Combinação de modelos de previsão de séries
temporais por meio de otimização multiobjetivo para
alocação eficiente de recursos na nuvem

Valter Rogério Messias

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Valter Rogério Messias

Combinação de modelos de previsão de séries temporais
por meio de otimização multiobjetivo para alocação eficiente
de recursos na nuvem

Tese apresentada ao Instituto de Ciências
Matemáticas e de Computação – ICMC-USP,
como parte dos requisitos para obtenção do título
de Doutor em Ciências – Ciências de Computação e
Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e
Matemática Computacional

Orientador: Prof. Dr. Julio César Estrella

USP – São Carlos
Julho de 2016

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

M634c Messias, Valter Rogério
 Combinação de modelos de previsão de séries
temporais por meio de otimização multiobjetivo para
alocação eficiente de recursos na nuvem / Valter
Rogério Messias; orientador Julio César Estrella. -
São Carlos - SP, 2016.
 207 p.

 Tese (Doutorado - Programa de Pós-Graduação em
Ciências de Computação e Matemática Computacional)
- Instituto de Ciências Matemáticas e de Computação,
Universidade de São Paulo, 2016.

 1. Otimização Multiobjetivo. 2. Modelos de
Previsão de Séries Temporais. 3. Elasticidade.
4. Computação em Nuvem. I. Estrella, Julio César,
orient. II. Título.

Valter Rogério Messias

Combination of time series forecasting models through
multi-objective optimization for efficient allocation of
resources in the cloud

Doctoral dissertation submitted to the Instituto de
Ciências Matemáticas e de Computação – ICMC-
USP, in partial fulfillment of the requirements for the
degree of the Doctorate Program in Computer Science
and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and
Computational Mathematics

Advisor: Prof. Dr. Julio César Estrella

USP – São Carlos
July 2016

*Este trabalho é dedicado à minha avó Catarina (in memoriam),
que sempre me acompanhou durante toda a sua vida.*

AGRADECIMENTOS

À minha mãe Ilda, minha avó Catarina (in memorian) e meu avô Benedito (in memorian), pelo carinho, incentivo e confiança, sem os quais esta conquista não seria possível.

Em especial, aos meus orientadores, Prof. Marcos José Santana e Júlio Cezar Estrella, e aos professores Ricardo Ehlers e Alexandre Delbem pela ajuda no desenvolvimento do trabalho.

A todos os professores e amigos que passaram pelo LaSDPC desde 2005 (início do mestrado) até este momento, pela ajuda, discussões e brincadeiras que contribuíram para a minha formação.

Ao Instituto de Ciências Matemáticas e de Computação - ICMC, formado por todos os funcionários, pela colaboração no desenvolvimento deste trabalho, que sempre se mostraram disponíveis nos momentos em que os solicitei.

Ao Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - IFSP, que disponibilizou o afastamento das minhas atividades docentes para que pudesse finalizar este trabalho.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES pelo apoio financeiro.

A todos aqueles que contribuíram direta e indiretamente com este trabalho.

*“A menos que modifiquemos a nossa maneira de pensar,
não seremos capazes de resolver os problemas
causados pela forma como nos acostumamos a ver o mundo.”
(Albert Einstein)*

RESUMO

MESSIAS, V. R.. **Combinação de modelos de previsão de séries temporais por meio de otimização multiobjetivo para alocação eficiente de recursos na nuvem**. 2016. 207 f. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

Em um ambiente de computação em nuvem, as empresas têm a capacidade de alocar recursos de acordo com a demanda. No entanto, há um atraso que pode levar alguns minutos entre o pedido de um novo recurso e o mesmo estar pronto para uso. Por esse motivo, as técnicas reativas, que solicitam um novo recurso apenas quando o sistema atinge um determinado limiar de carga, não são adequadas para o processo de alocação de recursos. Para resolver esse problema, é necessário prever as requisições que chegam ao sistema, no próximo período de tempo, para alocar os recursos necessários antes que o sistema fique sobrecarregado. Existem vários modelos de previsão de séries temporais para calcular as previsões de carga de trabalho com base no histórico de dados de monitoramento. No entanto, é difícil saber qual é o melhor modelo de previsão a ser utilizado em cada caso. A tarefa se torna ainda mais complicada quando o usuário não tem muitos dados históricos a serem analisados. A maioria dos trabalhos relacionados, considera apenas modelos de previsão isolados para avaliar os resultados. Outros trabalhos propõem uma abordagem que seleciona modelos de previsão adequados para um determinado contexto. Mas, neste caso, é necessário ter uma quantidade significativa de dados para treinar o classificador. Além disso, a melhor solução pode não ser um modelo específico, mas sim uma combinação de modelos. Neste trabalho propomos um método de previsão adaptativo, usando técnicas de otimização multiobjetivo, para combinar modelos de previsão de séries temporais. O nosso método não requer uma fase prévia de treinamento, uma vez que se adapta constantemente a medida em que os dados chegam ao sistema. Para avaliar a nossa proposta usamos quatro *logs* extraídos de servidores reais. Os resultados mostram que a nossa proposta frequentemente converge para o melhor resultado, e é suficientemente genérica para se adaptar a diferentes tipos de séries temporais.

Palavras-chave: Otimização Multiobjetivo, Modelos de Previsão de Séries Temporais, Elasticidade, Computação em Nuvem.

ABSTRACT

MESSIAS, V. R.. **Combinação de modelos de previsão de séries temporais por meio de otimização multiobjetivo para alocação eficiente de recursos na nuvem.** 2016. 207 f. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

In a cloud computing environment, companies have the ability to allocate resources according to demand. However, there is a delay that may take minutes between the request for a new resource and it is ready for using. The reactive techniques, which request a new resource only when the system reaches a certain load threshold, are not suitable for the resource allocation process. To address this problem, it is necessary to predict requests that arrive at the system in the next period of time to allocate the necessary resources, before the system becomes overloaded. There are several time-series forecasting models to calculate the workload predictions based on history of monitoring data. However, it is difficult to know which is the best time series forecasting model to be used in each case. The work becomes even more complicated when the user does not have much historical data to be analyzed. Most related work considers only single methods to evaluate the results of the forecast. Other work propose an approach that selects suitable forecasting methods for a given context. But in this case, it is necessary to have a significant amount of data to train the classifier. Moreover, the best solution may not be a specific model, but rather a combination of models. In this work we propose an adaptive prediction method using multi-objective optimization techniques to combine time-series forecasting models. Our method does not require a previous phase of training, because it constantly adapts the extent to which the data is coming. To evaluate our proposal we use four logs extracted from real servers. The results show that our proposal often brings the best result, and is generic enough to adapt to various types of time series.

Key-words: Multi-objective Optimization, Time Series Prediction Models, Elasticity, Cloud Computing.

LISTA DE ILUSTRAÇÕES

Figura 1 – Arquitetura de Serviços na Nuvem (WHATELY <i>et al.</i> , 2010).	8
Figura 2 – Interação entre Clientes e Serviços na Nuvem (WHATELY <i>et al.</i> , 2010). . .	9
Figura 3 – Problema modelado de maneira hierárquica.	17
Figura 4 – Etapas de um algoritmo genético.	20
Figura 5 – Aplicação Web hospedada na infraestrutura da nuvem.	34
Figura 6 – Exemplo de Avaliação	42
Figura 7 – Exemplo de cruzamento	43
Figura 8 – Séries Utilizadas.	49
Figura 9 – Clarknet (CMPL) - Primeiro Cenário	52
Figura 10 – Clarknet (CMPL) - Segundo Cenário	53
Figura 11 – Clarknet (CMPL) - Terceiro Cenário	54
Figura 12 – Clarknet (CMPL) - Quarto Cenário	55
Figura 13 – Nasa (CMPL) - Primeiro Cenário	57
Figura 14 – Nasa (CMPL) - Segundo Cenário	58
Figura 15 – Nasa (CMPL) - Terceiro Cenário	59
Figura 16 – Nasa (CMPL) - Quarto Cenário	60
Figura 17 – Copa do Mundo (CMPL) - Primeiro Cenário	61
Figura 18 – Copa do Mundo (CMPL) - Segundo Cenário	62
Figura 19 – Copa do Mundo (CMPL) - Terceiro Cenário	63
Figura 20 – Copa do Mundo (CMPL) - Quarto Cenário	64
Figura 21 – Google (CMPL) - Primeiro Cenário	66
Figura 22 – Google (CMPL) - Segundo Cenário	67
Figura 23 – Google (CMPL) - Terceiro Cenário	68
Figura 24 – Google (CMPL) - Quarto Cenário	69
Figura 25 – Clarknet (CMPLR) - Primeiro Cenário	72
Figura 26 – Clarknet (CMPLR) - Segundo Cenário	73
Figura 27 – Clarknet (CMPLR) - Terceiro Cenário	74
Figura 28 – Clarknet (CMPLR) - Quarto Cenário	75
Figura 29 – Nasa (CMPLR) - Primeiro Cenário	76
Figura 30 – Nasa (CMPLR) - Segundo Cenário	77
Figura 31 – Nasa (CMPLR) - Terceiro Cenário	78
Figura 32 – Nasa (CMPLR) - Quarto Cenário	79
Figura 33 – Copa do Mundo (CMPLR) - Primeiro Cenário	80

Figura 34 – Copa do Mundo (CMPLR) - Segundo Cenário	81
Figura 35 – Copa do Mundo (CMPLR) - Terceiro Cenário	82
Figura 36 – Copa do Mundo (CMPLR) - Quarto Cenário	83
Figura 37 – Google (CMPLR) - Primeiro Cenário	85
Figura 38 – Google (CMPLR) - Segundo Cenário	86
Figura 39 – Google (CMPLR) - Terceiro Cenário	87
Figura 40 – Google (CMPLR) - Quarto Cenário	88
Figura 41 – Clarknet (CMAHP) - Primeiro Cenário	90
Figura 42 – Clarknet (CMAHP) - Segundo Cenário	91
Figura 43 – Clarknet (CMAHP) - Terceiro Cenário	92
Figura 44 – Clarknet (CMAHP) - Quarto Cenário	93
Figura 45 – Nasa (CMAHP) - Primeiro Cenário	95
Figura 46 – Nasa (CMAHP) - Segundo Cenário	96
Figura 47 – Nasa (CMAHP) - Terceiro Cenário	97
Figura 48 – Nasa (CMAHP) - Quarto Cenário	98
Figura 49 – Copa do Mundo (CMAHP) - Primeiro Cenário	100
Figura 50 – Copa do Mundo (CMAHP) - Segundo Cenário	101
Figura 51 – Copa do Mundo (CMAHP) - Terceiro Cenário	102
Figura 52 – Copa do Mundo (CMAHP) - Quarto Cenário	103
Figura 53 – Google (CMAHP) - Primeiro Cenário	105
Figura 54 – Google (CMAHP) - Segundo Cenário	106
Figura 55 – Google (CMAHP) - Terceiro Cenário	107
Figura 56 – Google (CMAHP) - Quarto Cenário	108
Figura 57 – Clarknet (CMAGM) - Primeiro Cenário	110
Figura 58 – Clarknet (CMAGM) - Segundo Cenário	111
Figura 59 – Clarknet (CMAGM) - Terceiro Cenário	112
Figura 60 – Clarknet (CMAGM) - Quarto Cenário	113
Figura 61 – Nasa (CMAGM) - Primeiro Cenário	114
Figura 62 – Nasa (CMAGM) - Segundo Cenário	115
Figura 63 – Nasa (CMAGM) - Terceiro Cenário	116
Figura 64 – Nasa (CMAGM) - Quarto Cenário	117
Figura 65 – Copa do Mundo (CMAGM) - Primeiro Cenário	119
Figura 66 – Copa do Mundo (CMAGM) - Segundo Cenário	120
Figura 67 – Copa do Mundo (CMAGM) - Terceiro Cenário	121
Figura 68 – Copa do Mundo (CMAGM) - Quarto Cenário	122
Figura 69 – Google (CMAGM) - Primeiro Cenário	123
Figura 70 – Google (CMAGM) - Segundo Cenário	124
Figura 71 – Google (CMAGM) - Terceiro Cenário	125
Figura 72 – Google (CMAGM) - Quarto Cenário	126

Figura 73 – Séries Utilizadas.	143
Figura 74 – Série FIFA World Cup (CMPL) - Primeiro Cenário	144
Figura 75 – Série FIFA World Cup (CMPL) - Segundo Cenário	145
Figura 76 – Série FIFA World Cup (CMPL) - Terceiro Cenário	146
Figura 77 – Série FIFA World Cup (CMPL) - Quarto Cenário	147
Figura 78 – Série Nasa (CMPL) - Primeiro Cenário	148
Figura 79 – Série Nasa (CMPL) - Segundo Cenário	149
Figura 80 – Série Nasa (CMPL) - Terceiro Cenário	150
Figura 81 – Série Nasa (CMPL) - Quarto Cenário	151
Figura 82 – Série Clarknet (CMPL) - Primeiro Cenário	152
Figura 83 – Série Clarknet (CMPL) - Segundo Cenário	153
Figura 84 – Série Clarknet (CMPL) - Terceiro Cenário	154
Figura 85 – Série Clarknet (CMPL) - Quarto Cenário	155
Figura 86 – Série Google (CMPL) - Primeiro Cenário	156
Figura 87 – Série Google (CMPL) - Segundo Cenário	157
Figura 88 – Série Google (CMPL) - Terceiro Cenário	158
Figura 89 – Série Google (CMPL) - Quarto Cenário	159
Figura 90 – Série FIFA World Cup (CMPLR) - Primeiro Cenário	160
Figura 91 – Série FIFA World Cup (CMPLR) - Segundo Cenário	161
Figura 92 – Série FIFA World Cup (CMPLR) - Terceiro Cenário	162
Figura 93 – Série FIFA World Cup (CMPLR) - Quarto Cenário	163
Figura 94 – Série Nasa (CMPLR) - Primeiro Cenário	164
Figura 95 – Série Nasa (CMPLR) - Segundo Cenário	165
Figura 96 – Série Nasa (CMPLR) - Terceiro Cenário	166
Figura 97 – Série Nasa (CMPLR) - Quarto Cenário	167
Figura 98 – Série Clarknet (CMPLR) - Primeiro Cenário	168
Figura 99 – Série Clarknet (CMPLR) - Segundo Cenário	169
Figura 100 – Série Clarknet (CMPLR) - Terceiro Cenário	170
Figura 101 – Série Clarknet (CMPLR) - Quarto Cenário	171
Figura 102 – Série Google (CMPLR) - Primeiro Cenário	172
Figura 103 – Série Google (CMPLR) - Segundo Cenário	173
Figura 104 – Série Google (CMPLR) - Terceiro Cenário	174
Figura 105 – Série Google (CMPLR) - Quarto Cenário	175
Figura 106 – Série FIFA World Cup (CMAHP) - Primeiro Cenário	176
Figura 107 – Série FIFA World Cup (CMAHP) - Segundo Cenário	177
Figura 108 – Série FIFA World Cup (CMAHP) - Terceiro Cenário	178
Figura 109 – Série FIFA World Cup (CMAHP) - Quarto Cenário	179
Figura 110 – Série Nasa (CMAHP) - Primeiro Cenário	180
Figura 111 – Série Nasa (CMAHP) - Segundo Cenário	181

Figura 112–Série Nasa (CMAHP) - Terceiro Cenário	182
Figura 113–Série Nasa (CMAHP) - Quarto Cenário	183
Figura 114–Série Clarknet (CMAHP) - Primeiro Cenário	184
Figura 115–Série Clarknet (CMAHP) - Segundo Cenário	185
Figura 116–Série Clarknet (CMAHP) - Terceiro Cenário	186
Figura 117–Série Clarknet (CMAHP) - Quarto Cenário	187
Figura 118–Série Google (CMAHP) - Primeiro Cenário	188
Figura 119–Série Google (CMAHP) - Segundo Cenário	189
Figura 120–Série Google (CMAHP) - Terceiro Cenário	190
Figura 121–Série Google (CMAHP) - Quarto Cenário	191
Figura 122–Série FIFA World Cup (CMAGM) - Primeiro Cenário	192
Figura 123–Série FIFA World Cup (CMAGM) - Segundo Cenário	193
Figura 124–Série FIFA World Cup (CMAGM) - Terceiro Cenário	194
Figura 125–Série FIFA World Cup (CMAGM) - Quarto Cenário	195
Figura 126–Série Nasa (CMAGM) - Primeiro Cenário	196
Figura 127–Série Nasa (CMAGM) - Segundo Cenário	197
Figura 128–Série Nasa (CMAGM) - Terceiro Cenário	198
Figura 129–Série Nasa (CMAGM) - Quarto Cenário	199
Figura 130–Série Clarknet (CMAGM) - Primeiro Cenário	200
Figura 131–Série Clarknet (CMAGM) - Segundo Cenário	201
Figura 132–Série Clarknet (CMAGM) - Terceiro Cenário	202
Figura 133–Série Clarknet (CMAGM) - Quarto Cenário	203
Figura 134–Série Google (CMAGM) - Primeiro Cenário	204
Figura 135–Série Google (CMAGM) - Segundo Cenário	205
Figura 136–Série Google (CMAGM) - Terceiro Cenário	206
Figura 137–Série Google (CMAGM) - Quarto Cenário	207

LISTA DE ALGORITMOS

Algoritmo 1 – ALOCAÇÃO DE RECURSOS USANDO SOMA PONDERADA	36
Algoritmo 2 – ALOCAÇÃO DE RECURSOS USANDO O MÉTODO ϵ -RESTRITO	38
Algoritmo 3 – ALOCAÇÃO DE RECURSOS USANDO O AHP	41
Algoritmo 4 – ALOCAÇÃO DE RECURSOS USANDO AG	44

LISTA DE TABELAS

Tabela 1 – Escalas de comparação segundo Saaty	17
Tabela 2 – Matrizes de julgamento das alternativas	18
Tabela 3 – Matriz de julgamento dos critérios	18
Tabela 4 – Vetor de prioridades global	19
Tabela 5 – Recursos sub e superprovisionados	39
Tabela 6 – Matrizes de julgamento dos modelos	39
Tabela 7 – Peso de cada modelo na previsão (PG)	40
Tabela 8 – Cenários considerados nos experimentos	50
Tabela 9 – Clarknet (CMPL) - Primeiro Cenário	52
Tabela 10 – Clarknet (CMPL) - Segundo Cenário	53
Tabela 11 – Clarknet (CMPL) - Terceiro Cenário	54
Tabela 12 – Clarknet (CMPL) - Quarto Cenário	55
Tabela 13 – Nasa (CMPL) - Primeiro Cenário	57
Tabela 14 – Nasa (CMPL) - Segundo Cenário	58
Tabela 15 – Nasa (CMPL) - Terceiro Cenário	59
Tabela 16 – Nasa (CMPL) - Quarto Cenário	60
Tabela 17 – Copa do Mundo (CMPL) - Primeiro Cenário	61
Tabela 18 – Copa do Mundo (CMPL) - Segundo Cenário	62
Tabela 19 – Copa do Mundo (CMPL) - Terceiro Cenário	63
Tabela 20 – Copa do Mundo (CMPL) - Quarto Cenário	64
Tabela 21 – Google (CMPL) - Primeiro Cenário	66
Tabela 22 – Google (CMPL) - Segundo Cenário	67
Tabela 23 – Google (CMPL) - Terceiro Cenário	68
Tabela 24 – Google (CMPL) - Quarto Cenário	69
Tabela 25 – Clarknet (CMPLR) - Primeiro Cenário	72
Tabela 26 – Clarknet (CMPLR) - Segundo Cenário	73
Tabela 27 – Clarknet (CMPLR) - Terceiro Cenário	74
Tabela 28 – Clarknet (CMPLR) - Quarto Cenário	75
Tabela 29 – Nasa (CMPLR) - Primeiro Cenário	76
Tabela 30 – Nasa (CMPLR) - Segundo Cenário	77
Tabela 31 – Nasa (CMPLR) - Terceiro Cenário	78
Tabela 32 – Nasa (CMPLR) - Quarto Cenário	79
Tabela 33 – Copa do Mundo (CMPLR) - Primeiro Cenário	80

Tabela 34 – Copa do Mundo (CMPLR) - Segundo Cenário	81
Tabela 35 – Copa do Mundo (CMPLR) - Terceiro Cenário	82
Tabela 36 – Copa do Mundo (CMPLR) - Quarto Cenário	83
Tabela 37 – Google (CMPLR) - Primeiro Cenário	85
Tabela 38 – Google (CMPLR) - Segundo Cenário	86
Tabela 39 – Google (CMPLR) - Terceiro Cenário	87
Tabela 40 – Google (CMPLR) - Quarto Cenário	88
Tabela 41 – Clarknet (CMAHP) - Primeiro Cenário	90
Tabela 42 – Clarknet (CMAHP) - Segundo Cenário	91
Tabela 43 – Clarknet (CMAHP) - Terceiro Cenário	92
Tabela 44 – Clarknet (CMAHP) - Quarto Cenário	93
Tabela 45 – Nasa (CMAHP) - Primeiro Cenário	95
Tabela 46 – Nasa (CMAHP) - Segundo Cenário	96
Tabela 47 – Nasa (CMAHP) - Terceiro Cenário	97
Tabela 48 – Nasa (CMAHP) - Quarto Cenário	98
Tabela 49 – Copa do Mundo (CMAHP) - Primeiro Cenário	100
Tabela 50 – Copa do Mundo (CMAHP) - Segundo Cenário	101
Tabela 51 – Copa do Mundo (CMAHP) - Terceiro Cenário	102
Tabela 52 – Copa do Mundo (CMAHP) - Quarto Cenário	103
Tabela 53 – Google (CMAHP) - Primeiro Cenário	105
Tabela 54 – Google (CMAHP) - Segundo Cenário	106
Tabela 55 – Google (CMAHP) - Terceiro Cenário	107
Tabela 56 – Google (CMAHP) - Quarto Cenário	108
Tabela 57 – Clarknet (CMAGM) - Primeiro Cenário	110
Tabela 58 – Clarknet (CMAGM) - Segundo Cenário	111
Tabela 59 – Clarknet (CMAGM) - Terceiro Cenário	112
Tabela 60 – Clarknet (CMAGM) - Quarto Cenário	113
Tabela 61 – Nasa (CMAGM) - Primeiro Cenário	114
Tabela 62 – Nasa (CMAGM) - Segundo Cenário	115
Tabela 63 – Nasa (CMAGM) - Terceiro Cenário	116
Tabela 64 – Nasa (CMAGM) - Quarto Cenário	117
Tabela 65 – Copa do Mundo (CMAGM) - Primeiro Cenário	119
Tabela 66 – Copa do Mundo (CMAGM) - Segundo Cenário	120
Tabela 67 – Copa do Mundo (CMAGM) - Terceiro Cenário	121
Tabela 68 – Copa do Mundo (CMAGM) - Quarto Cenário	122
Tabela 69 – Google (CMAGM) - Primeiro Cenário	123
Tabela 70 – Google (CMAGM) - Segundo Cenário	124
Tabela 71 – Google (CMAGM) - Terceiro Cenário	125
Tabela 72 – Google (CMAGM) - Quarto Cenário	126

Tabela 73 – Adequabilidade Genérica	127
---	-----

LISTA DE ABREVIATURAS E SIGLAS

AHP		Analytic Hierarchy Process
AR		AutoRegressive
ARIMA	..	AutoRegressive Integrated Moving Average
ARMA	...	Autoregressive–Moving-Average
CMAGM	.	Combinação de Modelos por meio de Algoritmo Genético Multiobjetivo
CMAHP	..	Combinação de Modelos por meio do Método de Análise Hierárquica
CMPL		Combinação de Modelos por meio de Programação Linear
CMPLR	..	Combinação de Modelos por meio de Programação Linear com Restrição
EC2		Elastic Compute Cloud
ETS		Extended Exponential Smoothing
IaaS		Infrastructure as a Service
MA		Moving Average
NIST		National Institute of Standards and Technology
NSGA		<i>Non-dominated Sorted Genetic Algorithm</i>
PaaS		Platform as a Service
RL		Reinforcement Learning
S3		Simple Storage Service
SaaS		Software as a Service

SUMÁRIO

1	INTRODUÇÃO	1
1.1	Contextualização	1
1.2	Motivação	2
1.3	Objetivos	3
1.4	Estrutura do trabalho	3
2	REFERENCIAL TEÓRICO	5
2.1	Considerações Iniciais	5
2.2	Computação em Nuvem	5
2.2.1	Arquitetura de Serviços	7
2.2.2	Aplicações elásticas	10
2.3	Técnicas de autoscaling	10
2.3.1	Técnicas Reativas	10
2.3.1.1	<i>Técnicas baseadas em threshold</i>	11
2.3.1.2	<i>Teoria do Controle</i>	11
2.3.2	Técnicas Proativas	12
2.3.2.1	<i>Aprendizado por Reforço</i>	12
2.3.2.2	<i>Modelos de Previsão de Séries Temporais</i>	12
2.4	Otimização Multiobjetivo	14
2.4.1	Métodos de Otimização Multiobjetivo	15
2.4.1.1	<i>Método da soma ponderada</i>	15
2.4.1.2	<i>Método ε-restrito</i>	16
2.4.1.3	<i>Método de análise hierárquica (AHP)</i>	16
2.4.1.4	<i>Algoritmo Genético Multiobjetivo</i>	19
2.5	Considerações Finais	21
3	TRABALHOS RELACIONADOS	23
3.1	Considerações Iniciais	23
3.2	Técnicas Reativas	23
3.2.1	Técnicas baseadas em threshold	23
3.2.2	Teoria de Controle	26
3.3	Técnicas Proativas	28
3.3.1	Aprendizado por reforço	28

3.3.2	<i>Séries Temporais</i>	29
3.4	Considerações Finais	31
4	MODELAGEM DO SISTEMA	33
4.1	Considerações Iniciais	33
4.2	Autoscaling proposto	33
4.2.1	Previsão da demanda	34
4.2.1.1	Método da soma ponderada	35
4.2.1.1.1	Solução proposta usando soma ponderada	36
4.2.1.2	Método ε -restrito	37
4.2.1.2.1	Solução proposta usando o método ε -restrito	38
4.2.1.3	Método de Análise Hierárquica (AHP)	38
4.2.1.3.1	Solução proposta usando o AHP	40
4.2.1.4	Algoritmo Genético Multiobjetivo	40
4.2.1.4.1	Solução proposta usando AG	44
4.2.2	Alocação de recursos	45
4.3	Considerações Finais	46
5	AVALIAÇÃO E RESULTADOS	47
5.1	Considerações Iniciais	47
5.2	Experimentos e Avaliação	47
5.2.1	Metotologia	47
5.2.2	Logs Utilizados	48
5.3	Resultados	49
5.3.1	Algoritmo CMPL	50
5.3.1.1	Série Clarknet	50
5.3.1.2	Série Nasa	51
5.3.1.3	Série Copa do Mundo	56
5.3.1.4	Série Google	56
5.3.2	Algoritmo CMPLR	65
5.3.2.1	Série Clarknet	65
5.3.2.2	Série Nasa	71
5.3.2.3	Série Copa do Mundo	71
5.3.2.4	Série Google	84
5.3.3	Algoritmo CMAHP	84
5.3.3.1	Série Clarknet	84
5.3.3.2	Série Nasa	94
5.3.3.3	Série Copa do Mundo	94
5.3.3.4	Série Google	99
5.3.4	Algoritmo CMAGM	104

5.3.4.1	<i>Série Clarknet</i>	104
5.3.4.2	<i>Série Nasa</i>	109
5.3.4.3	<i>Série Copa do Mundo</i>	109
5.3.4.4	<i>Série Google</i>	118
5.3.5	Análise dos Resultados	127
5.3.6	Sobrecarga dos modelos	128
5.4	Considerações Finais	128
6	CONCLUSÃO	131
6.1	Contribuições	132
6.2	Trabalhos Futuros	133
REFERÊNCIAS		135
APÊNDICE A	PESOS DOS MODELOS DE PREVISÃO EM CADA CENÁRIO	143

INTRODUÇÃO

1.1 Contextualização

A computação em nuvem é uma tecnologia emergente que está se tornando cada vez mais popular porque, entre outras vantagens, permite aos clientes implantar facilmente aplicações elásticas, simplificando enormemente o processo de aquisição e liberação de recursos em tempo de execução. A elasticidade é uma das principais características da computação em nuvem, pois permite aos usuários adicionar e remover recursos quando necessário, pagando apenas pelos recursos realmente alocados. Os recursos, geralmente, são disponibilizados sob a forma de máquinas virtuais (VMs). As empresas podem usar a nuvem para uma variedade de propósitos, desde para a execução de trabalhos em lote, até para a hospedagem de aplicações web ([LORIDO-BOTRAN; MIGUEL-ALONSO; LOZANO, 2014](#)).

Na literatura, a arquitetura da nuvem é comumente dividida em três camadas: Infraestrutura como um Serviço (IaaS); Plataforma como um Serviço (PaaS); e Software como um Serviço (SaaS). Na camada de IaaS, são providos recursos de tecnologia da informação, tais como processamento, memória, armazenamento e largura de banda. Exemplos de provedores de IaaS são o Amazon EC2 ([AMAZON, 2010](#)), RackSpace ([RACKSPACE, 2010](#)) e o Google Compute Engine ([KRISHNAN; GONZALEZ, 2015](#)). A camada de PaaS disponibiliza ambientes de programação e ferramentas que podem ser utilizados pelos consumidores para construir e implementar aplicativos para a camada de infraestrutura de nuvem. Como exemplo podemos citar o Amazon Elastic Beanstalk ([VLIET *et al.*, 2011](#)), o Google App Engine ([ZAHARIEV, 2009](#)), e o Microsoft Windows Azure ([REDKAR; GUIDICI, 2009](#)). Por fim, a camada de SaaS fornece aos usuários aplicativos de software. Exemplos de provedores de SaaS são o Google Apps ([CONNER, 2008](#)), Microsoft Office 365 ([WILSON, 2014](#)) e Salesforce.com ([BARBARA, 2010](#)).

Neste trabalho nos concentramos na perspectiva do cliente de IaaS. Um cenário típico

poderia ser uma empresa, tal como um *e-commerce*, que queira hospedar o seu site de vendas, e para este fim aloca recursos de um provedor de IaaS, como o Amazon EC2, por exemplo.

1.2 Motivação

Como dito anteriormente, uma das principais características da computação em nuvem é a elasticidade. Essa característica permite que as aplicações possam adquirir e liberar recursos de forma dinâmica, ajustando às mudanças na demanda. No entanto, decidir a quantidade certa de recursos não é uma tarefa fácil. Para o usuário, é desejável ter um sistema que ajusta automaticamente os recursos, de acordo com a carga de trabalho que chega à aplicação, com o mínimo de intervenção possível. Esse sistema é chamado de Autoscaling, e é o foco deste trabalho.

Os recursos podem ser alocados de maneira horizontal ou vertical. No caso da alocação horizontal, cada recurso é uma réplica de um servidor (executando em uma VM). Novas réplicas são adicionadas ou liberadas conforme a necessidade. Por outro lado, a alocação vertical consiste em alterar os recursos atribuídos a uma VM, em tempo de execução. Por exemplo, o aumento ou redução da quantidade de núcleos de CPU alocadas, ou a quantidade de memória. O problema da alocação vertical é que sistemas operacionais mais comuns não permitem tais alterações sem ter que reiniciar o sistema. Esse é um dos motivos pelo qual a maioria dos provedores de nuvem oferecem apenas a alocação horizontal ([LORIDO-BOTRAN; MIGUEL-ALONSO; LOZANO, 2014](#)).

Um *autoscaling* deve procurar, em suas decisões, encontrar o melhor custo benefício para o usuário. Essa decisão depende, entre outros fatores, do regime de preços utilizados pelo provedor de nuvem. Geralmente, o modelo de precificação inclui tipos de VMs, e é cobrado por hora de uso. Um *autoscaling* deve, também, se preocupar em manter os acordos de nível de serviço (SLAs) com os clientes da aplicação. O SLA pode ser, por exemplo, o tempo máximo de resposta acordado com o cliente ou a quantidade mínima de requisições atendidas em determinado período de tempo.

As técnicas utilizadas para a construção de um *autoscaling* podem ser divididas em dois tipos: reativas e proativas. As técnicas reativas monitoram os eventos do sistema (uso da CPU, uso da memória, tamanho da fila, etc.) e decidem por alocar ou desalocar recursos quando esses eventos excedem um certo valor limite (*threshold*). Por outro lado, as técnicas proativas tentam prever a quantidade de recursos necessária no próximo período de tempo, para se antecipar a eventos indesejados.

O problema das técnicas reativas é que, frequentemente, o tempo para reagir é insuficiente. Isso ocorre porque o tempo para um recurso na nuvem (nesse caso, uma máquina virtual) estar pronto para uso após ter sido solicitado pode levar minutos ([LORIDO-BOTRÁN; MIGUEL-ALONSO; LOZANO, 2012](#)), tempo suficiente para que o sistema fique sobrecarregado. No caso

das técnicas proativas, existem vários modelos de previsão de séries temporais que podem ser utilizados para prever a demanda. No entanto, não existe um modelo que seja adequado para todos os tipos de séries temporais, geradas por diferentes tipos de aplicações. Além disso, uma mesma aplicação pode apresentar mudanças nas características de sua demanda ao longo do tempo, fazendo com que um modelo que estava prevendo bem passe a não ser mais adequado.

Para solucionar os problemas supracitados, é necessária a construção de um *autoscaling* proativo, baseado na combinação de modelos de previsão de séries temporais. Para determinar a importância (peso) de cada modelo de previsão na combinação utilizamos técnicas de otimização multiobjetivo. No nosso caso, os objetivos são: minimizar a quantidade de recursos subprovisionados e, conseqüentemente, o número de requisições não atendidas; e minimizar o número de recursos superprovisionados, ou ociosos, evitando gastos desnecessários.

1.3 Objetivos

A hipótese desta Tese é que é possível construir um *autoscaling* proativo, genérico, capaz de se adaptar a diferentes demandas e diferentes cenários de utilização, assim como a diferentes objetivos dos usuários.

Para comprovar a hipótese desta tese, o principal objetivo a ser alcançado é mostrar que a combinação de modelos de previsão de séries temporais, por meio de otimização multiobjetivo, é capaz de prover uma solução genérica, capaz de se adaptar a diferentes tipos de cargas de trabalho e diferentes objetivos dos usuários.

Para atingir tais objetivos, será realizado um estudo envolvendo: cargas de trabalho de diferentes tipos, extraídas de servidores reais; diferentes modelos de previsão de séries temporais; diferentes métodos de otimização multiobjetivo e diferentes cenários de avaliação.

1.4 Estrutura do trabalho

Esta Tese está organizada em seis capítulos. No próximo Capítulo, 2, o referencial teórico será apresentado, incluindo: uma breve explanação sobre computação em nuvem; descrição dos principais modelos de previsão de séries temporais; e apresentação dos métodos de otimização multiobjetivo utilizados.

No Capítulo 3, serão apresentados os trabalhos relacionados, divididos em seções de acordo com a técnica utilizada na construção do *autoscaling*.

O Capítulo 4 descreve a proposta de solução para o problema, explorando os algoritmos utilizados para combinar os modelos de previsão de séries temporais, com base em quatro métodos de otimização multiobjetivo: soma ponderada; método ϵ -restrito; método de análise hierárquica; e *Non-dominated Sorted Genetic Algorithm* (NSGA).

No Capítulo 5, a avaliação da proposta será apresentada, assim como a metodologia utilizada e os cenários considerados nos experimentos. Por fim, os resultados serão apresentados e discutidos.

Finalmente, no Capítulo 6, serão apresentadas a conclusão e as contribuições desta Tese e, ainda, algumas propostas de trabalhos futuros.

REFERENCIAL TEÓRICO

2.1 Considerações Iniciais

O objetivo deste capítulo é fornecer ao leitor o embasamento teórico necessário para o entendimento da proposta da Tese. Os temas abordados nas próximas seções serão: computação em nuvem; aplicações elásticas; modelos de previsão de séries temporais; e métodos de otimização multiobjetivo. De maneira alguma pretendemos esgotar todos esses assuntos no capítulo, mas sim fornecer uma introdução básica. Caso seja de interesse se aprofundar em algum tópico, o leitor poderá usar as referências que embasaram a escrita do capítulo.

2.2 Computação em Nuvem

A idéia essencial da computação em nuvem é permitir que recursos computacionais, como, armazenamento, processamento, banda, sejam disponibilizados através de serviços. Processo semelhante ocorreu no final do século XIX, durante o período da Revolução Industrial, quando era comum que grandes fábricas fossem responsáveis pela produção de sua própria energia elétrica. Hoje em dia as grandes fábricas consomem energia como um serviço, e pagam pela quantidade utilizada. A computação em nuvem tem uma proposta similar: recursos computacionais passarão a ser de responsabilidade de algumas empresas especializadas, que ficarão responsáveis por sua gestão e comercialização através de serviços ([CARR, 2008](#)).

Segundo [Breitman e Viterbo \(2010\)](#), o termo computação em nuvem foi introduzido em 2006, quando o então CEO da Google, Eric Schmidt, utilizou o termo para descrever os serviços da própria empresa, e, pouco tempo depois, quando a Amazon utilizou o mesmo termo para lançar seu serviço Elastic Compute Cloud (EC2). Mas foi popularizado através de um artigo na edição de Outubro da Wired, do autor George Gilder, intitulado “The Information Factories” (as fábricas de informação) ([GILDER, 2006](#)).

O National Institute of Standards and Technology (NIST) (MELL; GRANCE, 2009) define computação em nuvem como um modelo que possibilita acesso, de modo conveniente e sob demanda, a um conjunto de recursos computacionais configuráveis (redes, servidores, armazenamento, aplicações e serviços) que podem ser rapidamente adquiridos e liberados com mínimo esforço gerencial ou interação com o provedor de serviços. Existem diferentes modelos de implantação para a Computação em Nuvem. Os mais citados na literatura são: nuvem pública, privada, híbrida e comunitária (MELL; GRANCE, 2009).

Na nuvem pública todos os recursos disponíveis são abertos ao público em geral sob um contrato, onde se paga pelo montante utilizado. Os serviços comercializados são geralmente chamados de computação utilitária (*Utility Computing*). Alguns exemplos são as plataformas da Amazon, Google AppEngine e Microsoft Azure. Nuvem privada, difere de uma nuvem pública, no sentido de ser um ambiente disponibilizado por empresas privadas. Seus serviços são oferecidos apenas para usuários internos, e não para o público em geral.

Além dos modelos de nuvem pública e privada, existe outro modelo que combina os dois conceitos. Esse modelo é chamado de nuvem híbrida, que prevê uma utilização mista, porém integrada, dos dois paradigmas, combinando serviços de computação em nuvem externos com recursos internos. Esta colaboração deve ser realizada de forma coordenada, de forma a garantir integração no nível dos dados, processos e camadas de segurança (HASSAN; SONG; HUH, 2009).

Algumas características centrais da computação em nuvem, citadas na maioria dos textos relevantes sobre o assunto são:

Serviço sob demanda

O usuário pode adquirir unilateralmente recursos computacionais, como processamento no servidor ou armazenamento na rede, na medida em que necessite e sem precisar de interação humana com os provedores de cada serviço. O hardware e o software dentro de uma nuvem podem ser automaticamente reconfigurados, orquestrados e estas modificações são apresentadas de forma transparente para os usuários. Os recursos passam a ser disponibilizados e consumidos como utilitários. Em uma arquitetura baseada em serviços, características do consumidor são abstraídas das características do provedor através de interfaces de serviços bem definidas.

Independência de localização e Amplo acesso

Recursos são disponibilizados e acessados através de mecanismos padronizados que possibilitam o uso por plataformas *thin client*, tais como celulares, *laptops* e PDAs. A interface de acesso à nuvem não obriga os usuários a mudar suas condições e ambientes de trabalho, como por exemplo, linguagens de programação e sistema operacional. Os sistemas de software clientes instalados localmente para o acesso à nuvem são leves, como um navegador de Internet.

Os recursos físicos e virtuais, são dinamicamente atribuídos e ajustados de acordo com a demanda dos usuários, que não precisam ter conhecimento da localização física dos recursos

computacionais. Basta que se tenha acesso a infraestrutura de rede, para que se possa acessar os recursos independentemente da localização.

Elasticidade rápida

A elasticidade é, talvez, a característica mais inovadora de computação em nuvem. No entanto, é preciso distinguir a diferença entre elasticidade e escalabilidade. Escalabilidade é a habilidade de satisfazer um requisito de aumento da capacidade de trabalho pela adição proporcional da quantidade de recursos (BREITMAN; VITERBO, 2010). Uma arquitetura escalável é construída tipicamente com base em uma infraestrutura básica passível de repetição, cujo crescimento pode ser alcançado simplesmente com a adição repetida do mesmo conjunto de hardware básico. Quando se fala em escalabilidade, geralmente, não há preocupação com a remoção de recursos nem a preocupação se os recursos são plenamente utilizados, uma vez que os recursos adquiridos já foram pagos e não podem ter seus valores recuperados.

Ao contrário da escalabilidade, a elasticidade possibilita alocar e desalocar rapidamente grandes quantidades de recursos em tempo de execução. Esses recursos podem ser adquiridos automaticamente, de forma rápida, caso haja a necessidade de escalar com o aumento da demanda, e liberados na retração dessa demanda. Para os usuários, os recursos disponíveis para uso parecem ser ilimitados e podem ser adquiridos em qualquer quantidade e a qualquer momento. Isso é possível, devido a virtualização, que cria várias instâncias de recursos requisitados utilizando um único recurso real (ABOULNAGA *et al.*, 2009). A virtualização também é responsável por abstrair características físicas de uma plataforma computacional dos usuários, exibindo outro hardware virtual e emulando um ou mais ambientes.

Pagamento pelo uso

Outra característica central da computação em nuvem é o modelo de pagamento de acordo com a utilização dos recursos, como ocorre com os serviços de água, eletricidade, gás e telefonia, tão comuns e essenciais nas rotinas diárias. Dentro do paradigma de computação em nuvem, os consumidores de serviços e recursos computacionais necessitariam pagar aos provedores apenas quando se utilizarem de tais serviços. É fácil perceber que este modelo traz muitos benefícios aos usuários, que só contratarão novos recursos à medida em que estes se tornem necessários, fazendo com que grandes investimentos em infraestrutura e manutenção sejam descartados.

2.2.1 Arquitetura de Serviços

O ambiente de computação em nuvem é composto de três modelos de serviço: Infrastructure as a Service (IaaS); Platform as a Service (PaaS); e Software as a Service (SaaS). A figura 1 ilustra esses modelos.

Infraestrutura como um Serviço (IaaS)

Figura 1 – Arquitetura de Serviços na Nuvem (WHATELY *et al.*, 2010).

A IaaS é a parte responsável por prover toda a infraestrutura necessária para a PaaS e o SaaS. Essa infraestrutura é compreendida por processamento, *storage*, rede e outros recursos computacionais fundamentais, onde o cliente está apto a implantar e rodar qualquer software, o que pode incluir sistemas operacionais e aplicações. Os recursos oferecidos pela IaaS podem ser obtidos como se fossem um serviço. Exemplos são os serviços Elastic Compute Cloud (EC2) para processamento e Simple Storage Service (S3) para armazenamento da Amazon Web Services (ROBINSON, 2008).

A infraestrutura disponibilizada pela IaaS pode escalar dinamicamente, aumentando ou diminuindo os recursos de acordo com as necessidades das aplicações. Os recursos são disponibilizados como servidores virtuais, onde pode-se definir a quantidade exata de recursos necessários, gerando assim economia.

Plataforma como um Serviço (PaaS)

Na PaaS uma camada de alto nível é colocada acima da IaaS. O objetivo é integrar, implementar e testar aplicações na nuvem. O usuário não administra ou controla a infraestrutura localizada na IaaS, como rede, servidores, sistemas operacionais ou armazenamento, mas tem controle sobre as aplicações implantadas, que fazem uso dessa infraestrutura. A PaaS fornece um sistema operacional, linguagens de programação e ambientes de desenvolvimento para as aplicações.

Os recursos disponibilizados pela PaaS são alvo dos desenvolvedores de software que podem escrever suas aplicações de acordo com as especificações de uma plataforma em particular sem a necessidade de se preocuparem com a camada subjacente de infraestrutura de hardware (IaaS). Como exemplos de PaaS podemos destacar o Google App Engine (CIURANA, 2009).

Software como um Serviço (SaaS)

O modelo de SaaS proporciona sistemas de software com propósitos específicos que estão disponíveis para os usuários através da Internet. Os sistemas de software são acessíveis a partir de vários dispositivos do usuário, que tenham acesso a um navegador Web. No SaaS, o usuário não administra ou controla a infraestrutura da PaaS, acessando somente os softwares que estão disponíveis como serviços na camada acima da PaaS.

Como o software está na Web, ele pode ser acessado pelos usuários de qualquer lugar e a qualquer momento, permitindo maior integração entre unidades de uma mesma empresa ou outros serviços de software. Novos recursos podem ser incorporados automaticamente aos sistemas de software sem que os usuários precisem fazer atualização dos sistemas, nem se preocupar com a aquisição de licenças de softwares. Como exemplo de SaaS pode-se destacar o serviço Google Docs ([CIURANA, 2009](#)).

Cada uma das camadas citadas anteriormente é disponibilizada para interagir com usuários específicos, de acordo com o papel representado por estes. Os papéis são importantes para definir responsabilidades, acesso e perfil para os diferentes usuários que fazem parte e estão envolvidos em uma solução de computação em nuvem. A Figura 2 destaca estes papéis.

Figura 2 – Interação entre Clientes e Serviços na Nuvem ([WHATELY et al., 2010](#)).

Na figura 2 pode-se observar três tipos de usuários. O provedor, que é responsável por disponibilizar, gerenciar e monitorar toda a estrutura para a solução de computação em nuvem, deixando o desenvolvedor e o usuário final sem esse tipo de responsabilidade. O provedor pode interagir com as três camadas. Os desenvolvedores utilizam os recursos fornecidos na PaaS e disponibilizam softwares como serviços para os usuários finais. Estes, por sua vez, consomem os serviços disponíveis na camada SaaS.

2.2.2 Aplicações elásticas

Uma aplicação elástica é capaz de aumentar e diminuir seus recursos computacionais, horizontalmente ou verticalmente, para se adaptar a demanda. Esse tipo de aplicação, geralmente, possui um balanceador de carga como um *front-end* que recebe as requisições e as repassa para um conjunto de servidores no *back-end* que serão responsáveis por processá-las. Em um ambiente de computação em nuvem, esses servidores são representados por máquinas virtuais (VMs). Um *autoscaling* é responsável por tomar decisões a respeito de quando e como aumentar e diminuir a quantidade de recursos, de maneira automática, sem intervenção humana.

Um *autoscaling* pode ser implementado especificamente para determinada aplicação, ou ser implementado de maneira genérica, onde pode tratar de vários tipos de aplicações. Os provedores de infraestrutura na nuvem costumam oferecer esse último tipo de *autoscaling*. Em ambos os casos, o objetivo é encontrar o melhor *trade-off* entre atender o SLA (Acordo de nível de serviço) da aplicação e economizar recursos alugados junto ao provedor de infraestrutura na nuvem.

Em resumo, um *autoscaling* deve lidar com dois objetivos conflitantes: minimizar o subprovisionamento de recursos; e minimizar o superprovisionamento de recursos. O subprovisionamento de recursos ocorre quando não há recursos suficientes para atender todas as requisições que chegam ao sistema, cumprindo os acordos de níveis de serviço (SLAs). No caso do superprovisionamento há mais recursos disponíveis do que seriam necessários para atender todas as requisições, respeitando seus SLAs. Fica claro que, por serem conflitantes, a medida que se melhora um dos objetivos, a tendência é que o outro piore.

2.3 Técnicas de autoscaling

As técnicas de *autoscaling*, inicialmente, podem ser divididas em duas categorias: reativas; e proativas. Cada uma dessas categorias pode ser dividida em subcategorias, de acordo com a técnica de *autoscaling* utilizada: threshold; teoria de controle; aprendizado de máquina; e modelos de previsão de séries temporais.

2.3.1 Técnicas Reativas

Nesta categoria de técnicas, o sistema reage, apenas, quando detecta alguma mudança em relação a carga de trabalho. Para tanto, algumas variáveis (uso da CPU, uso da memória, tamanho da fila, etc.) são monitoradas continuamente. Quando os valores de tais variáveis atingem valores indesejados, alguma ação é tomada para tentar manter a estabilidade do sistema. O problema das técnicas reativas é que, frequentemente, o tempo para reagir é insuficiente. Isso ocorre porque o tempo para um recurso na nuvem (nesse caso, uma máquina virtual) estar pronto para uso após ter sido solicitado pode levar minutos ([LORIDO-BOTRAN](#); [MIGUEL-ALONSO](#); [LOZANO](#),

2014), tempo suficiente para que o sistema fique sobrecarregado. A seguir, apresentaremos duas técnicas pertencentes a essa categoria: *threshold*; e teoria de controle.

2.3.1.1 Técnicas baseadas em *threshold*

Técnicas baseadas em *threshold* são muito populares entre os provedores de infraestrutura na nuvem, tais como o Amazon EC2, e, também, em ferramentas disponibilizadas por terceiros, como o RightScale (CLARK, 2010). O motivo dessa popularidade se deve à simplicidade da técnica, onde é necessário definir uma ou mais variáveis a serem monitoradas, valores limites (*thresholds*) para tais variáveis, e ações a serem tomadas quando tais valores forem ultrapassados. Entretanto, definir *thresholds* adequados para as variáveis monitoradas não é uma tarefa fácil, pois requer profundo conhecimento da aplicação em questão.

Cada regra, definida pela técnica, tem duas partes: uma condição e uma ação a ser executada quando a condição for alcançada. A condição é baseada em uma ou mais métricas, tais como: taxa de chegada de requisições, carga da CPU, tempo médio de resposta, etc. Para cada métrica é definido um limiar superior e outro inferior. Se o limiar superior é ultrapassado por determinado período de tempo, a ação correspondente, aumentar recursos nesse caso, é executada. O mesmo vale para o limiar inferior, só que nesse caso a ação a ser realizada é a liberação de recursos. No caso do *Autoscaling* horizontal, os recursos são VMs. Já no caso do *Autoscaling* vertical, os recursos são núcleos de CPU ou quantidade de memória.

Como um exemplo do uso de técnicas baseadas em *threshold* podemos citar: adicione duas instâncias de máquina virtual quando a média do uso da CPU ultrapassar 70% por mais de cinco minutos; não faça nada por 10 minutos.

2.3.1.2 Teoria do Controle

A Teoria do Controle tem sido utilizada para automatizar o gerenciamento de diferentes tipos de sistemas de processamento de informação, tais como servidores web, sistemas de armazenamento de dados e clusters. O Controle tem por objetivo manter o valor de uma variável controlada (carga da CPU, por exemplo), perto de um determinado valor definido pelo usuário (valor de referência). Para tanto, ele ajusta o valor de outra variável, denominada variável manipulada (número de VMs, por exemplo). A variável manipulada é a entrada do Sistema alvo, enquanto a variável controlada, medida por um sensor, é considerada a saída do sistema.

Podemos classificar um Sistema de controle em três tipos principais: *open-loop*, *feedback* e *feed-forward*. Um controle do tipo *open-loop*, calcula a entrada para o Sistema alvo usando apenas o estado atual do sistema. Nesse caso não é usado *feedback* para determinar se a saída do Sistema atingiu o valor esperado. Por outro lado, um controle do tipo *feedback* monitora a saída do sistema, e é capaz de corrigir qualquer desvio em relação ao valor desejado. Já um controle do tipo *feed-forward* tenta antecipar os erros da saída, usando um modelo de previsão para prever o

comportamento do sistema e reagir antes que o erro ocorra. Como, geralmente, a previsão não é totalmente confiável, controles do tipo *feedback* e *feed-forward* costumam ser combinados.

2.3.2 Técnicas Proativas

As técnicas proativas tentam prever a quantidade de recursos necessária no próximo período de tempo, para se antecipar a eventos indesejados. Existem vários modelos de previsão de séries temporais que podem ser utilizados para prever a demanda. No entanto, não existe um modelo que seja adequado para todos os tipos de séries temporais, geradas por diferentes tipos de aplicações. Além disso, uma mesma aplicação pode apresentar mudanças nas características de sua demanda ao longo do tempo, fazendo com que um modelo que estava prevendo bem passe a não ser mais adequado. A seguir, apresentaremos duas técnicas pertencentes a essa categoria: aprendizado de máquina; e modelos de previsão de séries temporais.

2.3.2.1 Aprendizado por Reforço

Aprendizado por Reforço ou Reinforcement Learning (RL) (SUTTON; BARTO, 1998) é um método de tomada de decisão automática que não exige nem um conhecimento *a priori*. Essa técnica decide a melhor ação a ser tomada dependendo do estado atual da aplicação e da carga de trabalho que chega à aplicação. Nesse tipo de técnica, o foco do aprendizado é por meio da interação direta entre um agente (nesse caso, o Autoscaling) e seu ambiente (nesse caso, a aplicação). O Autoscaling aprenderá a partir da experiência com o ambiente (método de tentativa e erro) a melhor ação a ser tomada. Após executar a ação, o Autoscaling avalia o resultado da mesma no sistema, em relação ao alcance dos objetivos desejados. Dessa maneira será criado um mapa de ações de acordo com cada estado do sistema e com a demanda a qual esse está submetido.

2.3.2.2 Modelos de Previsão de Séries Temporais

Uma série temporal é uma sequência de pontos ordenados, observados em intervalos uniformes de tempo. Uma série temporal é frequentemente representada por uma função matemática que determina sua geração. Seus componentes podem ser determinísticos ou estocásticos. São determinísticos quando podem ser representados por uma função matemática, e estocásticos caso contrário.

O propósito de um modelo de previsão de séries temporais é: separar os componentes determinísticos e aleatórios da série; encontrar uma função matemática que represente os componentes determinísticos; e estimar os componentes aleatórios usando regras de probabilidade. A seguir serão descritos os principais modelos de previsão de séries temporais.

NAIVE

O modelo NAIVE, ou ingênuo, é muito simples. Este modelo assume que a próxima observação será igual a anterior. Necessita de apenas um ponto (último valor observado) para funcionar.

Auto-Regressivo (AR)

Neste modelo, a previsão do próximo ponto da série é calculada baseando-se em uma combinação linear das previsões anteriores (HYNDMAN; ATHANASOPOULOS, 2014). O usuário precisa informar um parâmetro (p) que define quantas previsões passadas devem ser consideradas pelo modelo.

Médias Móveis (MA)

O modelo de médias móveis é similar ao modelo auto-regressivo. A diferença é que ao invés de usar as previsões passadas, os erros de previsão são usados como base para um modelo de combinação linear (HYNDMAN; ATHANASOPOULOS, 2014). Esse modelo possui um parâmetro (q), que determina quantos erros passados devem ser considerados no modelo.

Auto-Regressivo de Médias Móveis (ARMA)

O modelo ARMA é uma combinação dos modelos AR e MA. Este modelo possui dois parâmetros. O primeiro (p) define a ordem do modelo auto-regressivo e o segundo (q) a ordem do modelo de médias móveis.

Auto-Regressivo Integrado com Médias Móveis (ARIMA)

O modelo ARIMA, proposto em (BOX; JENKINS; REINSEL, 2013), é definido por seis parâmetros (p, d, q) e (P, D, Q). Os primeiros três definem um modelo para os componentes de ruído e tendência da série. Os três últimos são opcionais, e definem um modelo para o componente de sazonalidade. P ou p definem a ordem do modelo auto-regressivo, D ou d definem a ordem da integração (necessária quando a série não é estacionária), e Q ou q definem a ordem do modelo de médias móveis (HERBST *et al.*, 2014). Em (HYNDMAN; KHANDAKAR, 2007) é proposta uma função para estimar de maneira automática esses parâmetros.

Alisamento Exponencial Simples (SES)

Alisamento exponencial é uma técnica que atribui pesos exponencialmente decrescentes ao longo do tempo para as previsões passadas. Desse modo, como os maiores pesos são atribuídos às previsões mais recentes, este modelo é mais sensível a capturar tendências em uma série temporal.

Alisamento Exponencial de Holt-Winters

Este modelo é uma extensão do modelo SES para capturar sazonalidade. Seu trabalho é dividir a série em três componentes: nível, tendência e sazonalidade. Para cada um dos componentes separados é, então, aplicado o SES. Em seguida, a série é reconstruída. Há duas variações para este modelo, aditivo e multiplicativo, que diferem de acordo com o padrão do componente sazonal (HYNDMAN; ATHANASOPOULOS, 2014).

Alisamento Exponencial Estendido (ETS)

O modelo ETS, proposto em (HYNDMAN; KHANDAKAR, 2007), tem por objetivo escolher, de acordo com a série, o melhor modelo entre o SES, Holt-Winters aditivo ou multiplicativo.

2.4 Otimização Multiobjetivo

Analisando o problema de alocação de recursos na nuvem, fica claro que o mesmo envolve a obtenção de duas metas que devem ser atingidas simultaneamente (minimizar, ao mesmo tempo, o subprovisionamento e superprovisionamento de recursos). Essas metas são conflitantes, ou seja, não existe uma solução única que otimize ambas ao mesmo tempo. Para esse tipo de problema, devemos buscar um conjunto de soluções eficientes.

Problemas dessa natureza são chamados de problemas de otimização multiobjetivo, por envolverem minimização (ou maximização) simultânea de um conjunto de objetivos satisfazendo a um conjunto de restrições. Neste caso, a tomada de decisão deverá ponderar os objetivos globais do problema e escolher uma entre as soluções do conjunto de soluções eficientes (ARROYO, 2002).

A otimização multiobjetivo pode ser definida como o problema de achar um vetor de variáveis de decisão cujos elementos representam as funções objetivos. Essas funções formam uma descrição matemática do critério de otimalidade que estão em conflito umas com as outras. Neste caso, o termo “otimizar” significa encontrar um conjunto de soluções que não podem ser melhoradas simultaneamente (COELLO; LAMONT; VELDHUIZEN, 2007).

Diferente da otimização mono-objetivo, onde uma solução ótima é claramente identificada, pois o espaço de solução é ordenado, na otimização multiobjetivo há um conjunto de alternativas, geralmente conhecidas como soluções Pareto-ótimas, que podem ser consideradas soluções eficientes, ou conjunto admissível do problema (FONSECA; FLEMING, 1995).

O conceito de Pareto-ótimo constitui a origem da busca na otimização multiobjetivo. Pela definição, uma solução é considerada Pareto-ótima se não existe outra solução viável que possa melhorar algum objetivo, sem causar uma piora em pelo menos um outro objetivo. Ou

seja, uma solução pertence ao conjunto de soluções Pareto-ótimo se não existe nenhuma outra solução que a domine (PARETO, 1964).

Para exemplificar o conceito de dominância, vamos imaginar um problema de minimização com dois objetivos. Imaginemos que uma solução do problema seja o vetor $X(x_1, x_2)$, e o vetor $Y(y_1, y_2)$ seja outra solução. Podemos dizer que X domina Y se e somente se $x_1 < y_1$ e $x_2 \leq y_2$ ou se $x_1 \leq y_1$ e $x_2 < y_2$. O inverso também é verdadeiro no caso de Y dominar X . Caso contrário, uma solução não pode ser identificada como melhor que a outra, a menos que informações de preferência em relação aos objetivos sejam incluídas.

2.4.1 Métodos de Otimização Multiobjetivo

Os métodos de otimização multiobjetivo tem por objetivo encontrar soluções com prioridades, ou pesos, associados aos objetivos. Esses métodos, geralmente, escalarizam os objetivos formando um único objetivo. Desse modo, o problema multiobjetivo é transformado em um problema mono-objetivo (OSIADACZ, 1989). A seguir, ilustraremos alguns métodos utilizados para resolver esta gama de problemas.

2.4.1.1 Método da soma ponderada

Neste método, o problema multiobjetivo é transformado em um problema mono-objetivo através da atribuição de pesos para cada objetivo. Desse modo, temos uma nova função que representa a relação linear entre todos os objetivos. As equações 2.1 e 2.2 ilustram o método para um problema de minimização.

$$F = \text{minimizar} \sum_{i=1}^n w_i F_i(x) \quad (2.1)$$

sujeito a:

$$x \in X^* \quad (2.2)$$

onde:

n = número de objetivos

F_i = função objetivo i

w_i = peso do objetivo i

x = valor da solução

X^* = valores aceitos para solução

Os pesos w_i devem estar todos em uma mesma escala ou ordem de grandeza para que reflitam a importância dos objetivos. Para tanto, o decisor deve normalizá-los para que cada peso de um objetivo expresse sua importância em relação aos demais.

2.4.1.2 Método ε -restrito

Este método consiste na otimização do objetivo mais importante sujeitando-se às restrições dos outros objetivos. Matematicamente, podemos definir o problema de acordo com as equações 2.3 e 2.4, considerando um problema de minimização.

$$F_1 = \text{minimizar } F_1(x) \quad (2.3)$$

sujeito a:

$$F_i(x) \leq \varepsilon_i, i = 2, 3, \dots, n \quad (2.4)$$

onde:

F_1 = objetivo 1

ε_i = limite superior do objetivo i

n = número de objetivos

Para construir o conjunto Pareto-ótimo deve-se apenas variar o limite superior i, independente se o espaço objetivo for ou não convexo. No entanto, caso o limite escolhido não seja adequado, o subconjunto de possíveis soluções obtido pode ser vazio, ou seja, não existe solução viável.

2.4.1.3 Método de análise hierárquica (AHP)

O Analytic Hierarchy Process (AHP), proposto por Tomas L. Saaty no início da década de 70, é um método de apoio a decisão multicritério muito utilizado na resolução de problemas que envolvem mais de um objetivo.

O AHP busca tratar a complexidade de um problema dividindo-o em fatores e níveis, e estabelecendo relações de prioridade entre esses fatores. Para a aplicação dessa metodologia é necessário estruturar os critérios (considerados no processo de decisão) e as alternativas (objeto da decisão) de forma hierárquica, sendo que no primeiro nível da hierarquia está o propósito geral do problema, no segundo os critérios e no terceiro as alternativas.

Para exemplificar esse processo, vamos considerar que uma determinada família deseja comprar um carro novo. Cada membro da família tem preferência por determinado modelo, de acordo com o critério que mais lhes agrada. O pai prefere o modelo A, porque é mais

econômico. A mãe prefere o modelo B, porque é mais confortável. Já o filho adolescente prefere o modelo C, porque é mais estiloso. Dessa maneira, a família precisa tomar uma decisão que envolve múltiplos critérios e alternativas. A Figura 3 ilustra o problema de maneira hierárquica, como determina o AHP.

Figura 3 – Problema modelado de maneira hierárquica.

Após a definição da hierarquia, o próximo passo é definir as relações de prioridade entre as alternativas, dentro de cada critério, e entre os critérios. Para estabelecer essas relações é necessário comparar par a par as alternativas, dentro do contexto de determinado critério, e definir a importância relativa de uma sobre a outra. Em (SAATY, 1988) é proposto uma tabela de escalas para ser usada na definição das relações de prioridade. A Tabela 1 ilustra as escalas propostas.

Tabela 1 – Escalas de comparação segundo Saaty

Escala	Descrição
1	Elementos tem a mesma importância entre si
3	Um dos elementos tem moderada importância em relação a outro
5	Um dos elementos tem forte importância em relação a outro
7	Um dos elementos tem importância muito forte em relação a outro
9	Um dos elementos tem extrema importância em relação a outro

Incrementos de 0.1 são permitidos entre os valores propostos na Tabela 1, quando se desejar ajustes mais finos.

Baseado nas escalas propostas na Tabela 1, são criadas matrizes de julgamentos para as alternativas, dentro de cada critério, e para os critérios. No nosso exemplo, serão criadas quatro matrizes de julgamentos: uma para definir a relação de importância entre as alternativas dentro do critério “economia”; uma para definir a relação de importância entre as alternativas dentro do critério “conforto”; outra para definir a relação de importância entre as alternativas dentro do critério “estilo”; e outra para definir a relação de importância entre os critérios. As Tabelas 2 e 3 ilustram essas matrizes de julgamento.

Tabela 2 – Matrizes de julgamento das alternativas

Economia	Carro A	Carro B	Carro C	PML
Carro A	1	3	7	0.5889
Carro B	1/3	1	5	0.3391
Carro C	1/7	1/5	1	0.0720
Conforto	Carro A	Carro B	Carro C	PML
Carro A	1	1/5	1/2	0.1211
Carro B	5	1	3	0.6414
Carro C	2	1/3	1	0.2375
Estilo	Carro A	Carro B	Carro C	PML
Carro A	1	2	1/5	0.1615
Carro B	1/2	1	1/9	0.0813
Carro C	5	9	1	0.7572

Tabela 3 – Matriz de julgamento dos critérios

Critério	Economia	Conforto	Estilo	VPC
Economia	1	3	5	0.6414
Conforto	1/3	1	2	0.2375
Estilo	1/5	1/2	1	0.1211

Na Tabela 2, podemos notar que o Carro A tem importância moderadamente superior (3) ao Carro B e muito fortemente superior (7) ao Carro C no critério “economia”. Desse modo, a importância do Carro B e do Carro C sobre o Carro A, no mesmo critério, é de 1/3 e 1/7 respectivamente, ou seja, o inverso. Essa regra vale para a criação de todas as matrizes de julgamento. Na Tabela 3 vemos que “economia” é moderadamente (3) mais importante que “conforto” e fortemente (5) mais importante que “estilo”. Já “conforto” é levemente (2) melhor que “estilo”.

Após a criação das matrizes de julgamento, é necessário calcular as prioridades médias locais (PML's) e o vetor de prioridades dos critérios (VPC). Esse cálculo é feito somando os elementos de cada linha da matriz e dividindo pela soma de todos os elementos da matriz. Por fim, o cálculo das prioridades globais é efetuado, multiplicando os PML's pelo VPC. O vetor de prioridades global (PG), armazena a prioridade associada a cada alternativa em relação ao foco principal. A Tabela 4 ilustra os PML's, o VPC e o PG para o nosso problema.

Tabela 4 – Vetor de prioridades global

	PML 1	PML 2	PML 3		VPC		PG
A	0.5889	0.1211	0.1615	x	0.6414	=	0.4260
B	0.3391	0.6414	0.0813		0.2375		0.3797
C	0.0720	0.2375	0.7572		0.1211		0.1943

De acordo com a Tabela 4, o carro A teve o maior índice no vetor de prioridades global (PG). Ou seja, em uma escala de 0 a 1, ele satisfaz 0.4260 dos anseios da família. Desse modo ele será o carro escolhido para compra.

2.4.1.4 Algoritmo Genético Multiobjetivo

Algoritmos genéticos (AGs) utilizam conceitos do princípio da seleção natural para lidar com uma ampla gama de problemas, em particular otimização. Eles são robustos, genéricos e facilmente adaptáveis. Inspirado pela forma como o *darwinismo* explica o processo de evolução das espécies, o AG está dividido nas seguintes etapas: inicialização, avaliação, seleção, cruzamento, mutação, atualização e finalização. A figura 4 ilustra essas etapas.

Basicamente, o que um algoritmo genético faz é criar uma população de possíveis respostas para o problema a ser tratado, e, em seguida, submetê-lo ao processo de evolução. A seguir, será descrito cada passo:

- **Inicialização:** é criada uma população inicial com indivíduos (possíveis soluções) candidatos a solucionar o problema;
- **Avaliação:** a capacidade das soluções é avaliada (indivíduos da população) por meio de uma análise a fim de determinar quais os indivíduos são mais aptos dentro da população (melhor solução para o problema);
- **Seleção:** os indivíduos são selecionados para a reprodução. A probabilidade de um indivíduo ser selecionado é proporcional à sua aptidão;
- **Cruzamento:** características das soluções escolhidas são recombinadas, gerando novos indivíduos;
- **Mutação:** características dos indivíduos resultantes do processo de reprodução são alteradas, aumentando assim a variedade da população;
- **Atualização:** indivíduos criados nessa geração são inseridos na população;
- **Finalização:** é verificado se foi alcançado as condições para o fim da evolução. Em caso positivo, a execução é finalizada. Caso contrário, a execução retorna para a fase de avaliação.

Figura 4 – Etapas de um algoritmo genético.

Os indivíduos são a unidade fundamental de um algoritmo genético: eles codificam possíveis soluções para o problema a ser tratado, e é através da sua manipulação (pelo processo de evolução) que as respostas são encontradas. A escolha de representação para os indivíduos é o passo mais importante no desenvolvimento de um AG, uma vez que essa representação terá a responsabilidade principal no desempenho do programa.

A principal diferença entre um AG multiobjetivo e um AG mono-objetivo está na forma como é atribuído a aptidão (*fitness*) às soluções. No caso de um AG mono-objetivo, é possível identificar uma única solução como sendo a mais apta. No caso do AG multiobjetivo existem várias soluções que podem ser consideradas, do ponto de vista matemático, como sendo igualmente aptas. Assim, é necessário definir algumas estratégias para calcular o *fitness* das soluções e para selecionar as soluções com maior probabilidade de gerarem indivíduos (soluções)

melhores por meio da reprodução.

Existem diversas implementações de AGs para problemas multiobjetivos: VEGA (Vector Evaluated Genetic Algorithm) (SCHAFFER, 1985); MOGA (Multiobjective Genetic Algorithm) (FONSECA; FLEMING *et al.*, 1993); NPGA (Niche Pareto Genetic Algorithm) (HORN; NAFPLIOTIS; GOLDBERG, 1994); NSGA (Non-dominated Sorting Genetic Algorithm) (SRINIVAS; DEB, 1994); SPEA (Strength Pareto Evolutionary Algorithm) (ZITZLER, 1999); e o NSGA II (Non-dominated Sorting Genetic Algorithm II) (DEB *et al.*, 2000). Neste trabalho utilizaremos uma solução baseada no NSGA II.

O NSGA II foi proposto por Deb *et al.* (2000), baseado em uma ordenação elitista por dominância, ou *ranking* de Pareto. Esse procedimento de ordenação, conhecido como *fast non dominated sorting*, consiste em classificar os indivíduos de um conjunto em diversos níveis, de acordo com o grau de dominância de tais indivíduos. Assim, o primeiro nível contém os indivíduos dominantes (não dominados) de todo o conjunto. O segundo nível possui os indivíduos dominados apenas pelos indivíduos do primeiro nível, e assim sucessivamente.

Para escolher qual indivíduo deve ser utilizado para gerar descendentes na nova população, o NSGA II emprega um processo de seleção por torneio, considerando que a aptidão (fitness) de cada indivíduo depende da fronteira à qual esse pertence. O restante do algoritmo funciona de maneira similar a um algoritmo genético tradicional.

2.5 Considerações Finais

Nesse capítulo foram abordados os tópicos relacionados ao desenvolvimento deste trabalho. Destaque para as características das aplicações elásticas hospedadas na nuvem, dos modelos de previsão de séries temporais, e dos métodos de otimização multiobjetivo. No próximo capítulo, uma revisão bibliográfica será apresentada, abordando trabalhos relacionados a esta Tese.

TRABALHOS RELACIONADOS

3.1 Considerações Iniciais

Neste capítulo é apresentada uma revisão da literatura sobre técnicas de *autoscaling*. A revisão é inicialmente dividida em duas categorias: reativas; e proativas. Cada uma dessas categorias será dividida em subcategorias, de acordo com a técnica de *autoscaling* utilizada: *threshold*; teoria de controle; aprendizado de máquina; e modelos de previsão de séries temporais. Em seguida, analisaremos trabalhos desenvolvidos em cada subcategoria, apontando suas vantagens e desvantagens.

3.2 Técnicas Reativas

Nessa categoria de técnicas, o sistema reage, apenas, quando detecta alguma mudança em relação a carga de trabalho. Para tanto, algumas variáveis (uso da CPU, uso da memória, tamanho da fila, etc.) são monitoradas continuamente. Quando os valores de tais variáveis atingem valores indesejados, alguma ação é tomada para tentar manter a estabilidade do sistema. O problema das técnicas reativas é que, frequentemente, o tempo para reagir é insuficiente. Isso ocorre porque o tempo para um recurso na nuvem (nesse caso, uma máquina virtual) estar pronto para uso após ter sido solicitado pode levar minutos (LORIDO-BOTRAN; MIGUEL-ALONSO; LOZANO, 2014), tempo suficiente para que o sistema fique sobrecarregado. A seguir, apresentaremos trabalhos relacionados a duas técnicas pertencentes a essa categoria: *threshold*; e teoria de controle.

3.2.1 Técnicas baseadas em *threshold*

Regras baseadas em limites (*threshold*) constituem uma maneira fácil de implementar um *autoscaling* capaz de adaptar dinamicamente os recursos para a demanda de entrada. No entanto, a criação de regras exige um esforço do gerenciador da aplicação, que precisa selecionar

a métrica adequada de desempenho ou a combinação lógica de métricas, além dos valores de vários parâmetros, principalmente limiares (*threshold*).

Os experimentos realizados em (KOPEREK; FUNIKA, 2012) mostram que os indicadores específicos da aplicação (por exemplo, o tempo médio de espera em fila) apresentam um melhor desempenho que os indicadores específicos do sistema (por exemplo, a carga da CPU). O Gerenciador da aplicação precisa definir o limite superior (por exemplo, 70%) e inferior (por exemplo, 30%) para a variável de desempenho (por exemplo, a carga da CPU). Esses limiares são a chave para o funcionamento correto das regras. Nesse caso, os autores propõem um *autoscaling* reativo, baseado em regras, com o objetivo de prover elasticidade horizontal, que utiliza como métricas a carga da CPU e o tempo médio de espera em fila. Os autores não deixam claro qual o SLA considerado. Os experimentos foram conduzidos em uma nuvem pública usando carga gerada de forma sintética.

De acordo com Dutreilh *et al.* (2010), os *thresholds* devem ser cuidadosamente ajustados, a fim de evitar as oscilações no sistema (por exemplo, no número de máquinas virtuais ou na quantidade de CPUs alocadas). Para evitar esse problema, é aconselhável definir um tempo de espera. Durante esse tempo não há decisões de aumento ou diminuição dos recursos do sistema.

A maioria dos trabalhos e provedores de nuvem usam apenas dois limiares (*thresholds*) por métrica de desempenho. No entanto, em (HASAN *et al.*, 2012) um conjunto de quatro limiares são considerados: *ThrU*, o limite superior; *ThrbU*, que é ligeiramente inferior ao limite superior; *ThrL*, o limite inferior; *ThroL*, que é ligeiramente superior ao limite inferior. Usando essa combinação é possível determinar a tendência da métrica de desempenho (por exemplo, tendendo para cima ou para baixo), e, em seguida, executar decisões de *autoscaling* mais precisas. As métricas utilizadas foram: a carga da CPU; o tempo de resposta; a carga na rede; o *jitter* e o *delay*. Os autores não deixam claro qual o SLA considerado e nem como o sistema foi monitorado. Também não foi descrito nenhum experimento, apenas o algoritmo foi descrito.

As condições estabelecidas nas regras são geralmente baseadas em uma única ou no máximo em duas métricas de desempenho, sendo as mais populares: a carga média da CPU nas VMs, o tempo de resposta, e a taxa de chegada das requisições. Ambos, Dutreilh *et al.* (2010) e Han *et al.* (2012) usam o tempo médio de resposta da aplicação. Por outro lado, Hasan *et al.* (2012) preferem usar as métricas de desempenho de vários domínios (computação, armazenamento e rede) ou mesmo uma correlação de várias delas. Em (HAN *et al.*, 2012), os autores propõem um *autoscaling* reativo, baseado em regras, com o objetivo de prover elasticidade horizontal e vertical, que utiliza como métricas a carga da CPU, uso da memória e I/O. O sistema é monitorado a cada minuto e o SLA considerado é o tempo de resposta. Os experimentos foram conduzidos por meio de um *testbed* customizado, submetido a uma carga gerada de forma sintética.

Na maioria dos casos, as regras usam os valores das métricas diretamente, tal como obtidos a partir do monitor, atuando, assim, de uma forma puramente reativa. No entanto, é

possível realizar uma análise prévia dos dados monitorados, a fim de prever o comportamento futuro (esperado) do sistema e executar regras de uma forma proativa (KHATUA; GHOSH; MUKHERJEE, 2010). Este tópico será discutido mais adiante, na seção dedicada à análise de séries temporais.

O algoritmo de *autoscaling* da RightScale (CLARK, 2010) se propõe a combinar as regras reativas por meio de um processo de votação. Se a maioria das VMs concordarem que devem aumentar ou diminuir a capacidade do sistema, então são tomadas medidas para tal fim; caso contrário, nenhuma ação é tomada. Cada voto das VMs para aumentar ou diminuir a capacidade do sistema é baseado em um conjunto de regras avaliadas individualmente. Depois de cada ação que aumenta ou diminui a capacidade do sistema, o RightScale recomenda um período de inatividade, devido ao fato de uma nova máquina levar, geralmente, entre 5 a 10 minutos para estar operacional (LORIDO-BOTRAN; MIGUEL-ALONSO; LOZANO, 2014). Esta técnica de *autoscaling* tem sido adotada por vários autores. Chieu *et al.* (2009) propôs inicialmente um conjunto de regras reativas com base no número de sessões ativas, mas este trabalho foi estendido em (CHIEU *et al.*, 2011) seguindo a abordagem do RightScale: se todas as VMs têm sessões ativas acima do limite superior determinado, uma nova VM é provisionada; se houver VMs com sessões ativas abaixo do limite inferior determinado, e pelo menos uma VM que não tem nenhuma sessão ativa, então uma VM será liberada. Os experimentos foram conduzidos por meio de um *testbed* customizado submetido a uma carga gerada de forma sintética.

Segundo Kupferman *et al.* (2009), a principal desvantagem do RightScale se deve ao fato do seu sistema de votação ser baseado em regras, e, dessa maneira, se torna altamente dependente dos valores de limiares definidos pelo gerente da aplicação, e também, das características da carga de trabalho de entrada. Simmons *et al.* (2011) tentam superar esses problemas por meio de uma árvore de estratégia, a qual avalia o conjunto de políticas implantadas e alterna entre estratégias alternativas ao longo do tempo, de uma maneira hierárquica. Os autores criaram políticas de escalonamento diferentes, personalizadas para diferentes cargas de trabalho. O objetivo da árvore de estratégias é alternar entre essas políticas, com base na tendência da carga de trabalho (analisadas com uma técnica baseada em regressão). O sistema era monitorado a cada 4 minutos. Os experimentos foram conduzidos por meio do Amazon EC2 submetido a uma carga gerada por meio do *log* FIFA World Cup 1998 (CUP, 1998).

A fim de reduzir custos, Casalicchio e Silvestri (2013) e Menasce *et al.* (2004) introduziram a idéia de “matança inteligente”. Muitos provedores de IaaS cobram horas de utilização parcial como horas cheias. Portanto, é aconselhável não finalizar uma VM antes de completar uma hora de uso, mesmo se a carga estiver baixa. Além de reduzir custos, a “matança inteligente” pode também melhorar o desempenho do sistema: no caso de uma carga de trabalho que oscila muito (comum em aplicações Web), os custos ao desligar e reiniciar VMs são evitados, além de melhorar o tempo de resposta e reduzir violações de SLA.

A popularidade das técnicas baseadas em *threshold* se deve a sua simplicidade e ao fato de

serem fáceis de entender pelos usuários. No entanto, esse tipo de técnica mostra dois problemas principais: a dificuldade de selecionar um conjunto adequado de métricas de desempenho e seus limites correspondentes; e a eficácia desses limites, que é altamente dependente das alterações de carga de trabalho, e pode exigir ajustes frequentes. A fim de resolver o problema dos limites estáticos, [Lorido-Bostrán, Miguel-Alonso e Lozano \(2013\)](#) introduziram o conceito de limites dinâmicos. Nesse caso, são definidos valores iniciais, mas eles podem ser automaticamente alterados em consequência das violações de SLAs observados. Meta-regras são utilizadas para definir como o limiar (*threshold*) utilizado nas regras de *autoscaling* pode mudar para melhor se adaptar à carga de trabalho. O sistema é monitorado a cada minuto. Os experimentos foram conduzidos por meio de um simulador customizado submetido a uma carga baseada no *log* ClarkNet ([CLARKNET, 1995](#)).

Em ([GHANBARI et al., 2011](#)), os autores propõem um *autoscaling* reativo usando o RightScale, com o objetivo de prover elasticidade horizontal. A métrica utilizada é a carga na CPU. Os experimentos foram conduzidos por meio do Amazon EC2 submetido a uma carga gerada por meio do *log* FIFA World Cup 1998 ([CUP, 1998](#)). O sistema foi monitorado usando o Amazon CloudWatch ([CLOUDWATCH, 2013](#)).

Em ([MAURER; BRANDIC; SAKELLARIOU, 2011](#)), os autores propõem um *autoscaling* reativo, baseado em regras, com o objetivo de prover elasticidade vertical, que utiliza como métricas a carga da CPU, uso da memória e I/O. Os experimentos foram conduzidos por meio de um simulador customizado submetido a uma carga gerada de forma sintética.

Desse modo, podemos concluir que, devido a sua simplicidade, as técnicas baseadas em *threshold* podem ser usadas para automatizar facilmente o processo de *autoscaling* de uma aplicação específica, especialmente no caso de aplicações com padrões previsíveis e bastante regulares. No entanto, em caso de cargas de trabalho em rajadas, ou com muitas variações (comum em aplicações Web), seria interessante considerar uma técnica de *autoscaling* mais avançada.

3.2.2 Teoria de Controle

Os tipos de controles mais simples e mais utilizados na literatura são os controles do tipo *fixed-gain*, incluindo PID (Proportional–Integral–Derivative), PI (Proportional–Integral) e I (Integral). Em ([LIM et al., 2009](#)) e ([LIM; BABU; CHASE, 2010](#)), os autores propõem um controle do tipo I para ajustar o número de VMs, tendo como base a taxa de utilização média da CPU. Os parâmetros de ganho K_p e K_i foram definidos manualmente, usando o método de tentativa e erro. Os experimentos foram realizados por meio de um *testbed* customizado usando carga de trabalho sintética. Os autores não deram maiores informações a respeito dos SLAs considerados. Em ([PARK; HUMPHREY, 2009](#)), os autores usam um controle do tipo PI para gerenciar os recursos necessários para execução de *jobs* do tipo *batch*. O progresso de execução de tais *jobs* é utilizado como variável controlada. Os parâmetros de ganho K_p e K_i

foram definidos usando um modelo de aplicação específico. O monitoramento do sistema foi feito por meio de uma biblioteca de sensores e o SLA considerado foi o deadline para os *jobs*. Os experimentos foram realizados por meio de um *testbed* customizado. Em (ZHU; AGRAWAL, 2010), um agente do tipo RL é usado para estimar o termo derivado de um controle do tipo PID. O agente RL é treinado por meio do método de tentativa e erro.

Técnicas de controle adaptativo também são bastante populares. Em (ALI-ELDIN; TORDSSON; ELMROTH, 2012), os autores propõem a combinação de dois controles adaptativos e proativos para decidir sobre a liberação de recursos (*scaling down*), usando parâmetros de ganho dinâmico, baseado na carga que chega ao sistema. Uma abordagem reativa é usada para decidir sobre o aumento dos recursos (*scaling up*). Os experimentos foram realizados por meio de um simulador customizado e carga de trabalho real extraída do log FIFA World Cup 1998. Os mesmos autores propõem em (ALI-ELDIN *et al.*, 2012) um controle adaptativo e proporcional, usando uma abordagem proativa, tanto para *scaling up* como para *scaling down*, levando em conta o tempo de inicialização da VM. Os experimentos, novamente, foram realizados por meio de um simulador customizado, e a carga de trabalho considerada, além do log FIFA World Cup 1998, foi o log Google Cluster Data (WILKES, 2011). Em (PADALA *et al.*, 2009) é proposto um controle adaptativo do tipo MIMO que usa um modelo ARMA de segunda ordem para modelar o relacionamento entre a alocação de recursos e o desempenho normalizado do sistema. O controle é capaz de ajustar o uso da CPU e de dispositivos de I/O, como o disco. O sistema é monitorado a cada 20 segundos e o SLA considerado foi o tempo de resposta. Os experimentos foram realizados por meio de um *testbed* customizado e carga de trabalho sintética. Em (BODIK *et al.*, 2009) os autores combinam *smoothing splines* (usado para mapear a relação entre carga de trabalho e número de servidores necessários para manter o desempenho do sistema) com um controle de escalonamento de ganho adaptativo. O SLA considerado foi o tempo de resposta e os experimentos foram realizados por meio do provedor Amazon EC2 (AMAZON, 2010), usando carga de trabalho sintética.

Kalyvianaki, Charalambous e Hand (2009) propuseram diferentes controles do tipo SISO e MIMO para determinar a alocação de CPUs dentro das VMs, usando filtro de kalman. O SLA considerado foi o tempo de resposta das requisições e os experimentos realizados em um *testbed* customizado por meio de carga de trabalho sintética. Gambi e Toffetti (2012) utilizaram um modelo *Kriging* para prever o tempo de término de um *job* em função do número de VMs, do número de requisições que chegam ao sistema e do número de *jobs* na fila. Os experimentos foram realizados em um *testbed* customizado, em uma nuvem privada, por meio de carga de trabalho sintética.

Modelos baseados em lógica fuzzy tem sido usados como modelos de desempenho em sistemas de controle visando relacionar a carga de trabalho (variável de entrada) e os recursos necessários (variável de saída). Nesse caso, as variáveis de entrada e saída do sistema são mapeadas em conjuntos *fuzzy*. Esse mapeamento é definido por uma função que determina um

valor dentro do intervalo $[0,1]$. Um modelo *fuzzy* é baseado em um conjunto de regras que relaciona as variáveis de entrada (precondição da regra) com as variáveis de saída (consequência da regra). O processo de tradução das variáveis de entrada em conjuntos *fuzzy* é chamado de *fuzzificação*. Por outro lado, a *defuzzificação* é o processo inverso, onde as variáveis *fuzzy* são transformadas em variáveis de saída. Um sistema de controle baseado no modelo *fuzzy* é chamado de controle *fuzzy*. Geralmente, o conjunto de regras e a função de transformação de um modelo *fuzzy* são definidas em tempo de projeto. Dessa maneira, o controle terá dificuldades de se adaptar a cargas altamente dinâmicas. Para lidar com esse problema, [Xu et al. \(2007\)](#) propuseram um controle *fuzzy* adaptativo para a camada de negócio de uma aplicação web, e estimou a capacidade da CPU necessária para atender a carga de trabalho que chega ao sistema. O SLA considerado foi o tempo de resposta e os experimentos foram realizados em um *testbed* customizado, utilizando cargas de trabalho real (log FIFA World Cup 1998) e sintética. Uma proposta parecida é apresentada em ([WANG et al., 2011b](#)), com a diferença que, neste caso, os autores focam na camada de banco de dados.

[Lama e Zhou \(2010\)](#) propõem um controle *fuzzy* baseado em rede neural, que é capaz de mudar sua estrutura (tanto em relação as regras *fuzzy* como em relação a função de mapeamento) e se adaptar em tempo de execução. Os experimentos foram conduzidos por meio de simulação, utilizando carga de trabalho sintética representada por uma distribuição de Pareto. [Roy, Dubey e Gokhale \(2011\)](#) combinaram um modelo ARMA para previsão da carga de trabalho, com um controle do tipo *look-ahead* para otimizar o problema de alocação de recursos. Modelos *fuzzy* também podem ser usados para criar um modelo preditivo ([WANG et al., 2011a](#)).

A adequação de controles para a tarefa de *autoscaling* é altamente dependente do tipo de controle e da dinâmica do sistema alvo. A idéia de ter um controle que automatiza o processo de alocação de recursos é muito atraente. No entanto, não é fácil criar um modelo de desempenho confiável, que mapeia as variáveis de entrada e saída. Controles reativos simples poderiam ser usados em aplicações onde é fácil prever suas necessidades. Por outro lado, tais modelos pecam quando o desejo é produzir uma solução de *autoscaling* genérica.

3.3 Técnicas Proativas

3.3.1 Aprendizado por reforço

Três elementos básicos devem ser considerados para aplicar técnicas de aprendizado por reforço (RL) em um *autoscaling*: o conjunto de ações A , o espaço de estados S , e a função de recompensa R ([LORIDO-BOTRAN; MIGUEL-ALONSO; LOZANO, 2014](#)). Os primeiros dois são altamente dependentes do tipo de *scaling*, horizontal ou vertical, enquanto que a função de recompensa, geralmente, leva em conta o custo de adquirir recursos (alocar VMs, por exemplo) e o custo das penalidades devido ao não cumprimento dos SLAs. No caso do *scaling* horizontal, o estado é comumente definido em termos da carga de entrada do sistema e do número de

VMs. Em (TESAURO *et al.*, 2006), os autores propõem usar $(w, ut-I, ut)$, onde w é o total de requisições observadas por período de tempo; ut e $ut-I$ são o número de VMs alocadas para a aplicação no atual período de tempo, e no intervalo anterior de tempo, respectivamente. O SLA considerado foi o tempo de resposta das requisições. Os experimentos foram conduzidos por meio de um *testbed* customizado e carga de trabalho sintética, baseadas nas distribuições de Poisson e Exponencial. Dutreilh *et al.* (2011) consideraram a definição de estado (w, u, p) , onde p é o desempenho em termos do tempo médio de resposta para as requisições, limitado por um valor P_{max} escolhido a partir de observações experimentais. Para executar os experimentos foi utilizado um *testbed* customizado e carga de trabalho sintética.

Os estados de ações para o *scaling* horizontal são geralmente três: adicionar uma nova VM, remover uma VM existente, ou não fazer nada. Por outro lado, as definições de estado para o *scaling vertical* levam em conta a quantidade de recursos associados a cada VM (CPU e memória, geralmente). Em (RAO *et al.*, 2009) e (XU; RAO; BU, 2012) os autores consideraram a seguinte definição de estado: $(memI, timeI, vcpuI, \dots, memi, timei, vcpi)$, onde $memi$, $timei$ e $vcpi$ são os i -ésimos tamanhos da memória, tempo de escalonamento e o número de CPUs virtuais da VM, respectivamente. As operações podem aumentar, diminuir ou manter o mesmo valor para cada uma das três variáveis. As ações para a tarefa de aprendizado por reforço são definidas como combinações de operações em cada variável. Devido ao fato das variáveis serem contínuas, as operações são discretizadas: a memória é reconfigurada em blocos de 256MB; mudanças no tempo de escalonamento em 256 unidades; e o número de CPUs virtuais em 1 unidade. Os experimentos foram executados por meio de um *testbed* customizado e carga de trabalho sintética, baseada em requisições do tipo *e-commerce*. Em (RAO *et al.*, 2011) é proposta uma abordagem distribuída, na qual existe um agente RL para cada VM. Neste caso, o estado é configurado como: CPU; bandwidth; memória; e swap. Os experimentos foram conduzidos por meio de um *testbed* customizado e carga de trabalho real, baseada no *log* Clarknet (CLARKNET, 1995).

A técnica de aprendizado por reforço apresenta alguns problemas, dentre os quais podemos destacar: desempenho insatisfatório nos primeiros passos e longo tempo de treinamento; espaço de estados excessivamente grande; e dificuldade para se adaptar a ambientes dinâmicos. Por outro lado, as vantagens são: essa técnica não exige nenhum conhecimento *a priori*; o cliente não precisa definir nenhum modelo específico para a aplicação; o aprendizado ocorre em tempo de execução. Devido a essas características, técnicas baseadas em RL pode ser uma abordagem útil para resolver o problema de construir um *autoscaling* genérico, especialmente se combinada com outras técnicas.

3.3.2 Séries Temporais

As técnicas de *autoscaling* baseadas em séries temporais podem ser usadas tanto para *autoscaling* vertical como para horizontal. Dutta *et al.* (2012) mostram que o *autoscaling* vertical

é mais limitado do que o horizontal, devido ao fato da quantidade de recursos possível de ser manipulada ser menor nesse tipo de *autoscaling*. Por outro lado, o *autoscaling* horizontal possui um custo maior para manipular tais recursos. Fang *et al.* (2012) usam um *scaling* vertical (CPU e memória) para regular mudanças na carga de trabalho, e *scaling* horizontal para manipular altas variações na carga.

Huang, Li e Yu (2012) apresentam um modelo de previsão de recursos (para utilização de CPU e memória) baseado em alisamento exponencial duplo, e compara com os modelos de média móvel (MA) e média móvel ponderada (WMA). O Alisamento Exponencial apresentou os melhores resultados. Mi *et al.* (2010) também usou alisamento exponencial duplo para prever a carga de trabalho que chega ao sistema, usando carga de trabalho real (*logs* FIFA World Cup 98 e ClarkNet). O modelo apresentou bons resultados para série ClarkNet e não tão bons para a série FIFA World Cup 98.

Roy, Dubey e Gokhale (2011) aplicaram o modelo autoregressivo (AR) para previsão da carga de trabalho, se baseando nas três últimas observações. O valor previsto é usado para estimar o tempo de resposta. Em seguida, um modelo de otimização usa o tempo de resposta para determinar a melhor alocação de recursos, considerando os acordos de níveis de serviço (SLAs). Kupferman *et al.* (2009) aplicaram um modelo AR de ordem 1 para prever a taxa de chegada de requisições. Os resultados mostraram que o desempenho da previsão depende de vários fatores, como: intervalo de monitoramento; e tamanho da série histórica.

Uma rede neural também pode ser usada na previsão de séries temporais (ISLAM *et al.*, 2012), (PRODAN; NAE, 2009). Nesse caso, o desempenho da previsão depende do tamanho da janela da série histórica. Outro parâmetro importante é o intervalo de previsão. Islam *et al.* (2012) propõem um intervalo de 12 minutos. Esse intervalo foi considerado porque o tempo de inicialização de uma VM na nuvem, geralmente, varia entre 5 à 15 minutos (LORIDO-BOTRAN; MIGUEL-ALONSO; LOZANO, 2014). Prodan e Nae (2009) usam uma rede neural para prever a carga de um jogo (número de jogadores, por exemplo) nos próximos dois minutos. A rede neural obteve melhores resultados do que o modelo de média móvel e alisamento exponencial simples. O problema do uso de redes neurais é que é necessária uma fase prévia de treinamento, com uma quantidade significativa de dados. Além disso, em cargas altamente dinâmicas, é necessário que a rede seja constantemente retreinada para manter um desempenho razoável, o que não é uma situação viável.

Outras técnicas de *autoscaling* se baseiam na identificação de padrão nas séries temporais (CARON; DESPREZ; MURESAN, 2010), (CARON; DESPREZ; MURESAN, 2011), (GONG; GU; WILKES, 2010), (SHEN *et al.*, 2011). Gong, Gu e Wilkes (2010) propõem o uso da transformada rápida de *fourier* (FFT) para identificar padrões repetidos na carga dos recursos (CPU, memória, I/O e rede), e compara os valores por meio de auto-correlação, auto-regressão e histograma. Em (CARON; DESPREZ; MURESAN, 2010) e (CARON; DESPREZ; MURESAN, 2011), a técnica proposta possui um grande número de parâmetros e requer muito tempo para

explorar o espaço de busca (série temporal), o que afeta o desempenho do algoritmo. Técnicas baseadas em histograma também tem sido usadas para prever a quantidade de recursos para as aplicações, considerando a média (CHANDRA; GONG; SHENOY, 2003) ou a frequência (GONG; GU; WILKES, 2010) da distribuição.

Fernandez *et al.* (2014) apresentam um autoscaling que suporta infraestruturas de nuvem heterogêneas. Uma árvore de decisão foi usada para determinar a melhor combinação de recursos entre as diferentes infraestruturas. Herbst *et al.* (2014) propõem um classificador para selecionar dinamicamente o melhor método de previsão para determinado tipo de série. Baseada nos objetivos dos usuários, uma árvore de decisão define o melhor método de previsão por meio de mecanismos de *feedback* que avaliam e comparam a acurácia de diferentes métodos de previsão. Balaji *et al.* (2014) apresentaram um estudo comparativo sobre o desempenho de dois modelos de previsão (Holt-Winters e ARIMA) usando uma carga de trabalho extraída dos servidores Web da NASA (NASA, 1996).

Técnicas baseadas em análise de séries temporais são muito usadas e atrativas para implementar *Autoscalings* proativos. Prevendo a carga de trabalho que chegará ao sistema, é possível lidar com o atraso existente entre a solicitação de uma VM e essa estar pronta para uso. Apesar do potencial dessa técnica na previsão de recursos, existem alguns pontos negativos que precisam ser considerados, como: a acurácia da previsão, que é altamente dependente do tipo de aplicação; o tipo de série temporal; a métrica selecionada; o intervalo de previsão; e o modelo de previsão utilizado. Atualmente, esforços tem sido realizados para construir uma ferramenta que automatize a escolha do melhor modelo de previsão para determinada aplicação ou para um conjunto de aplicações.

3.4 Considerações Finais

Nesse capítulo foi apresentada uma revisão da literatura sobre técnicas de *autoscaling*, divididas em várias categorias. Foram analisados trabalhos relevantes dentro de cada categoria, apontando suas vantagens e desvantagens. Um problema recorrente, e que abrange a quase totalidade dos trabalhos, é alta dependência do *autoscaling* em relação a aplicação alvo. Isso dificulta o uso da solução de maneira genérica. No próximo capítulo será apresentada a nossa solução para o problema, que objetiva a criação de um *autoscaling* genérico, baseado na combinação de modelos de previsão de séries temporais, por meio de otimização multiobjetivo.

MODELAGEM DO SISTEMA

4.1 Considerações Iniciais

Neste capítulo, é abordada a proposta deste trabalho que é a construção de um *autoscaling* genérico, capaz de se adaptar a diferentes tipos de demandas e diferentes objetivos dos usuários. Para alcançar esse objetivo usaremos uma combinação de modelos de previsão de séries temporais, por meio de métodos de otimização multiobjetivo. Assim, quatro desses métodos serão utilizados: soma ponderada; ϵ -restrito; análise hierárquica; e algoritmo genético multiobjetivo.

4.2 Autoscaling proposto

Uma típica arquitetura de aplicação Web hospedada na nuvem é mostrada na Figura 5. Nela, podemos notar um *Front-End* que é responsável por receber as requisições dos clientes e distribuí-las entre os servidores no *Back-End*. Cada servidor é uma máquina virtual (*Virtual Machine* - VM) alugada junto a um provedor de infraestrutura na nuvem. Para modelar a arquitetura apresentada utilizamos um modelo de fila $M/M/m$, baseado na teoria de filas (PENTA *et al.*, 2001).

O problema a ser resolvido é encontrar o número mínimo de servidores (m) para atender as requisições que chegam ao sistema, dentro do tempo limite prometido aos clientes. A ferramenta utilizada para atingir este objetivo é chamada de autoscaling. De acordo com a Figura 5, no passo 1, os clientes enviam requisições para o sistema. As requisições são recebidas por um balanceador de carga, que as distribuem entre os servidores virtuais. Em seguida, no passo 2, as requisições são registradas em um arquivo ou banco de dados, chamado de *log*. Em 3, o módulo de previsão de demanda do autoscaling extrai as informações do *log*, cria uma série temporal e faz a previsão de demanda para o próximo intervalo de tempo. Na sequência, em 4, a previsão é passada ao módulo de alocação de recursos, que calcula a quantidade de recursos necessária para

Figura 5 – Aplicação Web hospedada na infraestrutura da nuvem.

atender a demanda e, no passo 5, solicita esses recursos ao provedor de infraestrutura na nuvem.

O *Autoscaling* apresentado na Figura 5 é proativo, pois ele trabalha prevendo a demanda, a cada intervalo de tempo predefinido, para alocar os recursos necessários antes que o sistema fique sobrecarregado. O intervalo de tempo, também chamado de intervalo de reconfiguração, em que o autoscaling atua para alocar ou desalocar recursos foi configurado para uma hora. Esse valor foi escolhido porque a maioria dos provedores de infraestrutura na nuvem cobram pelos recursos alugados considerando o tempo mínimo de uma hora de uso. Assim sendo, para aproveitar ao máximo o investimento, é necessário planejar os recursos que serão usados a cada hora.

4.2.1 Previsão da demanda

Para prever a demanda é necessário utilizar algum modelo de previsão de séries temporais. O problema é que escolher esse modelo não é uma tarefa fácil, já que essa escolha está intimamente ligada às características da série em questão. Além disso, em uma aplicação Web, é comum existirem variações nas características da demanda ao longo do tempo, o que faz com que um modelo que estava prevendo bem, passe a não ser mais adequado.

Para solucionar o problema apresentado, propomos, neste trabalho, combinar modelos de previsão de séries temporais, com o objetivo de otimizar a previsão da demanda e, consequentemente, a alocação de recursos. Consideramos seis modelos de previsão: NAIVE; AutoRegressive

(AR); Moving Average (MA); Autoregressive–Moving–Average (ARMA); AutoRegressive Integrated Moving Average (ARIMA); e Extended Exponential Smoothing (ETS). Todos esses modelos foram descritos no capítulo 2.

Para atribuir pesos adequados a cada modelo propomos, neste trabalho, o uso de técnicas de otimização multiobjetivo. Consideramos seis modelos em nosso trabalho: NAIVE; AR(1); MA(1); ARMA(1,1); ARIMA; e ETS. Para implementar esses modelos nós usamos as funções *arima(p, d, q)*, *auto.arima()* e *ets()* do pacote *forecast* (HYNDMAN; KHANDAKAR, 2007) do software estatístico R.

Nosso objetivo é encontrar a importância de cada modelo de previsão (alternativa) de acordo com os critérios de subprovisionamento e superprovisionamento de recursos. A melhor solução será a que encontrar o melhor *trade-off* para esses dois critérios. Recursos subprovisionados (provisionados a menos do que o necessário) fazem com que o servidor não consiga atender todas as requisições dentro do tempo estabelecido nos acordos de níveis de serviço, o que gera multas e má reputação para a empresa. Por outro lado, recursos superprovisionados (provisionados a mais do que o necessário) geram gastos desnecessários para a empresa. A seguir serão descritas as técnicas de otimização multiobjetivo utilizadas neste trabalho.

4.2.1.1 Método da soma ponderada

O objetivo da programação linear é otimizar, por meio de um modelo matemático, a solução de problemas cuja função objetivo e restrições sejam lineares. A solução pode ser maximizar ou minimizar a função objetivo, respeitando as restrições do problema. As restrições do modelo determinam o conjunto de soluções viáveis para o problema. A melhor destas soluções é a solução ótima.

Quando um problema possui mais de um objetivo a ser otimizado, é necessária a criação de um modelo com múltiplas funções objetivo. Problemas reais são comumente multiobjetivos e, muitas vezes, esses objetivos são conflitantes. Por exemplo, queremos a solução que minimiza o custo, mas também que maximiza a qualidade, etc. Modelamos nosso problema como mostrado nas equações 4.1, 4.2, 4.3, 4.4 e 4.5.

$$Z_1 = \sum_{i=1}^N U_i x_i \quad (4.1)$$

$$Z_2 = \sum_{i=1}^N O_i x_i \quad (4.2)$$

$$Z = \text{minimizar} \sum_{i=1}^2 w_i Z_i \quad (4.3)$$

sujeito a:

$$x_i \in [0, 1] \quad (4.4)$$

$$\sum_{i=1}^N x_i = 1 \quad (4.5)$$

onde:

N = número de modelos considerados

U_i = número de recursos subprovisionados de acordo com as previsões do modelo i

O_i = número de recursos superprovisionados de acordo com as previsões do modelo i

w_i = peso do objetivo i

x_i = peso do modelo i na próxima previsão

4.2.1.1.1 Solução proposta usando soma ponderada

Algoritmo 1: ALOCAÇÃO DE RECURSOS USANDO SOMA PONDERADA

Entrada: log de requisições, μ, R, w_1, w_2

Saída: Número de servidores virtuais (recursos) a serem alocados (m)

1 **inicio**

2 $\vec{u} \leftarrow$ atribuir 0 para o valor de recursos subprovisionados de cada modelo

3 $\vec{o} \leftarrow$ atribuir 0 para o valor de recursos superprovisionados de cada modelo

4 **para cada hora faça**

5 $\vec{u} \leftarrow$ atualizar o número de recursos subprovisionados para cada modelo

6 $\vec{o} \leftarrow$ atualizar o número de recursos superprovisionados para cada modelo

7 $\vec{x} \leftarrow$ resolver a equação 4.3 sujeito as restrições 4.4 e 4.5

8 $ts \leftarrow$ criar série temporal a partir dos dados extraídos do log

9 **para cada modelo de previsão i faça**

10 $funcao_i \leftarrow$ estimar uma função matemática usando ts e o modelo i

11 $p_i \leftarrow$ prever a demanda para o próximo período usando a $funcao_i$

12 **fim**

13 $p_f \leftarrow \sum_{i=1}^6 p_i x_i$

14 $m \leftarrow$ calcular número de recursos usando μ, R, p_f por meio da equação 4.18

15 alocar m recursos junto ao provedor de infraestrutura na nuvem

16 **fim**

17 **fim**

O algoritmo 1 ilustra o processo de alocação de recursos usando o método da soma ponderada. Ele recebe como entrada o *log* de requisições, a taxa de processamento de cada servidor a ser alocado na nuvem (μ), o tempo de resposta máximo acordado com o cliente (R), e o peso de cada um dos objetivos (w_1, w_2). A saída será a quantidade de recursos a ser alocada

para o próximo intervalo de tempo (hora), de acordo com a previsão de demanda. Inicialmente, para cada modelo de previsão, é atribuído o valor zero para a quantidade de recursos sub e superprovisionados. Na sequência o algoritmo entra em um laço onde, a cada intervalo de tempo de uma hora, irá: atualizar o número de recursos sub e superprovisionados de acordo com as previsões passadas de cada modelo; definir o peso de cada modelo na previsão resolvendo o sistema de equações lineares das equações 4.3, 4.4 e 4.5; combinar a previsão futura de cada modelo, usando os pesos calculados no passo anterior; e, por fim, calcular e alocar a quantidade necessária de recursos para a próxima hora, de acordo com a previsão.

A atualização da quantidade de recursos sub e superprovisionados para cada modelo de previsão é feita da seguinte forma: supondo que o modelo A previu que, no próximo intervalo de tempo, seriam necessários 10 recursos (servidores virtuais) para atender a demanda, e o modelo B previu 5. No entanto, ao final do intervalo, foi verificado que eram necessários 8 recursos. Assim sendo, o modelo A irá acumular, em sua conta, 2 recursos superprovisionados. Já o modelo B irá acumular 3 recursos subprovisionados. Essa conta é atualizada a cada hora.

4.2.1.2 Método ε -restrito

Este método é bastante similar ao anterior, com a diferença que, nesse caso, um dos objetivos é usado como restrição do problema. Modelamos nosso problema como mostrado nas equações 4.6, 4.7, 4.8 e 4.9.

$$Z_1 = \text{minimizar} \sum_{i=1}^N D_i x_i \quad (4.6)$$

sujeito a:

$$\sum_{i=1}^N I_i x_i \leq W \quad (4.7)$$

$$x_i \in [0, 1] \quad (4.8)$$

$$\sum_{i=1}^N x_i = 1 \quad (4.9)$$

onde:

D_i = número de requisições não atendidas de acordo com as previsões do modelo i

I_i = número de recursos ociosos de acordo com as previsões do modelo i

x_i = peso do modelo i na próxima previsão

W = limite de recursos ociosos

4.2.1.2.1 Solução proposta usando o método ε -restrito**Algoritmo 2:** ALOCAÇÃO DE RECURSOS USANDO O MÉTODO ε -RESTRITO**Entrada:** log de requisições, μ, R, W **Saída:** Número de servidores virtuais (recursos) a serem alocados (m)

```

1  inicio
2   $\vec{D} \leftarrow$  atribuir 0 para o valor de requisições não atendidas de cada modelo
3   $\vec{I} \leftarrow$  atribuir 0 para o valor de recursos ociosos de cada modelo
4  para cada hora faça
5       $\vec{D} \leftarrow$  atualizar o número de requisições não atendidas para cada modelo
6       $\vec{I} \leftarrow$  atualizar o número de recursos ociosos para cada modelo
7       $\vec{x}_u \leftarrow$  resolver a equação 4.6 sujeito as restrições 4.7, 4.8 e 4.9
8       $ts \leftarrow$  criar série temporal a partir dos dados extraídos do log
9      para cada modelo de previsão  $i$  faça
10          $funcao_i \leftarrow$  estimar uma função matemática usando  $ts$  e o modelo  $i$ 
11          $p_i \leftarrow$  prever a demanda para o próximo periodo usando a  $funcao_i$ 
12      fim
13       $p_f \leftarrow \sum_{i=1}^6 p_i x_i$ 
14       $m \leftarrow$  calcular número de recursos usando  $\mu, R, p_f$  por meio da equação 4.18
15      alocar  $m$  recursos junto ao provedor de infraestrutura na nuvem
16  fim
17 fim

```

O algoritmo 2 ilustra o processo de alocação de recursos por meio do método ε -restrito. As entradas são: o log de requisições, a taxa de processamento de cada servidor a ser alocado na nuvem (μ), o tempo de resposta máximo acordado com o cliente (R) e limite de recursos ociosos desejado (W). A saída será a quantidade de recursos a ser alocada para o próximo intervalo de tempo (hora), de acordo com a previsão de demanda. Inicialmente, para cada modelo de previsão, é atribuído o valor zero para a quantidade de requisições não atendidas e recursos ociosos. Na sequencia o algoritmo entra em um laço onde, a cada intervalo de tempo de uma hora, irá: atualizar o número de requisições não atendidas e recursos ociosos de acordo com as previsões passadas de cada modelo; definir o peso de cada modelo na previsão resolvendo o sistema de equações lineares das equações 4.6, 4.7, 4.8 e 4.9; combinar a previsão futura de cada modelo, usando os pesos calculados no passo anterior; e, por fim, calcular e alocar a quantidade necessária de recursos para a próxima hora, de acordo com a previsão. A atualização da quantidade de requisições não atendidas e recursos ociosos para cada modelo de previsão é feita da mesma maneira que no algoritmo anterior.

4.2.1.3 Método de Análise Hierárquica (AHP)

O objetivo é encontrar o vetor de prioridades global (PG) que soluciona o problema, e, então, determinar o peso de cada modelo de previsão.

A cada intervalo de tempo (uma hora), criamos as matrizes de julgamento e calculamos os PML's e o PG. Para definir a relação de prioridade entre os modelos em cada critério, usamos a seguinte regra: a cada intervalo de tempo é calculada a quantidade de recursos sub e superprovisionados gerados por cada modelo, de acordo com suas previsões passadas; em seguida, os modelos são ordenados em ordem crescente, de acordo com a quantidade de recursos (sub ou superprovisionados, de acordo com o critério que está sendo considerado) que geraram até o momento. O modelo que estiver na primeira posição tem prioridade 3 (moderada) sobre o segundo, 5 (alta) sobre o terceiro, 7 (muita alta) sobre o quarto, e 9 (extrema) sobre o quinto. O segundo tem prioridade 3 sobre o terceiro, 5 sobre o quarto e 7 sobre o quinto. O terceiro tem prioridade 3 sobre o quarto e 5 sobre o quinto. O quarto tem prioridade 3 sobre o quinto. Caso existam dois modelos empatados na mesma posição, terão prioridade 1 (igual) entre eles.

Para exemplificar, vamos imaginar que até determinado momento, os nossos cinco modelos de previsão (A, B, C, D e E) apresentem quantidades, acumuladas, de recursos sub e superprovisionados de acordo com a Tabela 5.

Tabela 5 – Recursos sub e superprovisionados

Recursos	A	B	C	D	E
Subprovisionados	15	18	12	20	25
Superprovisionados	10	14	10	21	16

De acordo com o cenário apresentado na Tabela 5, as matrizes de julgamento ficaram como ilustradas na Tabela 6.

Tabela 6 – Matrizes de julgamento dos modelos

Subprovisionados	A	B	C	D	E
A	1	3	1/3	5	7
B	1/3	1	1/5	3	5
C	3	5	1	7	9
D	1/5	1/3	1/7	1	3
E	1/7	1/5	1/9	1/3	1
Superprovisionados	A	B	C	D	E
A	1	5	1	9	7
B	1/5	1	1/5	5	3
C	1	5	1	9	7
D	1/9	1/5	1/9	1	1/3
E	1/7	1/3	1/7	3	1

A partir das matrizes de julgamento mostradas na Tabela 6, calculamos os PML's e o PG, mostrado na Tabela 7. Consideramos, nesse exemplo, que os critérios subprovisionamento e superprovisionamento têm a mesma importância para o usuário, como mostra o VPC. Em seguida, usamos os valores do PG para combinar as previsões efetuadas por cada modelo. Ou seja, a previsão do modelo A será multiplicada por 0.3371, a do modelo B por 0.1568, e assim

sucessivamente. Em seguida os valores resultantes são somados e, assim, teremos a previsão final para o próximo intervalo de tempo.

Tabela 7 – Peso de cada modelo na previsão (PG)

	PML Sub	PML Super		VPC		PG
A	0.2909	0.3833	x	0.5	=	0.3371
B	0.1636	0.1500				0.1568
C	0.4545	0.3833				0.4189
D	0.0728	0.0167				0.0447
E	0.0181	0.0667				0.0425

4.2.1.3.1 Solução proposta usando o AHP

O algoritmo 3 ilustra o processo de alocação de recursos usando o AHP. O algoritmo recebe como entrada o log das requisições, a taxa de processamento de cada servidor a ser alocado na nuvem (μ), o tempo de resposta máximo acordado com o cliente (R), a prioridade do critério subprovisionamento ($priSub$), e a prioridade do critério superprovisionamento ($priSuper$). A saída será a quantidade de recursos a ser alocada para o próximo intervalo de tempo (hora), de acordo com a previsão de demanda. Inicialmente, para cada modelo de previsão, é atribuído o valor zero para a quantidade de recursos sub e superprovisionados. Na sequência o algoritmo entra em um *loop* onde, a cada intervalo de tempo de uma hora, irá: atualizar o número de recursos sub e superprovisionados de acordo com as previsões passadas de cada modelo; criar as matrizes de julgamento, como descrito na seção 2.4.1.3; calcular os PML's, como descrito na seção 2.4.1.3; definir o peso de cada modelo na próxima previsão calculando o PG, como descrito na seção 2.4.1.3; combinar a previsão futura de cada modelo, usando os pesos calculados no passo anterior; e, por fim, calcular e alocar a quantidade necessária de recursos para a próxima hora, de acordo com a previsão. A atualização da quantidade de recursos sub e superprovisionados para cada modelo de previsão é feita do mesmo modo que nos algoritmos anteriores.

4.2.1.4 Algoritmo Genético Multiobjetivo

Neste estudo, cada indivíduo é representado por um conjunto de seis números reais. Cada um desses números é chamado de gene. Cada gene representa um modelo de previsão de séries temporais. Os valores que cada gene podem assumir pertence ao intervalo entre zero e um. A soma dos valores de todos os genes deve ser igual a um. Cada indivíduo é composto de uma combinação de genes que representam cada um dos modelos de previsão utilizadas. Estes modelos foram descritos no capítulo 2. O valor de cada gene representa o peso de cada modelo no indivíduo. As Equações 4.10, 4.11 e 4.12 mostram a representação formal de genes e indivíduos.

$$gene \in [0, 1] \quad (4.10)$$

Algoritmo 3: ALOCAÇÃO DE RECURSOS USANDO O AHP**Entrada:** Log das requisições, $\mu, R, priSub, priSuper$ **Saída:** Número de servidores virtuais (recursos) a serem alocados (m)

```

1 inicio
2    $\vec{sub} \leftarrow$  atribuir 0 para o valor de recursos subprovisionados de cada modelo
3    $\vec{super} \leftarrow$  atribuir 0 para o valor de recursos superprovisionados de cada modelo
4   para cada hora faça
5      $\vec{sub} \leftarrow$  atualizar o número de recursos subprovisionados para cada modelo
6      $\vec{super} \leftarrow$  atualizar o número de recursos superprovisionados para cada modelo
7      $mjSub \leftarrow$  criar a matriz de julgamento dentro do critério subprovisionamento
      usando  $\vec{sub}$ 
8      $mjSuper \leftarrow$  criar a matriz de julgamento dentro do critério superprovisionamento
      usando  $\vec{super}$ 
9      $mjCriterios \leftarrow$  criar a matriz de julgamento entre os critérios usando  $priSub$  e
       $priSuper$ 
10     $pmlSub \leftarrow$  calcular o PML para o critério subprovisionamento usando  $mjSub$ 
11     $pmlSuper \leftarrow$  calcular o PML para o critério superprovisionamento usando
       $mjSuper$ 
12     $pmlCriterios \leftarrow$  calcular o PML entre os critérios usando  $mjCriterios$ 
13     $\vec{pg} \leftarrow$  calcular o vetor de prioridades global usando  $pmlSub, pmlSuper$  e
       $pmlCriterios$ 
14     $ts \leftarrow$  criar série temporal a partir dos dados extraídos do log
15    para cada modelo de previsão  $i$  faça
16       $funcao_i \leftarrow$  estimar uma função matemática usando  $ts$  e o modelo  $i$ 
17       $prev_i \leftarrow$  prever a demanda para o próximo periodo usando a  $funcao_i$ 
18    fim
19     $prev_f \leftarrow \sum_{i=1}^6 prev_i pg_i$ 
20     $m \leftarrow$  calcular número de recursos usando  $\mu, R, prev_f$  por meio da equação 4.18
21    alocar  $m$  recursos junto ao provedor de infraestrutura na nuvem
22  fim
23 fim

```

$$indivíduo = \{gene_1, gene_2, gene_3, gene_4, gene_5, gene_6\} \quad (4.11)$$

$$\sum_{i=1}^6 gene_i = 1 \quad (4.12)$$

Depois de escolher o método de representação, a etapa seguinte é escolher a dimensão da população e um método para a geração da população inicial. A população inicial deve ter uma variedade genética tão grande quanto possível, a fim de ser capaz de explorar todo o espaço de busca. Neste trabalho, a população inicial foi gerada considerando seis indivíduos iniciais. Cada indivíduo é um modelo de previsão de série temporal, ou seja, tem um dos seus genes com

valor igual a 1 e os outros genes com valor igual a 0. Na sequência, todos os indivíduos são combinados entre si (média aritmética) em grupos de dois, três, quatro, cinco e seis indivíduos. Assim, é formada a população inicial.

Para avaliar os indivíduos utilizaremos o nível de dominância de Pareto, que leva em consideração dois objetivos: o número de recursos subprovisionados; e o número de recursos superprovisionados. Os indivíduos são ordenados por meio do algoritmo NSGA II (DEB *et al.*, 2000) em níveis, de acordo com o grau de dominância de Pareto. Para desempate, entre os indivíduos com o mesmo grau de dominância, é utilizada uma função de custo, que é a soma ponderada dos objetivos. Cada objetivo tem seu próprio peso, de acordo com as prioridades do usuário. Assim, o melhor indivíduo (solução) é aquele que minimiza a soma ponderada dos objetivos, similar ao método da soma ponderada, descrito na seção 4.2.1.1. A Figura 6 ilustra o processo de avaliação.

$$\text{Custo} = \sum_{i=1}^N \text{custo}_i \quad (4.13)$$

Indivíduo A $\longrightarrow$ {0.2, 0.1, 0.25, 0.15, 0.2, 0.1}

Modelo de Previsão 1 Modelo de Previsão 2 Modelo de Previsão 3 Modelo de Previsão 4 Modelo de Previsão 5 Modelo de Previsão 6

↓ ↓ ↓ ↓ ↓ ↓
15 10 20 35 30 25

$\text{arredonda}(0.2 \cdot 15 + 0.1 \cdot 10 + 0.25 \cdot 20 + 0.15 \cdot 35 + 0.2 \cdot 30 + 0.1 \cdot 25) = 23$

Valor Observado $\longrightarrow$ 22

↑
Valor Previsto

Figura 6 – Exemplo de Avaliação

Na Figura 6, podemos ver como os modelos de previsão são combinados usando os genes do indivíduo como pesos para os seus valores. Depois disso, nós temos o valor previsto e podemos usar a Equação 4.18 para calcular o número de recursos necessários, e, consequentemente, o número de recursos sub e superprovisionados. A partir de então é possível calcular o custo gerado por cada indivíduo. Este processo é repetido para cada indivíduo em todas as previsões anteriores, e cada valor é adicionado ao custo total de cada indivíduo. O indivíduo com o menor custo será o mais capaz de resolver o problema. A equação 4.13 ilustra a função *fitness*.

Após a fase de avaliação, os 50 melhores indivíduos são selecionados para a fase de cruzamento. O cruzamento é necessário para explorar novas soluções. Este operador cria novos indivíduos contendo características de outros dois membros da população que são selecionados, e,

assim, cria novos pontos no espaço de busca. Neste trabalho, o operador de cruzamento escolhido foi a média aritmética dos valores designados para os genes de cada indivíduo participante do cruzamento. Após os pais serem selecionados como descrito anteriormente, na fase de seleção, será feita a combinação dos genes de cada um. Definimos, em nossa solução, que a probabilidade de cruzamento entre dois indivíduos é de 90%. A Figura 7 mostra uma operação de cruzamento.

Figura 7 – Exemplo de cruzamento

Após a fase de cruzamento, é realizada a etapa de mutação. O operador de mutação é crítico para o sucesso de algoritmos genéticos, uma vez que aumenta a variedade genética da população. No entanto, ao contrário do cruzamento, a mutação é, geralmente, feita por genes modificadores dentro de um indivíduo e sua probabilidade é baixa. Neste trabalho, utilizou-se o operador de *swap* (troca) como operador de mutação, com uma probabilidade de 10%. No operador de *swap*, dois genes selecionados aleatoriamente vão trocar seus valores.

Passada a fase de mutação, chega o momento de atualizar a população com novos indivíduos. Neste ponto, os indivíduos resultantes do processo de cruzamento e mutação são introduzidos na população de acordo com a política adotada pelo AG. Na forma mais tradicional, a população mantém um tamanho fixo e os indivíduos são criados no mesmo número que seus antecessores e os substituí por completo. No entanto, existem alternativas a esta abordagem, por exemplo, todos os N melhores indivíduos podem ser sempre mantidos. No nosso caso, mantemos os indivíduos originais e seus filhos resultantes do processo de cruzamento e mutação.

Por fim, chega um momento em que o AG deve parar sua execução. Para tanto, é realizado um teste que define se o processo de evolução do AG atingiu um ponto de parada predeterminado. Os critérios para a parada podem ser diferentes. No nosso caso, a regra de parada é quando os indivíduos param de evoluir, isto é, quando a melhor solução (indivíduo) da geração atual é o mesmo que o da geração anterior. No entanto, para evitar um número indefinido de iterações, definimos que o algoritmo deve parar sua execução quando atingir 100 iterações.

Algoritmo 4: ALOCAÇÃO DE RECURSOS USANDO AG**Entrada:** log de requisições, μ, R, w_1, w_2 **Saída:** Número de servidores virtuais (recursos) a serem alocados (m)

```

1  inicio
2   $\vec{u} \leftarrow$  atribuir 0 para o valor de recursos subprovisionados de cada modelo
3   $\vec{o} \leftarrow$  atribuir 0 para o valor de recursos superprovisionados de cada modelo
4  para cada hora faça
5       $\vec{u} \leftarrow$  atualizar o número de recursos subprovisionados para cada modelo
6       $\vec{o} \leftarrow$  atualizar o número de recursos superprovisionados para cada modelo
7      inicializar população
8      enquanto não alcançar a condição de parada faça
9          avaliar população
10         selecionar reprodutores
11         cruzar selecionados
12         mutar resultantes
13         avaliar resultantes
14         atualizar população
15     fim
16      $\vec{x} \leftarrow$  solução do AG
17      $ts \leftarrow$  criar série temporal a partir dos dados extraídos do log
18     para cada modelo de previsão  $i$  faça
19          $funcao_i \leftarrow$  estimar uma função matemática usando  $ts$  e o modelo  $i$ 
20          $p_i \leftarrow$  prever a demanda para o próximo período usando a  $funcao_i$ 
21     fim
22      $p_f \leftarrow \sum_{i=1}^6 p_i x_i$ 
23      $m \leftarrow$  calcular número de recursos usando  $\mu, R, p_f$  por meio da equação 4.18
24     alocar  $m$  recursos junto ao provedor de infraestrutura na nuvem
25 fim
26 fim

```

4.2.1.4.1 Solução proposta usando AG

O algoritmo 4 ilustra o processo de alocação de recursos usando algoritmo genético multiobjetivo. O algoritmo recebe como entrada o *log* de requisições, a taxa de processamento de cada servidor a ser alocado na nuvem (μ), o tempo de resposta máximo acordado com o cliente (R), e o peso de cada um dos objetivos (w_1, w_2). A saída será a quantidade de recursos a ser alocada para o próximo intervalo de tempo (hora), de acordo com a previsão de demanda. Inicialmente, para cada modelo de previsão, é atribuído o valor zero para a quantidade de recursos sub e superprovisionados. Na sequência o algoritmo entra em um laço onde, a cada intervalo de tempo de uma hora, irá: atualizar o número de recursos sub e superprovisionados de acordo com as previsões passadas de cada modelo; definir o peso de cada modelo na previsão por meio do algoritmo genético multiobjetivo, como descrito na seção 4.2.1.4; combinar a previsão futura de cada modelo, usando os pesos calculados no passo anterior; e, por fim, calcular e alocar a quantidade necessária de recursos para a próxima hora, de acordo com a previsão. A atualização

da quantidade de recursos sub e superprovisionados para cada modelo de previsão é feita da mesma forma que nos três algoritmos anteriores.

4.2.2 Alocação de recursos

Para determinar a quantidade de recursos a ser alocada, a partir da previsão de demanda, modelamos nosso sistema como uma fila $M/M/m$. Neste tipo de fila a utilização do sistema é modelada de acordo com a equação 4.14 (GROSS *et al.*, 2013).

$$\rho = \frac{\lambda}{m\mu} \quad (4.14)$$

Na equação 4.14, ρ é a utilização do sistema, λ é a taxa de chegada de requisições, μ é a taxa de processamento do servidor e m é o número de servidores.

Para o sistema ser estável, ρ precisa ser menor do que 1 (GROSS *et al.*, 2013). Nosso objetivo é encontrar o menor valor para m que mantenha o sistema estável. Escrevendo a equação 4.14 em função de m temos a equação 4.15.

$$m = \lceil \frac{\lambda}{\rho\mu} \rceil \quad (4.15)$$

Para encontrar um valor adequado para ρ é necessário considerar o valor de μ e o tempo de resposta de uma requisição no servidor. O tempo de resposta do servidor, pode ser calculado como mostra a equação 4.16 (GROSS *et al.*, 2013):

$$R = \frac{\frac{1}{\mu}}{1 - \rho} \quad (4.16)$$

Colocando a equação 4.16 em função de ρ obtemos a equação 4.17.

$$\rho = 1 - \frac{1}{R\mu} \quad (4.17)$$

Juntando as equações 4.15 e 4.17 obtemos a equação 4.18, que representa o número de servidores, m , cada um capaz de processar μ requisições por segundo, necessário para atender uma taxa de chegada de λ requisições por segundo, com tempo de resposta máximo de R segundos (MENASCÉ; ALMEIDA; DOWDY, 2002).

$$m = \lceil \frac{R\lambda}{R\mu - 1} \rceil \quad (4.18)$$

4.3 Considerações Finais

Nesse capítulo foi apresentado uma solução para o problema da alocação de recursos para aplicações hospedadas na nuvem. O objetivo é criar um *autoscaling* proativo, baseado na combinação de modelos de previsão de séries temporais, por meio de métodos de otimização multiobjetivo. Foram discutidas quatro soluções baseadas em otimização multiobjetivo: método da soma ponderada; método ε -restrito; método de análise hierárquica; e algoritmos genéticos. No próximo capítulo serão discutidos os resultados obtidos e uma comparação entre as propostas.

AVALIAÇÃO E RESULTADOS

5.1 Considerações Iniciais

No capítulo anterior foi apresentada a proposta de solução para o problema de alocação de recursos na nuvem. Quatro algoritmos, baseados em métodos de otimização multiobjetivo, foram considerados para tal finalidade. Neste capítulo serão apresentados os resultados obtidos oriundos das propostas, assim como uma comparação entre as mesmas. Os experimentos foram realizados por meio de simulação, e carga de trabalho foi baseada em quatro *logs* extraídos de servidores web reais.

5.2 Experimentos e Avaliação

5.2.1 Metodologia

Para atingir os objetivos e comprovar a hipótese desta Tese deve-se analisar os resultados obtidos em experimentos que considerem diferentes tipos de séries temporais, extraídas de *logs* de servidores com diferentes características de carga de trabalho, além de diferentes cenários das características das requisições e diferentes acordos de nível de serviço (SLAs). É necessário também considerar os diferentes objetivos dos usuários (preferência maior por recursos subprovisionados ou superprovisionados). Por fim, deve-se comparar os resultados obtidos pelos algoritmos propostos neste trabalho com os resultados apresentados pelo uso de outros modelos de previsão de séries temporais, mostrando qual solução é mais eficiente para a maioria dos cenários considerados. Deste modo, a metodologia utilizada para desenvolvimento desta Tese possui como base a investigação da literatura e a utilização de modelos de previsão de séries temporais combinados por meio de otimização multiobjetivo, para construção de um modelo genérico que se adapte a diferentes tipos de carga de trabalho, diferentes cenários de utilização e diferentes objetivos dos usuários.

Para realizar os experimentos, os algoritmos propostos no capítulo anterior foram implementados utilizando a linguagem Java, e também realizada a simulação da chegada de requisições com base em *logs* de servidores Web reais. A biblioteca *RJava* (URBANEK; URBANEK, 2014) foi utilizada para se comunicar com o pacote estatístico R (BRAUN; MURDOCH, 2007), onde estão implementadas as funções *arima(p, d, q)*, *auto.arima()* e *ets()*, utilizadas no trabalho. O modelo AR(1) foi implementado por meio da função *arima(1,0,0)*, o modelo MA(1) por meio da função *arima(0,0,1)*, o modelo ARMA(1,1) por meio da função *arima(1,0,1)*, o modelo ARIMA pela função *auto.arima()* e o ETS pela função *ets()*. O modelo NAIVE foi implementado usando a linguagem Java. Para efetuar a previsão para o próximo intervalo de tempo ($t + 1$), todos os modelos são treinados considerando os dados extraídos do *log* até o instante t . Para resolver os modelos de programação linear propostos nos algoritmos CMPL (algoritmo 1 do capítulo 4) e CMPLR (algoritmo 2 do capítulo 4) foi utilizada a classe *SimplexSolver* da biblioteca Apache Commons Math (MATH, 2014). Essa classe implementa o algoritmo *Simplex* de duas fases (DANTZIG, 1998). O computador utilizado para executar os experimentos foi um Intel Quad Core I5 3.2 GHz, 16 GB de RAM, 64 bits.

5.2.2 Logs Utilizados

O presente trabalho considerou a utilização de quatro *logs* extraídos de servidores reais: o *log* Fifa World Cup 98 (CUP, 1998); o *log* Nasa (NASA, 1996); o *log* ClarkNet (CLARKNET, 1995); e o Google Cluster Data *log* (WILKES, 2011). O *log* Fifa World Cup é composto por todas as requisições recebidas pelo site da copa do mundo de 1998, entre 30 de abril de 1998 e 26 de julho de 1998. Durante esse período o site recebeu 1.352.804.107 requisições. O *log* Nasa é composto por dois meses de requisições recebidas pelo servidor da NASA Kennedy Space Center na Florida. O *log* ClarkNet é composto por duas semanas de requisições recebidas pelo servidor ClarkNet, que é um site do Metro da área metropolitana de Baltimore-Washington. O *Log* de dados do Google Cluster fornece dados de uma célula de 12 mil máquinas ao longo de um período de um mês, em maio de 2011. Foram utilizadas as requisições de *jobs*, extraídas a partir desses dados. Para criar as séries temporais, extraímos dos *logs* citados, a quantidade máxima de requisições por segundo a cada hora. A Figura 8 ilustra as séries utilizadas. Para trazer as séries mais próximas da realidade atual, considerando o aumento no número de usuários da Internet no mundo, aplicamos um fator de ajuste nas mesmas. A série FIFA World Cup foi multiplicada por 9 (fator de ajuste), a série Nasa foi multiplicada por 6, a série Clarknet foi multiplicada por 3 e a série Google não foi ajustada, pois é bastante recente (2011). Esses fatores de ajuste foram baseados em (GROUP, 2015). Os resultados que serão apresentados consideram a quantidade de recursos sub e superprovisionados gerados durante todo o período de duração do *log*.

Figura 8 – Séries Utilizadas.

5.3 Resultados

As próximas seções ilustram os resultados obtidos, considerando as quatro séries utilizadas neste trabalho e quatro cenários distintos descritos na Tabela 8. Para cada cenário foram executados experimentos considerando: a mesma prioridade entre recursos subprovisionados e superprovisionados (Custo(1,1)); maior prioridade para recursos subprovisionados (Custo(2,1)); e maior prioridade para recursos superprovisionados (Custo(1,2)). O eixo X representa o número de recursos subprovisionados e o eixo Y representa o número de recursos superprovisionados, de acordo com cada modelo de previsão.

O motivo da escolha dos cenários apresentados na Tabela 8 foi variar a capacidade de processamento do servidor (μ) e o tempo de resposta máximo acordado com o cliente (R) com o objetivo de analisar como nossa proposta se comporta tanto quando as condições de processamento são mais rígidas (baixo tempo de resposta e baixa taxa de processamento) como quando as condições são menos exigentes (alta taxa de processamento e alto tempo de resposta).

Tabela 8 – Cenários considerados nos experimentos

Cenário	Taxa de processamento (μ)	Tempo de resposta (R)
Primeiro	30 requisições/segundo	0.2 segundo
Segundo	30 requisições/segundo	0.8 segundo
Terceiro	60 requisições/segundo	0.2 segundo
Quarto	60 requisições/segundo	0.8 segundo

Para analisar a otimalidade de uma solução multiobjetivo, como é o caso, é utilizado o conceito de solução não dominada, ou ótima de Pareto, que é uma solução para a qual não existe outra solução viável que melhore, ao mesmo tempo, todos os objetivos. Todas as soluções com essas características pertencem ao conjunto ótimo de Pareto.

5.3.1 Algoritmo CMPL

5.3.1.1 Série Clarknet

As Figuras 9, 10, 11 e 12 mostram os resultados para a série ClarkNet. As soluções que estão sob a linha são as chamadas soluções ótimas de Pareto ou soluções não dominadas. A linha representa a fronteira ótima de Pareto. As Tabelas 9, 10, 11 e 12 mostram a quantidade de recursos subprovisionados e superprovisionados gerados por cada modelo e solução.

No primeiro cenário, Figura 9, podemos observar que a proposta (CMPL) está no conjunto ótimo de Pareto, ao lado dos modelos Naive, AR e ARMA. Destaca-se também, que a solução CMPL(2,1) foi dominada pela solução CMPL(1,2). Isso mostra que, nem sempre, aumentar a prioridade de determinado critério vai levar a melhores resultados para o mesmo. É preciso considerar que, por se tratar de uma otimização estocástica (uma vez que estamos trabalhando com previsões), não há como garantir que o comportamento passado se repetirá exatamente igual no futuro.

Na Tabela 9 podemos observar o custo gerado por cada solução, considerando os seguintes cenários: Custo(1, 1), onde o preço de um recurso subprovisionado é o mesmo de um recurso superprovisionado e igual a 1; Custo(2, 1), onde o preço de um recurso subprovisionado é igual a 2 e o preço de um recurso superprovisionado é igual a 1; e Custo(1, 2), onde o preço de um recurso subprovisionado é igual a 1 e o preço de um recurso superprovisionado é igual a 2. A solução CMPL(1,1) foi a melhor nos cenários Custo(1, 1) e Custo(2, 1). No cenário Custo(1, 2) o modelo ARMA foi o melhor, seguido pelas soluções CMPL(1,1) e CMPL(1,2).

Na Figura 10, que ilustra os resultados para o segundo cenário, podemos observar que a solução CMPL(1,2) está no conjunto ótimo de Pareto. Por outro lado, as soluções CMPL(1,1) e CMPL(1,2) foram dominadas pelo modelo ETS. O modelo Naive também foi dominado neste cenário. Na Tabela 10 observa-se o custo gerado por cada solução. O modelo ARIMA apresentou os melhores resultados para o cenário Custo(1, 1), seguido pelo modelo ARMA, ETS e pela solução CMPL(1, 1). No cenário Custo(2, 1), os modelos ETS e ARIMA apresentaram os

menores custos, seguidos pelas soluções CMPL(2,1) e CMPL(1,1). No cenário Custo(1, 2) o modelo ARIMA foi novamente o melhor, seguido pela solução CMPL(1,2) e pelo modelo AR.

Na Figura 11 podemos observar os resultados para o terceiro cenário. As soluções CMPL(2,1) e CMPL(1,2) estão no conjunto ótimo de Pareto. Já a solução CMPL(1,1) foi dominada pela solução CMPL(2,1). Os modelos Naive e ARIMA também foram dominados neste cenário. Observando a Tabela 11 notamos que o modelo ARMA apresentou o menor custo para o cenário Custo(1,1), seguido pela solução CMPL(2,1) e pela CMPL(1,1). No cenário Custo(2,1), a solução CMPL(2,1) apresentou o menor custo. Já no cenário Custo(1, 2) o modelo ARMA apresentou melhor desempenho. A solução CMPL(1,2) apresentou o quarto menor custo.

No quarto cenário, ilustrado na Figura 12, as soluções CMPL(1,1) e CMPL(1,2) estão no conjunto ótimo de Pareto, e a solução CMPL(2,1) foi dominada pelo modelo ETS. Além desta, os modelos Naive e AR(1) também foram dominados neste cenário. Na Tabela 12 observa-se que o modelo ARIMA apresentou o menor custo para o cenário Custo(1,1), seguido pela solução CMPL(1,1). No cenário Custo(2,1), os modelos ETS e ARIMA apresentaram o menor custo, seguidos pelas soluções CMPL(1,1) e CMPL(2,1). Já no cenário Custo(1, 2) o modelo MA apresentou o menor custo. A solução CMPL(1,2) não obteve bom desempenho, superando apenas o modelo Naive.

5.3.1.2 Série Nasa

Nas Figuras 13, 14, 15 e 16 podemos observar os resultados para a série Nasa. Já as tabelas 13, 14, 15 e 16 ilustram a quantidade de recursos sub e superprovisionados para cada modelo e solução.

No primeiro cenário, Figura 13, podemos observar que a solução CMPL(1,1) e CMPL(1,2) estão no conjunto ótimo de Pareto, ao lado dos modelos ARIMA e MA. Podemos notar, também, que a solução CMPL(2,1) foi dominada pela solução CMPL(1,1). Na Tabela 13 podemos observar que o modelo ARIMA foi a melhor no cenário Custo(1, 1), seguido pela solução CMPL(1,1). No cenário Custo(2, 1) a solução CMPL(1,1) apresentou o menor custo, superando a solução CMPL(2,1). No cenário Custo(1, 2) o modelo ARIMA foi novamente o melhor, seguido pela solução CMPL(1,2).

A Figura 14 mostra os resultados para o segundo cenário, podemos observar que as soluções CMPL(1,1), CMPL(2,1) e CMPL(1,2) estão no conjunto ótimo de Pareto. Os modelos ETS, AR(1) e ARMA(1,1) foram dominados neste cenário. De acordo com a Tabela 14, o modelo ARIMA apresentou os melhores resultados para o cenário Custo(1, 1), seguido pela solução CMPL(1,1). No cenário Custo(2, 1), a solução CMPL(1,1) apresentou o menor custo, seguida pela solução CMPL(2,1). No cenário Custo(1, 2) o modelo MA foi o melhor, seguido pela solução CMPL(1,2).

No terceiro cenário, Figura 15, as soluções CMPL(1,1) e CMPL(1,2) estão no conjunto

Figura 9 – Clarknet (CMPL) - Primeiro Cenário

Tabela 9 – Clarknet (CMPL) - Primeiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	105	64	169	274	233
ARMA(1,1)	113	49	162	275	211
MA(1)	131	56	187	318	243
ARIMA	112	73	185	297	258
AR(1)	110	56	166	276	222
Naive	101	101	202	303	303
CMPL(1,1)	104	58	162	266	220
CMPL(2,1)	107	62	169	276	231
CMPL(1,2)	106	57	163	269	220

Figura 10 – Clarknet (CMPL) - Segundo Cenário

Tabela 10 – Clarknet (CMPL) - Segundo Cenário

	Recursos sub- provisionados	Recursos super- provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	76	62	138	214	200
ARMA(1,1)	87	49	136	223	185
MA(1)	123	27	150	273	177
ARIMA	79	56	135	214	191
AR(1)	101	46	147	248	193
Naive	87	87	174	261	261
CMPL(1,1)	78	66	144	222	210
CMPL(2,1)	77	68	145	222	213
CMPL(1,2)	119	37	156	275	193

Figura 11 – Clarknet (CMPL) - Terceiro Cenário

Tabela 11 – Clarknet (CMPL) - Terceiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	56	44	100	156	144
ARMA(1,1)	62	36	98	160	134
MA(1)	102	19	121	223	140
ARIMA	57	44	101	158	145
AR(1)	69	35	104	173	139
Naive	55	55	110	165	165
CMPL(1,1)	55	45	100	155	145
CMPL(2,1)	54	45	99	153	144
CMPL(1,2)	59	42	101	160	143

Figura 12 – Clarknet (CMPL) - Quarto Cenário

Tabela 12 – Clarknet (CMPL) - Quarto Cenário

	Recursos sub- provisionados	Recursos super- provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	49	44	93	142	137
ARMA(1,1)	60	34	94	154	128
MA(1)	96	14	110	206	124
ARIMA	53	36	89	142	125
AR(1)	75	34	109	184	143
Naive	53	53	106	159	159
CMPL(1,1)	52	40	92	144	132
CMPL(2,1)	51	46	97	148	143
CMPL(1,2)	79	33	112	191	145

ótimo de Pareto. Por outro lado, a solução CMPL(2,1) foi dominada pela solução CMPL(1,1). Os modelos AR(1) e ARMA(1,1) também foram dominados. De acordo com a Tabela 15, o modelo ARIMA apresentou o menor custo para o cenário Custo(1,1), seguido pela solução CMPL(1,1). No cenário Custo(2,1), o modelo ARIMA, novamente, apresentou o menor custo, seguido pela solução CMPL(1,1). A solução CMPL(2,1) foi a quarta melhor. No cenário Custo(1, 2) o modelo MA apresentou melhor desempenho, seguido pelo modelo ARIMA e pela solução CMPL(1,2).

No quarto cenário, ilustrado na Figura 16, podemos observar que as soluções CMPL(1,1), CMPL(2,1) e CMPL(1,2) estão no conjunto ótimo de Pareto. O único modelo dominado foi o ETS. Na Tabela 16 observamos que o modelo ARIMA apresentou o menor custo para o cenário Custo(1,1), seguido pela solução CMPL(1,1). No cenário Custo(2,1), o modelo Naive apresentou o menor custo, seguido pela solução CMPL(2,1). Já no cenário Custo(1, 2) o modelo MA apresentou o menor custo, seguido pela solução CMPL(1,2).

5.3.1.3 Série Copa do Mundo

As Figuras 17, 18, 19 e 20 mostram os resultados para a série FIFA World Cup. Nas Figuras 17, 18, 19 e 20 podemos observar a quantidade de recursos subprovisionados e superprovisionados para a referida série.

A Figura 17 ilustra os resultados para o primeiro cenário. Podemos observar que a solução CMPL(1,1), CMPL(2,1) e CMPL(1,2) estão no conjunto ótimo de Pareto, ao lado dos modelos Naive, ARIMA, AR e MA. Observando a Tabela 17, notamos que o modelo Naive foi a melhor no cenário Custo(1, 1), seguido de perto pelas soluções CMPL(2,1) e CMPL(1,1). A mesma situação se repetiu no cenário Custo(2, 1). No cenário Custo(1, 2) o modelo AR foi o melhor, seguido pela solução CMPL(1,2).

Os próximos cenários, ilustrados pelas Figuras 18, 19 e 20, e pelas Tabelas 18, 19 e 20, apresentam resultados muito similares aos discutidos no primeiro cenário. A diferença é que houve uma diminuição no número de recursos subprovisionados e superprovisionados, e consequentemente no custo, devido ao fato de, tanto a taxa de processamento quanto o SLA serem maiores.

5.3.1.4 Série Google

As Figuras 21, 22, 23 e 24 ilustram os resultados para a série Google job. Nas Tabelas 21, 22, 23 e 24 podemos observar a quantidade de recursos sub e superprovisionados para a referida série.

No primeiro cenário, Figura 21, nota-se que praticamente todas as soluções e modelos estão no conjunto ótimo de Pareto, exceto a solução CMPL(1,1). Na Tabela 21 podemos observar que o modelo MA foi a melhor no cenário Custo(1, 1), seguido pela solução CMPL(1,2). A solução CMPL(1,1) foi a que apresentou o quarto menor custo. No cenário Custo(2, 1) o modelo

Figura 13 – Nasa (CMPL) - Primeiro Cenário

Tabela 13 – Nasa (CMPL) - Primeiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	186	174	360	546	534
ARMA(1,1)	215	135	350	565	485
MA(1)	367	97	464	831	561
ARIMA	196	133	329	525	462
AR(1)	187	173	360	547	533
Naive	210	210	420	630	630
CMPL(1,1)	180	154	334	514	488
CMPL(2,1)	180	163	343	523	506
CMPL(1,2)	186	149	335	521	484

Figura 14 – Nasa (CMPL) - Segundo Cenário

Tabela 14 – Nasa (CMPL) - Segundo Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	201	137	338	539	475
ARMA(1,1)	219	96	315	534	411
MA(1)	274	52	326	600	378
ARIMA	189	101	290	479	391
AR(1)	202	108	310	512	418
Naive	168	168	336	504	504
CMPL(1,1)	175	116	291	466	407
CMPL(2,1)	172	130	302	474	432
CMPL(1,2)	218	83	301	519	384

Figura 15 – Nasa (CMPL) - Terceiro Cenário

Tabela 15 – Nasa (CMPL) - Terceiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	113	123	236	349	359
ARMA(1,1)	159	80	239	398	319
MA(1)	249	15	264	513	279
ARIMA	134	74	208	342	282
AR(1)	162	74	236	398	310
Naive	121	121	242	363	363
CMPL(1,1)	123	97	220	343	317
CMPL(2,1)	125	106	231	356	337
CMPL(1,2)	164	69	233	397	302

Figura 16 – Nasa (CMPL) - Quarto Cenário

Tabela 16 – Nasa (CMPL) - Quarto Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	158	91	249	407	340
ARMA(1,1)	203	44	247	450	291
MA(1)	266	3	269	535	272
ARIMA	182	48	230	412	278
AR(1)	207	41	248	455	289
Naive	121	121	242	363	363
CMPL(1,1)	155	79	234	389	313
CMPL(2,1)	123	118	241	364	359
CMPL(1,2)	222	27	249	471	276

Figura 17 – Copa do Mundo (CMPL) - Primeiro Cenário

Tabela 17 – Copa do Mundo (CMPL) - Primeiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	32403	27511	59914	92317	87425
ARMA(1,1)	39412	21283	60695	100107	81978
MA(1)	112525	5475	118000	230525	123475
ARIMA	35674	24295	59969	95643	84264
AR(1)	39446	18066	57512	96958	75578
Naive	26756	26754	53510	80266	80264
CMPL(1,1)	28415	25589	54004	82419	79593
CMPL(2,1)	26932	26611	53543	80475	80154
CMPL(1,2)	39264	18480	57744	97008	76224

Figura 18 – Copa do Mundo (CMPL) - Segundo Cenário

Tabela 18 – Copa do Mundo (CMPL) - Segundo Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	28161	23910	52071	80232	75981
ARMA(1,1)	34254	18500	52754	87008	71254
MA(1)	97827	4765	102592	200419	107357
ARIMA	30994	21138	52132	83126	73270
AR(1)	34299	15698	49997	84296	65695
Naive	23260	23258	46518	69778	69776
CMPL(1,1)	24701	22244	46945	71646	69189
CMPL(2,1)	23424	23112	46536	69960	69648
CMPL(1,2)	34077	16039	50116	84193	66155

Figura 19 – Copa do Mundo (CMPL) - Terceiro Cenário

Tabela 19 – Copa do Mundo (CMPL) - Terceiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	14722	12488	27210	41932	39698
ARMA(1,1)	17894	9669	27563	45457	37232
MA(1)	51165	2504	53669	104834	56173
ARIMA	16194	11042	27236	43430	38278
AR(1)	17910	8211	26121	44031	34332
Naive	12151	12150	24301	36452	36451
CMPL(1,1)	12927	11592	24519	37446	36111
CMPL(2,1)	12253	12073	24326	36579	36399
CMPL(1,2)	17736	8413	26149	43885	34562

Figura 20 – Copa do Mundo (CMPL) - Quarto Cenário

Tabela 20 – Copa do Mundo (CMPL) - Quarto Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	13807	11723	25530	39337	37253
ARMA(1,1)	16767	9072	25839	42606	34911
MA(1)	47867	2329	50196	98063	52525
ARIMA	15176	10338	25514	40690	35852
AR(1)	16790	7696	24486	41276	32182
Naive	11387	11386	22773	34160	34159
CMPL(1,1)	12078	10907	22985	35063	33892
CMPL(2,1)	11466	11320	22786	34252	34106
CMPL(1,2)	16721	7867	24588	41309	32455

ETS apresentou o menor custo, seguido pela solução CMPL(2,1). No cenário Custo(1, 2) a solução CMPL(1,2) apresentou o menor custo.

A Figura 22 mostra os resultados para o segundo cenário, no qual podemos observar resultados muito similares ao primeiro cenário. De acordo com a Tabela 22, o modelo AR apresentou os melhores resultados para o cenário Custo(1, 1), seguido pelo modelo MA e pelas soluções CMPL(2,1) e CMPL(1,1). No cenário Custo(2, 1), os modelos ARMA e ARIMA apresentaram os menores custos, seguidos pela solução CMPL(1,1). A solução CMPL(2,1) apresentou o sétimo menor custo. No cenário Custo(1, 2) o modelo MA foi o melhor, seguido pelos modelos ARMA, ARIMA e pela solução CMPL(1,2).

No terceiro cenário, Figura 23, as soluções CMPL(2,1) e CMPL(1,2) estão no conjunto ótimo de Pareto. Já a solução CMPL(1,1) foi dominada, assim como os modelos ARIMA e ARMA(1,1). De acordo com a Tabela 23, a solução CMPL(1,2) foi a melhor, ao lado dos modelos AR e MA, para o cenário Custo(1,1). A solução CMPL(1,1) apresentou o quinto menor custo. No cenário Custo(2,1), a solução CMPL(1,2) apresentou o menor custo. Já a solução CMPL(2,1) apresentou o sétimo menor custo. Já no cenário Custo(1, 2) os modelos MA e AR apresentaram os menores custos, seguidos pela solução CMPL(1,2).

O quarto cenário, ilustrado na Figura 24 e na Tabela 24, apresenta, praticamente, os mesmos resultados do cenário anterior. A diferença é que o modelo ETS também foi dominado neste cenário.

De acordo com os resultados observados para o algoritmo CMPL, percebe-se que a combinação de modelos de previsão sempre converge para a melhor solução. Isso não significa que será sempre a melhor solução, mas que vai sempre se aproximar da melhor solução. Outro ponto importante a ser observado, é que nem sempre a preferência do usuário em relação a recursos subprovisionados e superprovisionados se traduz no resultado final. Por exemplo, nem sempre a solução CMPL(2,1) foi melhor que a CMPL(1,1) no cenário Custo(2,1), como era esperado. Isso acontece porque, como já destacado anteriormente, não estamos trabalhando com otimização determinística, mas sim estocástica. Ou seja, baseando-se na solução que minimizou o custo no passado, assumimos que essa solução é mais apta para minimizar o custo, também, no futuro. No entanto, não podemos garantir que essa solução será sempre a melhor. Como trabalhos futuros, a sugestão é ajustar esses parâmetros de peso entre recursos subprovisionados e superprovisionados de maneira dinâmica.

5.3.2 Algoritmo CMPLR

5.3.2.1 Série Clarknet

As Figuras 25, 26, 27 e 28 mostram os resultados para a série ClarkNet. O eixo X representa o número de requisições não atendidas e o eixo Y representa o número de recursos ociosos, de acordo com cada modelo de previsão. Um cuidado especial a ser tomado é na escolha

Figura 21 – Google (CMPL) - Primeiro Cenário

Tabela 21 – Google (CMPL) - Primeiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	193	56	249	442	305
ARMA(1,1)	214	27	241	455	268
MA(1)	229	6	235	464	241
ARIMA	200	47	247	447	294
AR(1)	212	30	242	454	272
Naive	151	151	302	453	453
CMPL(1,1)	215	27	242	457	269
CMPL(2,1)	182	82	264	446	346
CMPL(1,2)	223	13	236	459	249

Figura 22 – Google (CMPL) - Segundo Cenário

Tabela 22 – Google (CMPL) - Segundo Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	129	39	168	297	207
ARMA(1,1)	139	10	149	288	159
MA(1)	154	0	154	308	154
ARIMA	133	22	155	288	177
AR(1)	146	7	153	299	160
Naive	99	99	198	297	297
CMPL(1,1)	141	15	156	297	171
CMPL(2,1)	128	45	173	301	218
CMPL(1,2)	149	6	155	304	161

Figura 23 – Google (CMPL) - Terceiro Cenário

Tabela 23 – Google (CMPL) - Terceiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	16	5	21	37	26
ARMA(1,1)	17	3	20	37	23
MA(1)	18	0	18	36	18
ARIMA	17	3	20	37	23
AR(1)	18	0	18	36	18
Naive	13	13	26	39	39
CMPL(1,1)	17	4	21	38	25
CMPL(2,1)	15	8	23	38	31
CMPL(1,2)	17	1	18	35	19

Figura 24 – Google (CMPL) - Quarto Cenário

Tabela 24 – Google (CMPL) - Quarto Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	14	4	18	32	22
ARMA(1,1)	15	1	16	31	17
MA(1)	15	0	15	30	15
ARIMA	15	1	16	31	17
AR(1)	15	0	15	30	15
Naive	10	10	20	30	30
CMPL(1,1)	14	3	17	31	20
CMPL(2,1)	12	8	20	32	28
CMPL(1,2)	14	1	15	29	16

do limite máximo de recursos ociosos (W). Esse valor deve estar entre a quantidade mínima e máxima de recursos ociosos gerados pelos modelos de previsão. Um número inferior ao limite mínimo torna o problema sem solução. Um valor superior ao limite máximo torna o problema mono-objetivo, pois nesse caso não existe preocupação com recursos ociosos. Nos experimentos, consideramos três valores para o parâmetro (W), representados pelas linhas pontilhadas. A linha inferior representa o primeiro quartil, considerando os valores ordenados do número de recursos superprovisionados, gerados por cada modelo de previsão, de acordo com suas previsões passadas. A linha do meio representa a mediana e a linha superior representa o terceiro quartil.

No primeiro cenário, Figura 25, podemos observar que a solução CMPLR(1) está no conjunto ótimo de Pareto. Por outro lado, a solução ultrapassou o limite máximo de recursos superprovisionados estipulado (linha pontilhada inferior). Neste caso, é necessário fazermos uma consideração: esse limite estabelecido não é um limite rígido, sendo apenas um valor de referência a ser perseguido, e sua ultrapassagem não inviabiliza a solução. Mais uma vez, devemos lembrar que, por se tratar de uma otimização estocástica, não há como garantir que o comportamento passado se repetirá de maneira exatamente igual no futuro. Na Tabela 25 podemos observar que o modelo ARMA apresentou o menor custo, seguido pelo modelo AR. A solução CMPLR(1) apresentou o terceiro menor custo. Por outro lado, no cenário Custo(2,1), a mesma solução apresentou o menor custo. No cenário Custo(1, 2) o modelo ARMA foi o melhor, seguido pelo modelo AR e pelo ETS. A solução CMPLR(1) foi a quarta melhor. Percebe-se, neste cenário, que o aumento do limite máximo de recursos superprovisionados não trouxe vantagens para a solução, pois as soluções CMPLR(2) e CMPLR(3) foram sempre inferiores à solução CMPLR(1).

A Figura 26 ilustra os resultados para o segundo cenário. Podemos observar que todos os modelos, exceto o Naive e o ARMA, e soluções estão no conjunto ótimo de Pareto. Na Tabela 26 podemos observar que a solução CMPLR(1) apresentou o melhor resultado para o cenário Custo(1, 1), ao lado do modelo ARIMA. No cenário Custo(2, 1), os modelos ETS e ARIMA apresentaram os menores custos, seguidos pelas soluções CMPLR(2) e CMPLR(3). No cenário Custo(1, 2) o modelo MA foi o melhor, seguido pela solução CMPLR(1).

Na Figura 27 podemos observar os resultados para o terceiro cenário. As soluções CMPLR(1), CMPLR(2) e CMPLR(3) estão no conjunto ótimo de Pareto. Observando a Tabela 27, notamos que a solução CMPLR(3) apresentou o menor custo para o cenário Custo(1,1). No cenário Custo(2,1), a solução CMPLR(3) apresentou o menor custo. Já no cenário Custo(1, 2) a solução CMPLR(1) apresentou o menor custo, ao lado do modelo ARMA.

No quarto cenário, ilustrado na Figura 28, podemos observar que a solução CMPLR(1) está no conjunto ótimo de Pareto. Por outro lado, as soluções CMPLR(2) e CMPLR(3) foram dominadas. Na Tabela 28 observamos que o modelo ARIMA apresentou o menor custo para o cenário Custo(1,1). No cenário Custo(2,1), os modelos ETS e ARIMA apresentaram o menor custo, seguidos pela solução CMPLR(3). Já no cenário Custo(1, 2) o modelo MA apresentou o

menor custo. A solução CMPLR(1) foi a quarta melhor.

5.3.2.2 *Série Nasa*

Nas Figuras 29, 30, 31 e 32 podemos observar os resultados para a série Nasa. No primeiro cenário, Figura 29, podemos observar que somente a solução CMPLR(2) está no conjunto ótimo de Pareto, ao lado dos modelos ETS, ARIMA, AR e MA. Na Tabela 29 podemos observar que o modelo ARIMA foi a melhor no cenário Custo(1, 1), seguido pela solução CMPLR(1). No cenário Custo(2, 1) a solução CMPLR(2) apresentou o menor custo. No cenário Custo(1, 2) o modelo ARIMA foi novamente o melhor, seguido pelo modelo ARMA e pela solução CMPLR(1).

A Figura 30 mostra os resultados para o segundo cenário, podemos observar que as soluções CMPLR(1) e CMPLR(2) estão no conjunto ótimo de Pareto. De acordo com a Tabela 30, o modelo ARIMA apresentou os melhores resultados para os cenários Custo(1, 1) e Custo(2, 1), seguido pela solução CMPLR(2). No cenário Custo(1, 2) o modelo MA foi o melhor, seguido pelo modelo ARIMA e pela solução CMPLR(1).

No terceiro cenário, Figura 31, as soluções CMPLR(1) e CMPLR(3) estão no conjunto ótimo de Pareto. Já a solução CMPLR(2) foi dominada pelo modelo ARIMA. De acordo com a Tabela 31, o modelo ARIMA apresentou o menor custo para o cenário Custo(1,1), seguido pela solução CMPLR(2). No cenário Custo(2,1), a solução CMPLR(3) apresentou o menor custo, ao lado do modelo ARIMA. Já no cenário Custo(1, 2) o modelo MA apresentou melhor desempenho, seguido pelo modelo ARIMA e pela solução CMPLR(1).

No quarto cenário, ilustrado na Figura 32, podemos observar que todas as soluções estão no conjunto ótimo de Pareto. Na Tabela 32 observamos que o modelo ARIMA apresentou o menor custo para o cenário Custo(1,1), seguido pela solução CMPLR(2). No cenário Custo(2,1), o modelo Naive apresentou o menor custo, seguido pela solução CMPLR(3). Já no cenário Custo(1, 2) o modelo MA apresentou o menor custo, seguido pelo modelo ARIMA e pela solução CMPLR(1).

5.3.2.3 *Série Copa do Mundo*

As Figuras 33, 34, 35 e 36 mostram os resultados para a série FIFA World Cup. Todos os cenários apresentam resultados similares. As soluções CMPLR(1), CMPLR(2) e CMPLR(3) estão no conjunto ótimo de Pareto, em todos os cenários. Observando as Tabelas 33, 34, 35 e 35, notamos que o modelo Naive foi a melhor no cenário Custo(1, 1), seguido de perto pela solução CMPLR(3). A mesma situação se repetiu no cenário Custo(2, 1). No cenário Custo(1, 2) a solução CMPLR(2) apresentou o menor custo.

Figura 25 – Clarknet (CMPLR) - Primeiro Cenário

Tabela 25 – Clarknet (CMPLR) - Primeiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	105	64	169	274	233
ARMA(1,1)	113	49	162	275	211
MA(1)	131	56	187	318	243
ARIMA	112	73	185	297	258
AR(1)	116	48	164	280	212
Naive	101	101	202	303	303
CMPLR(2)	103	70	173	276	243
CMPLR(3)	104	83	187	291	270
CMPLR(1)	103	66	169	272	235

Figura 26 – Clarknet (CMPLR) - Segundo Cenário

Tabela 26 – Clarknet (CMPLR) - Segundo Cenário

	Recursos sub- provisionados	Recursos super- provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	76	62	138	214	200
ARMA(1,1)	87	49	136	223	185
MA(1)	123	27	150	273	177
ARIMA	79	56	135	214	191
AR(1)	103	44	147	250	191
Naive	87	87	174	261	261
CMPLR(2)	77	61	138	215	199
CMPLR(3)	74	67	141	215	208
CMPLR(1)	87	48	135	222	183

Figura 27 – Clarknet (CMPLR) - Terceiro Cenário

Tabela 27 – Clarknet (CMPLR) - Terceiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	56	44	100	156	144
ARMA(1,1)	62	36	98	160	134
MA(1)	102	19	121	223	140
ARIMA	57	44	101	158	145
AR(1)	80	33	113	193	146
Naive	55	55	110	165	165
CMPLR(2)	57	41	98	155	139
CMPLR(3)	54	42	96	150	138
CMPLR(1)	64	35	99	163	134

Figura 28 – Clarknet (CMPLR) - Quarto Cenário

Tabela 28 – Clarknet (CMPLR) - Quarto Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	49	44	93	142	137
ARMA(1,1)	60	34	94	154	128
MA(1)	96	14	110	206	124
ARIMA	53	36	89	142	125
AR(1)	76	32	108	184	140
Naive	53	53	106	159	159
CMPLR(2)	57	41	98	155	139
CMPLR(3)	51	44	95	146	139
CMPLR(1)	74	29	103	177	132

Figura 29 – Nasa (CMPLR) - Primeiro Cenário

Tabela 29 – Nasa (CMPLR) - Primeiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	186	174	360	546	534
ARMA(1,1)	215	135	350	565	485
MA(1)	367	97	464	831	561
ARIMA	196	133	329	525	462
AR(1)	260	132	392	652	524
Naive	210	210	420	630	630
CMPLR(2)	170	183	353	523	536
CMPLR(3)	174	198	372	546	570
CMPLR(1)	202	145	347	549	492

Figura 30 – Nasa (CMPLR) - Segundo Cenário

Tabela 30 – Nasa (CMPLR) - Segundo Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	201	137	338	539	475
ARMA(1,1)	219	96	315	534	411
MA(1)	274	52	326	600	378
ARIMA	189	101	290	479	391
AR(1)	224	94	318	542	412
Naive	168	168	336	504	504
CMPLR(2)	188	114	302	490	416
CMPLR(3)	188	138	326	514	464
CMPLR(1)	223	88	311	534	399

Figura 31 – Nasa (CMPLR) - Terceiro Cenário

Tabela 31 – Nasa (CMPLR) - Terceiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	113	123	236	349	359
ARMA(1,1)	159	80	239	398	319
MA(1)	249	15	264	513	279
ARIMA	134	74	208	342	282
AR(1)	178	64	242	420	306
Naive	121	121	242	363	363
CMPLR(2)	136	79	215	351	294
CMPLR(3)	115	112	227	342	339
CMPLR(1)	191	46	237	428	283

Figura 32 – Nasa (CMPLR) - Quarto Cenário

Tabela 32 – Nasa (CMPLR) - Quarto Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	158	91	249	407	340
ARMA(1,1)	203	44	247	450	291
MA(1)	266	3	269	535	272
ARIMA	182	48	230	412	278
AR(1)	210	39	249	459	288
Naive	121	121	242	363	363
CMPLR(2)	172	62	234	406	296
CMPLR(3)	146	92	238	384	330
CMPLR(1)	203	42	245	448	287

Figura 33 – Copa do Mundo (CMPLR) - Primeiro Cenário

Tabela 33 – Copa do Mundo (CMPLR) - Primeiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	32403	27511	59914	92317	87425
ARMA(1,1)	39412	21283	60695	100107	81978
MA(1)	112525	5475	118000	230525	123475
ARIMA	35674	24295	59969	95643	84264
AR(1)	39505	18038	57543	97048	75581
Naive	26756	26754	53510	80266	80264
CMPLR(2)	41120	16616	57736	98856	74352
CMPLR(3)	32144	22367	54511	86655	76878
CMPLR(1)	69271	8486	77757	147028	86243

Figura 34 – Copa do Mundo (CMPLR) - Segundo Cenário

Tabela 34 – Copa do Mundo (CMPLR) - Segundo Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	28161	23910	52071	80232	75981
ARMA(1,1)	34254	18500	52754	87008	71254
MA(1)	97827	4765	102592	200419	107357
ARIMA	30994	21138	52132	83126	73270
AR(1)	34320	15682	50002	84322	65684
Naive	23260	23258	46518	69778	69776
CMPLR(2)	35746	14468	50214	85960	64682
CMPLR(3)	27911	19444	47355	75266	66799
CMPLR(1)	60093	7408	67501	127594	74909

Figura 35 – Copa do Mundo (CMPLR) - Terceiro Cenário

Tabela 35 – Copa do Mundo (CMPLR) - Terceiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	14722	12488	27210	41932	39698
ARMA(1,1)	17894	9669	27563	45457	37232
MA(1)	51165	2504	53669	104834	56173
ARIMA	16194	11042	27236	43430	38278
AR(1)	17934	8202	26136	44070	34338
Naive	12151	12150	24301	36452	36451
CMPLR(2)	18666	7551	26217	44883	33768
CMPLR(3)	14582	10138	24720	39302	34858
CMPLR(1)	31451	3860	35311	66762	39171

Figura 36 – Copa do Mundo (CMPLR) - Quarto Cenário

Tabela 36 – Copa do Mundo (CMPLR) - Quarto Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	13807	11723	25530	39337	37253
ARMA(1,1)	16767	9072	25839	42606	34911
MA(1)	47867	2329	50196	98063	52525
ARIMA	15176	10338	25514	40690	35852
AR(1)	16808	7690	24498	41306	32188
Naive	11387	11386	22773	34160	34159
CMPLR(2)	17527	7057	24584	42111	31641
CMPLR(3)	13687	9499	23186	36873	32685
CMPLR(1)	29536	3589	33125	62661	36714

5.3.2.4 Série Google

Nas Figuras 37, 38, 39 e 40 estão ilustrados os resultados para a série Google *job*. No primeiro cenário, Figura 37, notamos que praticamente todas as soluções e modelos estão no conjunto ótimo de Pareto. De acordo com a Tabela 37 o modelo MA foi a melhor no cenário Custo(1, 1), seguido pelo modelo AR e pela solução CMPLR(1). No cenário Custo(2, 1) a solução CMPLR(3) apresentou o menor custo, seguida pela solução CMPLR(2). No cenário Custo(1, 2) o modelo MA apresentou o menor custo. A solução CMPLR(1) foi a quarta melhor.

A Figura 38 mostra os resultados para o segundo cenário, onde podemos observar que a solução CMPLR(1) foi dominada pelo modelo ARIMA. Observando a Tabela 38, notamos que o modelo ARMA apresentou os melhores resultados para o cenário Custo(1, 1). A solução CMPLR(1) apresentou o quinto menor custo. No cenário Custo(2, 1), os modelos ARMA e ARIMA apresentaram os menores custos, seguidos pelas soluções CMPLR(2) e CMPLR(3). No cenário Custo(1, 2) o modelo MA foi o melhor, seguido pelos modelos AR e ARMA. A solução CMPLR(1) foi a quinta melhor.

No terceiro cenário, Figura 39, as soluções CMPLR(2) e CMPLR(3) foram dominadas. Já a solução CMPLR(1) está no conjunto ótimo de Pareto. De acordo com a Tabela 39, a solução CMPLR(1) foi a melhor em todos os cenários.

O quarto cenário, ilustrado na Figura 40 e na Tabela 40, é, praticamente, uma repetição do cenário anterior.

De acordo com os resultados observados para o algoritmo CMPLR, podemos perceber que, nem sempre, aumentar a quantidade de recursos superprovisionados tolerada leva a uma diminuição da quantidade de recursos subprovisionados. Por exemplo, nem sempre a solução CMPLR(2) resultou em uma quantidade menor de recursos subprovisionados do que a solução CMPLR(1), como era esperado. Isso se deve ao fato de, como já dito em outros pontos desta Tese, estarmos trabalhando com otimização estocástica. Outro ponto a ser considerado no algoritmo é a dificuldade de definir um limiar para um dos objetivos (neste caso a quantidade de recursos superprovisionados). Como pudemos observar nos resultados, às vezes, aumentar o limite de um dos objetivos não gera nem um benefício, pois nem sempre o outro objetivo é melhorado, como era de se esperar. Desse modo, como continuação deste trabalho, poderia ser considerado o ajuste dinâmico desse limiar.

5.3.3 Algoritmo CMAHP

5.3.3.1 Série Clarknet

As Figuras 41, 42, 43 e 44 mostram os resultados para a série ClarkNet. Para cada cenário foram executados experimentos considerando: a mesma prioridade entre recursos sub e superprovisionados (CMAHP(1,1)); forte prioridade para recursos subprovisionados (CMAHP(5,1)); e forte prioridade para recursos superprovisionados (CMAHP(1,5)).

Figura 37 – Google (CMPLR) - Primeiro Cenário

Tabela 37 – Google (CMPLR) - Primeiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	193	56	249	442	305
ARMA(1,1)	214	27	241	455	268
MA(1)	229	6	235	464	241
ARIMA	200	47	247	447	294
AR(1)	219	19	238	457	257
Naive	151	151	302	453	453
CMPLR(2)	176	77	253	429	330
CMPLR(3)	162	97	259	421	356
CMPLR(1)	203	42	245	448	287

Figura 38 – Google (CMPLR) - Segundo Cenário

Tabela 38 – Google (CMPLR) - Segundo Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	129	39	168	297	207
ARMA(1,1)	139	10	149	288	159
MA(1)	154	0	154	308	154
ARIMA	133	22	155	288	177
AR(1)	146	6	152	298	158
Naive	99	99	198	297	297
CMPLR(2)	125	43	168	293	211
CMPLR(3)	112	69	181	293	250
CMPLR(1)	134	27	161	295	188

Figura 39 – Google (CMPLR) - Terceiro Cenário

Tabela 39 – Google (CMPLR) - Terceiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	16	5	21	37	26
ARMA(1,1)	17	3	20	37	23
MA(1)	18	0	18	36	18
ARIMA	17	3	20	37	23
AR(1)	18	0	18	36	18
Naive	13	13	26	39	39
CMPLR(2)	15	3	18	33	21
CMPLR(3)	15	6	21	36	27
CMPLR(1)	15	1	16	31	17

Figura 40 – Google (CMPLR) - Quarto Cenário

Tabela 40 – Google (CMPLR) - Quarto Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	14	4	18	32	22
ARMA(1,1)	15	1	16	31	17
MA(1)	15	0	15	30	15
ARIMA	15	1	16	31	17
AR(1)	15	0	15	30	15
Naive	10	10	20	30	30
CMPLR(2)	12	2	14	26	16
CMPLR(3)	12	5	17	29	22
CMPLR(1)	12	1	13	25	14

No primeiro cenário, Figura 41, podemos observar que as soluções CMAHP(1,1) e CMAHP(1,5) estão no conjunto ótimo de Pareto. Por outro lado, a solução CMAHP(2,1) foi dominada pela solução CMAHP(1,1). Na Tabela 41 podemos observar o custo gerado por cada solução, considerando os seguintes cenários: Custo(1, 1), onde o preço de um recurso subprovisionado é o mesmo de um recurso superprovisionado e igual a 1; Custo(2, 1), onde o preço de um recurso subprovisionado é igual a 2 e o preço de um recurso superprovisionado é igual a 1; e Custo(1, 2), onde o preço de um recurso subprovisionado é igual a 1 e o preço de um recurso superprovisionado é igual a 2. O modelo ARMA apresentou o menor custo, seguido pela solução CMAHP(1,5), no cenário Custo(1, 1). A solução CMAHP(1,1) foi a quarta melhor. No cenário Custo(2, 1), a solução CMAHP(1,5) apresentou o menor custo, seguida pela solução CMAHP(1,1). A solução CMAHP(5,1) apresentou o sexto menor custo. Já no cenário Custo(1, 2) o modelo ARMA foi novamente o melhor, seguido pelo modelo AR e pela solução CMAHP(1,5).

A Figura 42 ilustra os resultados para o segundo cenário, onde podemos observar que todas as soluções CMAHP estão no conjunto ótimo de Pareto. Os modelos Naive, ETS e AR(1) foram dominados neste cenário. Na Tabela 42 podemos observar o custo gerado por cada solução. A solução CMAHP(5,1) apresentou o menor custo para os cenários Custo(1, 1) e Custo(2, 1). A solução CMAHP(1,1) apresentou o quarto menor custo no cenário Custo(1, 1). No cenário Custo(1, 2) o modelo MA obteve melhor desempenho, seguido pela solução CMAHP(1,5).

Na Figura 43 podemos observar os resultados para o terceiro cenário. As soluções CMAHP(5,1) e CMAHP(1,1) estão no conjunto ótimo de Pareto. Já a solução CMAHP(5,1) foi dominada pelo modelo ARMA. Os modelos Naive, ETS e ARIMA também foram dominados no cenário. Observando a Tabela 43 notamos que a solução CMAHP(5,1) apresentou o menor custo, ao lado do modelo ARMA, para o cenário Custo(1,1). A solução CMAHP(1,1) apresentou o segundo menor custo. No cenário Custo(2,1), a solução CMAHP(5,1) apresentou o menor custo. Já no cenário Custo(1, 2) o modelo ARMA apresentou melhor desempenho. A solução CMAHP(1,5) apresentou o segundo menor custo.

No quarto cenário, ilustrado na Figura 44, podemos observar que as soluções CMAHP(5,1) e CMAHP(1,5) estão no conjunto ótimo de Pareto, e a solução CMAHP(1,1) foi dominada pelo modelo ARIMA. Os modelos Naive e AR(1) também foram dominados. Na Tabela 44 observamos que o modelo ARIMA apresentou o menor custo para o cenário Custo(1,1). A solução CMAHP(1,1) apresentou o terceiro menor custo. No cenário Custo(2,1), os modelos ETS e ARIMA apresentaram o menor custo, seguidos pela solução CMAHP(5,1). Já no cenário Custo(1, 2) o modelo MA apresentou o menor custo. A solução CMAHP(1,5) não obteve bom desempenho, superando, apenas, os modelos Naive e AR.

Figura 41 – Clarknet (CMAHP) - Primeiro Cenário

Tabela 41 – Clarknet (CMAHP) - Primeiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	105	64	169	274	233
ARMA(1,1)	113	49	162	275	211
MA(1)	131	56	187	318	243
ARIMA	112	73	185	297	258
AR(1)	110	56	166	276	222
Naive	101	101	202	303	303
CMAHP(1,1)	104	64	168	272	232
CMAHP(5,1)	105	72	177	282	249
CMAHP(1,5)	106	59	165	271	224

Figura 42 – Clarknet (CMAHP) - Segundo Cenário

Tabela 42 – Clarknet (CMAHP) - Segundo Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	76	62	138	214	200
ARMA(1,1)	87	49	136	223	185
MA(1)	123	27	150	273	177
ARIMA	79	56	135	214	191
AR(1)	101	46	147	248	193
Naive	87	87	174	261	261
CMAHP(1,1)	84	54	138	222	192
CMAHP(5,1)	72	59	131	203	190
CMAHP(1,5)	97	43	140	237	183

Figura 43 – Clarknet (CMAHP) - Terceiro Cenário

Tabela 43 – Clarknet (CMAHP) - Terceiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	56	44	100	156	144
ARMA(1,1)	62	36	98	160	134
MA(1)	102	19	121	223	140
ARIMA	57	44	101	158	145
AR(1)	69	35	104	173	139
Naive	55	55	110	165	165
CMAHP(1,1)	59	40	99	158	139
CMAHP(5,1)	54	44	98	152	142
CMAHP(1,5)	67	36	103	170	139

Figura 44 – Clarknet (CMAHP) - Quarto Cenário

Tabela 44 – Clarknet (CMAHP) - Quarto Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	49	44	93	142	137
ARMA(1,1)	60	34	94	154	128
MA(1)	96	14	110	206	124
ARIMA	53	36	89	142	125
AR(1)	75	34	109	184	143
Naive	53	53	106	159	159
CMAHP(1,1)	57	37	94	151	131
CMAHP(5,1)	52	43	95	147	138
CMAHP(1,5)	75	32	107	182	139

5.3.3.2 Série Nasa

Nas Figuras 45, 46, 47 e 48 podemos observar os resultados para a série Nasa. As Tabelas 45, 46, 47 e 48 ilustram a quantidade de recursos sub e superprovisionados para a série.

No primeiro cenário, Figura 45, podemos observar que somente a solução CMAHP(5,1) está no conjunto ótimo de Pareto, ao lado dos modelos ARIMA e MA. Na Tabela 45 podemos observar que o modelo ARIMA foi a melhor no cenário Custo(1, 1), seguido pelo modelo ARMA e pela solução CMAHP(1,1). No cenário Custo(2, 1) o modelo ARIMA, novamente, apresentou o menor custo, seguido pela solução CMAHP(5,1). No cenário Custo(1, 2) o modelo ARIMA foi novamente o melhor. A solução CMAHP(1,5) apresentou o quarto menor custo.

A Figura 46 mostra os resultados para o segundo cenário, podemos observar que somente a solução CMAHP(1,5) está no conjunto ótimo de Pareto, ao lado dos modelos Naive, ARIMA, ARMA(1,1) e MA(1). De acordo com a Tabela 46, o modelo ARIMA apresentou os melhores resultados para o cenário Custo(1, 1), seguido pela solução CMAHP(1,1). A mesma situação se repete no cenário Custo(2, 1). No cenário Custo(1, 2) o modelo MA foi o melhor, seguido pela solução CMAHP(1,5).

No terceiro cenário, Figura 47, as soluções CMAHP(5,1) e CMAHP(1,5) estão no conjunto ótimo de Pareto. Já a solução CMAHP(1,1) foi dominada pelo modelo ARIMA. De acordo com a Tabela 47, o modelo ARIMA apresentou o menor custo para o cenário Custo(1,1), seguido pela solução CMAHP(1,1). No cenário Custo(2,1), o modelo ARIMA, novamente, apresentou o menor custo, seguido pelo modelo ETS e pela solução CMAHP(5,1). Já no cenário Custo(1, 2) o modelo MA apresentou melhor desempenho, seguido pelo modelo ARIMA e pela solução CMAHP(1,5).

No quarto cenário, ilustrado na Figura 48, podemos observar que, assim como no cenário anterior, as soluções CMAHP(5,1) e CMAHP(1,5) estão no conjunto ótimo de Pareto. Na Tabela 48 observamos que o modelo ARIMA apresentou o menor custo para o cenário Custo(1,1), seguido pela solução CMAHP(5,1). A solução CMAHP(1,1) apresentou o quarto menor custo. No cenário Custo(2,1), o modelo Naive apresentou o menor custo, seguido pela solução CMAHP(5,1). Já no cenário Custo(1, 2) o modelo MA apresentou o menor custo, seguido pela solução CMAHP(1,5).

5.3.3.3 Série Copa do Mundo

As Figuras 49, 50, 51 e 52 mostram os resultados para a série FIFA World Cup. Podemos observar que em todos os cenários os resultados foram bastante similares. As soluções CMAHP(1,1), CMAHP(5,1) e CMAHP(1,5) estão no conjunto ótimo de Pareto, ao lado dos modelos Naive, AR e MA. Observando as Tabelas 49, 50, 51 e 52, notamos que o modelo Naive foi o melhor no cenário Custo(1, 1), seguido pela solução CMAHP(5,1). A solução CMAHP(1,1) não apresentou bom desempenho, superando apenas a solução CMAHP(1,5) e o modelo MA(1).

Figura 45 – Nasa (CMAHP) - Primeiro Cenário

Tabela 45 – Nasa (CMAHP) - Primeiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	186	174	360	546	534
ARMA(1,1)	215	135	350	565	485
MA(1)	367	97	464	831	561
ARIMA	196	133	329	525	462
AR(1)	187	173	360	547	533
Naive	210	210	420	630	630
CMAHP(1,1)	205	146	351	556	497
CMAHP(5,1)	179	173	352	531	525
CMAHP(1,5)	238	133	371	609	504

Figura 46 – Nasa (CMAHP) - Segundo Cenário

Tabela 46 – Nasa (CMAHP) - Segundo Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	201	137	338	539	475
ARMA(1,1)	219	96	315	534	411
MA(1)	274	52	326	600	378
ARIMA	189	101	290	479	391
AR(1)	202	108	310	512	418
Naive	168	168	336	504	504
CMAHP(1,1)	194	110	304	498	414
CMAHP(5,1)	194	127	321	515	448
CMAHP(1,5)	223	84	307	530	391

Figura 47 – Nasa (CMAHP) - Terceiro Cenário

Tabela 47 – Nasa (CMAHP) - Terceiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	113	123	236	349	359
ARMA(1,1)	159	80	239	398	319
MA(1)	249	15	264	513	279
ARIMA	134	74	208	342	282
AR(1)	162	74	236	398	310
Naive	121	121	242	363	363
CMAHP(1,1)	136	93	229	365	322
CMAHP(5,1)	121	117	238	359	355
CMAHP(1,5)	174	61	235	409	296

Figura 48 – Nasa (CMAHP) - Quarto Cenário

Tabela 48 – Nasa (CMAHP) - Quarto Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	158	91	249	407	340
ARMA(1,1)	203	44	247	450	291
MA(1)	266	3	269	535	272
ARIMA	182	48	230	412	278
AR(1)	207	41	248	455	289
Naive	121	121	242	363	363
CMAHP(1,1)	190	54	244	434	298
CMAHP(5,1)	163	71	234	397	305
CMAHP(1,5)	220	28	248	468	276

No cenário Custo(2, 1), o modelo Naive foi novamente o melhor, desta vez seguido pelo modelo ETS. A solução CMAHP(5,1) apresentou o terceiro menor custo. No cenário Custo(1, 2) o modelo AR foi o melhor, seguido pela solução CMAHP(1,1). A solução CMAHP(1,5) apresentou o sétimo menor custo.

5.3.3.4 Série Google

As Figuras 53, 54, 55 e 56 ilustram os resultados para a série Google *job*. Nas Tabelas 53, 54, 55 e 56 podemos observar a quantidade de recursos sub e superprovisionados gerados por cada modelo e solução.

No primeiro cenário, Figura 53, notamos que todas as soluções e modelos estão no conjunto ótimo de Pareto. Na Tabela 53 podemos observar que o modelo MA foi a melhor no cenário Custo(1, 1). A solução CMAHP(1,1) apresentou o sexto menor custo. No cenário Custo(2, 1) a solução CMAHP(5,1) apresentou o menor custo. No cenário Custo(1, 2) o modelo MA apresentou o menor custo. A solução CMAHP(1,5) apresentou o quinto menor custo.

A Figura 54 mostra os resultados para o segundo cenário, no qual podemos observar que a solução CMAHP(1,5) é a única fora do conjunto ótimo de Pareto. De acordo com a Tabela 54, o modelo ARMA apresentou o melhor resultado para o cenário Custo(1, 1), seguido pela solução CMAHP(1,5). A solução CMAHP(1,1) apresentou o sexto menor custo. No cenário Custo(2, 1), os modelos ARMA e ARIMA apresentaram os menores custos, seguidos pela solução CMAHP(5,1). No cenário Custo(1, 2) o modelo MA foi o melhor, seguido pelos modelos ARMA, AR e pela solução CMAHP(1,5).

No terceiro cenário, Figura 55, as soluções CMAHP(1,1) e CMAHP(1,5) estão no conjunto ótimo de Pareto, ao lado dos modelos Naive e AR(1). Já a solução CMAHP(5,1) foi dominada. De acordo com a Tabela 55, a solução CMAHP(1,1) foi a melhor para os cenários Custo(1,1) e Custo(2,1). Nesse último cenário a solução CMAHP(5,1) apresentou o segundo menor custo. Já no cenário Custo(1, 2) os modelos MA e AR apresentaram os menores custos, seguidos pela solução CMAHP(1,1). A solução CMAHP(1,5) apresentou o quarto menor custo.

O quarto cenário, ilustrado na Figura 56, percebemos que as soluções CMAHP(1,1) e CMAHP(1,5) estão fora do conjunto ótimo de Pareto, que é composto, além da solução CMAHP(5,1), pelos modelos Naive, AR(1) e MA(1). Na Tabela 56, podemos observar que a solução CMAHP(5,1) foi a melhor nos cenários Custo(1, 1) e Custo(2, 1). A solução CMAHP(1,1) apresentou o quarto menor custo no cenário Custo(1, 1). No cenário Custo(1, 2) os modelos MA e AR apresentaram o menor custo, seguidos de perto pela solução CMAHP(5,1). A solução CMAHP(1,5) apresentou o quarto menor custo.

De acordo com os resultados observados para o algoritmo CMAHP, podemos perceber, assim como nos casos anteriores, que nem sempre a preferência do usuário em relação a recursos sub e superprovisionados se traduz no resultado final. Por exemplo, nem sempre a solução

Figura 49 – Copa do Mundo (CMAHP) - Primeiro Cenário

Tabela 49 – Copa do Mundo (CMAHP) - Primeiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	32403	27511	59914	92317	87425
ARMA(1,1)	39412	21283	60695	100107	81978
MA(1)	112525	5475	118000	230525	123475
ARIMA	35674	24295	59969	95643	84264
AR(1)	39446	18066	57512	96958	75578
Naive	26756	26754	53510	80266	80264
CMAHP(1,1)	48567	14204	62771	111338	76975
CMAHP(5,1)	35651	21352	57003	92654	78355
CMAHP(1,5)	63788	9377	73165	136953	82542

Figura 50 – Copa do Mundo (CMAHP) - Segundo Cenário

Tabela 50 – Copa do Mundo (CMAHP) - Segundo Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	28161	23910	52071	80232	75981
ARMA(1,1)	34254	18500	52754	87008	71254
MA(1)	97827	4765	102592	200419	107357
ARIMA	30994	21138	52132	83126	73270
AR(1)	34299	15698	49997	84296	65695
Naive	23260	23258	46518	69778	69776
CMAHP(1,1)	42209	12390	54599	96808	66989
CMAHP(5,1)	30977	18566	49543	80520	68109
CMAHP(1,5)	55463	8187	63650	119113	71837

Figura 51 – Copa do Mundo (CMAHP) - Terceiro Cenário

Tabela 51 – Copa do Mundo (CMAHP) - Terceiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	14722	12488	27210	41932	39698
ARMA(1,1)	17894	9669	27563	45457	37232
MA(1)	51165	2504	53669	104834	56173
ARIMA	16194	11042	27236	43430	38278
AR(1)	17910	8211	26121	44031	34332
Naive	12151	12150	24301	36452	36451
CMAHP(1,1)	22067	6451	28518	50585	34969
CMAHP(5,1)	16171	9696	25867	42038	35563
CMAHP(1,5)	29004	4261	33265	62269	37526

Figura 52 – Copa do Mundo (CMAHP) - Quarto Cenário

Tabela 52 – Copa do Mundo (CMAHP) - Quarto Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	13807	11723	25530	39337	37253
ARMA(1,1)	16767	9072	25839	42606	34911
MA(1)	47867	2329	50196	98063	52525
ARIMA	15176	10338	25514	40690	35852
AR(1)	16790	7696	24486	41276	32182
Naive	11387	11386	22773	34160	34159
CMAHP(1,1)	20681	6035	26716	47397	32751
CMAHP(5,1)	15175	9094	24269	39444	33363
CMAHP(1,5)	27151	3989	31140	58291	35129

CMAHP(5,1) foi melhor que a CMAHP(1,1) no cenário Custo(2,1), como era esperado. Além disso, existe a dificuldade de definir as prioridades entre recursos sub e superprovisionados. Isso se deve ao fato dos valores usados pelo método de análise hierárquica (CMAHP) para definir tais prioridades são um pouco subjetivos. Assim como nos casos anteriores (CMPL e CMPLR), como trabalhos futuros, poderia ser considerado o ajuste dinâmico dessas prioridades.

5.3.4 Algoritmo CMAGM

5.3.4.1 Série Clarknet

As Figuras 57, 58, 59 e 60 mostram os resultados para a série ClarkNet. Nas Tabelas 57, 58, 59 e 60 podemos observar a quantidade de recursos sub e superprovisionados gerados na série.

No primeiro cenário, Figura 57, podemos observar que a solução CMAGM(1,1) e CMAGM(1,2) estão no conjunto ótimo de Pareto, ao lado dos modelos Naive e ARMA. Por outro lado, a solução CMAGM(2,1) foi dominada pela solução CMAGM(1,1). Na Tabela 57 podemos observar que a solução CMAGM(1,1) foi a melhor nos cenários Custo(1, 1) e Custo(2, 1). Nesse último cenário a solução CMAGM(2,1) apresentou o terceiro menor custo. No cenário Custo(1, 2) o modelo ARMA foi o melhor, seguido pela solução CMAGM(1,2).

A Figura 58 ilustra os resultados para o segundo cenário, podemos observar que a solução CMAGM(1,2) está no conjunto ótimo de Pareto. Por outro lado, as soluções CMAGM(1,1) e CMAGM(1,2) foram dominadas. Na Tabela 58 podemos observar o custo gerado por cada solução. O modelo ARIMA apresentou os melhores resultados para o cenário Custo(1, 1), seguido pelo modelo ARMA e pela solução CMAGM(1,2). No cenário Custo(2, 1), os modelos ETS e ARIMA apresentaram os menores custos, seguidos pela solução CMAGM(2,1). No cenário Custo(1, 2) o modelo MA foi o melhor, seguido pela solução CMAGM(1,2) e pelo modelo ARMA.

Na Figura 59 podemos observar os resultados para o terceiro cenário. A solução CMAGM(2,1) está no conjunto ótimo de Pareto, enquanto as soluções CMAGM(1,1) e CMAGM(1,2) foram dominadas. Observando a Tabela 59 notamos que a solução CMAGM(2,1) apresentou o menor custo para os cenários Custo(1,1) e Custo(2,1). Já no cenário Custo(1, 2) o modelo ARMA apresentou melhor desempenho. A solução CMAGM(2,1) apresentou o terceiro menor custo.

No quarto cenário, ilustrado na Figura 60, podemos observar que as soluções CMAGM(1,1) e CMAGM(1,2) estão no conjunto ótimo de Pareto, e a solução CMAGM(2,1) foi dominada pela solução CMAGM(1,1). Na Tabela 60 observamos que o modelo ARIMA apresentou o menor custo para o cenário Custo(1,1), seguido pela solução CMAGM(1,1). No cenário Custo(2,1), os modelos ETS e ARIMA apresentaram o menor custo, seguidos pela solução CMAGM(1,1). Já no cenário Custo(1, 2) o modelo MA apresentou o menor custo, seguido pelo modelo ARIMA e

Figura 53 – Google (CMAHP) - Primeiro Cenário

Tabela 53 – Google (CMAHP) - Primeiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	193	56	249	442	305
ARMA(1,1)	214	27	241	455	268
MA(1)	229	6	235	464	241
ARIMA	200	47	247	447	294
AR(1)	212	30	242	454	272
Naive	151	151	302	453	453
CMAHP(1,1)	189	60	249	438	309
CMAHP(5,1)	167	87	254	421	341
CMAHP(1,5)	210	34	244	454	278

Figura 54 – Google (CMAHP) - Segundo Cenário

Tabela 54 – Google (CMAHP) - Segundo Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	129	39	168	297	207
ARMA(1,1)	139	10	149	288	159
MA(1)	154	0	154	308	154
ARIMA	133	22	155	288	177
AR(1)	146	7	153	299	160
Naive	99	99	198	297	297
CMAHP(1,1)	130	34	164	294	198
CMAHP(5,1)	120	50	170	290	220
CMAHP(1,5)	142	11	153	295	164

Figura 55 – Google (CMAHP) - Terceiro Cenário

Tabela 55 – Google (CMAHP) - Terceiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	16	5	21	37	26
ARMA(1,1)	17	3	20	37	23
MA(1)	18	0	18	36	18
ARIMA	17	3	20	37	23
AR(1)	18	0	18	36	18
Naive	13	13	26	39	39
CMAHP(1,1)	15	2	17	32	19
CMAHP(5,1)	15	3	18	33	21
CMAHP(1,5)	18	1	19	37	20

Figura 56 – Google (CMAHP) - Quarto Cenário

Tabela 56 – Google (CMAHP) - Quarto Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	14	4	18	32	22
ARMA(1,1)	15	1	16	31	17
MA(1)	15	0	15	30	15
ARIMA	15	1	16	31	17
AR(1)	15	0	15	30	15
Naive	10	10	20	30	30
CMAHP(1,1)	14	2	16	30	18
CMAHP(5,1)	12	2	14	26	16
CMAHP(1,5)	15	1	16	31	17

pela solução CMAGM(1,2).

5.3.4.2 *Série Nasa*

Nas Figuras 61, 62, 63 e 64 podemos observar os resultados para a série Nasa. No primeiro cenário, a Figura 61, podemos observar que todas as soluções estão no conjunto ótimo de Pareto, ao lado dos modelos ARIMA e MA. Na Tabela 61 podemos observar que o modelo ARIMA foi a melhor no cenário Custo(1, 1), seguido pela solução CMAGM(1,1). No cenário Custo(2, 1) a solução CMAGM(1,1) apresentou o menor custo. No cenário Custo(1, 2) o modelo ARIMA foi novamente o melhor, seguido pela solução CMAGM(1,2).

A Figura 62 mostra os resultados para o segundo cenário, podemos observar que as soluções CMAGM(1,1), CMAGM(1,2) estão no conjunto ótimo de Pareto. Observando a Tabela 62, o modelo ARIMA apresentou os melhores resultados para o cenário Custo(1, 1), seguido pela solução CMAGM(1,1). No cenário Custo(2, 1), a solução CMAGM(1,1) apresentou o menor custo, seguida pelo modelo ARIMA. No cenário Custo(1, 2) o modelo MA foi o melhor, seguido pelo modelo ARIMA e pela solução CMAGM(1,2).

No terceiro cenário, Figura 63, todas as soluções estão no conjunto ótimo de Pareto. De acordo com a Tabela 63, o modelo ARIMA apresentou o menor custo para o cenário Custo(1,1), seguido pela solução CMAGM(1,1). O mesmo se repetiu no cenário Custo(2,1). Já no cenário Custo(1, 2) o modelo MA apresentou melhor desempenho, seguido pelo modelo ARIMA e pela solução CMAGM(1,2).

No quarto cenário, ilustrado na Figura 64, podemos observar que as soluções CMAGM(1,1), CMAGM(2,1) e CMAGM(1,2) estão no conjunto ótimo de Pareto. Na Tabela 64 observamos que, assim como no cenário anterior, o modelo ARIMA apresentou o menor custo para o cenário Custo(1,1), seguido pela solução CMAGM(1,1). No cenário Custo(2,1), o modelo Naive apresentou o menor custo, seguido pelas soluções CMAGM(1,1) e CMAGM(2,1). Já no cenário Custo(1, 2) a solução CMAGM(1,2) apresentou o menor custo.

5.3.4.3 *Série Copa do Mundo*

As Figuras 65, 66, 67 e 68 mostram os resultados para a série FIFA World Cup. A Figura 65 ilustra os resultados para o primeiro cenário. Podemos observar que as soluções CMAGM(1,1), CMAGM(2,1) e CMAGM(1,2) estão no conjunto ótimo de Pareto. Observando a Tabela 65, notamos que o modelo Naive foi a melhor no cenário Custo(1, 1), seguido de perto pela solução CMAGM(1,1). O modelo Naive foi novamente o melhor no cenário Custo(2, 1), desta vez seguido pela solução CMAGM(2,1). No cenário Custo(1, 2) o modelo AR foi o melhor, seguido pela solução CMAGM(1,2).

No segundo cenário, Figura 66, notamos uma situação muito parecida a do primeiro cenário. A diferença é que, de acordo com a Tabela 66, podemos observar que, no cenário

Figura 57 – Clarknet (CMAGM) - Primeiro Cenário

Tabela 57 – Clarknet (CMAGM) - Primeiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	105	64	169	274	233
ARMA(1,1)	113	49	162	275	211
MA(1)	131	56	187	318	243
ARIMA	112	73	185	297	258
AR(1)	110	56	166	276	222
Naive	101	101	202	303	303
CMAGM(1,1)	102	60	162	264	222
CMAGM(2,1)	104	65	169	273	234
CMAGM(1,2)	108	56	164	272	220

Figura 58 – Clarknet (CMAGM) - Segundo Cenário

Tabela 58 – Clarknet (CMAGM) - Segundo Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	76	62	138	214	200
ARMA(1,1)	87	49	136	223	185
MA(1)	123	27	150	273	177
ARIMA	79	56	135	214	191
AR(1)	101	46	147	248	193
Naive	87	87	174	261	261
CMAGM(1,1)	85	55	140	225	195
CMAGM(2,1)	79	62	141	220	203
CMAGM(1,2)	91	47	138	229	185

Figura 59 – Clarknet (CMAGM) - Terceiro Cenário

Tabela 59 – Clarknet (CMAGM) - Terceiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	56	44	100	156	144
ARMA(1,1)	62	36	98	160	134
MA(1)	102	19	121	223	140
ARIMA	57	44	101	158	145
AR(1)	69	35	104	173	139
Naive	55	55	110	165	165
CMAGM(1,1)	54	45	99	153	144
CMAGM(2,1)	52	44	96	148	140
CMAGM(1,2)	69	37	106	175	143

Figura 60 – Clarknet (CMAGM) - Quarto Cenário

Tabela 60 – Clarknet (CMAGM) - Quarto Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	49	44	93	142	137
ARMA(1,1)	60	34	94	154	128
MA(1)	96	14	110	206	124
ARIMA	53	36	89	142	125
AR(1)	75	34	109	184	143
Naive	53	53	106	159	159
CMAGM(1,1)	50	43	93	143	136
CMAGM(2,1)	52	47	99	151	146
CMAGM(1,2)	69	29	98	167	127

Figura 61 – Nasa (CMAGM) - Primeiro Cenário

Tabela 61 – Nasa (CMAGM) - Primeiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	186	174	360	546	534
ARMA(1,1)	215	135	350	565	485
MA(1)	367	97	464	831	561
ARIMA	196	133	329	525	462
AR(1)	187	173	360	547	533
Naive	210	210	420	630	630
CMAGM(1,1)	176	158	334	510	492
CMAGM(2,1)	175	164	339	514	503
CMAGM(1,2)	190	147	337	527	484

Figura 62 – Nasa (CMAGM) - Segundo Cenário

Tabela 62 – Nasa (CMAGM) - Segundo Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	201	137	338	539	475
ARMA(1,1)	219	96	315	534	411
MA(1)	274	52	326	600	378
ARIMA	189	101	290	479	391
AR(1)	202	108	310	512	418
Naive	168	168	336	504	504
CMAGM(1,1)	178	113	291	469	404
CMAGM(2,1)	180	125	305	485	430
CMAGM(1,2)	216	89	305	521	394

Figura 63 – Nasa (CMAGM) - Terceiro Cenário

Tabela 63 – Nasa (CMAGM) - Terceiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	113	123	236	349	359
ARMA(1,1)	159	80	239	398	319
MA(1)	249	15	264	513	279
ARIMA	134	74	208	342	282
AR(1)	162	74	236	398	310
Naive	121	121	242	363	363
CMAGM(1,1)	126	95	221	347	316
CMAGM(2,1)	123	113	236	359	349
CMAGM(1,2)	178	61	239	417	300

Figura 64 – Nasa (CMAGM) - Quarto Cenário

Tabela 64 – Nasa (CMAGM) - Quarto Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	158	91	249	407	340
ARMA(1,1)	203	44	247	450	291
MA(1)	266	3	269	535	272
ARIMA	182	48	230	412	278
AR(1)	207	41	248	455	289
Naive	121	121	242	363	363
CMAGM(1,1)	143	88	231	374	319
CMAGM(2,1)	132	111	243	375	354
CMAGM(1,2)	206	30	236	442	266

Custo(1, 1), a solução CMAGM(2,1) superou a solução CMAGM(1,1) como a segunda de menor custo, ficando atrás do modelo Naive.

No terceiro cenário, ilustrado pela Figura 67 e pela tabela 67, apresenta resultados praticamente iguais aos apresentados no cenário anterior, mudando apenas a proporção.

Por fim, no quarto cenário, Figura 67 e tabela 67, os resultados foram similares aos do primeiro cenário.

5.3.4.4 Série Google

As Figuras 69, 70, 71 e 72 ilustram os resultados para a série Google *job*. No primeiro cenário, Figura 69, notamos que todas as soluções e modelos estão no conjunto ótimo de Pareto. Na Tabela 69 podemos observar que o modelo MA foi a melhor no cenário Custo(1, 1), seguido pelo modelo ARMA e pela solução CMAGM(1,2). No cenário Custo(2, 1) o a solução CMAGM(2,1) foi a que apresentou o menor custo. No cenário Custo(1, 2) o modelo MA apresentou melhor desempenho, seguido pela solução CMAGM(1,2).

A Figura 70 mostra os resultados para o segundo cenário. Podemos observar que apenas o modelo AR não está no conjunto ótimo de Pareto. De acordo com a Tabela 70, o modelo ARMA apresentou os melhores resultados para o cenário Custo(1, 1), seguido pela solução CMAGM(1,1). No cenário Custo(2, 1) a solução CMAGM(1,1) apresentou o menor custo, e no cenário Custo(1, 2) a solução CMAGM(1,2).

No terceiro cenário, Figura 71, apenas a solução CMAGM(1,1) está no conjunto ótimo de Pareto. Já as soluções CMAGM(2,1) e CMAGM(1,2) foram dominadas. De acordo com a Tabela 71, a solução CMAGM(1,1) foi a melhor, ao lado dos modelos AR e MA, para o cenário Custo(1,1). No cenário Custo(2,1), a solução CMAGM(1,1), novamente, apresentou o menor custo seguida pela solução CMAGM(2,1). Já no cenário Custo(1, 2) os modelos MA e AR apresentaram os menores custos, seguidos pela solução CMAGM(1,1).

O quarto cenário, ilustrado na Figura 72, é muito similar ao cenário anterior. De acordo com a Tabela 72, a solução CMAGM(1,1) apresentou o menor custo para os cenários Custo(1, 1) e Custo(2, 1). Por fim, no cenário Custo(1, 2), os modelos MA e AR apresentaram os menores custos, seguidos pela solução CMAGM(1,1).

Observando os resultados apresentados pelo algoritmo CMAGM, podemos perceber que a combinação de modelos de previsão se mostrou eficiente no sentido de atender a diferentes objetivos dos usuários. Assim como nos algoritmos anteriores, nem sempre a preferência do usuário em relação a recursos sub e superprovisionados foi satisfeita. Ou seja, nem sempre a solução CMAGM(2,1) foi melhor que a CMAGM(1,1) no cenário Custo(2,1), como era esperado. Para tentar melhorar essa questão, assim como nos casos anteriores, sugere-se ajustar esses parâmetros de peso entre recursos sub e superprovisionados de maneira dinâmica.

Figura 65 – Copa do Mundo (CMAGM) - Primeiro Cenário

Tabela 65 – Copa do Mundo (CMAGM) - Primeiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	32403	27511	59914	92317	87425
ARMA(1,1)	39412	21283	60695	100107	81978
MA(1)	112525	5475	118000	230525	123475
ARIMA	35674	24295	59969	95643	84264
AR(1)	39446	18066	57512	96958	75578
Naive	26756	26754	53510	80266	80264
CMAGM(1,1)	29643	25424	55067	84710	80491
CMAGM(2,1)	29105	26264	55369	84474	81633
CMAGM(1,2)	39115	18888	58003	97118	76891

Figura 66 – Copa do Mundo (CMAGM) - Segundo Cenário

Tabela 66 – Copa do Mundo (CMAGM) - Segundo Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	28161	23910	52071	80232	75981
ARMA(1,1)	34254	18500	52754	87008	71254
MA(1)	97827	4765	102592	200419	107357
ARIMA	30994	21138	52132	83126	73270
AR(1)	34299	15698	49997	84296	65695
Naive	23260	23258	46518	69778	69776
CMAGM(1,1)	27171	21704	48875	76046	70579
CMAGM(2,1)	25451	22518	47969	73420	70487
CMAGM(1,2)	34017	16407	50424	84441	66831

Figura 67 – Copa do Mundo (CMAGM) - Terceiro Cenário

Tabela 67 – Copa do Mundo (CMAGM) - Terceiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	14722	12488	27210	41932	39698
ARMA(1,1)	17894	9669	27563	45457	37232
MA(1)	51165	2504	53669	104834	56173
ARIMA	16194	11042	27236	43430	38278
AR(1)	17910	8211	26121	44031	34332
Naive	12151	12150	24301	36452	36451
CMAGM(1,1)	13536	11472	25008	38544	36480
CMAGM(2,1)	13282	11571	24853	38135	36424
CMAGM(1,2)	17790	8663	26453	44243	35116

Figura 68 – Copa do Mundo (CMAGM) - Quarto Cenário

Tabela 68 – Copa do Mundo (CMAGM) - Quarto Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	13807	11723	25530	39337	37253
ARMA(1,1)	16767	9072	25839	42606	34911
MA(1)	47867	2329	50196	98063	52525
ARIMA	15176	10338	25514	40690	35852
AR(1)	16790	7696	24486	41276	32182
Naive	11387	11386	22773	34160	34159
CMAGM(1,1)	12967	10790	23757	36724	34547
CMAGM(2,1)	12769	11134	23903	36672	35037
CMAGM(1,2)	16708	8080	24788	41496	32868

Figura 69 – Google (CMAGM) - Primeiro Cenário

Tabela 69 – Google (CMAGM) - Primeiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	193	56	249	442	305
ARMA(1,1)	214	27	241	455	268
MA(1)	229	6	235	464	241
ARIMA	200	47	247	447	294
AR(1)	212	30	242	454	272
Naive	151	151	302	453	453
CMAGM(1,1)	208	40	248	456	288
CMAGM(2,1)	162	100	262	424	362
CMAGM(1,2)	219	23	242	461	265

Figura 70 – Google (CMAGM) - Segundo Cenário

Tabela 70 – Google (CMAGM) - Segundo Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	129	39	168	297	207
ARMA(1,1)	139	10	149	288	159
MA(1)	154	0	154	308	154
ARIMA	133	22	155	288	177
AR(1)	146	7	153	299	160
Naive	99	99	198	297	297
CMAGM(1,1)	136	15	151	287	166
CMAGM(2,1)	120	54	174	294	228
CMAGM(1,2)	143	5	148	291	153

Figura 71 – Google (CMAGM) - Terceiro Cenário

Tabela 71 – Google (CMAGM) - Terceiro Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	16	5	21	37	26
ARMA(1,1)	17	3	20	37	23
MA(1)	18	0	18	36	18
ARIMA	17	3	20	37	23
AR(1)	18	0	18	36	18
Naive	13	13	26	39	39
CMAGM(1,1)	15	3	18	33	21
CMAGM(2,1)	15	4	19	34	23
CMAGM(1,2)	16	3	19	35	22

Figura 72 – Google (CMAGM) - Quarto Cenário

Tabela 72 – Google (CMAGM) - Quarto Cenário

	Recursos sub-provisionados	Recursos super-provisionados	Custo (1, 1)	Custo (2, 1)	Custo (1, 2)
ETS	14	4	18	32	22
ARMA(1,1)	15	1	16	31	17
MA(1)	15	0	15	30	15
ARIMA	15	1	16	31	17
AR(1)	15	0	15	30	15
Naive	10	10	20	30	30
CMAGM(1,1)	12	2	14	26	16
CMAGM(2,1)	12	4	16	28	20
CMAGM(1,2)	13	2	15	28	17

5.3.5 Análise dos Resultados

Para demonstrar que as soluções baseadas em otimização multiobjetivo atendem a hipótese desta Tese, que é criar uma solução genérica, capaz de se adaptar a vários tipos de demandas e cenários, criamos uma metodologia para avaliar a adequabilidade genérica de cada modelo e solução. Nessa metodologia estabelecemos uma pontuação, de acordo com o desempenho de cada modelo e solução, dentro de cada cenário. A pontuação é a seguinte: a solução ou modelo melhor colocado em cada cenário recebe 9 pontos; a segunda melhor solução ou modelo recebe 8; a terceira 7; e assim sucessivamente, com a pior solução não recebendo nenhum ponto. Lembrando que, consideramos a melhor solução aquela que apresenta o menor custo para o usuário. No cenário Custo(1,1), apenas as soluções CMPL(1,1), CMPLR(2), CMAHP(1,1) e CMAGM(1,1) foram consideradas. No cenário Custo(2,1), apenas as soluções CMPL(2,1), CMPLR(3), CMAHP(5,1) e CMAGM(2,1). Já no cenário Custo(1,2), somente consideramos as soluções CMPL(1,2), CMPLR(1), CMAHP(1,5) e CMAGM(1,2). A Tabela 73 ilustra os resultados, contendo a soma da pontuação de cada solução e modelo, em cada tipo de carga de trabalho e no total.

Tabela 73 – Adequabilidade Genérica

Pontuação					
	Série				
Modelo	ClarkNet	Nasa	World Cup	Google	Total
ETS	64	30	39	27	160
ARMA(1,1)	77	35	32	64	208
MA(1)	27	29	0	76	132
ARIMA	60	92	37	52	241
AR(1)	39	35	68	73	215
Naive	9	27	96	13	145
CMPL	62	88	96	48	294
CMPLR	71	69	52	72	264
CMAHP	73	62	36	72	243
CMAGM	79	85	84	82	330

Podemos observar na Tabela 73 que a solução CMAGM obteve a maior pontuação na série ClarkNet, seguida pelo modelo ARMA. Na série Nasa, o modelo ARIMA obteve o melhor desempenho, seguido pela solução CMPL. Já na série FIFA World Cup, houve um empate entre o modelo Naive e a solução CMPL. Por fim, na série Google, a solução CMAGM, assim como na série ClarkNet, foi novamente a melhor, seguida pelo modelo MA. Observando a pontuação total, notamos que as quatro soluções propostas neste trabalho, baseadas em otimização multiobjetivo, apresentaram melhores resultados que qualquer outro modelo individual. Isso mostra que elas atendem o propósito deste trabalho, que é criar uma solução genérica e adaptativa para o problema de alocação de recursos na Nuvem. Podemos observar que, entre as quatro soluções propostas, a que se mostrou mais apta ao propósito desta Tese foi a CMAGM. Em seguida, aparece a solução CMPL. A terceira melhor foi a CMPLR. A solução CMAHP foi a quarta melhor.

A definição de prioridades entre os critérios é um fator importante em todas as soluções multiobjetivo, e é determinante para o sucesso da mesma. Por exemplo, nas soluções CMAGM e CMPL essa prioridade é definida de maneira mais objetiva, ou seja, a prioridade é diretamente proporcional ao custo que o usuário tem com cada um dos critérios. Por exemplo, se o custo de um recurso subprovisionado for o dobro do custo de um recurso superprovisionado basta definir a prioridade do primeiro como sendo 2 e a do segundo como sendo 1. Isso facilita a convergência para um resultado adequado para o usuário. As soluções CMAHP e CMPLR, por outro lado, apresentam uma maior subjetividade na definição das prioridades entre os objetivos. As relações de prioridades propostas pelo AHP, por exemplo, não representam a diferença literal entre os objetivos. Isto é, se o usuário definir um dos objetivos como prioridade 5 e o outro como prioridade 1, isso não significa, de acordo com o AHP, que o primeiro tem importância cinco vezes maior que o segundo. Significa, apenas, que o primeiro tem forte importância em relação ao segundo. Já na CMPLR, o usuário precisa definir o limite máximo do custo de determinado objetivo. Essa definição não trivial, já que os resultados mostraram que, nem sempre, aumentar esse valor limite levará a resultados melhores para o usuário.

5.3.6 Sobrecarga dos modelos

A sobrecarga apresentada pelos algoritmos Combinação de Modelos por meio de Programação Linear (CMPL), Combinação de Modelos por meio de Programação Linear com Restrição (CMPLR), Combinação de Modelos por meio do Método de Análise Hierárquica (CMAHP) e Combinação de Modelos por meio de Algoritmo Genético Multiobjetivo (CMAGM) para fazer uma previsão, considerando o ambiente descrito na seção sobre a metodologia, foi de, aproximadamente, 400 milissegundos para uma série com 300 pontos de dados. Esse *overhead* é desprezível, considerando que será feita apenas uma previsão a cada hora. Comparando com o *overhead* gerado por cada modelo individualmente, o modelo ETS demora pouco mais de 250 milissegundos para fazer uma previsão, considerando uma série com 300 pontos de dados. O modelo ARIMA leva em torno de 100 milissegundos. O modelo ARMA(1, 1) gasta em torno de 8 milissegundos e os modelos AR(1) e MA(1) em torno de 5 milissegundos. No caso do modelo NAIVE, praticamente não existe *overhead*.

5.4 Considerações Finais

Este capítulo apresentou os resultados obtidos com nossas propostas, assim como uma comparação entre as mesmas. Os experimentos foram realizados por meio de simulação, e carga de trabalho foi baseada em quatro *logs* extraídos de servidores web reais. De acordo com os resultados, as soluções baseadas em combinação de modelos de previsão por meio de otimização multiobjetivo, são mais eficazes no propósito de se adaptar a diferentes séries e cenários, do que um único modelo de previsão. Dentre as soluções estudadas, a que obteve o melhor desempenho

foi a CMAGM. No próximo capítulo apresentaremos as conclusões a respeito deste Trabalho, assim como sugestões para trabalhos futuros.

CONCLUSÃO

O foco deste trabalho foi realizar um estudo sobre o uso de técnicas de otimização multiobjetivo para combinar modelos de previsão de séries temporais, a fim de otimizar a previsão de demanda e, conseqüentemente, a alocação dos recursos para aplicações hospedadas na infraestrutura da nuvem. A previsão de demanda é importante porque as técnicas reativas não são capazes de lidar com variações de demanda sem causar instabilidade no sistema. Isso se deve ao fato da alocação de recursos na nuvem não ser instantânea, ou seja, existe um atraso entre a solicitação de um recurso e o mesmo estar pronto para uso. Dessa maneira, é necessário prever a demanda futura, para solicitar os recursos necessários com antecedência.

No entanto, a escolha do melhor modelo de previsão é uma tarefa difícil. Como pode ser visto a partir dos resultados alcançados com o desenvolvimento deste trabalho, não há um modelo de previsão melhor para todos os casos. Um modelo que tem um bom desempenho para uma determinada série temporal, pode apresentar um mau desempenho para outra. A tarefa torna-se ainda mais complicada quando o usuário não possui muitos dados históricos para analisar. Por exemplo, uma empresa que está iniciando um negócio e quer hospedar sua aplicação na nuvem. A presente proposta considera precisamente esse problema, sendo capaz de se adaptar a vários tipos de séries temporais. Além disso, a proposta apresentada não requer fase prévia de treinamento, não precisando de uma grande quantidade de dados históricos para funcionar. Ela é capaz de se adaptar a medida em que os dados vão chegando para serem processados. Assim, o método proposto neste trabalho é genérico o suficiente para ser usado por qualquer cliente que deseja hospedar sua aplicação na nuvem e ter a garantia de obter um bom resultado.

Quatro algoritmos foram propostos para resolver o problema: CMPL, baseado no método da soma ponderada; CMPLR, baseado no método ϵ -restrito; CMAHP, baseado no método de análise hierárquica; e CMAGM, baseado em algoritmo genético multiobjetivo. Todos os algoritmos foram avaliados por meio de quatro diferentes tipos de cargas de trabalho, extraídas de *logs* de servidores reais. De acordo com os resultados, o que mostrou melhor desempenho,

ou seja, maior capacidade de se adaptar a diferentes tipos de carga de trabalho e diferentes cenários, foi o CMAGM. O CMPL obteve o segundo melhor desempenho, seguido pelo CMPLR. O AHP teve o pior desempenho entre os quatro algoritmos. De qualquer modo, todos os quatro algoritmos propostos apresentaram melhor desempenho genérico do que qualquer modelo de previsão usado isoladamente.

6.1 Contribuições

A realização deste projeto propiciou algumas contribuições diretamente relacionadas com a hipótese desta Tese:

1. Desenvolvimento de um algoritmo para combinação de modelos de previsão de séries temporais por meio de programação linear, usando o método da soma ponderada.
2. Desenvolvimento de um algoritmo para combinação de modelos de previsão de séries temporais por meio de programação linear, usando o método ε -restrito.
3. Desenvolvimento de um algoritmo para combinação de modelos de previsão de séries temporais por meio do método de análise hierárquica (AHP).
4. Desenvolvimento de um algoritmo para combinação de modelos de previsão de séries temporais por meio de algoritmo genético, mono e multiobjetivo.

As publicações geradas, oriundas do desenvolvimento deste trabalho são descritas a seguir:

1. Combining time series prediction models using genetic algorithm to autoscaling web applications hosted in the cloud infrastructure. *Neural Computing and Applications*, Springer London, p. 1–24, 2015. ISSN 0941-0643. (MESSIAS *et al.*, 2015).
2. Efficient resource allocation for web applications hosted in the cloud by means of weighted multi-objective linear programming. In: ACM. Proceedings of the 21st Brazilian Symposium on Multimedia and the Web. [S.l.], 2015. p. 57–64. (MESSIAS; ESTRELLA; EHLERS, 2015b).
3. Optimal cloud resource allocation by means of the analytic hierarchy process. In: Computing Conference (CLEI), 2015 Latin American . [S.l.:s.n.], 2015. p. 1–12. (MESSIAS; ESTRELLA; EHLERS, 2015a).

Artigos aceitos e não publicados:

1. A proactive fuzzy-based auto-scaling to optimize resource allocation in the cloud. *International Conference on Economics of Grids, Clouds, Systems and Services*. 2015.

6.2 Trabalhos Futuros

O presente trabalho de doutorado não finaliza as possibilidades de estudo relacionadas à busca de um *autoscaling* genérico, capaz de prover elasticidade às aplicações hospedadas na nuvem. Outros estudos devem ser conduzidos a partir dos resultados e constatações encontradas durante o desenvolvimento desta tese. Dentre os possíveis trabalhos futuros, destacam-se os seguintes estudos:

1. **Criação de modelos de previsão para tipos específicos de aplicações, como aplicações Web, por exemplo:** os modelos de previsão de séries temporais utilizados neste trabalho, apesar de serem utilizados para análise e previsão de séries temporais em geral, são originalmente projetados para a área econômica. Desse modo, tais modelos tendem a não funcionar bem para séries com características específicas, como as séries que representam a taxa de chegada de requisições em rajadas, típicas de algumas aplicações Web. Neste trabalho isso ficou claro, pois na série Copa de Mundo FIFA, o modelo que apresentou o melhor desempenho foi o Naive, apesar da sua extrema simplicidade. Isso mostra que, na realidade, os outros modelos não foram capazes de lidar com altas variações na chegada de requisições. Desse modo, seria interessante criar modelos de previsão específicos para tais tipos de séries. Quanto mais modelos de previsão especializados estiverem disponíveis, melhor será o resultado da combinação dos mesmos para criar uma solução genérica;
2. **Ajustar os parâmetros dos métodos de otimização multiobjetivo dinamicamente, afim de obter melhores resultados:** como mostrado nos resultados, nem sempre os parâmetros que definem a prioridade dos objetivos geram os resultados desejados. Por exemplo, nem sempre que o usuário definir uma prioridade maior para determinado objetivo, significa que esse objetivo terá melhores resultados do que se o usuário tivesse definido a mesma prioridade entre os objetivos. Isso ocorre porque estamos trabalhando com previsões, ou seja, estamos otimizando uma solução baseando-se nos resultados passados, sem, no entanto, ter a garantia que o mesmo resultado será repetido no futuro. Desse modo, seria interessante monitorar e ajustar os parâmetros de prioridades dos objetivos de maneira dinâmica, de modo a otimizar os resultados para o usuário;
3. **Estudar técnicas de aprendizagem em fluxo contínuo de dados:** à medida que uma série temporal evolui ao longo do tempo, muitas vezes, dados passados não representam mais as suas características. Assim sendo, é necessário descartar os dados antigos que não representam mais o estado atual da série. Existe um ramo da área de aprendizado de máquina que estuda técnicas para identificar qual é o momento de esquecer os dados passados para melhorar o desempenho do algoritmo. Esse ramo é o aprendizado em fluxo contínuo de dados ou detecção de novidades em fluxo contínuo de dados. A utilização dessas técnicas pode melhorar o desempenho dos algoritmos propostos neste trabalho,

especialmente em séries com características similares à série Copa do Mundo FIFA, em que há evidentes mudanças na estrutura da série ao longo do tempo;

4. **Combinar mais de uma técnica de *autoscaling*:** Neste trabalho, foram utilizados apenas a combinação de modelos de previsão de séries temporais. No entanto, nada impede de se utilizar os algoritmos propostos para combinar outras técnicas de *autoscaling*, como, por exemplo, modelos de previsão de séries temporais com uma rede neural e uma solução baseada em teoria de controle.

REFERÊNCIAS

ABOULNAGA, A.; SALEM, K.; SOROR, A.; MINHAS, U.; KOKOSIELIS, P.; KAMATH, S. Deploying database appliances in the cloud. **IEEE Data Eng. Bull.**, v. 32, n. 1, p. 13–20, 2009. Citado na página 7.

ALI-ELDIN, A.; KIHIL, M.; TORDSSON, J.; ELMROTH, E. Efficient provisioning of bursty scientific workloads on the cloud using adaptive elasticity control. In: ACM. **Proceedings of the 3rd workshop on Scientific Cloud Computing Date**. [S.l.], 2012. p. 31–40. Citado na página 27.

ALI-ELDIN, A.; TORDSSON, J.; ELMROTH, E. An adaptive hybrid elasticity controller for cloud infrastructures. In: IEEE. **Network Operations and Management Symposium (NOMS), 2012 IEEE**. [S.l.], 2012. p. 204–212. Citado na página 27.

AMAZON, E. Amazon elastic compute cloud (amazon ec2). **Amazon Elastic Compute Cloud (Amazon EC2)**, 2010. Citado 2 vezes nas páginas 1 e 27.

ARROYO, J. E. C. Heurísticas e metaheurísticas para otimização combinatória multiobjetivo. Biblioteca Digital da Unicamp, 2002. Citado na página 14.

BALAJI, M.; RAO, G.; KUMAR, C. *et al.* A comparative study of predictive models for cloud infrastructure management. In: IEEE. **Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on**. [S.l.], 2014. p. 923–926. Citado na página 31.

BARBARA, B. Salesforce. com: Raising the level of networking. **Inf. Today**, v. 27, p. 45–45, 2010. Citado na página 1.

BODIK, P.; GRIFFITH, R.; SUTTON, C.; FOX, A.; JORDAN, M.; PATTERSON, D. Statistical machine learning makes automatic control practical for internet datacenters. In: **Proceedings of the 2009 conference on Hot topics in cloud computing**. [S.l.: s.n.], 2009. p. 12–12. Citado na página 27.

BOX, G. E.; JENKINS, G. M.; REINSEL, G. C. **Time series analysis: forecasting and control**. [S.l.]: John Wiley & Sons, 2013. Citado na página 13.

BRAUN, J.; MURDOCH, D. J. **A first course in statistical programming with R**. [S.l.]: Cambridge University Press Cambridge, 2007. v. 25. Citado na página 48.

BREITMAN, K.; VITERBO, J. Computação na nuvem: uma visão geral. In: **Amãpytuna: Computação em nuvem: serviços livres para a sociedade do conhecimento**. [S.l.]: Funag, 2010. v. 3, p. 17–45. Citado 2 vezes nas páginas 5 e 7.

CARON, E.; DESPREZ, F.; MURESAN, A. Forecasting for cloud computing on-demand resources based on pattern matching. 2010. Citado na página 30.

_____. Pattern matching based forecast of non-periodic repetitive behavior for cloud clients. **Journal of Grid Computing**, Springer, v. 9, n. 1, p. 49–64, 2011. Citado na página 30.

- CARR, N. **The big switch: Rewiring the world, from Edison to Google**. [S.l.]: WW Norton & Company, 2008. Citado na página 5.
- CASALICCHIO, E.; SILVESTRI, L. Autonomic management of cloud-based systems: The service provider perspective. In: **Computer and Information Sciences III**. [S.l.]: Springer, 2013. p. 39–47. Citado na página 25.
- CHANDRA, A.; GONG, W.; SHENOY, P. Dynamic resource allocation for shared data centers using online measurements. In: **Quality of Service—IWQoS 2003**. [S.l.]: Springer, 2003. p. 381–398. Citado na página 31.
- CHIEU, T. C.; MOHINDRA, A.; KARVE, A.; SEGAL, A. *et al.* Dynamic scaling of web applications in a virtualized cloud computing environment. In: IEEE. **e-Business Engineering, 2009. ICEBE'09. IEEE International Conference on**. [S.l.], 2009. p. 281–286. Citado na página 25.
- CHIEU, T. C.; MOHINDRA, A.; KARVE, A. *et al.* Scalability and performance of web applications in a compute cloud. In: IEEE. **e-Business Engineering (ICEBE), 2011 IEEE 8th International Conference on**. [S.l.], 2011. p. 317–323. Citado na página 25.
- CIURANA, E. **Developing with Google App Engine**. [S.l.]: Springer, 2009. Citado 2 vezes nas páginas 8 e 9.
- CLARK, T. Quantifying the benefits of the rightscale cloud management platform. **Fact Point Group Whitepaper, funded by Rightscale**, 2010. Citado 2 vezes nas páginas 11 e 25.
- CLARKNET, L. **ClarkNet-HTTP - Two Weeks of HTTP Logs from the ClarkNet WWW Server**. 1995. <http://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html>. Accessed: 2014-10-15. Citado 3 vezes nas páginas 26, 29 e 48.
- CLOUDWATCH, A. **monitoring for AWS cloud resources**. 2013. <https://aws.amazon.com/pt/cloudwatch/>. Accessed: 2015-10-15. Citado na página 26.
- COELLO, C. C.; LAMONT, G. B.; VELDHUIZEN, D. A. V. **Evolutionary algorithms for solving multi-objective problems**. [S.l.]: Springer Science & Business Media, 2007. Citado na página 14.
- CONNER, N. **Google Apps: The Missing Manual: The Missing Manual**. [S.l.]: "O'Reilly Media, Inc.", 2008. Citado na página 1.
- CUP, L. F. W. **1998 World Cup Web Site Access Logs**. 1998. <http://ita.ee.lbl.gov/html/contrib/WorldCup.html>. Accessed: 2014-10-15. Citado 3 vezes nas páginas 25, 26 e 48.
- DANTZIG, G. B. **Linear programming and extensions**. [S.l.]: Princeton university press, 1998. Citado na página 48.
- DEB, K.; AGRAWAL, S.; PRATAP, A.; MEYARIVAN, T. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. **Lecture notes in computer science**, Springer, v. 1917, p. 849–858, 2000. Citado 2 vezes nas páginas 21 e 42.
- DUTREILH, X.; KIRGIZOV, S.; MELEKHOVA, O.; MALENFANT, J.; RIVIERRE, N.; TRUCK, I. Using reinforcement learning for autonomic resource allocation in clouds: Towards a fully automated workflow. In: **ICAS 2011, The Seventh International Conference on Autonomic and Autonomous Systems**. [S.l.: s.n.], 2011. p. 67–74. Citado na página 29.

- DUTREILH, X.; RIVIERRE, N.; MOREAU, A.; MALENFANT, J.; TRUCK, I. From data center resource allocation to control theory and back. In: IEEE. **Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on**. [S.l.], 2010. p. 410–417. Citado na página 24.
- DUTTA, S.; GERA, S.; VERMA, A.; VISWANATHAN, B. Smartscale: Automatic application scaling in enterprise clouds. In: IEEE. **Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on**. [S.l.], 2012. p. 221–228. Citado na página 29.
- FANG, W.; LU, Z.; WU, J.; CAO, Z. Rpps: a novel resource prediction and provisioning scheme in cloud data center. In: IEEE. **Services Computing (SCC), 2012 IEEE Ninth International Conference on**. [S.l.], 2012. p. 609–616. Citado na página 30.
- FERNANDEZ, H.; PIERRE, G.; KIELMANN, T. *et al.* Autoscaling web applications in heterogeneous cloud infrastructures. In: **IEEE International Conference on Cloud Engineering**. [S.l.: s.n.], 2014. Citado na página 31.
- FONSECA, C. M.; FLEMING, P. J. An overview of evolutionary algorithms in multiobjective optimization. **Evolutionary computation**, MIT Press, v. 3, n. 1, p. 1–16, 1995. Citado na página 14.
- FONSECA, C. M.; FLEMING, P. J. *et al.* Genetic algorithms for multiobjective optimization: Formulation discussion and generalization. In: CITESEER. **ICGA**. [S.l.], 1993. v. 93, p. 416–423. Citado na página 21.
- GAMBI, A.; TOFFETTI, G. Modeling cloud performance with kriging. In: IEEE PRESS. **Proceedings of the 34th International Conference on Software Engineering**. [S.l.], 2012. p. 1439–1440. Citado na página 27.
- GHANBARI, H.; SIMMONS, B.; LITOIU, M.; ISZLAI, G. Exploring alternative approaches to implement an elasticity policy. In: IEEE. **Cloud Computing (CLOUD), 2011 IEEE International Conference on**. [S.l.], 2011. p. 716–723. Citado na página 26.
- GILDER, G. The information factories. **Wired magazine**, v. 14, n. 10, 2006. Citado na página 5.
- GONG, Z.; GU, X.; WILKES, J. Press: Predictive elastic resource scaling for cloud systems. In: IEEE. **Network and Service Management (CNSM), 2010 International Conference on**. [S.l.], 2010. p. 9–16. Citado 2 vezes nas páginas 30 e 31.
- GROSS, D.; SHORTLE, J. F.; THOMPSON, J. M.; HARRIS, C. M. **Fundamentals of queueing theory**. [S.l.]: John Wiley & Sons, 2013. Citado na página 45.
- GROUP, M. M. **Internet Users in the World by Regions**. 2015. <http://www.internetworldstats.com/stats.htm>. Accessed: 2016-01-15. Citado na página 48.
- HAN, R.; GUO, L.; GHANEM, M. M.; GUO, Y. Lightweight resource scaling for cloud applications. In: IEEE. **Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on**. [S.l.], 2012. p. 644–651. Citado na página 24.
- HASAN, M. Z.; MAGANA, E.; CLEMM, A.; TUCKER, L.; GUDREDDI, S. L. D. Integrated and autonomic cloud resource scaling. In: IEEE. **Network Operations and Management Symposium (NOMS), 2012 IEEE**. [S.l.], 2012. p. 1327–1334. Citado na página 24.

HASSAN, M.; SONG, B.; HUH, E. A framework of sensor-cloud integration opportunities and challenges. In: ACM. **Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication**. [S.l.], 2009. p. 618–626. Citado na página 6.

HERBST, N. R.; HUBER, N.; KOUNEV, S.; AMREHN, E. Self-adaptive workload classification and forecasting for proactive resource provisioning. **Concurrency and Computation: Practice and Experience**, Wiley Online Library, 2014. Citado 2 vezes nas páginas 13 e 31.

HORN, J.; NAFPLIOTIS, N.; GOLDBERG, D. E. A niched pareto genetic algorithm for multi-objective optimization. In: IEEE. **Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on**. [S.l.], 1994. p. 82–87. Citado na página 21.

HUANG, J.; LI, C.; YU, J. Resource prediction based on double exponential smoothing in cloud computing. In: IEEE. **Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on**. [S.l.], 2012. p. 2056–2060. Citado na página 30.

HYNDMAN, R. J.; ATHANASOPOULOS, G. **Forecasting: principles and practice**. [S.l.]: OTexts, 2014. Citado 2 vezes nas páginas 13 e 14.

HYNDMAN, R. J.; KHANDAKAR, Y. **Automatic time series for forecasting: the forecast package for R**. [S.l.], 2007. Citado 3 vezes nas páginas 13, 14 e 35.

ISLAM, S.; KEUNG, J.; LEE, K.; LIU, A. Empirical prediction models for adaptive resource provisioning in the cloud. **Future Generation Computer Systems**, Elsevier, v. 28, n. 1, p. 155–162, 2012. Citado na página 30.

KALYVIANAKI, E.; CHARALAMBOUS, T.; HAND, S. Self-adaptive and self-configured cpu resource provisioning for virtualized servers using kalman filters. In: ACM. **Proceedings of the 6th international conference on Autonomic computing**. [S.l.], 2009. p. 117–126. Citado na página 27.

KHATUA, S.; GHOSH, A.; MUKHERJEE, N. Optimizing the utilization of virtual resources in cloud environment. In: IEEE. **Virtual Environments Human-Computer Interfaces and Measurement Systems (VECIMS), 2010 IEEE International Conference on**. [S.l.], 2010. p. 82–87. Citado na página 25.

KOPEREK, P.; FUNIKA, W. Dynamic business metrics-driven resource provisioning in cloud environments. In: **Parallel Processing and Applied Mathematics**. [S.l.]: Springer, 2012. p. 171–180. Citado na página 24.

KRISHNAN, S.; GONZALEZ, J. L. U. Google compute engine. In: **Building Your Next Big Thing with Google Cloud Platform**. [S.l.]: Springer, 2015. p. 53–81. Citado na página 1.

KUPFERMAN, J.; SILVERMAN, J.; JARA, P.; BROWNE, J. Scaling into the cloud. **CS270-advanced operating systems**, 2009. Citado 2 vezes nas páginas 25 e 30.

LAMA, P.; ZHOU, X. Autonomic provisioning with self-adaptive neural fuzzy control for end-to-end delay guarantee. In: IEEE. **Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on**. [S.l.], 2010. p. 151–160. Citado na página 28.

LIM, H. C.; BABU, S.; CHASE, J. S. Automated control for elastic storage. In: ACM. **Proceedings of the 7th international conference on Autonomic computing**. [S.l.], 2010. p. 1–10. Citado na página 26.

LIM, H. C.; BABU, S.; CHASE, J. S.; PAREKH, S. S. Automated control in cloud computing: challenges and opportunities. In: ACM. **Proceedings of the 1st workshop on Automated control for datacenters and clouds**. [S.l.], 2009. p. 13–18. Citado na página 26.

LORIDO-BOTRÁN, T.; MIGUEL-ALONSO, J.; LOZANO, J. A. Auto-scaling techniques for elastic applications in cloud environments. **Department of Computer Architecture and Technology, University of Basque Country, Tech. Rep. EHU-KAT-IK-09**, v. 12, 2012. Citado na página 2.

_____. Comparison of auto-scaling techniques for cloud environments. 2013. Citado na página 26.

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**, Springer, v. 12, n. 4, p. 559–592, 2014. Citado 7 vezes nas páginas 1, 2, 11, 23, 25, 28 e 30.

MATH, C. **The Apache Commons Mathematics Library**. 2014. <http://commons.apache.org/proper/commons-math/>. Accessed: 2015-08-04. Citado na página 48.

MAURER, M.; BRANDIC, I.; SAKELLARIOU, R. Enacting slas in clouds using rules. In: **Euro-Par 2011 Parallel Processing**. [S.l.]: Springer, 2011. p. 455–466. Citado na página 26.

MELL, P.; GRANCE, T. Draft nist working definition of cloud computing. **Referenced on June. 3rd**, 2009. Citado na página 6.

MENASCÉ, D. A.; ALMEIDA, V. A.; DOWDY, L. W. **Capacity Planning for Web Services: metrics, models, and methods**. [S.l.]: Prentice Hall PTR Upper Saddle River, 2002. Citado na página 45.

MENASCE, D. A.; ALMEIDA, V. A.; DOWDY, L. W.; DOWDY, L. **Performance by design: computer capacity planning by example**. [S.l.]: Prentice Hall Professional, 2004. Citado na página 25.

MESSIAS, V.; ESTRELLA, J.; EHLERS, R. Optimal cloud resource allocation by means of the analytic hierarchy process. In: **Computing Conference (CLEI), 2015 Latin American**. [S.l.: s.n.], 2015. p. 1–12. Citado na página 132.

MESSIAS, V. R.; ESTRELLA, J. C.; EHLERS, R. Efficient resource allocation for web applications hosted in the cloud by means of weighted multi-objective linear programming. In: ACM. **Proceedings of the 21st Brazilian Symposium on Multimedia and the Web**. [S.l.], 2015. p. 57–64. Citado na página 132.

MESSIAS, V. R.; ESTRELLA, J. C.; EHLERS, R.; SANTANA, M. J.; SANTANA, R. C.; REIFF-MARGANIEC, S. Combining time series prediction models using genetic algorithm to autoscaling web applications hosted in the cloud infrastructure. **Neural Computing and Applications**, Springer London, p. 1–24, 2015. ISSN 0941-0643. Disponível em: <<http://dx.doi.org/10.1007/s00521-015-2133-3>>. Citado na página 132.

MI, H.; WANG, H.; YIN, G.; ZHOU, Y.; SHI, D.; YUAN, L. Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers. In: IEEE. **Services Computing (SCC), 2010 IEEE International Conference on**. [S.l.], 2010. p. 514–521. Citado na página 30.

NASA, L. **NASA-HTTP - Two Months of HTTP Logs from the KSC-NASA WWW Server**. 1996. <http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html>. Accessed: 2014-10-15. Citado 2 vezes nas páginas 31 e 48.

OSIADACZ, A. J. Multiple criteria optimization; theory, computation, and application, ralph e. steuer, wiley series in probability and mathematical statistics-applied, wiley, 1986, no. of pages 546, price f5 1.40, \$77.10. **Optimal Control Applications and Methods**, Wiley Online Library, v. 10, n. 1, p. 89–90, 1989. Citado na página 15.

PADALA, P.; HOU, K.-Y.; SHIN, K. G.; ZHU, X.; UYSAL, M.; WANG, Z.; SINGHAL, S.; MERCHANT, A. Automated control of multiple virtualized resources. In: ACM. **Proceedings of the 4th ACM European conference on Computer systems**. [S.l.], 2009. p. 13–26. Citado na página 27.

PARETO, V. **Cours d'économie politique**. [S.l.]: Librairie Droz, 1964. Citado na página 15.

PARK, S.-M.; HUMPHREY, M. Self-tuning virtual machines for predictable escience. In: IEEE COMPUTER SOCIETY. **Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid**. [S.l.], 2009. p. 356–363. Citado na página 26.

PENTA, M. D.; CASAZZA, G.; ANTONIOL, G.; MERLO, E. Modeling web maintenance centers through queue models. In: IEEE. **Software Maintenance and Reengineering, 2001. Fifth European Conference on**. [S.l.], 2001. p. 131–138. Citado na página 33.

PRODAN, R.; NAE, V. Prediction-based real-time resource provisioning for massively multi-player online games. **Future Generation Computer Systems**, Elsevier, v. 25, n. 7, p. 785–793, 2009. Citado na página 30.

RACKSPACE, U. Inc., “**The Rackspace Cloud**,”. 2010. <http://www.rackspace.com/>. Accessed: 2015-10-15. Citado na página 1.

RAO, J.; BU, X.; XU, C.-Z.; WANG, L.; YIN, G. Vconf: a reinforcement learning approach to virtual machines auto-configuration. In: ACM. **Proceedings of the 6th international conference on Autonomic computing**. [S.l.], 2009. p. 137–146. Citado na página 29.

RAO, J.; BU, X.; XU, C.-Z.; WANG, K. A distributed self-learning approach for elastic provisioning of virtualized cloud resources. In: IEEE. **Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on**. [S.l.], 2011. p. 45–54. Citado na página 29.

REDKAR, T.; GUIDICI, T. **Windows Azure Platform**. [S.l.]: Springer, 2009. Citado na página 1.

ROBINSON, D. **Amazon Web Services Made Simple: Learn how Amazon EC2, S3, SimpleDB and SQS Web Services enables you to reach business goals faster**. [S.l.]: Emereo Pty Ltd, 2008. Citado na página 8.

ROY, N.; DUBEY, A.; GOKHALE, A. Efficient autoscaling in the cloud using predictive models for workload forecasting. In: IEEE. **Cloud Computing (CLOUD), 2011 IEEE International Conference on**. [S.l.], 2011. p. 500–507. Citado 2 vezes nas páginas 28 e 30.

SAATY, T. L. **What is the analytic hierarchy process?** [S.l.]: Springer, 1988. Citado na página 17.

SCHAFFER, J. D. Multiple objective optimization with vector evaluated genetic algorithms. In: **Proceedings of the 1st International Conference on Genetic Algorithms, Pittsburgh, PA, USA, July 1985**. [S.l.: s.n.], 1985. p. 93–100. Citado na página 21.

SHEN, Z.; SUBBIAH, S.; GU, X.; WILKES, J. Cloudscale: elastic resource scaling for multi-tenant cloud systems. In: ACM. **Proceedings of the 2nd ACM Symposium on Cloud Computing**. [S.l.], 2011. p. 5. Citado na página 30.

SIMMONS, B.; GHANBARI, H.; LITOIU, M.; ISZLAI, G. Managing a saas application in the cloud using paas policy sets and a strategy-tree. In: INTERNATIONAL FEDERATION FOR INFORMATION PROCESSING. **Proceedings of the 7th International Conference on Network and Services Management**. [S.l.], 2011. p. 343–347. Citado na página 25.

SRINIVAS, N.; DEB, K. Muultiobjective optimization using nondominated sorting in genetic algorithms. **Evolutionary computation**, MIT Press, v. 2, n. 3, p. 221–248, 1994. Citado na página 21.

SUTTON, R. S.; BARTO, A. G. **Introduction to reinforcement learning**. [S.l.]: MIT Press Cambridge, 1998. v. 135. Citado na página 12.

TESAURO, G.; JONG, N. K.; DAS, R.; BENNANI, M. N. A hybrid reinforcement learning approach to autonomic resource allocation. In: IEEE. **Autonomic Computing, 2006. ICAC'06. IEEE International Conference on**. [S.l.], 2006. p. 65–73. Citado na página 29.

URBANEK, S.; URBANEK, M. S. Package rjava. 2014. Citado na página 48.

VLIET, J.; PAGANELLI, F.; WEL, S. V.; DOWD, D. **Elastic Beanstalk**. [S.l.]: "O'Reilly Media, Inc.", 2011. Citado na página 1.

WANG, L.; XU, J.; ZHAO, M.; FORTES, J. Adaptive virtual resource management with fuzzy model predictive control. In: ACM. **Proceedings of the 8th ACM international conference on Autonomic computing**. [S.l.], 2011. p. 191–192. Citado na página 28.

WANG, L.; XU, J.; ZHAO, M.; TU, Y.; FORTES, J. A. Fuzzy modeling based resource management for virtualized database systems. In: IEEE. **Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on**. [S.l.], 2011. p. 32–42. Citado na página 28.

WHATELY, L. L. A.; MAZONI, M. V. F.; BREITMAN, K.; VITERBO, J.; MARTINS, A.; TUJAL, L. C. P.; SOARES, L. F. G.; FERNANDES, A. **AMÃPYTUNA COMPUTAÇÃO EM NUVEM: SERVIÇOS LIVRES PARA A SOCIEDADE DO CONHECIMENTO**. 2010. Citado 3 vezes nas páginas 15, 8 e 9.

WILKES, J. **More Google cluster data**. 2011. Google research blog. Posted at <<http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html>>. Citado 2 vezes nas páginas 27 e 48.

WILSON, K. Microsoft office 365. In: **Using Office 365**. [S.l.]: Springer, 2014. p. 1–14. Citado na página [1](#).

XU, C.-Z.; RAO, J.; BU, X. Url: A unified reinforcement learning approach for autonomic cloud management. **Journal of Parallel and Distributed Computing**, Elsevier, v. 72, n. 2, p. 95–105, 2012. Citado na página [29](#).

XU, J.; ZHAO, M.; FORTES, J.; CARPENTER, R.; YOUSIF, M. On the use of fuzzy modeling in virtualized data center management. In: IEEE. **Autonomic Computing, 2007. ICAC'07. Fourth International Conference on**. [S.l.], 2007. p. 25–25. Citado na página [28](#).

ZAHARIEV, A. Google app engine. **Helsinki University of Technology**, 2009. Citado na página [1](#).

ZHU, Q.; AGRAWAL, G. Resource provisioning with budget constraints for adaptive applications in cloud environments. In: ACM. **Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing**. [S.l.], 2010. p. 304–307. Citado na página [27](#).

ZITZLER, E. **Evolutionary algorithms for multiobjective optimization: Methods and applications**. [S.l.]: Citeseer, 1999. v. 63. Citado na página [21](#).

PESOS DOS MODELOS DE PREVISÃO EM CADA CENÁRIO

Figura 73 – Séries Utilizadas.

(a) CMPL(1,1)

(b) CMPL(2,1)

(c) CMPL(1,2)

Figura 74 – Série FIFA World Cup (CMPL) - Primeiro Cenário

(a) CMPL(1,1)

(b) CMPL(2,1)

(c) CMPL(1,2)

Figura 75 – Série FIFA World Cup (CMPL) - Segundo Cenário

(a) CMPL(1,1)

(b) CMPL(2,1)

(c) CMPL(1,2)

Figura 76 – Série FIFA World Cup (CMPL) - Terceiro Cenário

(a) CMPL(1,1)

(b) CMPL(2,1)

(c) CMPL(1,2)

Figura 77 – Série FIFA World Cup (CMPL) - Quarto Cenário

(a) CMPL(1,1)

(b) CMPL(2,1)

(c) CMPL(1,2)

Figura 78 – Série Nasa (CMPL) - Primeiro Cenário

(a) CMPL(1,1)

(b) CMPL(2,1)

(c) CMPL(1,2)

Figura 79 – Série Nasa (CMPL) - Segundo Cenário

(a) CMPL(1,1)

(b) CMPL(2,1)

(c) CMPL(1,2)

Figura 80 – Série Nasa (CMPL) - Terceiro Cenário

(a) CMPL(1,1)

(b) CMPL(2,1)

(c) CMPL(1,2)

Figura 81 – Série Nasa (CMPL) - Quarto Cenário

(a) CMPL(1,1)

(b) CMPL(2,1)

(c) CMPL(1,2)

Figura 82 – Série Clarknet (CMPL) - Primeiro Cenário

(a) CMPL(1,1)

(b) CMPL(2,1)

(c) CMPL(1,2)

Figura 83 – Série Clarknet (CMPL) - Segundo Cenário

(a) CMPL(1,1)

(b) CMPL(2,1)

(c) CMPL(1,2)

Figura 84 – Série Clarknet (CMPL) - Terceiro Cenário

(a) CMPL(1,1)

(b) CMPL(2,1)

(c) CMPL(1,2)

Figura 85 – Série Clarknet (CMPL) - Quarto Cenário

(a) CMPL(1,1)

(b) CMPL(2,1)

(c) CMPL(1,2)

Figura 86 – Série Google (CMPL) - Primeiro Cenário

(a) CMPL(1,1)

(b) CMPL(2,1)

(c) CMPL(1,2)

Figura 87 – Série Google (CMPL) - Segundo Cenário

(a) CMPL(1,1)

(b) CMPL(2,1)

(c) CMPL(1,2)

Figura 88 – Série Google (CMPL) - Terceiro Cenário

(a) CMPL(1,1)

(b) CMPL(2,1)

(c) CMPL(1,2)

Figura 89 – Série Google (CMPL) - Quarto Cenário

(a) CMPLR(2)

(b) CMPLR(3)

(c) CMPLR(1)

Figura 90 – Série FIFA World Cup (CMPLR) - Primeiro Cenário

(a) CMPLR(2)

(b) CMPLR(3)

(c) CMPLR(1)

Figura 91 – Série FIFA World Cup (CMPLR) - Segundo Cenário

(a) CMPLR(2)

(b) CMPLR(3)

(c) CMPLR(1)

Figura 92 – Série FIFA World Cup (CMPLR) - Terceiro Cenário

(a) CMPLR(2)

(b) CMPLR(3)

(c) CMPLR(1)

Figura 93 – Série FIFA World Cup (CMPLR) - Quarto Cenário

(a) CMPLR(2)

(b) CMPLR(3)

(c) CMPLR(1)

Figura 94 – Série Nasa (CMPLR) - Primeiro Cenário

(a) CMPLR(2)

(b) CMPLR(3)

(c) CMPLR(1)

Figura 95 – Série Nasa (CMPLR) - Segundo Cenário

(a) CMPLR(2)

(b) CMPLR(3)

(c) CMPLR(1)

Figura 96 – Série Nasa (CMPLR) - Terceiro Cenário

(a) CMPLR(2)

(b) CMPLR(3)

(c) CMPLR(1)

Figura 97 – Série Nasa (CMPLR) - Quarto Cenário

(a) CMPLR(2)

(b) CMPLR(3)

(c) CMPLR(1)

Figura 98 – Série Clarknet (CMPLR) - Primeiro Cenário

(a) CMPLR(2)

(b) CMPLR(3)

(c) CMPLR(1)

Figura 99 – Série Clarknet (CMPLR) - Segundo Cenário

(a) CMPLR(2)

(b) CMPLR(3)

(c) CMPLR(1)

Figura 100 – Série Clarknet (CMPLR) - Terceiro Cenário

(a) CMPLR(2)

(b) CMPLR(3)

(c) CMPLR(1)

Figura 101 – Série Clarknet (CMPLR) - Quarto Cenário

(a) CMPLR(2)

(b) CMPLR(3)

(c) CMPLR(1)

Figura 102 – Série Google (CMPLR) - Primeiro Cenário

(a) CMPLR(2)

(b) CMPLR(3)

(c) CMPLR(1)

Figura 103 – Série Google (CMPLR) - Segundo Cenário

(a) CMPLR(2)

(b) CMPLR(3)

(c) CMPLR(1)

Figura 104 – Série Google (CMPLR) - Terceiro Cenário

(a) CMPLR(2)

(b) CMPLR(3)

(c) CMPLR(1)

Figura 105 – Série Google (CMPLR) - Quarto Cenário

(a) CMAHP(1,1)

(b) CMAHP(5,1)

(c) CMAHP(1,5)

Figura 106 – Série FIFA World Cup (CMAHP) - Primeiro Cenário

(a) CMAHP(1,1)

(b) CMAHP(5,1)

(c) CMAHP(1,5)

Figura 107 – Série FIFA World Cup (CMAHP) - Segundo Cenário

(a) CMAHP(1,1)

(b) CMAHP(5,1)

(c) CMAHP(1,5)

Figura 108 – Série FIFA World Cup (CMAHP) - Terceiro Cenário

(a) CMAHP(1,1)

(b) CMAHP(5,1)

(c) CMAHP(1,5)

Figura 109 – Série FIFA World Cup (CMAHP) - Quarto Cenário

(a) CMAHP(1,1)

(b) CMAHP(5,1)

(c) CMAHP(1,5)

Figura 110 – Série Nasa (CMAHP) - Primeiro Cenário

(a) CMAHP(1,1)

(b) CMAHP(5,1)

(c) CMAHP(1,5)

Figura 111 – Série Nasa (CMAHP) - Segundo Cenário

(a) CMAHP(1,1)

(b) CMAHP(5,1)

(c) CMAHP(1,5)

Figura 112 – Série Nasa (CMAHP) - Terceiro Cenário

(a) CMAHP(1,1)

(b) CMAHP(5,1)

(c) CMAHP(1,5)

Figura 113 – Série Nasa (CMAHP) - Quarto Cenário

(a) CMAHP(1,1)

(b) CMAHP(5,1)

(c) CMAHP(1,5)

Figura 114 – Série Clarknet (CMAHP) - Primeiro Cenário

(a) CMAHP(1,1)

(b) CMAHP(5,1)

(c) CMAHP(1,5)

Figura 115 – Série Clarknet (CMAHP) - Segundo Cenário

(a) CMAHP(1,1)

(b) CMAHP(5,1)

(c) CMAHP(1,5)

Figura 116 – Série Clarknet (CMAHP) - Terceiro Cenário

(a) CMAHP(1,1)

(b) CMAHP(5,1)

(c) CMAHP(1,5)

Figura 117 – Série Clarknet (CMAHP) - Quarto Cenário

(a) CMAHP(1,1)

(b) CMAHP(5,1)

(c) CMAHP(1,5)

Figura 118 – Série Google (CMAHP) - Primeiro Cenário

(a) CMAHP(1,1)

(b) CMAHP(5,1)

(c) CMAHP(1,5)

Figura 119 – Série Google (CMAHP) - Segundo Cenário

(a) CMAHP(1,1)

(b) CMAHP(5,1)

(c) CMAHP(1,5)

Figura 120 – Série Google (CMAHP) - Terceiro Cenário

(a) CMAHP(1,1)

(b) CMAHP(5,1)

(c) CMAHP(1,5)

Figura 121 – Série Google (CMAHP) - Quarto Cenário

(a) CMAGM(1,1)

(b) CMAGM(2,1)

(c) CMAGM(1,2)

Figura 122 – Série FIFA World Cup (CMAGM) - Primeiro Cenário

(a) CMAGM(1,1)

(b) CMAGM(2,1)

(c) CMAGM(1,2)

Figura 123 – Série FIFA World Cup (CMAGM) - Segundo Cenário

(a) CMAGM(1,1)

(b) CMAGM(2,1)

(c) CMAGM(1,2)

Figura 124 – Série FIFA World Cup (CMAGM) - Terceiro Cenário

(a) CMAGM(1,1)

(b) CMAGM(2,1)

(c) CMAGM(1,2)

Figura 125 – Série FIFA World Cup (CMAGM) - Quarto Cenário

(a) CMAGM(1,1)

(b) CMAGM(2,1)

(c) CMAGM(1,2)

Figura 126 – Série Nasa (CMAGM) - Primeiro Cenário

(a) CMAGM(1,1)

(b) CMAGM(2,1)

(c) CMAGM(1,2)

Figura 127 – Série Nasa (CMAGM) - Segundo Cenário

(a) CMAGM(1,1)

(b) CMAGM(2,1)

(c) CMAGM(1,2)

Figura 128 – Série Nasa (CMAGM) - Terceiro Cenário

(a) CMAGM(1,1)

(b) CMAGM(2,1)

(c) CMAGM(1,2)

Figura 129 – Série Nasa (CMAGM) - Quarto Cenário

(a) CMAGM(1,1)

(b) CMAGM(2,1)

(c) CMAGM(1,2)

Figura 130 – Série Clarknet (CMAGM) - Primeiro Cenário

(a) CMAGM(1,1)

(b) CMAGM(2,1)

(c) CMAGM(1,2)

Figura 131 – Série Clarknet (CMAGM) - Segundo Cenário

(a) CMAGM(1,1)

(b) CMAGM(2,1)

(c) CMAGM(1,2)

Figura 132 – Série Clarknet (CMAGM) - Terceiro Cenário

(a) CMAGM(1,1)

(b) CMAGM(2,1)

(c) CMAGM(1,2)

Figura 133 – Série Clarknet (CMAGM) - Quarto Cenário

(a) CMAGM(1,1)

(b) CMAGM(2,1)

(c) CMAGM(1,2)

Figura 134 – Série Google (CMAGM) - Primeiro Cenário

(a) CMAGM(1,1)

(b) CMAGM(2,1)

(c) CMAGM(1,2)

Figura 135 – Série Google (CMAGM) - Segundo Cenário

(a) CMAGM(1,1)

(b) CMAGM(2,1)

(c) CMAGM(1,2)

Figura 136 – Série Google (CMAGM) - Terceiro Cenário

(a) CMAGM(1,1)

(b) CMAGM(2,1)

(c) CMAGM(1,2)

Figura 137 – Série Google (CMAGM) - Quarto Cenário