) e Algoritmos Genéticos (AG). Como estudo de caso, são escolhidas aplicações de aprendizado de máquina baseadas na ferramenta Weka. Esta combinação é empregada para estimar o tempo de execução das aplicações e recomendar uma configuração viável e válida de recursos na nuvem, tendo como base o cálculo do tempo de execução e dos custos. Os resultados obtidos demonstram que os tempos previstos tiveram uma acurácia de 94,71% em relação aos tempos reais, levando assim a uma estimativa eficaz de tempo e custo e obteve-se, em alguns casos de execução em ambiente em nuvem, uma redução de tempo e custo de 38,8% e 45,62%, respectivamente.

Palavras-chaves: Computação em Nuvem, Algoritmo Genético, Regressão de Vetores de Suporte, Recomendador de configuração, Weka.

Abstract

The allocation of resources in Cloud Computing has been done in a reactive way, what can generate service guarantee failures and the charging of idle resources. In order to alleviate these problems, the objective of this work is to present a predictive resource allocation solution, in the form of a Configuration Recommender, using Support Vector Regression (SVR) and Genetic Algorithms (GA). As a case study, machine learning applications based on the Weka tool are chosen. This arrangement is used to compute application execution time and recommend a viable and valid configuration of cloud resources, based on the estimation of execution time and costs. The results show that the predicted times had an acuracy of 94.71% in relation to the real ones, thus leading to an efficient estimation of time and cost. In some cases of execution in the cloud environment, there was a reduction of time and cost of 38.8% and 45.62%, respectively.

Keywords: Cloud Computing, Genetic Algorithms, Support Vector Regression, Configuration Recommender, Weka.

Lista de figuras

Lista de tabelas

Lista de gráficos

Lista de abreviaturas e siglas

AG	Algoritmo Genético
Amazon AWS	*Amazon Web Services*
API	*Application Programming Interface*
ARIMA	*Autoregressive Integrated Moving Average*
AWS	*Amazon Web Services*
B	Modo *Fitness* de Custo Benefício
C	Modo *Fitness* de Custo
EC2	*Elastic Compute Cloud*
Eucalyptus	Elastic Utility Computing Architecture for Linking Your Programs to Useful Systems
IaaS	*Infraestruture as a Service*
JSON	*avaScript Object Notation*
KDD	*Knowledge Discovery in Database*
LR	*Linear Regression*
MV	*Máquina Virtual*
MVS	Máquina de Vetores de Suporte
N1	Instâncias com 1 núcleo
N2	Instâncias com 2 núcleos
N4	Instâncias com 4 núcleos
N8	Instâncias com 8 núcleos
PSO	*Particle Swarm Optimization*
QoS	*Quality of Service*

RBF	*Radial-Basis Function*
REST	*Representational State Transfer*
RL	Regressão Linear
RN	Redes Neurais
RVS	Regressor de Vetores de Suporte
SaaS	*Software as a Service*
SLA	*Service Level Agreement*
SRM	*Structural Risk Minimization*
SVM	*Suporte Vector Machine*
SVR	*Support Vector Regression*
T	Modo *Fitness* de Tempo
URI	*Uniform Resource Identifier*
V1	Custo de 1 segundo com instâncias com 1 núcleo
V2	Custo de 1 segundo com instâncias com 2 núcleos
V4	Custo de 1 segundo com instâncias com 4 núcleos
V8	Custo de 1 segundo com instâncias com 8 núcleos
VM	*Virtual Machine*

Sumário

1 Introdução

Até um passado recente, os serviços em nuvem eram contratados baseados na maior demanda que o usuário possuía, ficando uma grande parte do tempo com os recursos computacionais ociosos mas sendo pagos, aguardando que fossem utilizados. Esse modelo tem mudado, na medida em que os provedores de nuvem estão conseguindo analisar o uso, predizer e automatizar os seus serviços, gerando assim uma melhor redistribuição dos recursos, uma vez que a alocação se torna mais precisa, tanto em termos de recursos quanto de tempo, gerando assim um uso mais eficaz por parte do provedor e redução de custos ao cliente, que irá pagar realmente o necessário, sem desperdícios.

A redução de custos é o desafio, tanto dos provedores quanto dos usuários da computação em nuvem: como determinar quais recursos serão necessários e por quanto tempo? De acordo com Zhu et al. (2016), a maioria dos serviços de análise é reativa, ou seja, somente após a demanda se mostrar necessária é que os recursos serão provisionados, normalmente em até 5 minutos, e leva ainda um certo tempo até que seja identificado que os recursos não estão mais sendo usados, em média 15 minutos, sem levar em consideração o tempo ainda necessário para os recursos estarem disponíveis. Enquanto essa análise e disponibilidade são realizadas, existe o risco da qualidade do serviço (QoS) e do nível de serviço (SLA) contratados pelo cliente não serem garantidos, além da cobrança desnecessária de recursos.

Uma forma de resolver esse problema é a mudança da postura de análises reativas para análises preditivas ou pró-ativas, ou seja, identificar quais recursos serão necessários antes da execução e alocá-los automaticamente.

Aprendizado de Máquina, através de Regressão Linear (RL), Redes Neurais (RN) e Máquina de Vetores de Suporte (MVS) têm sido amplamente utilizados para solução de problemas de predição em várias áreas. As RNs podem ser usadas para gerar possíveis melhores soluções para problemas que não possuem uma única resposta. Já as MVS possuem a capacidade de gerar uma saída prevista com alto nível de confiança, em especial o *Support Vector Regression (SVR)* utilizado na análise de regressão com dados históricos (ZHU et al., 2016; SEMBIRING; BEYER, 2013; PARIKH, 2013).

Nesta dissertação, é apresentado um recomendador de configurações de máquinas virtuais em nuvem baseado em dados históricos de aplicações, tendo como estudo de caso o Weka (WEKA, 2017), usando SVR e Algoritmos Genéticos (AG) a fim de sugerir uma configuração ideal mediante o objetivo colocado e predizer o tempo e custo das execuções. Foi realizado

o levantamento dos dados históricos e consequente treinamento do regressor e otimização do mesmo. Para verificar a exatidão da predição, foram executadas em ambiente real as configurações sugeridas e comparadas com o tempo calculado a fim de validar a estimativa.

Apesar do modelo ter sido implementado em um ambiente de nuvem privada e utilizando uma base de dados para a aplicação específica do Weka, o modelo e o protótipo podem ser expandidos para ambientes de nuvem pública, bem como para outras aplicações que podem se beneficiar do modelo distribuído e paralelo da nuvem.

O uso de um recomendador, independente de qual plataforma está sendo utilizada, seja privada, pública ou híbrida, tem como principal benefício o uso mais eficiente e eficaz dos recursos. Isso permite um melhor aproveitamento do parque tecnológico bem como a otimização do tempo dos usuários. Mesmo que o uso não seja pago, por exemplo, no caso de uma nuvem privada em uma instituição, o uso do recomendador permite, além da melhor utilização, redução do tempo de execução das tarefas e maior quantidade de usuários utilizando os mesmos recursos, gerando assim, economia e benefícios em todos os aspectos.

1.1 Trabalhos Relacionados

O uso de arquitetura em nuvem, pública, híbrida ou privada, tem impulsionado o desenvolvimento de métodos computacionais para o uso eficiente dos recursos computacionais bem como a redução de custos, tando no Provedor de Serviços quanto para o usuário.

Dentre as formas de se analisar e estimar o uso dos recursos, tem se destacado o uso de Redes Neurais, Regressão Linear e Máquina de Vetor de Suporte. Na sequência, serão apresentados alguns trabalhos que utilizam Aprendizado de Máquina e Inteligência Computacional com esta finalidade em ambientes em nuvem.

Em Zhu et al. (2016), é apresentada uma estratégia de alocação de recursos pró-ativa baseada na predição de demanda de uso de recursos, utilizando *Support Vector Regression* (SVR), juntamente a *Particle Swarm Optimization* (PSO), com o objetivo de melhorar a acurácia da predição. O resultado desta pesquisa comprovou que a predição foi mais eficiente e precisa que a utilização de métodos tradicionais de SVR e Regressão Linear (LR).

Em Patel, Chaudhary e Garg (2016), demonstra-se a eficácia da utilização de técnicas de predição na migração on-line de máquinas virtuais. O comparativo foi realizado entre métodos estatísticos de probabilidade para definição das páginas de memória que deveriam ser migradas, usando o algoritmo ARIMA (*Autoregressive Integrated Moving Average*). O segundo estudo

utilizou aprendizado de máquina baseado em regressão usando o modelo SVR. O modelo ARIMA obteve uma acurácia de 91,74%, enquanto o modelo SVR alcançou 94,61% na predição de páginas de memórias sujas, constatando-se a superioridade com relação ao ARIMA.

O aprendizado de máquina também está sendo usado em diversas outras áreas, no caso Sembiring e Beyer (2013) o utilizaram na predição de recursos da computação em nuvem para aplicações multimídias, com o objetivo de manter um maior equilíbrio no uso dos recursos computacionais do ambiente em nuvem e para garantir a qualidade do serviço. Para tanto, foi desenvolvida uma solução de alocação de recursos dinâmicos através do uso de aprendizado de máquina para que em cada tarefa fosse feita uma predição dos recursos de acordo com o histórico de uso. Os experimentos consideraram a utilização de processador e mostraram um uso mais eficiente se comparado aos métodos convencionais de alocação.

Vários estudos na área da computação em nuvem vêm sendo realizados em aprendizado de máquina para melhorar a utilização dos recursos e consequentemente a redução de custos, tanto para o usuário quanto para o provedor. Em Prevost et al. (2011), foi feito um estudo usando Redes Neurais (RN), criando um algoritmo usando autorregressão linear e RN para prever a carga de rede das aplicações no serviço de nuvem. O modelo desenvolvido combina predição de carga de rede, modelos estocásticos e alocação de recursos, para minimizar o consumo de energia e manter o desempenho requerido.

Outra linha de pesquisa usando a predição e computação em nuvem é sobre a diminuição da intervenção humana. Isso pode ser observado em Niehorster et al. (2011), Bankole e Ajila (2013) e Hormozi et al. (2012), que utilizam aprendizado de máquina através do modelo SVM para alocação de recursos, configuração de aplicações e geração de estimativas de custos com alta precisão e para garantir o QoS (*Quality of Service*) exigido, resultando no uso mais eficaz e eficiente dos recursos e redução dos custos, tanto operacionais quanto de manutenção.

As Redes Neurais Artificiais também foram utilizadas por Mendonça et al. (2016) na otimização da análise estrutural de placas laminadas, substituindo o método dos Elementos Finitos, que possui um alto custo computacional. Essa pesquisa se mostrou eficiente, pois os resultados permitiram aproximar o cálculo da segurança à flambagem das placas com os valores reais.

Os trabalhos de Zhu et al. (2016), Patel, Chaudhary e Garg (2016), Sembiring e Beyer (2013), Prevost et al. (2011), Niehorster et al. (2011), Bankole e Ajila (2013), Hormozi et al. (2012) demonstram a eficiência do SVR para estimativa com séries históricas. Seguindo os trabalhos citados, a presente dissertação também o utiliza para a estimativa, mas com o objetivo

de predizer o tempo de execução nas mais diversas configurações possíveis em um ambiente em nuvem. Usando o Algoritmo Genético para criar essas possíveis configurações, associada à estimativa do tempo, construiu-se um protótipo capaz de sugerir uma ótima configuração, senão a melhor, para a execução de determinada aplicação, além de estimar o tempo de execução e o custo do ambiente.

Os trabalhos relacionados acima indicam serem promissoras as pesquisas de técnicas de predição no uso de recursos computacionais, principalmente em ambientes em nuvem, assim como a utilização de técnicas de Inteligência Computacional para alocação dos recursos necessários.

1.2 Objetivos

O objetivo deste trabalho é especificar e implementar um recomendador de configurações para recursos em nuvem baseado em dados históricos de aplicações usando SVR e Algoritmos Genéticos (AG) a fim de predizer o tempo e o custo de execuções a partir de uma configuração recomendada. Como estudo de caso, foi selecionado o software de análise e mineração de dados Weka, muito utilizado na área de aprendizagem de máquina.

1.2.1 Objetivos Específicos

Os objetivos específicos são:

- Configurar uma nuvem privada compatível com a Amazon AWS.
- Executar a aplicação Weka de forma distribuída e paralela no ambiente em nuvem.
- Gerar uma base de conhecimento de execução de aplicações Weka em nuvem.
- Disponibilizar a execução de aplicações Weka como um serviço em nuvem.

1.3 Contribuições do Trabalho

As principais contribuições deste trabalho podem ser resumidas em três partes:

- A geração da base de conhecimento de execução de aplicações Weka na nuvem.
- A utilização do SVR para estimativa do tempo e custo em ambiente em nuvem de execução de aplicações Weka.

- O desenvolvimento de um novo método de predição de configuração, tempo e custo através da construção do recomendador integrando SVR com AG.

Os resultados obtidos neste trabalho poderão tornar possível a incorporação do método proposto em uma ferramenta para várias áreas de pesquisa que utilizam a ferramenta Weka, de forma a auxiliar os pesquisadores a melhorar o planejamento tanto em termos de cronograma quanto de custos.

1.4 Organização do Trabalho

O restante deste trabalho está organizado da seguinte forma: O Capítulo 2 apresenta a fundamentação teórica necessária para seu embasamento. Serão descritos os conceitos associados à Computação em Nuvem, definições, características, modelos, algorítimos utilizados e seus componentes necessários, bem como a arquitetura do Eucalyptus para criação de nuvem privada. Na área de Mineração de Dados e da ferramenta Weka, são descritos conceitos, definições e características, além das técnicas de Reconhecimento de Padrões e Inteligência Computacional por meio de Máquina de Vetores de Suporte e os Algoritmos Genéticos.

O Capítulo 3 descreve as etapas do processo desenvolvido, com o objetivo de implementar o Recomendador que faz a sugestão de configuração e estimativa de tempo e custo da execução de aplicações. O Capítulo 4 apresenta os resultados obtidos com a prototipação e faz uma breve discussão destes resultados. Por fim, o Capítulo 5 apresenta as conclusões da dissertação.

2 Fundamentação Teórica

Neste capítulo é apresentada a fundamentação teórica utilizada no desenvolvimento desta dissertação, importante para a compreensão das técnicas e ferramentas utilizadas para alcançar os objetivos esperados.

2.1 Computação em Nuvem

A definição de computação em nuvem segundo Mell e Grance (2011), Herbst, Kounev e Reussner (2013) é ”um modelo para permitir acesso à rede de modo onipresente, conveniente e sob demanda a um conjunto compartilhado de recursos de computação configuráveis (redes, servidores, armazenagem, aplicativos e serviços) que possam ser rapidamente provisionados e liberados com o mínimo de esforço de gerenciamento ou interação do prestador de serviços.”

Ainda Parikh (2013) cita que ”computação em nuvem é um tipo de sistema distribuído e paralelo que consiste na interconexão de computadores virtualizados que são dinamicamente provisionados e apresentados como um ou mais recursos computacionais unificados baseados em acordos de nível de serviço, estabelecidos através de negociação entre o provedor de serviços e consumidores.”

2.1.1 Serviços em Nuvem

Ainda segundo Lorido-Botran, Miguel-Alonso e Lozano (2014), Kupferman Jeff Silverman (2016), Parikh (2013), existem quatro modelos básicos de serviços que podem ser oferecidos quando se trata de computação em nuvem, detalhados abaixo e representados na Figura 1:

- **SaaS (*Software as a Service*)**: *Software* (aplicativo) como serviço é provavelmente o mais comum e popular entre as empresas e usuários. No SaaS, o software é utilizado diretamente na nuvem e se torna, assim, livre do pagamento de licenças por sua utilização.
- **PaaS (*Platform as a Service*)**: O objetivo das Plataformas como Serviço, PaaS, é o de disponibilizar um ambiente de desenvolvimento em nuvem com ferramentas e bibliotecas.
- **IaaS (*Infrastructure as a Service*)**: Na Infraestrutura como Serviço, é colocado à disposição do usuário um conjunto de recursos de hardware, como CPUs, memória RAM,

roteadores, racks, *firewall*, os quais podem e devem ser configurados livremente pelo usuário contratante.

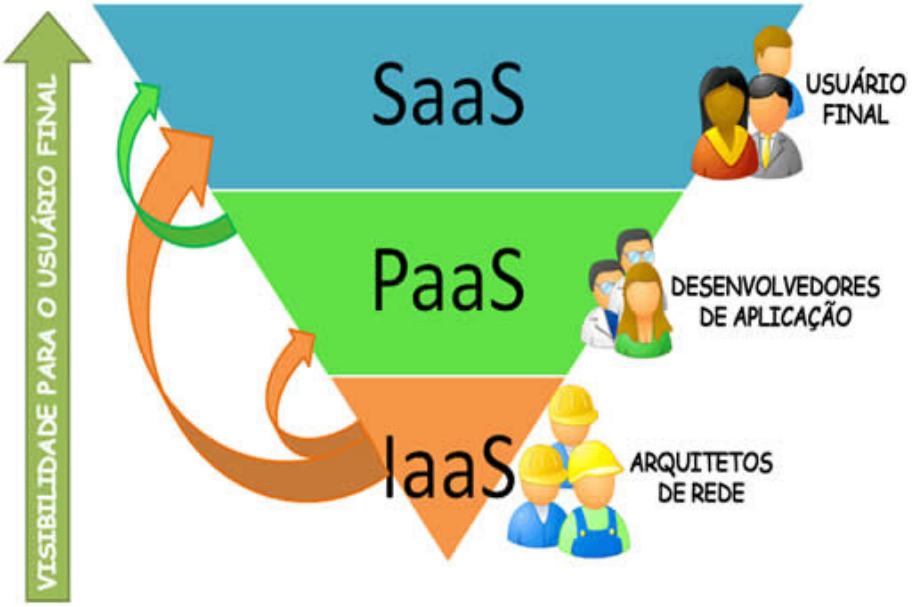


Figura 1: Modelos de Serviço em Nuvem. Fonte: (SEGURANçA, 2016)

2.1.2 Arquitetura em Nuvem

No contexto de Arquitetura em Nuvem, normalmente são aceitos quatro modelos, segundo Cunha et al. (2017), Parikh (2013), detalhados a seguir e representados na Figura 2:

- **Nuvem privada**: A ideia de nuvem privada é de que a infraestrutura esteja, se não na totalidade, na maior parte dentro de uma estrutura privada e que funcione somente para uma única organização. Alguns softwares utilizados para implementar essa arquitetura são o HPE Eucalyptus e Openstack.
- **Nuvem pública**: A nuvem pública é o exemplo mais clássico de nuvem, principalmente pelo fato de os serviços serem entregues através da internet. Sua principal característica é o compartilhamento de recursos na internet. Como exemplo tem-se a Amazon AWS e Google Cloud.
- **Nuvem híbrida**: Como forma de simplificar essa definição, a nuvem híbrida é uma mistura dos modelos acima. Utiliza recursos da nuvem privada e da nuvem pública, agregando serviços dos vários modelos. A grande questão nesse modelo é a de definir o que irá para a nuvem pública e o que ficará na nuvem privada. Por exemplo, podemos citar a utilização

da nuvem privada, com o HPE Eucalyptus, e quando não existir mais recursos disponíveis pode ser usado os recursos da pública, como da Amazon AWS.

- **Nuvem comunitária**: A nuvem comunitária possui partes em comum entre a nuvem privada e a pública, pois é uma nuvem privada porém utiliza estrutura do tipo pública para disponibilizar recursos para um grupo específico de organizações, com o mesmo objetivo. Ou seja, seria uma nuvem pública para um grupo restrito de usuários.

Figura 2: Modelos de Arquitetura em Nuvem. Fonte: (SCURRA, 2016)

Esses modelos podem ser utilizados de forma específica e isolada, ou podem ser combinados de acordo com a infraestrutura desejada, como, por exemplo, a criação de uma infraestrutura híbrida de nuvem privada e pública, conforme Johnson (2011), Sousa et al. (2016).

2.1.3 Características de computação em Nuvem

De acordo com Cunha et al. (2017), Herbst, Kounev e Reussner (2013), Lorido-Botran, Miguel-Alonso e Lozano (2014), existem cinco características essenciais para que se tenha um serviço em nuvem:

- **Autoserviço sob demanda**: Possibilidade de alterar os recursos disponibilizados, como armazenamento, memória e demais recursos através do próprio usuário.

- **Acesso amplo à rede**: Leva em consideração a independência da localização física dos recursos, ou seja, não interessa onde o recurso está sendo aprovisionado e sim que esteja disponível.
- **Agrupamento de recursos**: Possibilidade de compartilhar e agrupar vários recursos entre diversos usuários.
- **Elasticidade rápida**: Possibilidade de os recursos serem aprovisionados de forma automática ou com o mínimo de esforço possível, permitindo uma rápida resposta à demanda.
- **Serviço medido**: Os serviços são monitorados e medidos, como forma de cobrar somente pelos recursos realmente utilizados.

2.1.4 Escalabilidade e Elasticidade

A ideia básica de um sistema escalável é a possibilidade de se adicionar recursos, seja por meio de cópias de máquinas virtuais, seja pela expansão das já existentes, como processador ou memória, de acordo com a demanda exigida, de forma satisfatória, conforme afirma Lorido-Botran, Miguel-Alonso e Lozano (2014). Escalabilidade e Elasticidade são inter-relacionadas mas possuem conceitos distintos, apesar de a escalabilidade ser uma condição para que a elasticidade possa acontecer:

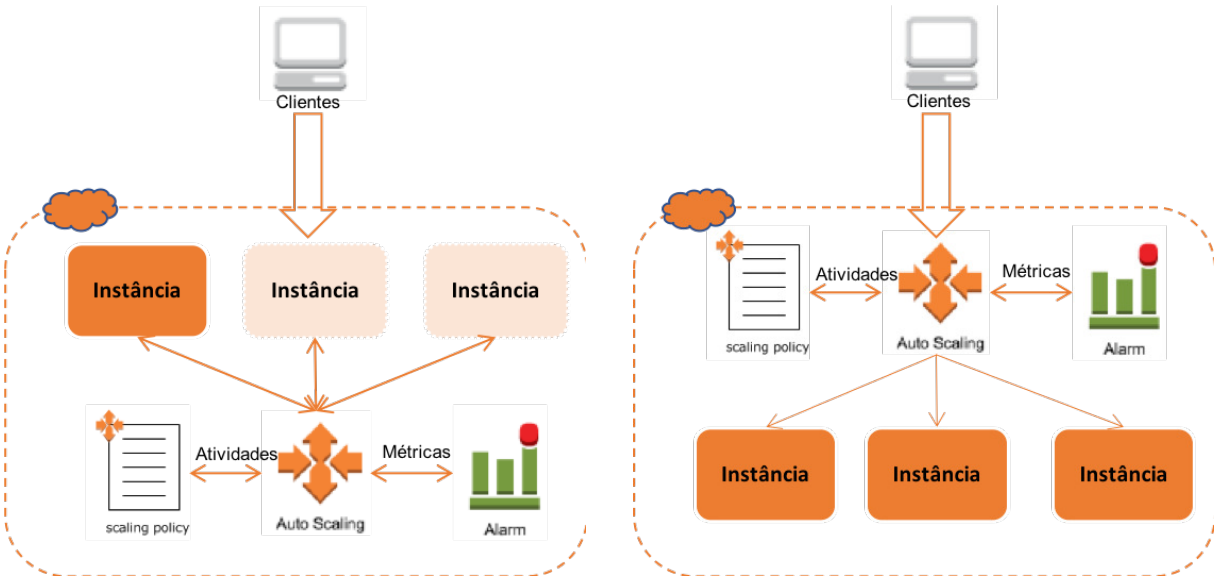
- **Elasticidade**: É a habilidade que um sistema tem de se adaptar às mudanças de carga de trabalho e de demanda de recursos, seja de forma manual ou automática, conforme Brebner (2012).
- **Escalabilidade**: Escalabilidade é o sistema que inclui o hardware, a virtualização e os softwares necessários para permitir a elasticidade, também de acordo com Brebner (2012).

O escalonamento pode ser realizado de duas formas no Sistema em Nuvem:

- **Horizontal**: Nesse caso adiciona-se novas máquinas virtuais ao Sistema, aumentando assim a quantidade de nós que fazem parte do mesmo.
- **Vertical**: Nessa situação são adicionados novos recursos às máquinas virtuais já existentes, em tempo de execução, como processadores, memória, etc.

Na Figura 3, têm-se uma representação do funcionamento dos algoritmos reativos e dos proativos. Nos Reativos (3a), a submissão é feita ao serviço em nuvem e executada com os nós

existentes. Ao longo do tempo, o autoescalonador monitora os recursos das instâncias através dos alarmes criados e as políticas de escalonamento. Caso um alarme seja disparado, são criadas ou destruídas instâncias ou adicionados ou removidos recursos ao grupo, e após essas tarefas estes ficam disponíveis ao usuário. Já no modelo Proativo (3b), antes da execução é feita uma análise pelo autoescalonador, juntamente com os alarmes e políticas de escalonamento, e a infraestrutura é preparada para a submissão das requisições. Somente após os recursos estarem disponíveis é que é iniciada a execução. Os modelos híbridos começam o seu funcionamento no modelo proativo e, durante a execução, têm o comportamento do modelo reativo, monitorando e adequando os recursos ao longo da execução, de acordo com Lorido-Botran, Miguel-Alonso e Lozano (2014).



(a) *Algoritmos Reativos.* (b) *Algoritmos Proativos.*

Figura 3: Algoritmos de Autoescalonamento.

2.1.5 Nuvem Privada e HPE Eucalyptus

A utilização do modelo de Nuvem Privada permite às empresas e instituições utilizarem o seu parque tecnológico e criarem um ambiente em nuvem local, com todas as funcionalidades existentes no modelo de nuvem pública, inclusive de mesclar os dois modelos, gerando assim o modelo híbrido, segundo Silva João; Vaz (2013), Lassenius (2016).

Existem duas soluções comumente utilizadas para o modelo privado, que é o Openstack (OPENSTACK, 2017) e o HPE Eucalyptus (HP, 2016b). As duas soluções possuem características semelhantes, mas para este trabalho foi selecionado o Eucalyptus, já que utiliza as mesmas

bibliotecas da Amazon AWS (AMAZON, 2016), com recursos semelhantes ao EC2 (*Elastic Compute Cloud*) e autoescalonamento, consequentemente facilita a migração de soluções entre as duas plataformas e a comunicação entre elas, bem como a implementação de uma futura nuvem híbrida, conforme Sousa et al. (2016), Lassenius (2016), HP (2016a).

O Eucalyptus tem o objetivo de permitir que as empresas criem ambientes de nuvem privada, utilizando o parque tecnológico já existente na organização e facilitando uma futura migração para nuvem pública ou híbrida, como a AWS, sem necessidade de grandes mudanças, já que ambas as arquiteturas de nuvem se utilizam dos mesmos modelos de comunicação, segundo Silva João; Vaz (2013), Lassenius (2016).

As principais características do HPE Eucalyptus são:

- **Auto-scaling**: Segue a mesma ideia do auto-escalonamento da Amazon e também fornece APIs que ajudam a definir regras e políticas que permitem escalonamento de forma dinâmica, com base em flutuações de cargas de trabalho, utilizando tanto algoritmos proativos quanto reativos.
- **Elastic Load Balancing**: É o responsável por distribuir a carga de trabalho entre as várias instâncias que fazem parte do mesmo grupo.
- **CloudWatch**: Responsável por monitorar as máquinas virtuais e os aplicativos, além de ser responsável pela execução de tarefas agendadas.
- **Resource tagging**: Permite que sejam atribuídos metadados significativos para determinados recursos da nuvem. Isso ajuda no rastreamento, no gerenciamento e monitoramento de coleções de recursos específicos utilizados em toda a nuvem.
- **Modo de manutenção**: Permite que sejam executadas tarefas de manutenção sem que haja indisponibilidade dos recursos.

Os componentes do Eucalyptus são descritos a seguir e representados na Figura 4:

- **Cloud Controller - CLC**: O CLC é a interface web do Eucalyptus, que permite acessar os recursos e gerenciar as instâncias. Permite também extrair relatórios e demais recursos de administração.
- **Walrus**: O Walrus é responsável pelo armazenamento persistente para as máquinas virtuais. É o sistema de discos do Eucalyptus.
- **Cluster Controller - CC**: É o responsável por monitorar e controlar os grupos de instâncias e assim manter os SLA's e aprovisionar recursos para os nós.

- **Storage Controller - SC**: O controlador de armazenamento fornece armazenamento em nível de bloco persistente para cada uma das máquinas virtuais, ou seja, é o responsável pelo gerenciamento dos discos virtuais das máquinas.
- **Node Controller - NC**: Os controladores de nó são servidores reais baseados em *hypervisor* que hospedam as máquinas virtuais.

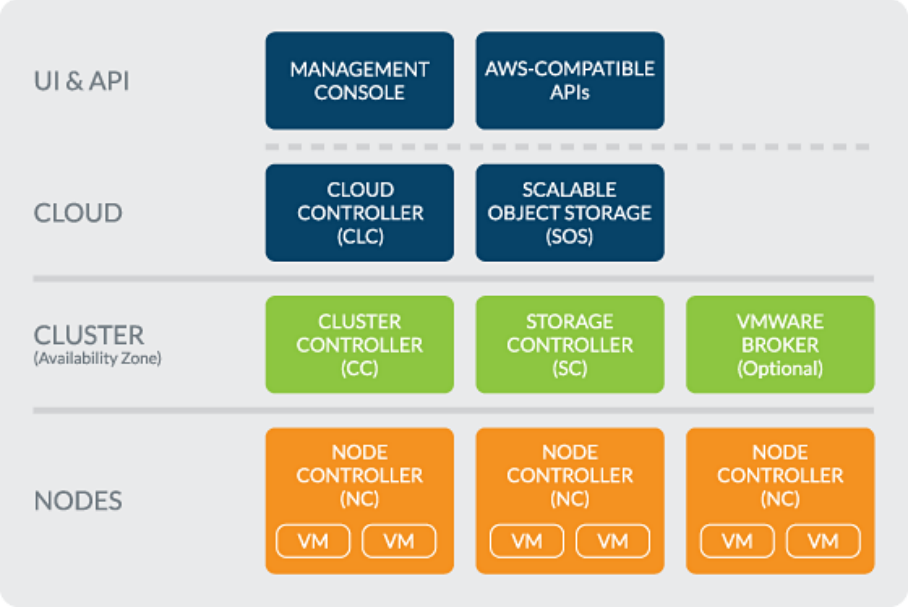


Figura 4: Componentes do HPE Eucalyptus. Fonte: (HP, 2016b)

Segundo Hsieh et al. (2015), Mohapatra et al. (2017), o Eucalyptus possui suporte às virtualizações do Xen,VMwar e KVM, sendo a última o padrão. Isso permite uma maior compatibilidade entre as diversas soluções já existentes, facilitando assim a migração das virtualizações legadas para o ambiente em nuvem. Ainda possui suporte para a virtualização de redes e switchs, através da arquitetura *Open Flow*, que permite um controle de fluxo de dados, bem como definição de prioridades, através da criação de redes virtuais e direcionamento de pacotes.

A arquitetura do Eucalyptus foi desenvolvida para prover suporte a escalonamento automático, transparência na migração de máquinas virtuais entre os nós, escalabilidade facilitada através do adicionamento somente de controladores de nós, autenticação integrada com o Amazon AWS, implementação da biblioteca AWS, suporte a vários Sistemas Operacionais para as máquinas virtuais, desenvolvimento em várias linguagens como Java e Python, prover SaaS, IaaS, PaaS e tolerância a falhas, dentre outros recursos, segundo Hsieh et al. (2015), Mohapatra et al. (2017), Silva João; Vaz (2013), Lassenius (2016), Sousa et al. (2016).

Este modelo desenvolvido permite a migração do modelo de nuvem privado para o modelo híbrido, uma vez que, caso não haja mais recursos disponíveis na nuvem privada, pode ser alocado recursos da pública de forma transparente e automática. E a medida que não há mais necessidade, os recursos da nuvem pública são desalocados. Isso permite um uso mais econômico do modelo e ao mesmo tempo, amplia as possibilidades e recursos disponíveis.

2.2 Mineração de Dados

A mineração de dados é uma etapa na Descoberta de Conhecimento (KDD - *Knowledge Discovery in Database*) que consiste em um processo não trivial de identificação de novos padrões válidos, úteis e compreensíveis, segundo Avelar, Rocha e Cruz (2017). Na Figura 5, tem-se as etapas ou tarefas que são realizadas na descoberta do conhecimento. A mineração de dados é uma das tarefas compreendidas no processo.

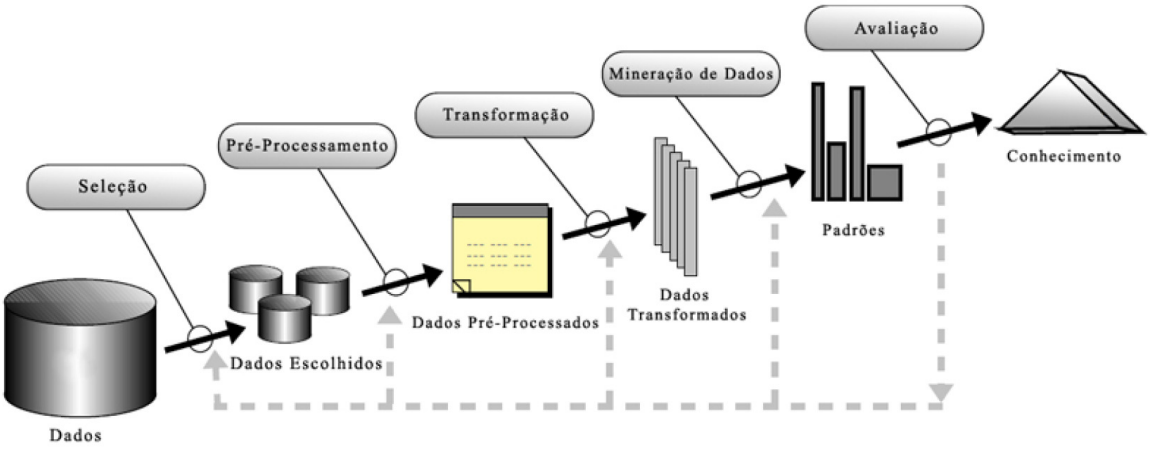


Figura 5: Processo do KDD. Fonte: (AVELAR; ROCHA; CRUZ, 2017)

Um dos objetivos da Mineração de Dados é a extração de novos conhecimentos a partir de uma base, normalmente para tomada de decisão, sendo muito utilizado o aprendizado de máquina para este fim. Segundo Rezende (2003), Witten et al. (2016), Sharma, Bajpai e Litoriya (2012), Aprendizado de Máquina é visto como um sistema no qual, a partir de um conjunto de dados, é possível obter conhecimentos e analisá-los. Para implementar um sistema deste tipo devem ser considerados requisitos da aprendizagem como, por exemplo, a aprendizagem dedutiva e indutiva. Ainda segundo Rezende (2003), o aprendizado de máquina somente utiliza o indutivo, uma vez que o uso dessa técnica permite que novos conhecimentos sejam obtidos

a partir de uma base de dados composta por fatos, enquanto que o método dedutivo utiliza a análise lógica para obter uma conclusão utilizando argumentos e premissas.

A técnica da Mineração de Dados, segundo Witten et al. (2016), Avelar, Rocha e Cruz (2017), é a descoberta de padrões, normalmente, desconhecidos, assim como de modelos descritivos, compreensíveis e preditivos a partir de dados em grande escala, através de técnicas de descobertas de correlações significativas, padrões e tendências, a partir da análise dos dados já coletados, usando reconhecimento de padrões, estatísticas e técnicas matemáticas.

Sendo assim, pode-se afirmar, de acordo com Rezende (2003), Avelar, Rocha e Cruz (2017), Sharma, Bajpai e Litoriya (2012), que os dois principais objetivos da Mineração de Dados são a descoberta de relação entre os dados e o fornecimento de subsídios para que se possa fazer uma previsão de tendências futuras baseadas em dados passados. A mineração tem sido utilizada nas áreas da medicina, marketing, governo, finança e outras, conforme Avelar, Rocha e Cruz (2017), Rezende (2003).

Ainda segundo Avelar, Rocha e Cruz (2017), Rezende (2003), é possível realizar predições numéricas utilizando métodos que têm o objetivo de descobrir um valor futuro em uma variável, normalmente para variáveis contínuas. Para tanto, modelam o relacionamento de variáveis independentes (preditoras) com uma variável dependente (resposta), onde as preditoras são os atributos dos dados e a resposta é a que se quer predizer. Já para as variáveis discretas, métodos de classificadores podem ser utilizados.

As regressões podem ser Linear ou Não-Linear, onde na Linear a relação entre as variáveis preditoras e a resposta segue um comportamento linear. Já na Regressão não-linear, a relação entre as variáveis preditoras e a resposta não segue um comportamento linear, como, por exemplo, a relação pode ser uma função polinomial. Existem técnicas que convertem regressões não-lineares para lineares.

As principais etapas realizadas na mineração são descritas a seguir, de acordo com Rezende (2003), Avelar, Rocha e Cruz (2017), Witten et al. (2016):

- **Descrição**: Consiste em descrever os padrões e tendências revelados pelos dados. Esta tarefa é comumente associada às técnicas de análise exploratória de dados, para comprovar as relações entre as variáveis e os resultados.
- **Classificação**: Construção de um modelo aplicado aos dados não classificados, com a finalidade de categorizar esses dados em classes e descobrir uma relação entre os atributos metas, que são os que interferem no resultado, e os atributos de previsão, que são os

que serão estimados na regressão, por exemplo, a relação entre o tamanho do arquivo, quantidade de processadores e o tempo de execução.

- **Estimativa ou regressão**: Usada para definir um valor para uma variável contínua desconhecida analisando os valores das demais, por exemplo, estimar o tempo de execução de uma aplicação a partir do tamanho do arquivo e da quantidade de processadores.
- **Associação**: Tem o objetivo de determinar quais itens estão inter-relacionados em uma mesma transação.
- **Segmentação ou Agrupamento**: Divisão dos dados heterogêneos em subgrupos e grupos mais homogêneos possíveis.
- **Sumarização**: Método para encontrar uma descrição para um subconjunto de dados.
- **Séries temporais**: Semelhante à etapa de Estimativa, mas utiliza medidas baseadas no tempo, explorando as variações que podem existir em períodos específicos.

Para a construção do regressor utilizou-se as etapas de Descrição e Classificação para a análise dos dados bem como a correlação entre os mesmos. Já a etapa de Estimativa, por ser a principal etapa a ser explorada, foi utilizada na construção, testes e validações além de ser utilizada durante o processo de execução do regressor. Como se trata de tipos de dados relativamente simples e conhecidos, os processos de Associação e Segmentação foi pouco utilizado.

2.2.1 Weka: Ferramenta de Mineração de Dados

A mineração de dados é amplamente utilizada nas mais diversas áreas quer seja por pesquisadores, professores e alunos. As ferramentas deste processo são utilizadas para pesquisas em visão computacional, computação gráfica, análise de imagens, dentre outras áreas e que normalmente, demandam muitos recursos computacionais e tempo de execução. Uma das utilizadas no KDD é a Weka, sendo que esta não é diferente das demais em relação aos recursos necessários. Com o objetivo de auxiliar os pesquisadores da UFMA na redução do tempo em suas pesquisas e permitir novas possibilidades de execução da Weka, esta foi escolhida como a ferramenta alvo para ser utilizada neste trabalho.

É uma ferramenta de código aberto multiplataforma desenvolvida pela Universidade Waikato, na Nova Zelândia, e utilizada mundialmente para aprendizado de máquina nas áreas de

visão computacional, interpretação de imagens e mineração de dados, conforme Alam e Pachauri (2017).

Proporciona suporte a todo o processo de KDD, representado na Figura 5, inclusive de mineração de dados experimental, inclui preparação de dados de entrada, avaliação de esquemas estatísticos de aprendizado e visualização dos dados de entrada e saída. Além dessas funcionalidades, conta com ferramentas de pré-processamento e uma interface de fácil manuseio para que usuários possam comparar resultados de diferentes métodos e identificar os mais apropriados para a solução, segundo Kalmegh (2015), Weka (2017), Hall et al. (2009).

Além de permitir adicionar novos algoritmos e extensões, também permite executá-los de forma remota e paralela, obtendo assim as vantagens no uso da Computação em Nuvem, sendo facilmente extensível, e também pode ser incorporada em soluções externas de forma transparente, segundo Hall et al. (2009). Esses recursos permitem uma ampla gama de possibilidades no uso da ferramenta, desde criar ou modificar classificadores, ou integrá-la ao código de sua aplicação, através de chamadas disponibilizadas pelo Weka, ou a possibilidade de executá-la no computador local ou até em uma infraestrutura em nuvem privada ou pública.

O Auto-Weka é um recurso amplamente utilizado, que executa diversos algoritmos otimizando os parâmetros e analisando os resultados gerados, devolvendo-os ao usuário e sugerindo quais os melhores algoritmos e parâmetros a utilizar, conforme Weka (2017), Hall et al. (2009). Esta ferramenta possui um alto custo computacional, necessitando de uso intenso de processamento e um longo tempo de execução, principalmente se estiver sendo usado com uma massa de dados complexa e grande. Nesse contexto, o Weka e o Auto-Weka se beneficiam da computação em nuvem, uma vez que podem ser disponibilizados recursos computacionais de forma mais eficaz e eficiente para executar os algoritmos em tempo e custo menores.

O Weka ainda possui a versão Servidor, que permite ser executada de forma distribuída. A versão padrão somente é executada no modo isolado, utilizando os núcleos do processador local e, dependendo dos parâmetros, utiliza somente um núcleo para todas as tarefas. No modelo distribuído, conforme demonstrado na Figura 6, é necessário ter pelo menos um nó que responderá como controlador e quantos nós forem necessários. Apesar do weka server também processar as tarefas, sua função principal é a de receber as requisições, controlar as execuções nos nós e devolver ao cliente o resultado. Quanto maior a quantidade de tarefas divididas maior o paralelismo e consequentemente mais rápida a execução.

As tarefas são criadas pela interface do usuário e são criadas a partir dos parâmetros de execução e da configuração do experimento. Nesse caso, são criadas tarefas distintas na utilização

de validação cruzada, execução de vários algoritmos, como exemplo o auto-weka, vários datasets, dentre outras possibilidades. Ou seja, dependendo do planejamento do experimento, é possível utilizar de forma mais eficiente a arquitetura do weka server. Esse recurso tanto pode ser utilizado através da execução diretamente pela ferramenta quanto pelo uso das suas bibliotecas em outras linguagens e aplicações.

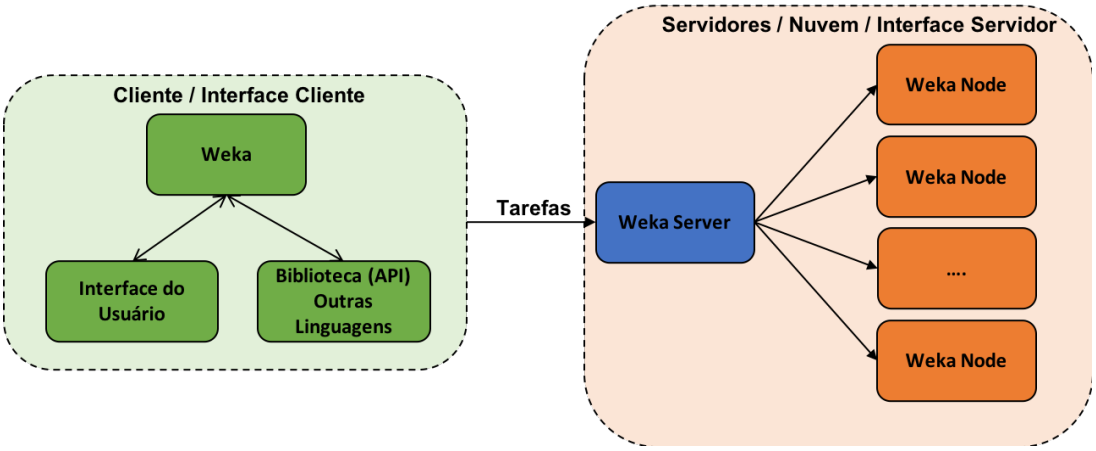


Figura 6: Modelo do Weka Server.

2.3 Máquina de Vetores de Suporte - MVS

De acordo com Lorena e Carvalho (2007), Sousa (2017), aprendizado de máquina usa um princípio de inferência denominado indução, onde se obtêm conclusões genéricas a partir de conjuntos específicos de exemplos. Esse processo pode ser dividido em dois tipos principais: aprendizado supervisionado (preditivo) e não-supervisionado (descritivo). A diferença entre os tipos de métodos é que os não-supervisionados não precisam de uma pré-categorização para os dados, ou seja, todas as informações serão obtidas a partir das amostras a serem analisadas sendo que não é necessário ter um atributo alvo. Estes geralmente utilizam uma medida de similaridade entre os atributos. As tarefas de agrupamento e associação são não-supervisionadas. Já nos métodos supervisionados, estes são providos com um conjunto de dados que possuem uma variável alvo pré-definida e os registros são categorizados em relação a ela, ou seja, exemplos já rotulados, classificados, seguido por uma etapa de treinamento. Quanto às tarefas de classificação, algumas são não-supervisionadas, enquanto as de regressão são do tipo supervisionado, de acordo com Avelar, Rocha e Cruz (2017), Rezende (2003).

A Máquina de Vetores de Suporte (MVS) é um modelo de aprendizado de máquina supervisionado, sendo utilizado em classificação, reconhecimento de padrões, análise e predição

em série histórica e análise de regressão. O MVS possui melhor desempenho se comparado a outros métodos de predição como regressão Linear (RL) e Redes Neurais (RN), porque é baseado no princípio da minimização do risco estrutural e não no princípio de minimização do risco empírico, conforme Lorena e Carvalho (2007), Negnevitsky (2011), Raschka e Mirjalili (2017).

O princípio básico do MVS é baseado na construção de um hiperplano como superfície de decisão, que mantenha a maior distância possível em relação aos elementos da amostra. Por hiperplano entende-se uma superfície de separação de duas regiões em espaços multidimensionais, segundo Sousa (2017), Negnevitsky (2011), Lorena e Carvalho (2007). Na Figura 7, tem-se a representação de três hiperplanos separando os círculos e os triângulos. As linhas separadoras dos planos, chamadas de fronteiras de decisão, são a representação das funções ou hipóteses utilizadas pelo classificador. Neste caso, para os mesmos elementos, têm-se 3 hipóteses diferentes.

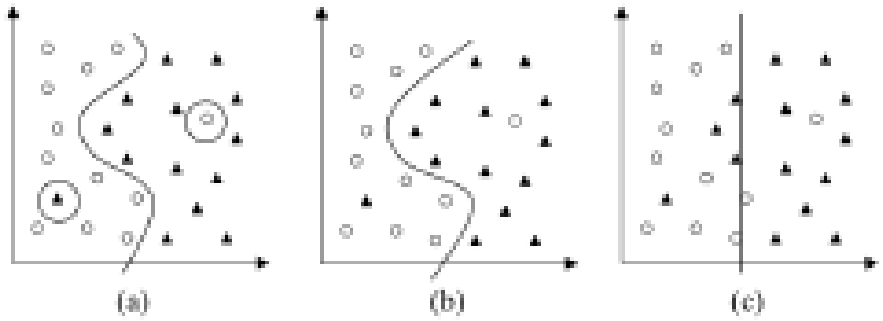


Figura 7: Exemplos de hiperplanos com três hipóteses diferentes. Fonte: (LORENA; CARVALHO, 2007)

Ainda segundo Sousa (2017), Lorena e Carvalho (2007), caso não seja possível separar completamente as classes, o MVS possibilita encontrar um hiperplano que minimize a ocorrência de erros através do uso do princípio da minimização do risco estrutural - SRM (*Structural Risk Minimization*), ou seja, encontrar a maior distância da margem de separação entre elas.

Para a construção do modelo, de acordo com Sousa (2017), Lorena e Carvalho (2007), Avelar, Rocha e Cruz (2017), não é necessário utilizar todos os registros do repositório, ou seja, toda a base de dados de conhecimento. Assim, utiliza-se uma amostra dividida em três conjuntos:

- **Conjunto de Treinamento (*Training Set*)**: Conjunto de registros usados no qual o modelo é desenvolvido.
- **Conjunto de Testes (*Test Set*)**: Conjunto de registros utilizado para testar o modelo.

- **Conjunto de Validação (*Validation Set*)**: Conjunto de Registros usado para validar o modelo construído.

Essa divisão, além de permitir trabalhar com uma quantidade menor de registros, pois utiliza somente uma amostra dos dados, permite também que o modelo não fique dependente dos dados específicos e que, ao ser submetido a dados diferentes, tenha resultados insatisfatórios. Este efeito é chamado de *Bias*. Ou seja, à medida que se aumenta a precisão do modelo para um conjunto de dados específico, perde-se a precisão para outros tipos de dados, como afirma Avelar, Rocha e Cruz (2017).

2.3.1 SVR - *Support Vector Regression*

De acordo com Lorena e Carvalho (2007), Sousa (2017), Rezende (2003), nem sempre os dados podem ser separados por um hiperplano de forma linear, como mostrado na Figura 8a, onde uma fronteira curva seria mais indicada. As MVS tratam os problemas não lineares transformando o seu plano original para um novo plano de maior dimensão, chamado de espaço de características (*feature space*), onde existe uma alta probabilidade de os dados serem linearmente separados, representado na Figura 8b.

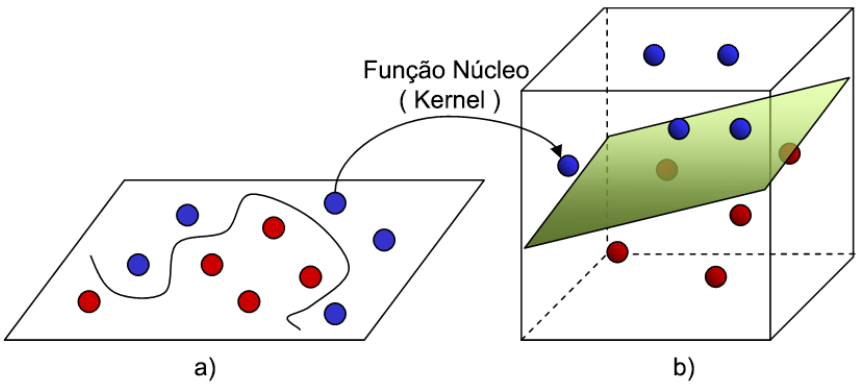


Figura 8: (a) Fronteira não linear no espaço de entrada; (b) Fronteira linear no espaço de características. Fonte: (QUORA, 2016)

O novo espaço de características deve permitir a separação das amostras usando uma função linear, que, nesse caso, será utilizado como a entrada de dados, Figura 8b, para todo o processo, ignorando o espaço original, Figura 8a. A construção utiliza uma função K de núcleo que pode realizar o mapeamento das amostras para um espaço de dimensão muito elevada sem aumentar a complexidade dos cálculos, pois é uma função que recebe dois pontos, para cada

amostra de treinamento, do espaço de entradas e computa o produto escalar desses dados no espaço de características.

Segundo Sousa (2017), Lorena e Carvalho (2007), Teixeira (2016), Almeida (2013), a Equação (1), onde $\omega \in \mathbb{R}^n$ são parâmetros. O ajuste é feito desde que o erro, ε, não seja maior que um valor definido $\varepsilon > 0$. Desta forma são encontrados pesos que satisfazem a Equação (2).

$$f(x) = \langle \omega, x \rangle + b \tag{1}$$

$$|y_i - f(x_i)| < \varepsilon \tag{2}$$

Onde todo `i = 1 ... n`. Para ter um efeito de regularização, a norma l_2 do vetor ω é minimizada. O problema é formulado como o problema de otimização representado na Equação (3):

$$\begin{aligned} minimizar \quad & \frac{1}{2}||\omega||^2 \\ sujeito\ a \quad & y_i - \langle \omega, x_i \rangle + b \leq \varepsilon \\ & \langle \omega, x_i \rangle + b - y_i \leq \varepsilon \end{aligned} \tag{3}$$

Sendo assim, define-se a função de regressão na Equação (4), como sendo:

$$\begin{aligned} Minimizar \quad & \frac{1}{2}||\omega||^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \\ sujeito\ a \quad & y_i - \langle \omega, x_i \rangle - b \leq \varepsilon + \xi_i \\ & \langle \omega, x_i \rangle + b - y_i \leq \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0, \forall_i = 1....l \end{aligned} \tag{4}$$

Onde ε é o valor máximo de desvio em relação aos valores alvos, e $y_i - \varepsilon, y_i + \varepsilon$ são as margens de erro máximas da função. A função de perda ε-insensível introduz variáveis de folga não negativas ξ_i, ξ_i^*, com a finalidade de penalizar dados fora da margem, Equação (2), e representada na Figura 9, de acordo com Almeida (2013).

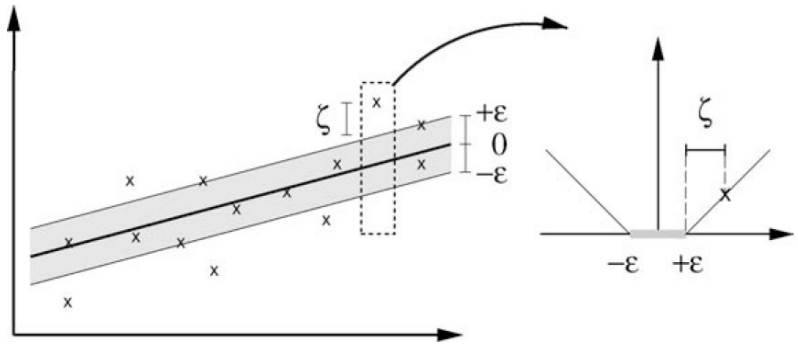


Figura 9: Função de perda ε-insensível. Fonte: (ALMEIDA, 2013)

A formulação dual deste problema pode ser escrito como a Equação (5):

$$\begin{aligned} Maximizar \quad & -\frac{1}{2}\sum_{i,j=1}^{m}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)\langle x_i,x_j\rangle \\ & -\varepsilon\sum_{i=1}^{l}(\alpha_i-\alpha_i^*)+\sum_{i=1}^{l}y_i(\alpha_i-\alpha_i^*) \\ sujeito\ a \quad & \sum_{i=1}^{l}y_i(\alpha_i-\alpha_i^*)=0 \\ & \alpha_i,\alpha_i\geq 0 \end{aligned} \tag{5}$$

É possível aplicar uma função núcleo (*kernel*) na formulação dual apresentada na Equação (5) que consiste em transformar um problema não-linear em um problema linear gerando um novo espaço de característica, como apresentado na Figura 8.

Ainda de acordo com Sousa (2017), Lorena e Carvalho (2007), alguns dos núcleos mais utilizados na prática são os Polinomiais, os Gaussianos ou RBF (*Radial-Basis Function*) e os Sigmoidais, listados na Tabela 1, sendo que possuem parâmetros que devem ser determinados pelo usuário, indicados também na tabela.

Tabela 1: Funções núcleos mais comuns.

Tipo de Kernel	**Função K(x1, x2)**	**Parâmetros**
Polinomial	$(\gamma(x_i.y_i)+\kappa)^{\delta})$	γ, κ, δ
Gaussiano	$exp(-\gamma\|x_i-y_i\|^2)$	γ
Sigmoidal	$tanh(\gamma(x_i.y_i)+\kappa))$	γ e κ

Afirmam Abreu e Neto (2017), Lorena e Carvalho (2007), Sousa (2017), Negnevitsky (2011), Raschka e Mirjalili (2017) que o SVR é amplamente utilizado para previsão de dados baseada em séries históricas, justamente pela sua abordagem relativamente diferente do MVS, pois sua metodologia analisa os dados e reconhece padrões mediante a inclusão de uma

função de perda e determina um hiperplano ótimo em que as amostras de treinamento estejam mais próximas possíveis, de acordo com Abreu e Neto (2017), Lorena e Carvalho (2007), independentemente de qual lado da superfície os pontos se encontram, e sim que a distância para a superfície seja a mínima possível, e que desvios sejam permitidos desde que estejam dentro de um limite estabelecido.

2.4 Algoritmos Genéticos

Os Algoritmos Genéticos são técnicas da computação evolutiva geralmente utilizadas para encontrar uma boa, às vezes ótima, solução para problemas que possuem bilhões de soluções possíveis. Utilizam processos semelhantes aos processos biológicos da evolução, gerando de forma contínua possíveis soluções e avaliando o quanto satisfaz o problema, selecionando sempre o melhor candidato, de acordo com Sheppard (2016).

Complementando a definição acima, Carvalho (2006) define AG como uma técnica de busca e otimização por meio de modelos computacionais baseados na teoria da evolução das espécies proposta por Charles Darwin e nos princípios básicos da herança genética, descritos por Gregor Mendel.

2.4.1 Componentes de um Algoritmo Genético

Segundo Sheppard (2016), Carvalho (2006), Pinho et al. (2013), os algoritmos genéticos possuem uma terminologia própria, onde indivíduos são as possíveis soluções do problema e são representados por cromossomos, que, no caso, um indivíduo pode possuir um ou mais cromossomos. Normalmente, são implementados por vetores, onde cada elemento do vetor é chamado de gene. Os valores que os genes podem assumir são denominados de alelos, possuindo um local fixo no cromossomo chamado de lócus.

Os Algoritmos Genéticos possuem alguns componentes básicos, ainda de acordo com Sheppard (2016), Carvalho (2006), Pinho et al. (2013), que são:

- **Representação Genética**: É a abstração do problema estudado e dos dados para a representação no AG. Ou seja, é a definição dos cromossomos, genes e alelos.
- **População Inicial**: A geração da população inicial é feita de duas formas. A mais comum é gerando indivíduos de forma aleatória e a outra é através de heurísticas relacionadas às

características do problema. Ao adotar a segunda opção, pode-se chegar a uma solução mais rápida, mas corre-se o risco de ocorrer uma convergência prematura que não é conveniente para o AG.

- **Função de Avaliação**: Função de Avaliação *(Fitness)*, que define se as soluções são válidas e o quanto uma solução é mais viável ou eficaz que outra solução.
- **Método de Seleção de Reprodução**: São os métodos utilizados para selecionar os indivíduos que serão submetidos às operações genéticas.
  - Roleta Simples: Neste método, cada indivíduo possui seu valor de aptidão, função de avaliação, representado proporcionalmente em relação ao valor total das aptidões da população. Sendo assim, os que possuírem maior participação, consequentemente possuem maiores chances de serem selecionados.
  - Seleção por Torneio: Neste método, é levado em consideração o valor da função de avaliação entre os indivíduos que participam do sorteio e é selecionado o que possuir melhor valor. A definição dos indivíduos que fazem parte do torneio é de forma aleatória e limitada a uma quantidade máxima de participantes, que é um parâmetro do AG.
  - Seleção Elitista: Apesar de alguns estudiosos não considerarem um método, pode ser utilizado com outros métodos existentes, pois esta técnica consiste em reintroduzir o melhor indivíduo da geração anterior na próxima, ou a utilização dos melhores indivíduos das gerações anteriores para a seleção da melhor solução. Pois o melhor resultado pode não estar presente na última geração do processo.
- **Operadores Genéticos**: Os operadores genéticos são as operações sofridas pelos indivíduos. Existem duas mais comuns, que são:
  - Cruzamento: O cruzamento, ou *crossover*, é o mais importante dos operadores e consiste na combinação de dois indivíduos para formar novos indivíduos. Nessa operação é escolhida aleatoriamente qual posição os indivíduos são quebrados e suas partes são recombinadas formando novos indivíduos. Com este processo espera-se que ao longo das gerações os melhores genes apareçam com maior frequência.
  - Mutação: A operação de mutação introduz mudanças aleatórias nos genes, gerando assim novos indivíduos, mas não muito diferentes dos originais. A principal função é a de reintrodução da variedade genética à população. A forma mais simples de

realizar a mutação é a partir da escolha aleatória de qual gene será realizada a mutação.

- **Critério de Parada**: Determina quais parâmetros devem ser observados para que o AG finalize sua execução e retorne a melhor solução encontrada. Pode ser a quantidade de gerações, quantidade de gerações sem sofrerem melhoras na função de avaliação ou outro critério definido a partir do problema.
- **Parâmetros do Algoritmo Genético**: Existem alguns parâmetros na execução do AG que influenciam na eficácia do mesmo, citados a seguir:
  - Tamanho da População: O número da população afeta diretamente o desempenho geral e a sua eficiência pois está diretamente relacionada às possíveis soluções disponíveis.
  - Numero de Gerações: O número de gerações varia de acordo com a complexidade do problema, pois o ideal seria o AG retornar a melhor solução, mas que caso haja uma quantidade insuficiente de gerações pode-se não chegar à solução recomendada.
  - Taxa de Cruzamento: Essa taxa determina se será feito o cruzamento entre dois indivíduos. Gera-se um número entre 0 e 1, e se esse valor for menor que a taxa, então o cruzamento será realizado.
  - Taxa de Mutação: Semelhante à taxa de cruzamento, esta determina se um indivíduo sofrerá mutação, e é aplicada a todos os indivíduos da população.

Ou seja, ao encontrar uma solução possível, o AG avalia se essa solução é mais viável que as demais. Além de utilizar o conceito de reprodução e herança, que nesse caso seleciona os indivíduos com melhor aptidão, há a evolução da população através de avaliação, seleção, recombinação e mutação sofrida pelos indivíduos. Estes são formados por cromossomos, que é na realidade uma estrutura de dados que representa uma possível solução. Após várias gerações, o indivíduo mais apto deverá ser encontrado, conforme Pacheco (1999), Sheppard (2016), Negnevitsky (2011), o que será consequentemente uma melhor solução.

2.4.2 Funcionamento do Algoritmo Genético

Segundo Negnevitsky (2011), Sheppard (2016), o AG inicialmente cria uma população aleatória e calcula para cada indivíduo a função de avaliação. Após este passo, os indivíduos são selecionados para reprodução. Os indivíduos selecionados são expostos aos operadores

genéticos, cruzamento e mutação, e gerados novos indivíduos. Ao final dessa geração, os mais aptos são selecionados para a próxima geração, repetindo-se assim todo o processo. Ao final de diversas gerações, é possível encontrar o indivíduo mais apto das diversas gerações, ou seja, o indivíduo mais apto é a melhor solução das encontradas. As novas gerações são criadas até que seja encontrado um critério de parada, podendo ser um valor de avaliação esperado ou simplesmente a quantidade de gerações criadas. No último caso, recomenda-se selecionar o indivíduo mais apto de todas as gerações criadas.

A Figura 10 representa o fluxo do AG citado anteriormente, e a Figura 11 representa o cromossomo adotado neste trabalho. O cromossomo é composto por duas partes distintas, uma que representa a configuração do servidor e outra que representa as configurações das estações escravas. Sendo assim, os valores possíveis para a parte dos servidores é somente 1 ou 2, ou seja, 1 ou 2 núcleos no servidor, e nos escravos podem ser entre 0 e 4, representando nenhuma máquina com determinada quantidade de núcleos até 4 máquinas com os núcleos especificados em N1, N2, N4 e N8. Consequentemente, o cromossomo representa a configuração do ambiente em nuvem que a aplicação Weka será executada, ou seja, uma solução para o problema.

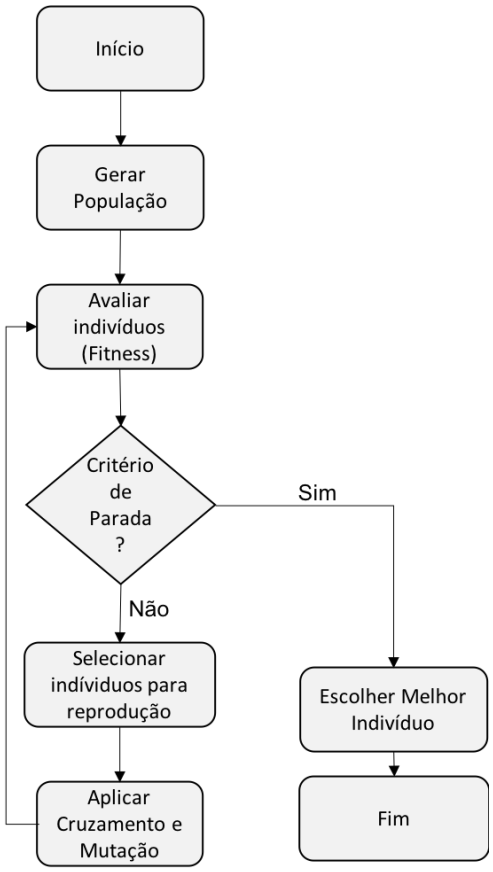


Figura 10: Fluxograma do Algoritmo Genético.

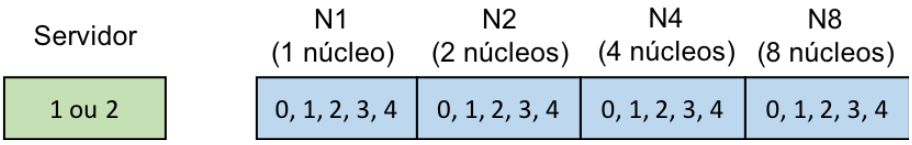


Figura 11: Cromossomo adotado.

Na Figura 12, tem-se um exemplo de cruzamento de cromossomos no ponto entre os genes N2 e N4 onde os genes à direita irão para o segundo cromossomo e serão substituídos pelos genes à esquerda do segundo, gerando assim novos indivíduos, filhos, baseados nos pais.

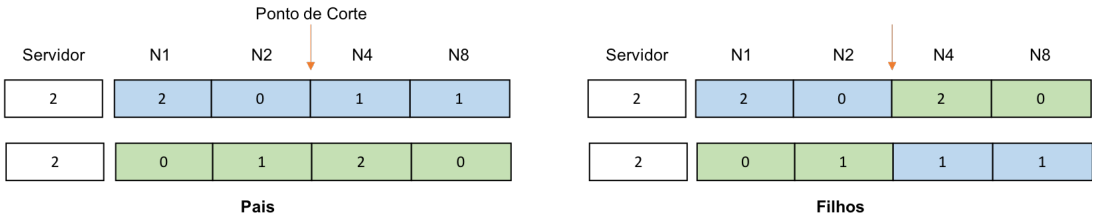


Figura 12: Exemplo de Cruzamento.

Já na Figura 13, observa-se um exemplo de mutação de cromossomos, onde a partir da taxa de mutação houve no gene servidor e no gene N4. No caso do gene servidor, houve a troca de 2 para 1 e no gene N4 de 1 para 2, uma vez que nesses exemplos os genes utilizam valores reais, gerando assim novos indivíduos diferentes do original.

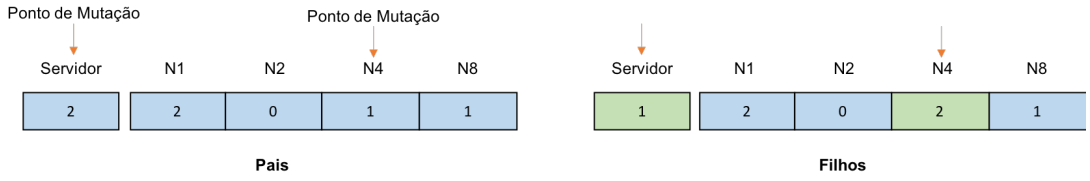


Figura 13: Exemplo de Mutação.

3 Arquitetura do Recomendador de Configuração de Nuvem

Este capítulo descreve os procedimentos utilizados na geração dos experimentos e captura dos dados para a criação do espaço de características utilizada no regressor na nuvem Eucalyptus com o Weka. Primeiramente, é apresentada a infraestrutura computacional utilizada nos experimentos. Em seguida, apresenta-se o experimento e a captura dos dados utilizados nos testes. Na sequência, são descritas as etapas necessárias para o treinamento, testes e validação do regressor proposto.

3.1 Diagrama Geral do Recomendador

Na Figura 14, tem-se o diagrama principal do Recomendador, onde o cliente submete o arquivo de *dataset* que será utilizado no Weka à interface do usuário, que pode ser através do *Web service* ou através de uma aplicação disponibilizada como SaaS. As informações de execução são enviadas ao Algoritmo Genético que faz a execução propriamente dita gerando as possíveis configurações.

O Regressor SVR possui a responsabilidade de predizer o tempo de execução levando em consideração o espaço de características de execuções, o arquivo de *Dataset* submetido e as configurações geradas pelo AG. Essa associação permite ao AG, através da sua função *fitness*, receber o tempo estimado pelo regressor e calcular o custo.

Ao término da execução do AG, o Recomendador acessa a IaaS do ambiente em nuvem e instancia as máquinas virtuais de acordo com a configuração sugerida com o serviço do Weka Server e submete o arquivo Dataset para execução no ambiente Weka em Nuvem. Ao término da execução, o cliente recebe o resultado e a configuração, tempo e custos reais da execução, que são monitoradas pelo ambiente, são alimentados ao espaço de características de execução Weka para que o regressor possa melhorar e ampliar as predições ao longo do tempo.

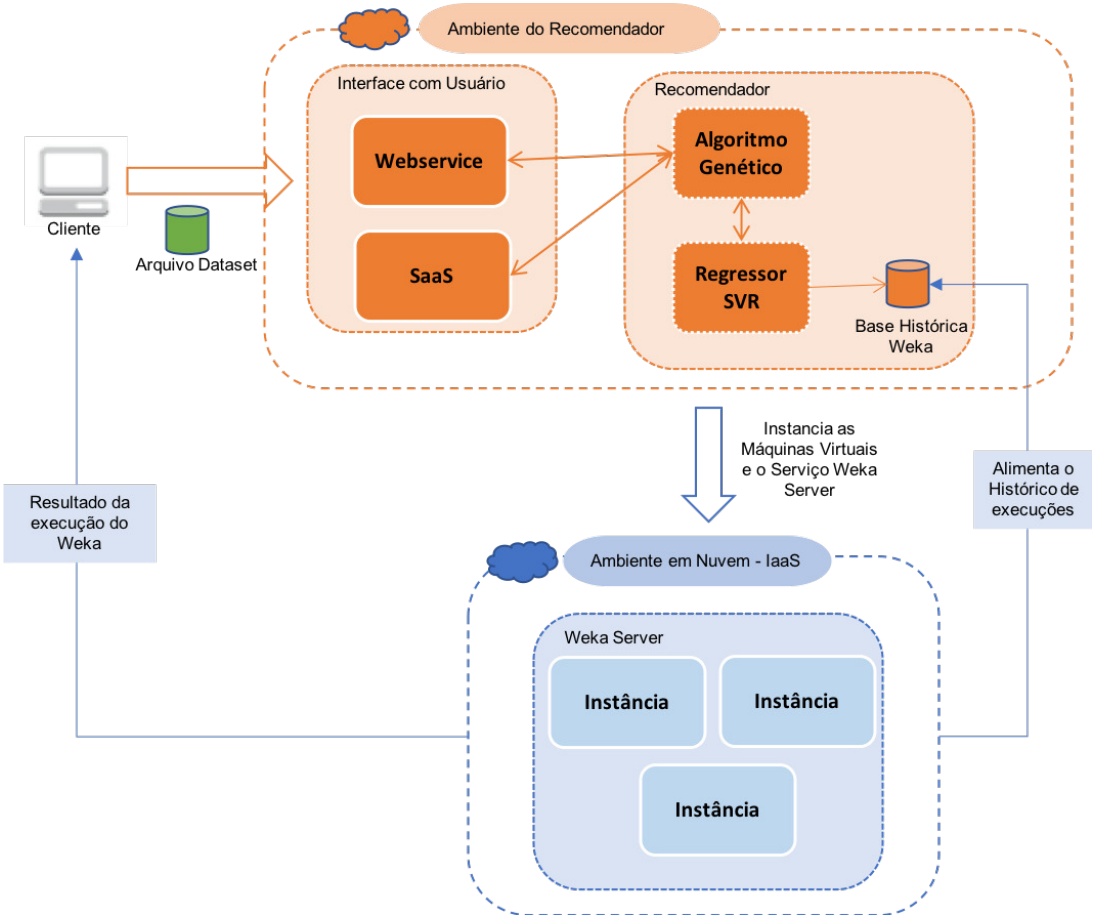


Figura 14: Diagrama de funcionamento do Recomendador.

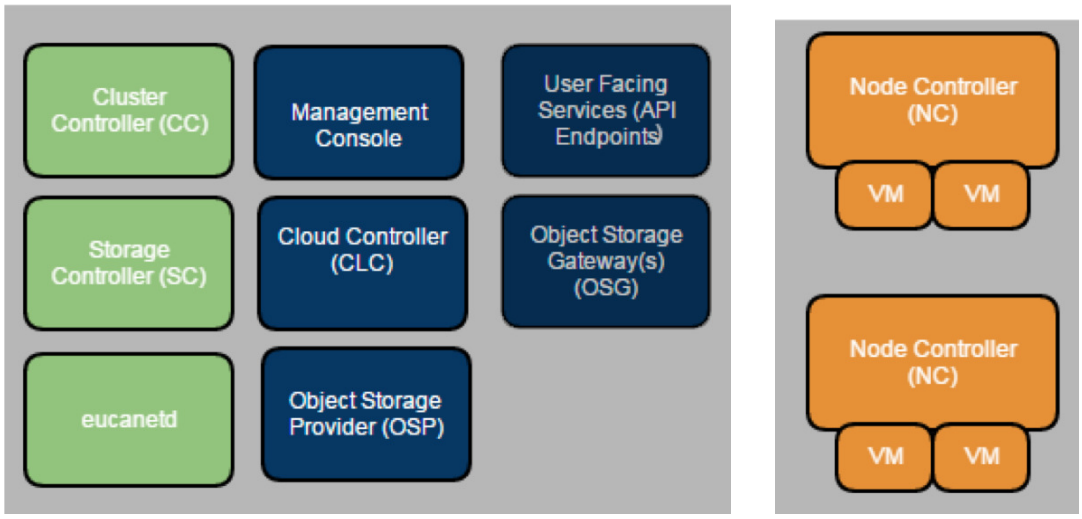
3.2 Prototipação do Recomendador

Para a implementação do protótipo e para geração de dados para o regressor descritas neste trabalho, foi utilizada a infraestrutura de nuvem privada Eucalyptus (HP, 2016b) com a ferramenta Weka (WEKA, 2017), com o módulo Weka Server habilitado (PENTAHO, 2017), que permite sua execução de forma distribuída utilizando assim os recursos em nuvem.

3.2.1 Nuvem Privada

Neste trabalho foi utilizado o ambiente de nuvem privada Eucalyptus, sendo implementada em um computador como controlador principal, contendo a maior parte dos módulos do mesmo, conforme mostra a Figura 15a, e dois computadores que serão responsáveis pelo gerenciamento das máquinas virtuais, chamados de controladores de nós, representados na Figura 15b. Caso haja necessidade de adicionar mais recursos à nuvem, podem ser adicionados

novos controladores de nós. Os módulos do Gerenciador principal também podem ser distribuídos e paralelizados de acordo com a necessidade, mas para este trabalho decidiu-se centralizá-los.



(a) *Gerenciador Principal* (b) *Controladores de Nós*

Figura 15: Distribuição dos módulos do Eucalyptus nos computadores do parque tecnológico. Fonte: (HP, 2016b)

Todas as máquinas possuem o sistema operacional CentOs 7 64bits, com Java 8 e Python 3, inclusive para as máquinas virtuais criadas para utilização do Weka.

Sendo assim, a infraestrutura criada irá tanto prover IaaS, na questão da própria infraestrutura de nuvem privada, com possibilidade de criação e gerenciamento de instâncias e recursos em nuvem, quanto SaaS, no momento em que se disponibiliza o Weka como serviço de rede. Este modelo de nuvem privada pode facilmente ser estendido ao modelo Híbrido, uma vez que, caso não haja mais recursos disponíveis, pode-se criar instâncias na Amazon AWS e integrá-las de forma transparente, bastando para isso ser configurada a comunicação entre os dois serviços. Essa possibilidade expande, praticamente de forma ilimitada, tanto os recursos disponíveis quanto os possíveis usos. Essa integração é nativa no Eucalyptus, sendo necessário somente ativar e configurar o recurso tando na Amazon quanto no Eucalyptus.

De acordo com os recursos existentes na nuvem privada, é possível alocar até 16 núcleos de processamento e 16gb de RAM em todas as instâncias. A quantidade de máquinas virtuais criadas são limitadas de acordo com os recursos disponíveis. Os sistemas operacionais utilizados não permitem escalonamento vertical sem reinicialização, tornando inviável esse tipo de implementação devido às características da aplicação Weka. Sendo assim, o escalonamento utilizado é somente o horizontal, ou seja, o escalonamento se dá através da adição de novas

instâncias. Todas as instâncias são definidas com o tamanho da memória e discos fixos, havendo somente variação da quantidade de processadores, pois a aplicação Weka é dependente mais do processamento que de memória e disco, uma vez que os arquivos de origem utilizados são definidos.

Como as características de execução do Weka Server são bem definidas, conforme se verá na Seção 3.2.3, os algoritmos reativos não são aplicáveis e os híbridos não têm uma eficácia melhor que os preditivos, uma vez que o modelo reativo não é possível ser utilizado nesse tipo de aplicação, principalmente pelo fato de não haver possibilidade de redistribuição das tarefas. Sendo assim, o melhor algoritmo de escalonamento a ser aplicado às aplicações Weka são os modelos preditivos, que, no caso deste trabalho, trata-se do *Time-series Analysis* (LORIDO-BOTRAN; MIGUEL-ALONSO; LOZANO, 2014), pois permite analisar e estimá-la antes de iniciar a execução e consequente alocação prévia dos recursos, permitindo assim uma execução mais viável. É importante ressaltar que os modelos híbridos podem ser utilizados, mas as funções reativas não teriam eficácia.

3.2.2 Máquinas Virtuais - Instâncias

Tendo em vista executar o Weka no ambiente em nuvem, foram criadas instâncias com o objetivo da execução paralela. Todas as instâncias possuíam a mesma configuração, com exceção da quantidade de núcleos de processamento. Nos experimentos foram utilizadas entre duas e cinco instâncias.

- **Núcleos**: 1, 2, 4 e 8 núcleos
- **Memória**: 2048 MB RAM
- **Sistema Operacional**: CentOs 7 64bits
- **WEKA**: 3.8.1 – Módulo Server instalado
- **JAVA**: Oracle JDK 8

De acordo com os dados submetidos ao Weka, a capacidade de memória das instâncias era suficiente para execução, não havendo a utilização de paginação de memória em nenhum momento.

Para referência do cálculo do custo foram utilizados os valores da Amazon AWS (AMAZON, 2016), conforme descrito na Tabela 2 e convertidos para custo em segundo. Para

manter uma referência internacional, foram adotados nesse trabalho os valores praticados nos servidores americanos, em relação à unidade de segundos, em dólares americanos.

Tabela 2: Custos de Instâncias na Amazon AWS.

Instância	**Qtd Núcleos**	**Estados Unidos**		**Brasil**	
		Custo Hora	**Custo Seg**	**Custo Hora**	**Custo Seg**
t2.small	1	0,023	0,000006389	0,041	0,000011389
t2.medium	2	0,047	0,000013056	0,081	0,000022500
t2.xlarge	4	0,188	0,000052222	0,324	0,000090000
m4.2xlarge	8	0,4	0,000111111	0,636	0,000176667

3.2.3 Weka

A execução Weka de forma distribuída exige que um dos nós realize a função de controlador, responsável por receber as requisições e distribuí-las e os restantes fazem a execução propriamente dita das tarefas. O controlador também executa as tarefas. Para tanto, um dos parâmetros do Weka Server é a quantidade de *slots* disponíveis para execução, que deve ser no máximo a quantidade de processadores disponíveis em cada máquina. Os *slots* são as unidades de processamento utilizadas pelo Weka e correspondem aos núcleos do processador disponíveis, ou seja, 1 *slot* é 1 núcleo do processador.

Em todas as execuções do Weka adotou-se a mesma configuração do algoritmo e classificação:

1. **Filtro**: NumericToNominal
    a) **Parâmetro do Filtro**: -R –last
2. **Método utilizado**: Classificação
    a) **Classificador**: MultilayerPerceptron
    b) **Parâmetros do Classificador**: -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a
    c) **Opções de Teste**: Percentage Split: 60
    d) **Folders**: 10

Ainda foram utilizados três arquivos de dados distintos, separados da mesma fonte de dados, composto de informações para detecção de massas em imagens mamográficas cedidas por Neto et al. (2016), mas com complexidades distintas na quantidade de atributos e na quantidade de instâncias, como forma de testar diferentes cargas de dados. Sendo assim, têm-se arquivos com

baixa complexidade, média complexidade e maior complexidade, separados de forma aleatória a partir da mesma origem.

Devido ao padrão do Weka dividir em dez *Folders* sua execução, gerando, assim, dez divisões de execução, adotou-se para este experimento o mesmo número como sendo o máximo de *slots* a serem usados, consequentemente a quantidade máxima de execuções em paralelo.

A infraestrutura do ambiente em nuvem privada implementada na plataforma Eucalyptus limitou o uso de 5 máquinas virtuais para os experimentos. Essa quantidade pode ser expandida transformando a estrutura em nuvem híbrida, uma vez que há a compatibilidade com a solução da Amazon AWS, pois utilizam a mesma API.

As execuções foram realizadas em catorze configurações diferentes, detalhadas na Tabela 3, utilizando entre 2 e 5 máquinas virtuais e o total de 4, 5, 7 e 10 núcleos disponíveis no ambiente, distribuídos nas VM's. A geração das configurações foi realizada levando em consideração a quantidade de núcleos disponíveis nos computadores usados pelos pesquisadores, neste caso 4 núcleos, e a quantidade padrão de 10 *folders* em que o Weka divide as tarefas para a execução do mesmo. Sendo que a quantidade de máquinas foi determinada pelo mínimo de 1 servidor e 1 escravo e a máxima de 5 máquinas que a infraestrutura utilizada suportava.

Tabela 3: Configurações utilizadas.

	Qtd de Núcleos				
	Servidor	**Escravo 1**	**Escravo 2**	**Escravo 3**	**Escravo 4**
Config 1	2	4	4		
Config 2	2	2	2	4	
Config 3	2	8			
Config 4	2	2	2	2	2
Config 5	1	2	2	2	
Config 6	1	2	4		
Config 7	1	1	1	2	2
Config 8	1	2	2		
Config 9	1	4			
Config 10	1	1	1	1	1
Config 11	1	1	1	2	
Config 12	1	1	1	1	
Config 13	1	1	2		
Config 14	2	2			

Em todas as execuções foram capturadas as informações de consumo de memória, processamento e tempo de execução:

- **Tempo de Execução**: As medições foram realizadas utilizando-se o próprio tempo de execução informado no log de execução do WEKA.

- **Memória**: Criou-se um programa que a cada 30 segundos executa o comando *vmstat* e salva o seu resultado.
- **Processamento**: Semelhante à memória, utilizou-se um programa que executa o comando *iostat* e salva o seu resultado também a cada 30 segundos.

A partir dessa execução percebeu-se que o uso de memória é diretamente relacionado à carga de dados e que não varia de uma execução para outra da mesma origem, variando somente pela complexidade e quantidade de dados analisados. O processamento possui uma particularidade decorrente da ferramente Weka, onde cada tarefa utiliza somente um núcleo de processamento e, quando em execução, o utiliza em sua totalidade, ou seja, 100% de utilização do núcleo por tarefa. O Weka não divide uma tarefa em vários processadores, sendo assim, neste trabalho analisou-se somente o tempo de execução como variável de estimativa para o regressor.

As características da aplicação Weka são bem definidas e são citadas a seguir:

- **Distribuição das tarefas**: A distribuição das tarefas é realizada no momento em que o Controlador as recebe, sendo utilizado o algoritmo *Round-Robin*, conforme Weka (2017), Hall et al. (2009), onde é distribuída uma tarefa para cada nó participante do grupo, não havendo redistribuição de tarefa.
- **Processamento**: Cada tarefa é executada por vez e em somente um núcleo, não havendo possibilidade de uma tarefa ser dividida em mais de um núcleo ou um núcleo de processamento receber duas ou mais tarefas simultâneas. Sendo assim, estes tipos de tarefas são mais bem explorados quando utilizados processadores de maior velocidade ou sistemas com maior quantidade de núcleos. Não existe impacto direto na escolha de quantidade de máquinas ou núcleos por máquina, e sim no total de núcleos disponíveis no ambiente. Essa decisão pode se relacionar diretamente aos custos dessas máquinas.
- **Falha de execução**: Caso haja uma falha no processamento de alguma tarefa ou em algum nó, as tarefas não são redistribuídas, sendo necessário reiniciar a execução por completo.
- **Execução**: Caso um nó tenha finalizado sua execução e existam tarefas pendentes em outros nós, o controlador não redistribui as mesmas. Devido a isso, o planejamento adequado para esse tipo de aplicação é fundamental para evitar, por exemplo, que todo o sistema fique apenas aguardando uma única tarefa.
- **Disco**: Basicamente, a aplicação Weka não utiliza armazenamento de dados. Sendo assim, um mínimo de espaço em disco é necessário para execução de suas tarefas.

- **Memória**: Como referido acima, a quantidade de memória utilizada é diretamente relacionada ao arquivo de dados de origem, não variando durante as execuções ou tarefas.

3.2.4 Regressor SVR

A partir destes dados gerados pelas execuções do Weka, têm-se as séries históricas, ou seja, valores que podem ser utilizados pelo regressor para realizar a predição do tempo de execução em uma determinada configuração para um arquivo específico.

O regressor foi implementado em Python usando a biblioteca Sklearn (SKLEARN, 2017), que permite usar e implementar aprendizado de máquina utilizando a linguagem de forma nativa. No regressor, usou-se o método GridSearchCV, que realiza pesquisa exaustiva sobre valores de parâmetros especificados para um estimador, implementando um método de ajuste (*fit*) e pontuação (*score*), que permite ir ajustando valores dos parâmetros e classificando o resultado da predição através de validação cruzada. Com este método, é possível testar vários parâmetros para o estimador de acordo com a base de dados submetida.

Inicialmente, o regressor foi submetido ao GridSearchCV com um conjunto de faixas de parâmetros, em itálico, e se obteve a configuração recomendada, em negrito, para cada parâmetro:

i. kernel: RBF

ii. epsilon (*0, 1, 0.05*): **1.0**

iii. gamma ($2^{(-15:3)}$): **0.25**

iv. cost ($2^{(-3:15)}$): **32768**

Após a execução do treinamento (70%), teste (30%) e validação cruzada, o regressor obteve uma acurácia de 94,35% na predição, confirmando a viabilidade do modelo.

As etapas seguintes consistiram em melhorar a acurácia do regressor, aumentando os intervalos dos parâmetros, chegando a uma combinação de 20 milhões de possibilidades na última configuração. Esses valores foram executados em um ambiente em nuvem, com processamento paralelo com 16 processadores, obtendo-se os seguintes resultados de configuração, chegando a 95,09% de acurácia.

i. kernel: RBF

ii. epsilon (*0, 1, 0.005*): **1.0**

iii. gamma ($2^{(-50:50)}$): **0.03125**

iv. cost ($2^{(-50:50)}$): **8388608**

Na tabela 4 têm-se as cinco configurações executadas do GridSearchCV com seus parâmetros obtidos e os resultados de acurácia calculados. Observa-se que a última execução, SVR 5, obteve melhor acurácia como consequência do maior intervalo de combinações possíveis e maior tempo no cálculo dos parâmetros.

Tabela 4: Resultados das configurações do SVR.

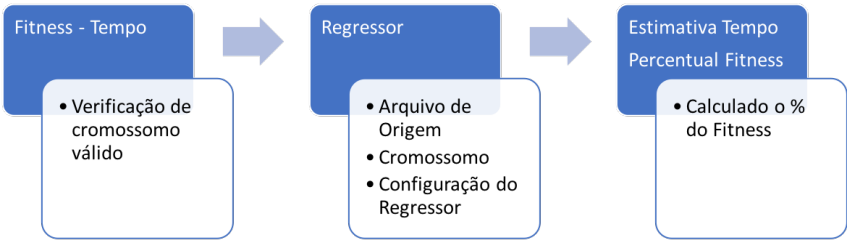
Resultados Parametrização do SVR (GridSearchCV)					
Configuração	**epsilon**	**gamma**	**cost**	**Variação Máxima %**	**Acurácia %**
SVR 1	1,0	0,25	32768,0	5,65	94,35
SVR 2	0	0,125	16384,0	6,06	93,94
SVR 3	1,0	0,5	8192,0	5,81	94,19
SVR 4	1,0	1,0	32768,0	5,84	94,16
SVR 5	1,0	0,03125	8388608,0	4,91	95,09

3.2.5 Algoritmo Genético

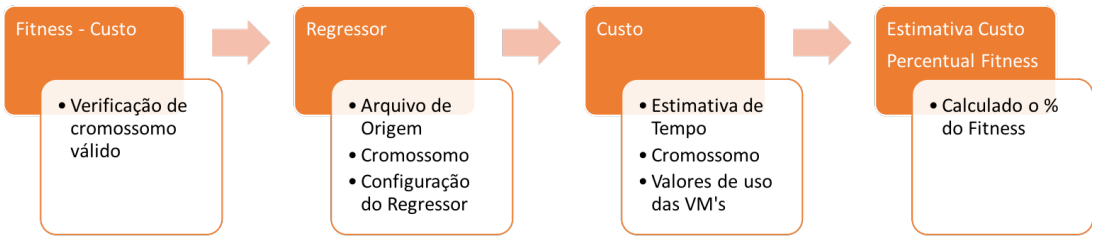
Após treinar o regressor, realiza-se a construção do algoritmo genético, cujo objetivo é gerar as possíveis configurações da nuvem, para então encontrar as melhores soluções (SHEPPARD, 2016). Para se definir qual a melhor configuração, adotaram-se três critérios:

1. Melhor Tempo
2. Melhor Custo
3. Melhor relação Custo / Tempo

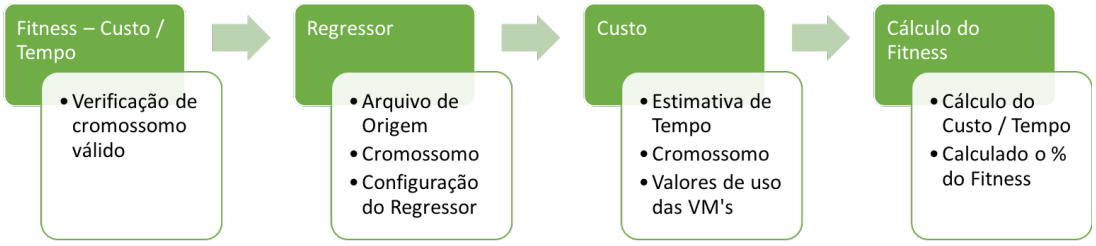
No AG, os critérios são definidos na Função de Avaliação *(Fitness)*, que define se as soluções são válidas e o quanto uma solução é mais viável ou eficaz que outra solução, de acordo com Negnevitsky (2011). Para tanto, foram criadas as três funções, sendo parametrizada qual será usada na chamada do Algoritmo, permitindo assim ao usuário definir limites, como, por exemplo qual a melhor configuração para que o execute mais rápido, com menor tempo até um limite de custo ou qual o menor custo com determinado limite de tempo de execução, conforme representado na Figura 16.



(a) *Fitness Tempo*



(b) *Fitness Custo*



(c) *Fitness Custo / Tempo*

Figura 16: Fluxograma das Funções *Fitness*.

Observa-se que, sempre que a função *Fitness* é chamada, o regressor também é chamado para fazer a estimativa do tempo, utilizando como parâmetro o arquivo de origem dos dados Weka e o cromossomo que possui a configuração da nuvem.

O cromossomo adotado possui dois tipos de genes, uma vez que para a execução do Weka em ambiente em nuvem é necessário um Servidor, que além de gerenciar as tarefas também processará a requisição, além dos Escravos, representados por N1, N2, N4 e N8, que

irão somente processar as requisições. Sendo assim, o cromossomo é dividido em duas partes, conforme visualizado na Figura 17, que representa alguns indivíduos da população, sendo que a primeira linha representa a identificação do gene:

1. Servidor
    a. Quantidade de núcleos: 1 ou 2
2. Escravos
    a. Existem 4 genes (N1, N2, N4 e N8) representando respectivamente 1, 2, 4 e 8 núcleos.
    b. Quantidade de núcleos: 0, 1, 2, 3 e 4.
    c. Somatório deve ser entre 1 e 4 – São no máximo 4 máquinas a serem usadas e no mínimo 1 máquina.
    d. Não há restrição à quantidade de núcleos a ser utilizada.

Indiv.	Servidor	Escravos			
		N1	N2	N4	N8
1	2	0	4	0	0
2	2	0	2	1	0
3	1	0	0	0	1
4	2	0	0	0	1
5	2	1	1	0	0

Figura 17: Exemplos de indivíduos da população.

O custo de uso das máquinas é calculado levando em consideração o tempo e as máquinas usadas:

Custo = Tempo * (N1 * V1 + N2 * V2 + N4 * V4 + N8 * V8)

onde V1, V2, V4 e V8 são os custos por segundo de uso para máquinas e N1, N2, N4 e N8 a quantidade de máquinas usadas com 1, 2, 4 e 8 núcleos respectivamente, conforme representado na Figura 18.

Sendo assim, o Algoritmo Genético tem 3 parâmetros obrigatórios na sua execução:

Indiv.	Servidor	Escravos				Tempo Estimado (s)	Custo Estimado (US$)
		N1	N2	N4	N8		
1	2	0	4	0	0	1122	0,0879
2	2	0	2	1	0	1468	0,1533
3	1	0	0	0	1	1472	0,1631
4	2	0	0	0	1	1191	0,1555
5	2	1	1	0	0	1299	0,0592

Figura 18: Exemplos de estimativas de tempo e custo de indivíduos da população.

1. Instâncias: Quantidade de instâncias do arquivo de entrada (linhas) que será utilizada no Weka
2. Atributos: Quantidade de Atributos do arquivo de entrada (colunas) que será utilizada no Weka
3. Modo Fitness: T, C ou B (Tempo, Custo ou Custo Benefício)

As mutações e os cruzamentos podem ocorrer nas duas partes do cromossomo, na parte do servidor e na parte dos escravos exclusivamente. Ou seja, somente podem ocorrer entre partes de servidores ou entre partes de escravos, não sendo permitidas essas operações entre as partes de servidores e escravos. Como forma de validar e otimizar o AG, adotou-se somente a função Fitness de Tempo, uma vez que as outras funções são diretamente dependentes desta.

Foram realizadas execuções do AG com taxas de cruzamento seguindo as recomendações de Rezende (2003), que sugere entre 0,6 e 0,99, sendo que o objetivo do algoritmo é de explorar a maior quantidade de possíveis soluções, adotou-se a taxa de 0,8.

Já para as taxas de mutações e tamanho da população, foram realizadas execuções do AG com taxas de 0,1%, 1% e 10% e com tamanhos de variando de 25 a 150 com incremento de 25 indivíduos. Para cada combinação dos parâmetros acima, foram realizadas 200 execuções e obtidos os seus valores de resultado de configuração e tempo para a melhor solução encontrada, para o *Fitness* Tempo.

Na Figura 19, têm-se o diagrama principal do processo, onde é submetido ao AG tanto o arquivo que será usado no Weka, representado em X(i), quanto N(i) que são os parâmetros de configuração do AG, como o tipo de Fitness, tamanho da população e taxa de mutação. Os indivíduos da população inicial são gerados aleatoriamente e os cromossomos devem ser

validados, atendendo aos critérios definidos na Seção 3.2.5, e caso seja gerado um inválido, é gerado um novo. Somente os válidos são submetidos ao regressor, que recebe como parâmetro o cromossomo c(i) e os dados do arquivo Weka X(i). O resultado do regressor SVR é o Tempo estimado t(i) que retorna ao AG para o cálculo do Custo C(i) juntamente com a configuração c(i). Todos os cromossomos gerados, que sofrerão cruzamento de um ponto e/ou mutação, são submetidos novamente ao regressor para a estimativa de tempo, chamado na função *Fitness* do AG.

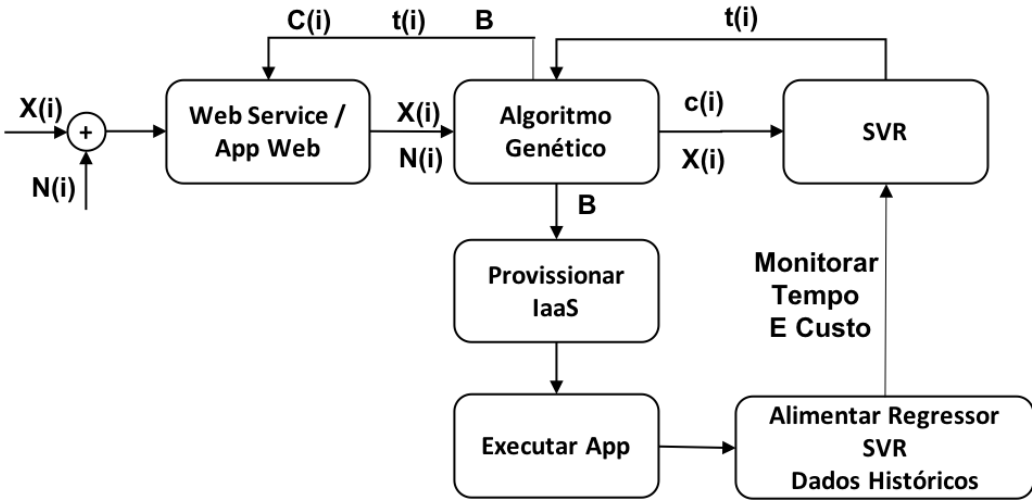


Figura 19: Diagrama do Processo Proposto.

Para a seleção dos cromossomos, utilizou-se o critério de Roleta Simples associado à Seleção Elitista, sendo assim, após as gerações criadas são definidos os melhores indivíduos de cada geração e o AG, baseado na função *Fitness*, seleciona o melhor indivíduo de todas, retornando como *B* a configuração recomendada, o tempo estimado e o custo estimado para o *Web Service*/Aplicação Web e para o IaaS.

Como forma de explorar o máximo possível de soluções, justamente por estar usando também a seleção elitista, adotou-se como critério de parada a centésima geração criada, com o objetivo de analisar e identificar o comportamento dos indivíduos e quando ocorria a convergência das soluções, sendo que o mesmo foi configurado para ser executado cinco vezes nos experimentos para cada validação.

Com a configuração definida, o serviço de nuvem provisiona as máquinas e serviços necessários e a aplicação Weka é executada. Ao término da execução da aplicação, as máquinas instanciadas podem ser desalocadas, liberando os recursos, gerando assim uma redução dos

custos, na medida em que estes serão somente provisionados e cobrados exclusivamente para a execução da aplicação.

No processo principal, a execução da aplicação Weka é monitorada e comparada com os dados previstos, gerando assim estatísticas, além do que os dados das execuções são utilizados para alimentar o regressor SVR, de modo que a predição tenha sua acurácia aumentada ao longo do tempo.

4 Validação da Arquitetura

Este capítulo apresenta os resultados obtidos pelo método proposto para um recomendador de alocação em Computação em Nuvem usando Algoritmos Genéticos e SVR. Visando alcançar o objetivo proposto neste estudo, nas próximas seções são apresentados e discutidos diversos experimentos realizados.

4.1 Cenário dos Experimentos

Conforme apresentado na Seção 3.2.3, os dados do espaço de características foram gerados baseados em três modelos de arquivos de dados e catorze configurações diferentes do ambiente em nuvem, sendo que todas foram executadas em uma nuvem privada utilizando a solução do Eucalyptus. Esses dados capturados foram utilizados para o treinamento, teste e validação do regressor, Seção 3.2.4, executados também no ambiente em nuvem.

A solução do Recomendado representado na Figura 14, Seção 3.1, foi implementada, quase que na totalidade, utilizando a linguagem Python (PYTHON, 2017), que se caracteriza pela sua simplicidade, por ser expansível e pela disponibilidade de recursos e independência de plataforma. O único módulo desenvolvido em PHP e HTML foi a interface da aplicação SaaS, mas que executa os módulos desenvolvidos em Python. Sendo assim, todo o modelo pode trabalhar de forma escalável, caso seja necessário.

Todas as instâncias criadas possuem os mesmos software e versões de bibliotecas, além do que, as suas configurações e custos foram padronizadas com relação às da Amazon AWS, conforme descrito na Seção 3.2.2.

4.2 Resultados Obtidos

São apresentados a seguir os resultados dos treinamentos, experimentos, simulações e execuções sobre o Regressor e sobre o Recomendador, usando o Algoritmo Genético e o SVR.

4.2.1 Regressor

A Tabela 5 apresenta os resultados dos testes, definidos na Seção 3.2.4, realizados com o regressor SVR na predição do tempo em segundos, mostrando sua viabilidade com erro máximo

de 5,65%, chegando à acurácia de 94,35%. A coluna **Tempo Real(s)** contém os valores obtidos na execução da aplicação Weka em nuvem, Seção 2.2.1, através da captura do tempo de execução. A coluna **Tempo Estimado(s)** contém os tempos calculados pelo regressor, enquanto as colunas **Diferença(s)** e **var %** são o erro em segundos e em percentual, que são a subtração do tempo calculado pelo tempo real da execução.

Tabela 5: Resultados iniciais do SVR.

Método 1					
#	**Tempo Real(s)**	**Tempo Estimado(s)**	**Diferença(s)**	**%**	**var %**
1	455	480,6976	25,6976	105,65%	5,65%
2	4025	4007,1128	17,8872	99,56%	-0,44%
3	3175	3119,9928	55,0072	98,27%	-1,73%
4	870	890,2641	20,2641	102,33%	2,33%
5	2974	3025,8053	51,8053	101,74%	1,74%
6	2143	2143,6356	0,6356	100,03%	0,03%
7	3677	3680,236	3,236	100,09%	0,09%
8	2527	2521,8074	5,1926	99,79%	-0,21%
9	2042	2143,6356	101,6356	104,98%	4,98%
10	854	862,5274	8,5274	101,00%	1,00%
11	459	480,6976	21,6976	104,73%	4,73%
12	448	449,7849	1,7849	100,40%	0,40%
13	1197	1190,1406	6,8594	99,43%	-0,57%
14	2242	2201,242	40,758	98,18%	-1,82%
15	4065	4041,3855	23,6145	99,42%	-0,58%
16	3337	3308,8268	28,1732	99,16%	-0,84%
17	2231	2249,8334	18,8334	100,84%	0,84%
18	1584	1575,955	8,045	99,49%	-0,51%
19	1466	1526,5028	60,5028	104,13%	4,13%
20	881	846,9477	34,0523	96,13%	-3,87%

Ainda foram realizados testes de melhoria na acurácia e validação do regressor, utilizando no SVR os parâmetros otimizados, representados na Tabela 4, Seção 3.2.4, onde a última configuração, SVR 5, obteve uma melhor acurácia, devido à maior amplitude de intervalos e consequentemente maior quantidade de combinações disponíveis para os parâmetros do regressor. Apesar de todas as configurações se mostrarem viáveis, adotou-se a SVR 5.

4.2.2 Algoritmo Genético

Os desafios com relação ao Algoritmo Genético são de encontrar os parâmetros viáveis e validar a sua eficácia na resolução do problema. No Gráfico 1, observa-se a evolução das taxas de mutação, pois foi mantida fixa a taxa de 0,8 para cruzamento, em relação ao tempo estimado, ao melhor tempo obtido e ao tempo médio calculado, representando no eixo Y, enquanto que no eixo X têm-se a quantidade de gerações. Naturalmente, na execução do AG os valores do *Fitness*

variam durante sua execução, mas tendem a estabilizar-se com relação ao resultado ótimo ao longo do tempo. Esse comportamento também foi observado com uma menor amplitude para a taxa de 1% ao longo das execuções, ou seja, foi a taxa que teve menor amplitude de variação, tornando-se assim o parâmetro de escolha para a taxa de mutação.

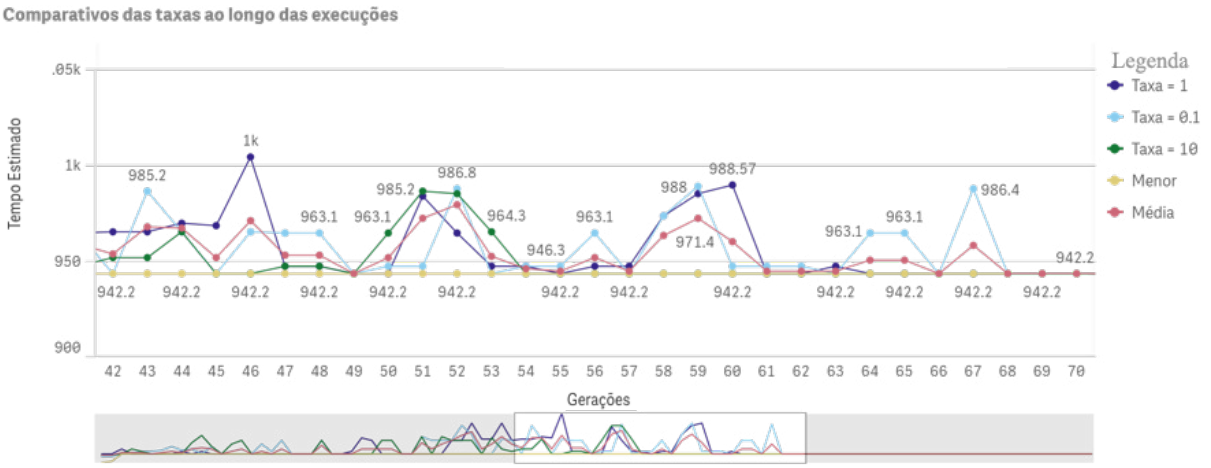


Gráfico 1: Taxas de mutação do AG.

Em relação ao tamanho da população, no Gráfico 2 analisam-se as execuções em relação aos tamanhos das populações. Pode-se observar que as populações de 125 e 150 indivíduos estão mais próximas do menor valor e em geral abaixo da média, além de que ambas convergem antes das demais. Sendo assim, as populações de 125 e 150 indivíduos são as melhores candidatas.

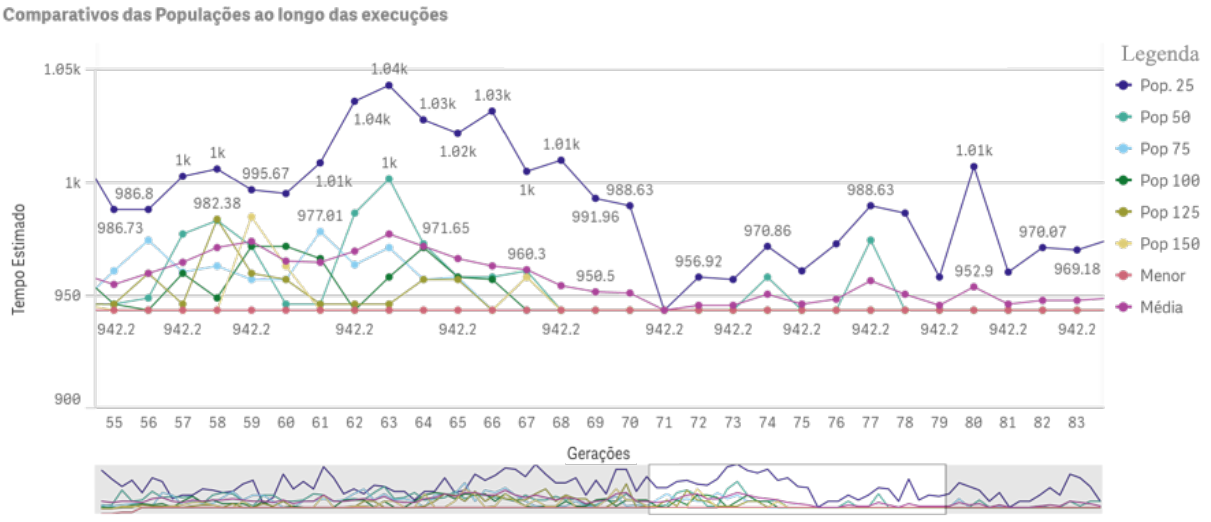


Gráfico 2: Tamanho da População do AG.

A população de 125 indivíduos se torna mais eficaz por produzir valores equivalentes à de 150, com a vantagem de ter um custo computacional menor. Desta forma, foi adotada a população de 125 indivíduos e 100 gerações.

4.2.3 Recomendador de Configurações

Devidamente configurado e validado, o Recomendador (AG + SVR) foi submetido a testes e os resultados obtidos sugerem uma configuração de nuvem, definindo a quantidade de núcleos para o servidor e a quantidade de máquinas com quantos núcleos cada uma, para que pudesse ser executada a aplicação o mais rápido possível e com o menor custo.

Foi realizado teste comparativo da configuração recomendada com o tempo previsto de execução e a execução da aplicação em ambiente de produção, em 10 recomendações aleatórias. Foram instanciadas as máquinas virtuais na infraestrutura de nuvem privada de acordo com as recomendações sugeridas pelo modelo e executada a aplicação Weka nas mesmas. Os resultados obtidos são apresentados na Tabela 6 e representados no Gráfico 3, onde se obtiveram um erro máximo de 5,29% e erro médio de 3,07%, demostrando a eficácia do processo.

Os percentuais de erro citados acima e os intervalos de diferença estão relacionados às variações existentes a qualquer ambiente, como tráfego de rede, link do cliente com o provedor, rota de acesso, demanda no provedor de IaaS, dentre outros fatores, tornando assim essa variação aceitável para o modelo. Os custos não estão representados, pois possuem a mesma variação percentual para essas execuções, já que o cálculo é diretamente proporcional à configuração e ao tempo de execução, possuindo assim a mesma variação.

Tabela 6: Resultados de validação dos tempos de execução.

Execução	Real(s)	Previsto(s)	Real / Previsto(%)	Erro(%)
1	1118	1108	99,11%	0,89%
2	1054	1016	96,39%	3,61%
3	1107	1062	95,93%	4,07%
4	1111	1067	96,04%	3,96%
5	435	412	94,71%	5,29%
6	400	412	103,00%	-3,00%
7	415	412	99,28%	0,72%
8	407	412	101,23%	-1,23%
9	1383	1330	96,17%	3,83%
10	1386	1314	94,81%	5,19%
Média	**881,6**	**854,5**	**96,93%**	**3,07%**

O Gráfico 4 apresenta a superfícies de execução dos tempos reais (a) e calculadas pelo Recomendador (b). Verifica-se que as duas são qualitativamente similares, sendo que o Recomendador possui a vantagem de permitir simulações de comportamentos de acordo com a necessidade, de forma rápida e confiável.

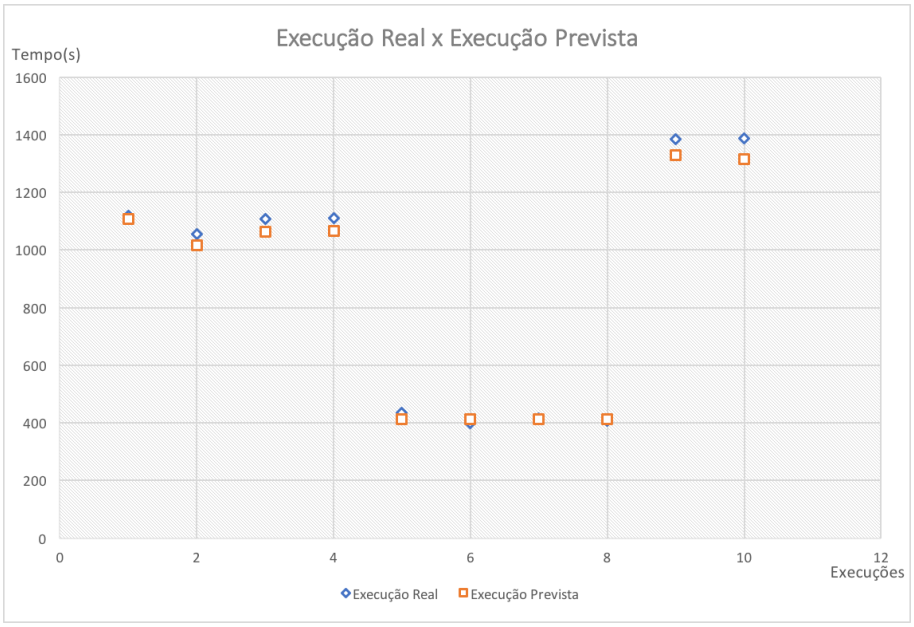


Gráfico 3: Comparativo de Execução Real x Previsto.

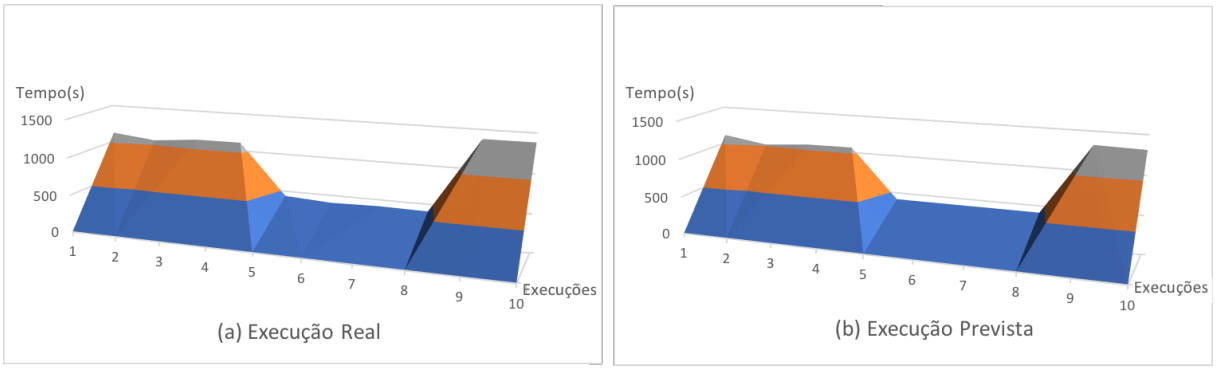


Gráfico 4: Superfície de resposta.

4.3 Redução de Tempo e Custo

Na Seção 3.2.3 foram realizadas as execuções iniciais do Weka, apresentada na Tabela 3 e levando em consideração a Config 1, servidor de 2 núcleos e 2 máquinas de 4 núcleos, totalizando 10 núcleos, para o arquivo de maior complexidade e a execução em ambiente real da configuração recomendada para esse mesmo arquivo de dados com melhor tempo de execução e custo, sugerido servidor com 1 núcleo, 3 máquinas de 2 núcleos e 1 máquina de 4 núcleos, obteve-se redução no tempo de 38,8% e 45,62% no custo. É importante mencionar que estes são valores médios obtidos a partir de 5 execuções para cada situação.

4.4 Recomendador de VM's

Como o Recomendador foi desenvolvido em Python, modular e de forma orientada a objetos, a sua integração a outras soluções e ferramentas se torna viável, podendo ser integrado diretamente na plataforma IaaS e/ou disponibilizado como SaaS, de acordo com Lorido-Botran, Miguel-Alonso e Lozano (2014) e Kupferman Jeff Silverman (2016).

4.4.1 Módulo de Administração da Nuvem

Um dos objetivos da Computação em Nuvem é a de provisionar recursos de forma automatizada. Sendo assim, o Recomendador pode facilmente se integrar à Nuvem, principalmente se o serviço de nuvem suportar as APIs do AWS utilizados pela Amazon e Eucalyptus, por exemplo, permitindo a utilização em nuvens privadas, públicas ou híbridas, conforme Cunha et al. (2017), Parikh (2013). Essa integração foi representada na Figura 19 no módulo Provisionar IaaS, detalhado na Seção 3.2.5, que permite uma automação na criação das máquinas virtuais, alocação dos recursos e consequente execução da aplicação Weka sem intervenção manual.

É importante salientar que o Recomendador é independente dessa integração com o Serviço em Nuvem, mas altamente sugerido como forma de automatizar o processo, tanto na alocação quanto na liberação dos recursos computacionais.

O tempo médio de resposta do recomendador, independente de qual plataforma está sendo utilizada, é de 00:01:51, ou seja, cerca de 111 segundos é o tempo de execução do recomendador para retornar a configuração e estimativa de tempo e custo.

4.4.2 Recomendador como SaaS - Aplicação Web

O Recomendador ainda pode ser disponibilizado como SaaS, nesse caso não havendo integração direta com o provedor de serviço, que poderia ser um consumidor, mas somente funcionaria como um recomendador de configuração e previsão de tempo e custo da execução. Mas seria de muita valia, pois com os seus resultados os pesquisadores poderiam se planejar na execução dos seus experimentos tanto em nível de tempo quanto de custos necessários para a execução.

Tanto com relação à subseção anterior quanto a esta, é de suma importância a alimentação da base de conhecimento do regressor para que o modelo possa realizar as predições com mais exatidão e ampliar sua própria base de conhecimento, que durante essa pesquisa está limitada aos serviços da Amazon AWS ou Eucalyptus, aplicações Weka e ao limite de até 5 máquinas virtuais. Com a inserção de novas informações de conhecimento, o SVR terá uma base mais ampla de uso, principalmente relacionadas aos *datasets*, e o AG novas configurações possíveis, não ficando limitado aos provedores de IaaS nem à quantidade de máquinas utilizadas. Esse modelo pode ser expandido para outros tipos de aplicações que utilizam a nuvem como ambiente de execução.

É importante frisar que este modelo pode ser usado para outros tipos de problema, para tanto sendo necessária a alimentação da base de conhecimento do SVR e a parametrização do AG.

Na Figura 20, tem-se a apresentação da tela de resultado do Recomendador na versão web disponibilizada como SaaS, exemplificando o seu uso. Neste caso, o usuário utiliza as configurações recomendadas para a criação manual em qualquer ambiente em nuvem, seja pública, seja privada. É importante ressaltar que toda a configuração e execução é de responsabilidade do usuário, o que não ocorre no modelo integrado ao IaaS.

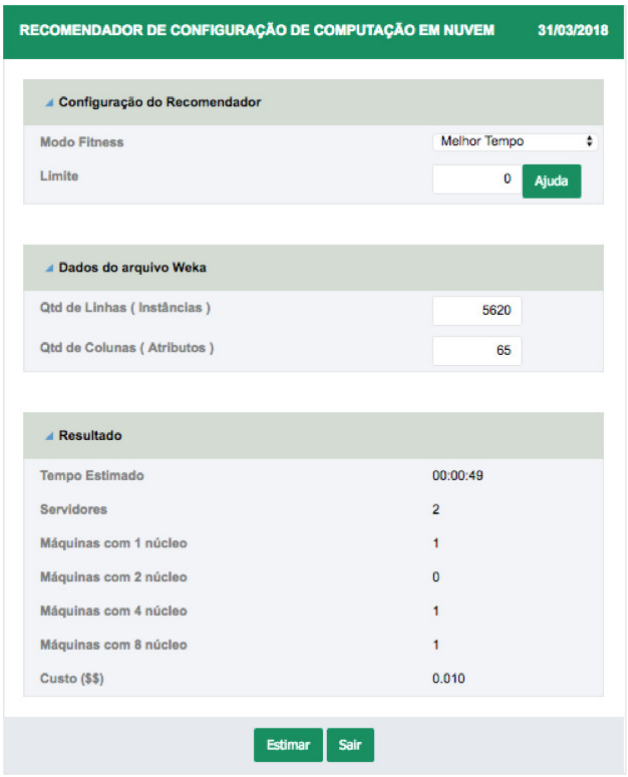


Figura 20: Recomendador no modelo SaaS - Aplicação Web.

4.4.3 Recomendador como SaaS - Web Service

Neste trabalho também foi implementada uma interface de comunicação com o Recomendador por meio de *Web Service REST*, permitindo assim uma ampla utilização pelas mais diversas aplicações que possam consumir o serviço de forma simples e eficiente, conforme mostrado na Figura 21. Possui três URI, que acessa o Recomendador com objetivo de Tempo, Custo, ou Custo Benefício, com três parâmetros obrigatórios, que são a quantidade de linhas e colunas do arquivo de origem de dados e o limite da função *Fitness*.

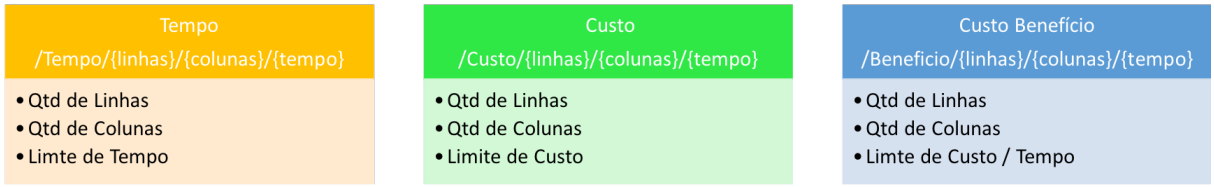


Figura 21: Recomendador no modelo SaaS - *Web Service REST*.

O retorno do *Web Service*, independentemente de qual URI está sendo acessada, é demonstrado abaixo, no formato JSON, que possui informações da execução tais como tipo de regressor, tipo de fitness usado, o resultado com a configuração recomendada em Servidor e Hosts e a estimativa em Tempo e Custo para a chamada http://recomendador.saas.api/Tempo/5620/65/0 :


```
{
    linhas: 5620,
    colunas: 65,
    tipo_regressor: N3,
    tipo_Fitness: Tempo,
    limite: 0,
    Populacao: 125,
    Mutacao: 1,
    Tempo: 49.81017331752264,
    Custo: 0.010722984681276321,
    Fitness: 0.010722984681276321,
    FitnessPercento: 0.038373161309446215,
    Servidor: 2,
    Host_1N: 1,
    Host_2N: 0,
    Host_4N: 1,
    Host_8N: 1
}
```

4.5 Discussão dos Resultados

Analisando os resultados obtidos com os experimentos, foi possível observar que o uso do SVR é eficaz na predição de valores baseados em séries históricas. Uma base de dados com maior variedade e com mais dados garante uma melhor acurácia, mas sendo que esta pode ser melhorada ao longo do tempo, sendo necessário realimentar o regressor com as informações das execuções realizadas, como mostram as Tabelas 5 e 6, tanto na realização da validação cruzada do regressor quanto na execução real da estimativa.

É possível que a acurácia do regressor possa ser melhorada, modificando a parametrização do mesmo, como demonstrado na Tabela 4, Seção 3.2.4. Para tanto, é necessária uma maior amplitude de parâmetros e maior tempo de execução do *GridSearchCV*, embora os valores encontrados com maior margem de erro de 6,06% e menor de 4,91% sejam aceitáveis para a predição do tempo.

A computação em nuvem tem muito a contribuir, principalmente no que diz respeito à melhor utilização dos recursos disponíveis, tanto em termos de nuvem pública quanto privada. Demonstrou-se o uso mais eficaz de computadores desktop agregados para prover mais recursos, como demonstrado na Seção 3.2.1. No caso da ferramenta Weka, Seção 3.2.3, pode-se utilizá-la de forma distribuída e paralelizada, situação até então pouco explorada, de forma a acelerar o tempo de execução pelos pesquisadores, consequentemente permitindo aos mesmos melhor aproveitamento dos recursos computacionais disponíveis para as pesquisas.

O uso do Algoritmo Genético, como apresentado neste trabalho, para obter as melhores configurações do ambiente em nuvem mostrou-se essencial para obtenção do melhor resultado. Os resultados presentes nas Seções 4.2.2 e seguintes comprovam o impacto que a escolha dos indivíduos que formarão a configuração do ambiente têm sobre o resultado final da execução.

Nos Gráficos 1 e 2, tem-se o comportamento das taxas de mutação e do tamanho da população na execução do AG, com taxa de cruzamento fixa de 0.8. Percebeu-se que os melhores parâmetros são os que permitem o AG convergir ao ótimo global o mais cedo possível, que no caso para a taxa definiu-se 1% e para a população de 125 indivíduos, tendo assim uma melhor relação entre custo computacional e acurácia da resposta.

Ainda na Seção 4.4, têm-se as várias formas em que o Recomendador pode ser utilizado e integrado, no modelo SaaS, sendo o mais simples, conforme visto na Figura 20, que nesse caso o Recomendador é somente uma aplicação Web, onde o pesquisador pode submeter as informações e tem como resultado a configuração recomendada, o tempo e o custo estimado.

Os outros modelos, representados na Figura 19, são os que integram com a Infraestrutura de Nuvem, onde o Recomendador provisiona as máquinas virtuais e os serviços automaticamente, diminuindo assim a intervenção humana e o modelo que alimenta automaticamente a base de dados utilizada para o Regressor SVR, contribuindo para uma melhor predição, também de forma automatizada. Independentemente da integração total ou parcial do modelo, o acesso ainda pode ser disponibilizado como *Web Service*, através da arquitetura *REST (Representational State Transfer)*.

Os dados históricos de execução ficam disponíveis para todos os pesquisadores que irão utilizar o modelo, podendo ser utilizados, inclusive, em outras ferramentas. É Importante citar que no modelo de integração com o IaaS, Seção 4.4, a execução é continuamente monitorada e o tempo de execução e as informações dos arquivos de dados são utilizados para alimentar a base do regressor, melhorando ao longo do tempo a sua predição.

Na Seção 4.2.3, demonstrou-se que é possível unir as três tecnologias, Computação em Nuvem, Regressão com SVR e Algoritmos Genéticos, de forma eficaz, gerando assim um Recomendador de Configuração, que é o objetivo deste trabalho. A eficácia representada na Tabela 6 e nos Gráficos 3 e 4 demonstram que a predição e a configuração sugeridas são qualitativamente similares à execução real no ambiente em nuvem. As variações máximas de 5,29% e erro médio de 3,07% são decorrentes das variações existentes no ambiente real, como tráfego de rede, links de comunicação e rotas de acesso, dentre outros fatores, não comprometendo a qualidade e eficiência do processo proposto.

4.6 Comparação com Outros Trabalhos

Comparar os resultados obtidos neste trabalho com aqueles relacionados na Seção 1.1 não foi tarefa fácil, devido ao fato de os trabalhos apresentarem diferentes metodologias, dados históricos e objetivos na predição dos seus experimentos. Entretanto, foi possível extrair algumas conclusões, listadas na sequência.

Comparando os resultados obtidos em Zhu et al. (2016), Patel, Chaudhary e Garg (2016), Sembiring e Beyer (2013), que tiveram uma acurácia maior utilizando a regressão através do SVR, os trabalhos que obtiveram redução de custos como em Prevost et al. (2011) e ainda os que levaram à redução da intervenção humana como em Niehorster et al. (2011), Bankole e Ajila (2013), Hormozi et al. (2012), com o método proposto neste trabalho, conforme apresentado na Tabela 7, vê-se que este mostra-se promissor, uma vez que consegue estimar com acurácia

confiável, gerando consequentemente redução de custos através do melhor aproveitamento dos recursos disponíveis e diminuição da intervenção humana, na integração do método através da API existente entre Recomendador, o Eucalyptus e o Amazon AWS, associando assim os três objetivos em uma única metodologia.

Tabela 7: Comparação com trabalhos relacionados.

Autores	**Objetivos**		
	Estimativa com SVR	**Redução de Custos**	**Redução de Intervenção Humana**
ZHU et al., 2016	X		
PATEL; CHAUDHARY; GARG, 2016	X		
SEMBIRING; BEYER, 2013	X		
PREVOST et al., 2011		X	
NIEHORSTER et al., 2011			X
BANKOLE; AJILA, 2013			X
HORMOZI et al., 2012			X
MÉTODO PROPOSTO	**X**	**X**	**X**

Apesar de este trabalho ter estudado o Recomendador na aplicação Weka, o modelo pode ser utilizado em outras soluções, sendo necessária uma base de dados histórica para submissão ao regressor e o treinamento do mesmo. Todo o conjunto funcionaria sem necessidade de alterações. As limitações de quantidade de máquinas e ou núcleos a ser utilizada são parâmetros de configuração do Algoritmo Genético.

Uma outra diferença entre os trabalhos relacionados e o método proposto nesta dissertação é a utilização de Algoritmos Genéticos. Aqueles utilizam outros métodos, como Redes Neurais e outros Algoritmos (PSO, ARIMA), ou somente utilizaram SVR para predição. O método proposto vai além da estimativa da execução, pois propõe melhores configurações a serem utilizadas, gerando assim redução de custos, tempos de intervenção e execução.

5 Conclusão

Estimar o uso de recursos computacionais é uma importante porém difícil tarefa, principalmente devido ao fato de que o uso desses recursos tem natureza e características muito distintas e comportamentos muitas vezes imprevisíveis, tanto para os provedores de infraestrutura quanto para os usuários. Por outro lado, o uso de dados históricos de utilizações permite mapear o uso e o comportamento das execuções.

O objetivo desta dissertação foi o desenvolvimento de um protótipo com o uso exclusivo do regressor SVR juntamente com Algoritmo Genético, de modo a permitir a sugestão de configuração viável e a estimativa do tempo e do custo de execução de aplicações Weka e assim contribuir para auxiliar os pesquisadores na realização e planejamento de suas pesquisas de forma mais precisa.

Para alcançar este objetivo, foram utilizadas técnicas de aprendizado de máquina e inteligência computacional através de máquina de vetores de suporte e algoritmo genético. O método proposto foi organizado em quatro etapas: obtenção de dados históricos, estimativa através do SVR, geração de possíveis configurações através do Algoritmo Genético e validação dos resultados.

No trabalho, foram realizados vários testes para comprovar o uso de cada técnica utilizada e também para verificar a eficácia do método proposto. O experimento de melhor resultado mostra que uma base de conhecimento ampla e bem treinada consegue predizer o tempo com segurança na execução das aplicações, de acordo com as configurações sugeridas pelo Algoritmo Genético, de forma rápida e eficaz, com erros próximos de 5% ao ambiente real, e tendo economia de, em alguns casos, até 38,8% no tempo e 45,62% no custo em relação ao executado inicialmente.

No decorrer deste trabalho podem ser observadas algumas contribuições. A primeira é a geração de um espaço de características de execução de aplicações Weka em nuvem. A segunda é a utilização do SVR para estimativa de tempo e custo de execução de aplicações em ambiente em nuvem. A terceira e principal contribuição é a criação de um protótipo de recomendação de configuração e estimativa de execução de aplicações em ambiente em nuvem.

Os resultados obtidos demonstraram que a execução real em relação à estimada é promissora, atingindo um índice de acerto superior a 94% na estimativa do tempo e custos.

5.1 Trabalhos Futuros

Existe um grande interesse da comunidade acadêmica em pesquisas de métodos computacionais na área da Computação em Nuvem, dada a relevância do tema. Assim, como possíveis trabalhos futuros, a fim de ampliar a validação do processo desenvolvido, pode-se apontar:

1. Pesquisar outras aplicações que possam ser avaliadas na arquitetura em nuvem e capturar uma base de espaço de características para submissão ao Recomendador.
2. Avaliar a utilização de outro método de estimativa, como Redes Neurais, e outros modelos, como o PSO, em substituição ao Algoritmo Genético.
3. Realizar a integração do Recomendador com a IaaS, ou seja, com os Provedores de Serviço em Nuvem e a retroalimentação da base de conhecimento e consequente retreinamento do Regressor.

Referências

ABREU, A. L. E. de; NETO, A. C. Estimação do grau de astigmatismo pelo método support vector regression correlacionado. *Cadernos do IME-Série Estatística*, v. 41, p. 15, 2017.

ALAM, F.; PACHAURI, S. Detection using weka. *Advances in Computational Sciences and Technology*, v. 10, n. 6, p. 1731–1743, 2017.

ALMEIDA, J. D. S. d. *Metodologia computacional para detecção e diagnóstico automáticos e planejamento cirúrgico do estrabismo*. Tese (Doutorado), 2013. DEPARTAMENTO DE ENGENHARIA DA ELETRICIDADE/CCET. Disponível em: <http://tedebc.ufma.br:8080/jspui/handle/tede/1823>.

AMAZON. *Amazon Web Service*. 2016. Disponível em: `http://amazon.com`. Acesso em: Junho 2016.

AVELAR, C. F. P. de; ROCHA, T. A. H.; CRUZ, F. J. S. Mineração de dados. *Revista Vianna Sapiens*, v. 8, n. 2, p. 25–25, 2017.

BANKOLE, A. A.; AJILA, S. A. Predicting cloud resource provisioning using machine learning techniques. In: IEEE. *Electrical and Computer Engineering (CCECE), 2013 26th Annual IEEE Canadian Conference on*. [S.l.], 2013. p. 1–4.

BREBNER, P. C. Is your cloud elastic enough?: Performance modelling the elasticity of infrastructure as a service (iaas) cloud applications. In: *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*. New York, NY, USA: ACM, 2012. (ICPE '12), p. 263–266. ISBN 978-1-4503-1202-8. Disponível em: <http://doi.acm.org/10.1145/2188286.2188334>.

CARVALHO, É. G. d. *Desenvolvimento de um sistema óptico para retinografia e angiografia digital*. Dissertação (Mestrado) — Universidade de São Paulo, Instituto de Física de São Carlos, São Carlos, 2006.

CUNHA, C. R. et al. The role of cloud computing in the development of information systems for smes. *IBIMA Publishing*, p. 1–7, 2017. ISSN 2326-6538. Disponível em: <http://hdl.handle.net/10198/14061>.

HALL, M. et al. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, ACM, New York, NY, USA, v. 11, n. 1, p. 10–18, nov. 2009. ISSN 1931-0145. Disponível em: <http://doi.acm.org/10.1145/1656274.1656278>.

HERBST, N. R.; KOUNEV, S.; REUSSNER, R. Elasticity in cloud computing: What it is, and what it is not. In: *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13)*. San Jose, CA: USENIX, 2013. p. 23–27. ISBN 978-1-931971-02-7. Disponível em: <https://www.usenix.org/conference/icac13/technical-sessions/presentation/herbst>.

HORMOZI, E. et al. Using of machine learning into cloud environment (a survey): managing and scheduling of resources in cloud systems. In: IEEE. *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2012 Seventh International Conference on*. [S.l.], 2012. p. 363–368.

HP. *Eucalyptus 4.3.1 Administration Guide*. 2016. Disponível em: `http://docs.hpcloud.com/eucalyptus/4.3.1/`. Acesso em: Junho 2016.

HP. *HP Eucalyptus*. 2016. Disponível em: `http://hphelion.com`. Acesso em: Junho 2016.

HSIEH, W.-K. et al. Load balancing virtual machines deployment mechanism in sdn open cloud platform. In: IEEE. *Advanced Communication Technology (ICACT), 2015 17th International Conference on*. [S.l.], 2015. p. 329–335.

JOHNSON, D. V. D. L. Auto-scaling and cloud bursting service for eucalyptus. *IBIMA Publishing*, 2011. Disponível em: <https://people.cs.uct.ac.za/˜djohnson/Files/autoscale.pdf>.

KALMEGH, S. Analysis of weka data mining algorithm reptree, simple cart and randomtree for classification of indian news. *International Journal of Innovative Science, Engineering and Technology*, v. 2, n. 2, p. 438–46, 2015.

KUPFERMAN JEFF SILVERMAN, P. J. J. Scaling into the cloud. *ADVANCED OPERATING SYSTEMS*, p. 1–8, 2016.

LASSENIUS, M. Hpe eucalyptus molnmiljö för arcada. Yrkeshögskolan Arcada, 2016.

LORENA, A. C.; CARVALHO, A. C. de. Uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada*, v. 14, n. 2, p. 43–67, 2007.

LORIDO-BOTRAN, T.; MIGUEL-ALONSO, J.; LOZANO, J. A. A review of auto-scaling techniques for elastic applications in cloud environments. *Journal of Grid Computing*, v. 12, n. 4, p. 559–592, 2014. ISSN 1572-9184. Disponível em: <http://dx.doi.org/10.1007/s10723-014-9314-7>.

MELL, P.; GRANCE, T. The nist definition of cloud computing. national institute of standards and technology special publication 800-145. *Gaithersburg: US Department of Commerce. Google Scholar*, 2011.

MENDONÇA, J. A. F. de et al. Aplicação de redes neurais artificiais na otimização de placas laminadas. *Iberian Latin-American Congress on Computational Methods in Engineering*, 2016.

MOHAPATRA, S. et al. Comparison of various platforms in cloud computing. *International Journal of Computer Applications*, Foundation of Computer Science, v. 162, n. 7, 2017.

NEGNEVITSKY, M. *Artificial Intelligence: A Guide to Intelligent Systems*. 3st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2011.

NETO, O. P. d. S. et al. Detecção automática de massas em imagens mamográficas usando particle swarm optimization (pso) e índice de diversidade funcional. Universidade Federal do Maranhão, 2016.

NIEHORSTER, O. et al. Autonomic resource management with support vector machines. In: IEEE COMPUTER SOCIETY. *Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing*. [S.l.], 2011. p. 157–164.

OPENSTACK. *OpenStack Queens release delivers new features for containers, edge and HPC*. 2017. Disponível em: `https://www.openstack.org`. Acesso em: Janeiro 2017.

PACHECO, M. A. C. Algoritmos genéticos: princípios e aplicações. *ICA: Laboratório de Inteligência Computacional Aplicada. Departamento de Engenharia Elétrica. Pontifícia Universidade Católica do Rio de Janeiro. Fonte desconhecida*, 1999.

PARIKH, S. M. A survey on cloud computing resource allocation techniques. In: IEEE. *Engineering (NUiCONE), 2013 Nirma University International Conference on*. [S.l.], 2013. p. 1–5.

PATEL, M.; CHAUDHARY, S.; GARG, S. Machine learning based statistical prediction model for improving performance of live virtual machine migration. *Journal of Engineering*, Hindawi Publishing Corporation, v. 2016, 2016.

PENTAHO. *Pentaho Weka Server Datamining*. 2017. Disponível em: `http://wiki.pentaho.com/display/DATAMINING/Weka+Server`. Acesso em: Janeiro 2017.

PINHO, A. F. de et al. Algoritmos genéticos: Fundamentos e aplicações. 2013.

PREVOST, J. J. et al. Prediction of cloud data center networks loads using stochastic and neural models. In: IEEE. *System of Systems Engineering (SoSE), 2011 6th International Conference on*. [S.l.], 2011. p. 276–281.

PYTHON. *Python is a programming language that lets you work quickly and integrate systems more effectively*. 2017. Disponível em: `https://www.python.org/`. Acesso em: Setembro 2017.

QUORA. *What are kernels in machine learning and SVM and why do we need them?* 2016. Disponível em: `https://www.quora.com/What-are-kernels-in-machine-learning-and-SVM-and-why-do-we-need-them`. Acesso em: Agosto 2016.

RASCHKA, S.; MIRJALILI, V. *Python machine learning*. [S.l.]: Packt Publishing Ltd, 2017.

REZENDE, S. O. *Sistemas Inteligentes: Fundamentos e Aplicações*. [S.l.]: Manole, 2003.

SCURRA. *Infraestrutura e Serviços para Cloud: Sobre a nuvem híbrida*. 2016. Disponível em: `http://www.scurra.com.br/blog/infraestrutura-e-servicos-para-cloud-sobre-nuvem-hibrida/`. Acesso em: Agosto 2016.

SEGURANçA de˙A Segurança em Computação nas Nuvens. 2016. Disponível em: `http://www.de-seguranca.com.br/a-seguranca-em-computacao-nas-nuvens/`. Acesso em: Junho 2016.

SEMBIRING, K.; BEYER, A. Dynamic resource allocation for cloud-based media processing. In: ACM. *Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. [S.l.], 2013. p. 49–54.

SHARMA, N.; BAJPAI, A.; LITORIYA, M. R. Comparison the various clustering algorithms of weka tools. *facilities*, v. 4, n. 7, 2012.

SHEPPARD, C. *Genetic Algorithms with Python*. 1st. ed. [S.l.]: CreateSpace Independent Publishing Platform, 2016. ISBN 1540324001.

SILVA JOãO; VAZ, F. Modelo de implantação de nuvem privada em infraestruturas de computação em nuvem baseadas no sistema eucalyptus. *ERBASE 2013*, 2013.

SKLEARN. *scikit-learn - Machine Learning in Python*. 2017. Disponível em: `http://scikit-learn.org/stable/`. Acesso em: Setembro 2017.

SOUSA, E. et al. Cloud infrastructure planning considering different redundancy mechanisms. In: *Computing*. [S.l.: s.n.], 2016.

SOUSA, J. A. d. *Diagnóstico de glaucoma em retinografias utilizando funções geoestatística*. Dissertação (Mestrado) — Universidade Federal do Maranhão, Departamento de Ciência da Computação, São Luis, 2017.

TEIXEIRA, L. de A. Métodos de regressão para aprendizado por reforço. 2016.

WEKA. *The University of Waikato. Weka 3: Data mining software in Java*. 2017. Disponível em: `http://www.cs.waikato.ac.nz/ml/weka`. Acesso em: Janeiro 2017.

WITTEN, I. H. et al. *Data Mining: Practical machine learning tools and techniques*. [S.l.]: Morgan Kaufmann, 2016.

ZHU, Z. et al. Pso-svr-based resource demand prediction in cloud computing. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, Fuji Technology Press Ltd., v. 20, n. 2, p. 324–331, 2016.