

Cloud Computing (CC) popularization unleashed the creation of a large number of new companies providing CC services, the Cloud Providers (CPs). Therefore, selecting which CPs are most suitable to satisfy each CC customer has become a complex problem with several criteria involved. In order to systematically quantify the quality of each available CP, the criteria used are the Performance Indicators (PIs). Thus, this dissertation proposes a new generalist PI-based architecture for solving the Cloud Provider Selection (CPS) problem. For such, the idea is to find out the smallest CP set that maximizes customer satisfaction with the lowest possible price. To accomplish that, robust deterministic, metaheuristics and hybrid selection methods are developed and used.

Advisor: Dr. Adriano Fiorese

Co-advisor: Dr. Rafael Stubs Parpinelli

Joinville, 2019

YEAR  
2019

LUCAS BORGES DE MORAES | AN ARCHITECTURE TO SELECT AND RANK CLOUD  
COMPUTING PROVIDERS USING PERFORMANCE INDICATORS



**UDESC**

SANTA CATARINA STATE UNIVERSITY – UDESC  
TECHNOLOGICAL SCIENCES CENTRE – CCT  
POSTGRADUATE PROGRAM IN APPLIED COMPUTING

MASTER'S DISSERTATION

# **AN ARCHITECTURE TO SELECT AND RANK CLOUD COMPUTING PROVIDERS USING PERFORMANCE INDICATORS**

LUCAS BORGES DE MORAES

JOINVILLE, 2019

**LUCAS BORGES DE MORAES**

**AN ARCHITECTURE TO SELECT AND RANK CLOUD COMPUTING  
PROVIDERS USING PERFORMANCE INDICATORS**

Dissertation submitted to the Applied Computing Graduate Program of the Technological Sciences Centre of the Santa Catarina State University, to obtain the Master's degree in Applied Computing.

Advisor: PhD. Adriano Fiorese

Co-advisor: PhD. Rafael Stubs Parpinelli

**JOINVILLE**

**2019**

**Ficha catalográfica elaborada pelo programa de geração automática da  
Biblioteca Setorial do CCT/UDESC,  
com os dados fornecidos pelo(a) autor(a)**

Moraes, Lucas Borges de  
An Architecture to Select and Rank Cloud Computing  
Providers using Performance Indicators / Lucas Borges de  
Moraes. -- 2019.  
138 p.

Orientador: Adriano Fiorese  
Coorientador: Rafael Stubs Parpinelli  
Dissertação (mestrado) -- Universidade do Estado de  
Santa Catarina, Centro de Ciências Tecnológicas, Programa  
de Pós-Graduação em Computação Aplicada, Joinville, 2019.

1. Seleção de Provedor de Nuvem. 2. Método de Decisão  
Multicritério. 3. Algoritmo Evolutivo. 4. Método de Seleção  
Híbrido. 5. Indicador de Desempenho. I. Fiorese, Adriano. II.  
Stubs Parpinelli, Rafael. III. Universidade do Estado de Santa  
Catarina, Centro de Ciências Tecnológicas, Programa de  
Pós-Graduação em Computação Aplicada. IV. Título.

# AN ARCHITECTURE TO SELECT AND RANK CLOUD COMPUTING PROVIDERS USING PERFORMANCE INDICATORS

por

**Lucas Borges de Moraes**

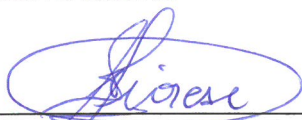
Esta dissertação foi julgada adequada para obtenção do título de

**Mestre em Computação Aplicada**

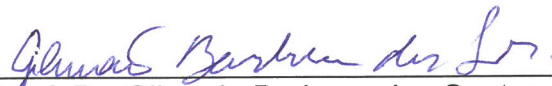
Área de concentração em “Ciência da Computação”,  
e aprovada em sua forma final pelo

CURSO DE MESTRADO ACADÊMICO EM COMPUTAÇÃO APLICADA  
DO CENTRO DE CIÊNCIAS TECNOLÓGICAS DA  
UNIVERSIDADE DO ESTADO DE SANTA CATARINA.

Banca Examinadora:



Prof. Dr. Adriano Fiorese  
CCT/UDESC (Orientador/Presidente)



Prof. Dr. Gilmario Barbosa dos Santos  
CCT/UDESC



Prof. Dr. Rafael de Santiago  
INE/UFSC

**Joinville, SC, 29 de março de 2019.**

I dedicate this work to my advisor and co-advisor, whose guidelines have been very valuable to me and helped to enhance this work. I also dedicate it to all my friends who helped me in this journey; and finally, I declare this work as a written testimony of the victory of the willpower and prayer against the discouragement and the soul depression that struck me in this life.

## **ACKNOWLEDGEMENTS**

Initially, I thank God above all, for allowing me to overcome all the difficulties that have appeared along this tortuous path, for the grace of resilience, health, intelligence and the willpower that was given to me to study. I thank Santa Catarina State University (UDESC) for providing the environment and partial conditions that made this work possible. I thank my advisor Adriano, for his support, his corrections in the texts, his patience and good will for guidance since the undergraduation time to this Master's Thesis. I thank my co-advisor Rafael, for his great knowledge and directives, both in the articles and in this work as well. Thanks to teachers Omir and Charles, for their good humor and encouragement to the scientific research, preparing me for this work. To my friends Luiz, Guilherme, Leonardo and Tiago, for the moments of relaxation, advice, partnership in studies and work, hitchhiking and friendship, relieving daily stress. Finally, I thank all the professors of the Department of Computer Science (DCC), for the knowledge and patience provided over the years that I have been in the Santa Catarina State University. Thank you very much everyone.

“The scientist is not the man who provides the real answers; is who asks the real questions.”

Claude Lévi-Strauss

“Sometimes we feel that what we do is nothing but a drop of water in the sea. But the sea would be smaller if it lacked a drop.”

Madre Teresa de Calcutá

## ABSTRACT

Cloud Computing (CC) is a service model that allows hosting and on demand distribution of computing resources all around the world via Internet. Popularization of cloud services inspired the emergence of a large number of new companies providing CC, also known as Cloud Providers (CPs). This abundance in quantity of CPs does not mean guaranteed growing quality and it can be difficult a precise selection by rigorous customers. Thus, selecting which CPs are the most suitable to satisfy each customer has become a complex problem. So, the need for an automated CP selection method becomes more evident. In order to quantify the quality of each available CP, it is necessary to use a metric, e.g. Performance Indicators or PIs. PI is a useful tool for systematic and synthesized information collection about some real aspect of an organization (e.g., CPs). Bearing this in mind, several works have been developed in the Cloud Provider Selection (CPS) area, but they are very disconnected and do not have a common background basis. In several of them, the top ranked CP is chosen as the best one for the customer. This approach is well suited when customer is interested or the context and conditions allow a single CP to attend the customer needs. However, if the customer needs to take into consideration possible relationships or inter-dependencies between eventually ranked CP (not only the top-most ranked), then, instead of a single CP, there is a set of them at the end of the selection process that can be used to attend the customer. When this happens, it makes the CPS problem much more complex because it creates a combinatorial explosion of possible different solution sets. To overcome such limitation, robust metaheuristics methods are used in this work. Among these metaheuristics, the most classical ones are the Simulated Annealing (SA) and the Evolutionary Algorithms (EAs). Therefore, the central idea adopted for the CPS problem is to find out the smallest set of CPs that maximizes customer satisfaction with the lowest possible price. Thus, the objective of this work is to present a general PI-based architecture for solving the CPS problem including some developed selection methods, either deterministic (using Multicriteria Decision Methods or MCDM), metaheuristics (using EA or SA) and hybrid ones (MCDM with EA and/or SA). Experimental results have shown that the most promising methods are the hybrid ones, especially the one that uses Discretized Differential Evolution (DDE).

**Key-words:** Cloud Provider Selection, Multicriteria Decision Method, Evolutionary Algorithm, Hybrid Selection Methods, Performance Indicators.



## RESUMO

A Computação em Nuvem (CN) é um modelo de serviço que permite a hospedagem e a distribuição sob demanda de recursos computacionais por todo o mundo via Internet. A popularização dos serviços em nuvem inspirou o surgimento de um grande número de novas empresas prestadoras de CN, também conhecidas como Provedores de Nuvem (PNs). Essa abundância da quantidade de PNs não significa uma garantia de qualidade crescente e pode dificultar uma seleção criteriosa de clientes mais rigorosos. Assim, selecionar quais PNs são os mais adequados para satisfazer cada cliente se tornou um problema complexo. Logo, a necessidade de um método automatizado de seleção de PNs torna-se mais evidente. Para quantificar a qualidade de cada PN disponível é necessário o uso de uma métrica, e.g. Indicadores de Desempenho ou PIs. Um PI é uma ferramenta útil para coleta sistemática e sintetizada de informações sobre algum aspecto real de uma organização (e.g., PNs). Tendo isso em mente, vários trabalhos têm sido desenvolvidos na área da Seleção de Provedores de Nuvem (SPN), mas eles são muito desconectados e não possuem uma base conceitual comum. Em vários trabalhos, o PN no topo desse ranking é escolhido como o melhor PN para àquele cliente. Essa abordagem é adequada quando o cliente assim desejar ou o contexto e as condições permitem que um único PN atenda às necessidades do cliente. Porém, se o cliente precisar levar em consideração possíveis relacionamentos ou interdependências entre eventuais PNs ranqueados (não somente o PN mais ao topo), então, em vez de um único PN, há um conjunto deles no final do processo de seleção, que podem ser usados para atender o cliente. Quando isso acontece, torna o problema SPN muito mais complexo, pois cria uma explosão combinatória de soluções possíveis. Para superar tal limitação, métodos metaheurísticos robustos são utilizados neste trabalho. Entre essas metaheurísticas, as mais clássicas são o Simulated Annealing (SA) e os Algoritmos Evolucionários (AEs). Finalmente, a ideia central adotada para o problema SPN é encontrar o menor conjunto de PNs que maximiza a satisfação do cliente com o menor preço possível. Portanto, o objetivo deste trabalho é apresentar uma arquitetura geral, baseada em PIs, para resolver o problema SPN incluindo métodos de seleção desenvolvidos para serem metodologias determinísticas (usando Métodos de Tomada de Decisão Multicritério ou MCDM), metaheurísticas (usando AE ou SA) e híbridas (MCDM com AE e/ou SA). Resultados experimentais mostraram que os métodos mais promissores são os híbridos, especialmente aquele que utiliza a Evolução Diferencial Discretizada (EDD).

**Palavras-chave:** Seleção de Provedor de Nuvem, Método de Decisão Multicritério, Algoritmo Evolutivo, Método de Seleção Híbrido, Indicador de Desempenho.

## LIST OF FIGURES

Figure 1 – NIST reference architecture based on cloud agents combined . . . .	27
Figure 2 – Categories of indicators . . . . .	29
Figure 3 – Utility variation of a quantitative indicator according to its numerical value . . . . .	30
Figure 4 – Utility variation of a qualitative indicator according to its categorical value . . . . .	31
Figure 5 – Example of the basic AHP hierarchy construction to select CPs using PIs as criteria . . . . .	35
Figure 6 – Geometric interpretation of the efficiency frontier for CRS and VRS models . . . . .	42
Figure 7 – SA flowchart . . . . .	44
Figure 8 – Simple GA flowchart . . . . .	47
Figure 9 – BDE flowchart . . . . .	49
Figure 10 – DDE flowchart . . . . .	51
Figure 11 – Proposed architecture overview and plausible utilization scenario . .	67
Figure 12 – Example of a customer interface frame . . . . .	70
Figure 13 – Selection methods utilization and its response types . . . . .	71
Figure 14 – Classification of priorities of the PIs by matching method . . . . .	74
Figure 15 – CPS-Matching method overview . . . . .	75
Figure 16 – CPS-DEA method overview . . . . .	78
Figure 17 – Example of a possible solution encoded in a binary vector . . . . .	84
Figure 18 – E.g.1: Convergence plots for request 3 with 200 CPs and price decrease . . . . .	99
Figure 19 – E.g.2: Convergence plots for request 5 with 200 CPs and price decrease . . . . .	106

## LIST OF TABLES

Table 1 – Standard scale values used to construct Judgement Matrix in AHP . .	36
Table 2 – AHP Judgement Matrix: Modelling of the priorities of each criterion for the main objective . . . . .	37
Table 3 – Normalization of the Judgement Matrix and calculation of each criterion score to the objective . . . . .	37
Table 4 – AHP Judgement Matrix: Modelling of the priorities of each value for the respective criterion 1 . . . . .	38
Table 5 – Normalization of the Judgement Matrix and calculation of each score value to the criterion 1 . . . . .	38
Table 6 – Overview and comparison between the related work and the proposed architecture . . . . .	62
Table 7 – Generic CP database . . . . .	68
Table 8 – Generic customer request . . . . .	69
Table 9 – Parameters used by GA, BDE and DDE algorithms . . . . .	91
Table 10 – E.g.1: Simulated CP database with their quantitative PIs . . . . .	93
Table 11 – E.g.1: Simulated customer requests . . . . .	94
Table 12 – E.g.1: Results for 200 CPs database with 5 quantitative PIs plus the price . . . . .	95
Table 13 – E.g.1: Results for 200 CPs database with arbitrary price decrease (single global optimal) . . . . .	97
Table 14 – E.g.1: Hit percentages of the hybrid methods for 200 CPs with arbitrary price decrease . . . . .	100
Table 15 – E.g.1: Execution time for the developed experiments with 200 CPs, in milliseconds . . . . .	101
Table 16 – E.g.2: Simulated CP database with their qualitative PIs . . . . .	102
Table 17 – E.g.2: Simulated customer requests . . . . .	102
Table 18 – E.g.2: Results for 200 CPs database with 4 qualitative PIs plus the price	103
Table 19 – E.g.2: Results for 200 CPs database with arbitrary price decrease (single global optimal) . . . . .	105
Table 20 – E.g.2: Hit percentages of the hybrid methods for 200 CPs with arbitrary price decrease . . . . .	107
Table 21 – E.g.2: Execution time for the developed experiments with 200 CPs, in milliseconds . . . . .	107
Table 22 – E.g.3: CPs database built with data from Cloudwards . . . . .	109
Table 23 – E.g.3: Proposed customer requests . . . . .	111
Table 24 – E.g.3: Results for the real CPs database and from Cloudwards . . . .	113

Table 25 – E.g.3: Execution time for the real use case, in milliseconds . . . . . 114

Table 26 – Recommended RI values for a Judgement Matrix with dimension  $n$  . 133

Table 27 – Judgement Matrix: Relations of importance between each different PI  
level . . . . . 133

Table 28 – Normalized Judgement Matrix for each importance level . . . . . 134

Table 29 – Ex.1: Dunn test with Bonferroni correction for the hits values at Table 13.136

Table 30 – Ex.1: Dunn’s test with Bonferroni correction for the hits values at Ta-  
ble 14. . . . . 136

Table 31 – Ex.2: Dunn’s test with Bonferroni correction for the hits values at Ta-  
ble 19. . . . . 137

Table 32 – Ex.2: Dunn test with Bonferroni correction for the hits values at Table 20.137

## LIST OF ABBREVIATIONS AND ACRONYMS

<b>AHP</b>	Analytic Hierarchy Process
<b>ANP</b>	Analytical Network Process
<b>BCC</b>	Banker, Charnes and Cooper
<b>BDE</b>	Binary Differential Evolution
<b>CC</b>	Cloud Computing
<b>CCR</b>	Charnes, Cooper and Rhodes
<b>CI</b>	Consistency Index
<b>CP</b>	Cloud Provider
<b>CPS</b>	Cloud Provider Selection
<b>CPS-BDE</b>	Cloud Provider Selection using Binary Differential Evolution
<b>CPS-DDE</b>	Cloud Provider Selection using Discretized Differential Evolution
<b>CPS-DEA</b>	Cloud Provider Selection using Data Envelopment Analysis
<b>CPS-GA</b>	Cloud Provider Selection using Genetic Algorithm
<b>CPS-Matching</b>	Cloud Provider Selection using PI Matching
<b>CPS-SA</b>	Cloud Provider Selection using Simulated Annealing
<b>CR</b>	Consistency Ratio
<b>CRS</b>	Constant Returns to Scale
<b>CSMIC</b>	Cloud Service Measurement Index Consortium
<b>DE</b>	Differential Evolution
<b>DEA</b>	Data Envelopment Analysis
<b>DDE</b>	Discretized Differential Evolution
<b>DMU</b>	Decision-Making Unit
<b>EA</b>	Evolutionary Algorithm
<b>ELECTRE</b>	Elimination et Choix Traduisant la Réalité

**GA** Genetic Algorithm

**HB** Higher is Better

**HLT** Higher and Lower are Tolerable

**HT** Higher is Tolerable

**IaaS** Infrastructure-as-a-Service

**ISYDS** Integrated System for Decision Support

**KPI** Key Performance Indicator

**LB** Lower is Better

**LP** Linear Programming

**LT** Lower is Tolerable

**Matching-BDE** PI Matching with Binary Differential Evolution hybrid

**Matching-DDE** PI Matching with Discretized Differential Evolution hybrid

**Matching-GA** PI Matching with Genetic Algorithm hybrid

**Matching-SA** PI Matching with Simulated Annealing hybrid

**MAUT** Multi Attribute Utility Theory

**MCD** Multicriteria Decision Analysis

**MCDM** Multicriteria Decision Method

**NB** Nominal is Best

**NIST** National Institute of Standards and Technology

**OS** Operational System

**PaaS** Platform-as-a-Service

**PROMETHEE** Preference Ranking Organization Method for Enriched Evaluation

**PI** Performance Indicator

**QoE** Quality of Experience

**QoS** Quality of Service

**RI** Random Index

**SA** Simulated Annealing

**SaaS** Software-as-a-Service

**SLA** Service Level Agreement

**SMI** Service Measurement Index

**TOPSIS** Technique for Order Preference by Similarity to Ideal Solution

**VM** Virtual Machine

**VRS** Variable Returns to Scale

## CONTENTS

<b>1</b>	<b>INTRODUCTION . . . . .</b>	<b>17</b>
1.1	MAIN CONTRIBUTIONS . . . . .	19
1.2	PUBLICATIONS . . . . .	20
1.3	OBJECTIVES . . . . .	21
1.4	RESEARCH METHODOLOGY . . . . .	22
1.5	WORK STRUCTURE . . . . .	23
<b>2</b>	<b>GENERAL BACKGROUND . . . . .</b>	<b>25</b>
2.1	CLOUD COMPUTING . . . . .	25
2.2	PERFORMANCE INDICATORS . . . . .	28
<b>2.2.1</b>	<b>Performance Indicators for Cloud Providers . . . . .</b>	<b>32</b>
2.3	SELECTION METHODS . . . . .	33
<b>2.3.1</b>	<b>Deterministic Approaches . . . . .</b>	<b>33</b>
2.3.1.1	<i>Analytic Hierarchy Process . . . . .</i>	<i>34</i>
2.3.1.2	<i>Data Envelopment Analysis . . . . .</i>	<i>39</i>
<b>2.3.2</b>	<b>Metaheuristic Approaches . . . . .</b>	<b>42</b>
2.3.2.1	<i>Simulated Annealing . . . . .</i>	<i>43</i>
2.3.2.2	<i>Genetic Algorithmic . . . . .</i>	<i>45</i>
2.3.2.3	<i>Binary Differential Evolution . . . . .</i>	<i>48</i>
2.3.2.4	<i>Discretized Differential Evolution . . . . .</i>	<i>50</i>
2.4	PARTIAL CONSIDERATIONS . . . . .	53
<b>3</b>	<b>RELATED WORK . . . . .</b>	<b>54</b>
3.1	CLOUD PROVIDER SELECTION USING NON MCDA METHODS . . . . .	54
3.2	CLOUD PROVIDER SELECTION USING MCDA METHODS . . . . .	57
3.3	RELATED WORK ASSESSMENT . . . . .	61
3.4	PARTIAL CONSIDERATIONS . . . . .	65
<b>4</b>	<b>SELECTING AND RANKING CLOUD PROVIDERS . . . . .</b>	<b>66</b>
4.1	CLOUD PROVIDER SELECTION ARCHITECTURE . . . . .	66
<b>4.1.1</b>	<b>Candidate Providers' Database . . . . .</b>	<b>67</b>
<b>4.1.2</b>	<b>Customer Request . . . . .</b>	<b>69</b>
<b>4.1.3</b>	<b>Customer Interface . . . . .</b>	<b>69</b>
4.2	SELECTION METHODS MODELLING . . . . .	71
<b>4.2.1</b>	<b>The PI Attendance Condition Concept . . . . .</b>	<b>72</b>



<b>4.2.2</b>	<b>Deterministic Methods Modelling</b>	<b>73</b>
4.2.2.1	<i>CPS-Matching</i>	73
4.2.2.2	<i>CPS-DEA</i>	78
<b>4.2.3</b>	<b>Metaheuristic Methods Modelling</b>	<b>82</b>
4.2.3.1	<i>Individual Modelling</i>	83
4.2.3.2	<i>Fitness Modelling</i>	84
4.2.3.3	<i>Energy Modelling</i>	87
<b>4.2.4</b>	<b>Hybrid Methods Modelling</b>	<b>87</b>
4.3	PARTIAL CONSIDERATIONS	88
<b>5</b>	<b>EXPERIMENTS AND RESULTS</b>	<b>90</b>
5.1	EXPERIMENTS SETUP	90
5.2	USE CASE – QUANTITATIVE PIs	92
<b>5.2.1</b>	<b>Results and Analysis – Quantitative PIs</b>	<b>94</b>
5.3	USE CASE 2 – QUALITATIVE PIs	100
<b>5.3.1</b>	<b>Results and Analysis – Qualitative PIs</b>	<b>103</b>
5.4	USE CASE – REAL PIs	108
<b>5.4.1</b>	<b>Results and Analysis – Real PIs</b>	<b>112</b>
5.5	PARTIAL CONSIDERATIONS	115
<b>6</b>	<b>FINAL CONSIDERATIONS</b>	<b>117</b>
6.1	RECOMMENDATIONS FOR FUTURE WORK	119
	<b>BIBLIOGRAPHY</b>	<b>121</b>
	<b>APPENDIX A – PERFORMANCE INDICATORS LISTS FOR CLOUD PROVIDERS</b>	<b>127</b>
	<b>APPENDIX B – CONSISTENCY VERIFICATION OF A JUDGEMENT MATRIX</b>	<b>132</b>
B.1	SOLVING AN AHP JUDGEMENT MATRIX	133
	<b>APPENDIX C – STATISTICAL EVALUATION OF EXPERIMENTAL RESULTS</b>	<b>135</b>

## 1 INTRODUCTION

Modernization of society and evolution of telecommunications technology (especially computer networks) has been provided a perfect environment for the rise of cloud computing paradigm. Cloud Computing (CC) is a service model that allows a large, on demand, hosting and distribution of computing resources through computer networks (HOGAN et al., 2013). CC also brought the benefit of better use of computational resources being a convenient service easily accessible via Internet (ZHANG; CHENG; BOUTABA, 2010). This model becomes a basis for innovation and efficiency on provisioning of computational services and spreading to all five continents. This popularization of CC usage inspired the emergence of a large number of new companies providing CC services (HÖFER; KARAGIANNIS, 2011), also known as Cloud Providers (CPs) (HOGAN et al., 2013).

This rapid growth in the quantity of CPs does not mean guaranteed growing CP service offering quality and can difficult a precise choice of CP by more careful and discerning customers, since the demand for better Quality of Service (QoS) was also been increased (GARG; VERSTEEG; BUYYA, 2013; BARANWAL; VIDYARTHI, 2014). Thus, selecting which CPs are the most suitable to satisfy each customer needs has become a complex problem to be solved manually. Thus, the need for an automated selection method became evident and justified. This method has to consider each individual customer requirements, based on his computing needs. To accomplish this, the method need to quantify the quality of each available CP, using a metric, e.g., Performance Indicator (PI).

PI is a useful tool for systematic information collection about some crucial aspects of an organization (like CPs) (POLLOCK, 2007). PIs are responsible for quantifying (assigning a value, whether quantitative or qualitative) a particular aspect regarding the objects of study. CC has a noticeable key set of PIs organized in a hierarchical framework divided into seven major categories called Service Measurement Index (SMI), developed by Cloud Service Measurement Index Consortium (CSMIC) (SIEGEL; PERDUE, 2012).

This is the origin of the called Cloud Provider Selection (CPS) problem. Regarding the solution for this problem, it stands out the influence of the Operational Research area called Multicriteria Decision Analysis (MCDA) (ISHIZAKA; NEMERY, 2013). There are several MCDA methods used to rank CPs according to multiple criteria (WHAIDUZZAMAN et al., 2014), like Analytic Hierarchy Process (AHP) (GARG; VERSTEEG; BUYYA, 2013), Elimination et Choix Traduisant la Realité (ELECTRE)

(MACEDO; SILVA; FERREIRA, 2013) and Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) (JAISWAL; MISHRA, 2017). Basically, the top ranked CP is chosen as the best CP for the customer. This approach is well suited when user is interested or the context and conditions allow a single CP to attend the user needs. On the other hand, other approach is needed when the user/customer needs take into consideration possible relationships or interdependencies between eventually ranked CPs (not only the top-most ranked), i.e, when a single CP will not be able to fully attend the entire customer request (MORAES; FIORESE; PARPINELLI, 2017). For example, if a customer needs a certain amount of processing power and storage, two different CPs can provide better amount of each one of these resources separately than a single CP including lower total price (sum of both CPs prices). Thus, in this case, instead of a single CP, there is a set of them at the end of the selection process that can be used to attend the customer. This fact makes the CPS problem much more complex because it creates a combinatorial explosion of possible different solution sets. To overcome such limitation, robust metaheuristics methods are used in this work. The metaheuristics used are the classical ones: SA (KIRKPATRICK; GELATT; VECCHI, 1983), thermodynamics-inspired algorithm, and the Evolutionary Algorithms (EAs), populational bio-inspired techniques, such as the Genetic Algorithm (GA) (GOLDBERG, 1989) and Differential Evolution (DE) (STORN; PRICE, 1995). Thus, for this work, the adopted central idea for the CPS problem is to find out the smallest set of CPs that maximizes customer satisfaction with the lowest possible price. Maximizing the customer satisfaction corresponds in fact to maximize the attendance of the customer needs comprising the CC services modelled by CP PIs expressed in a request. At the best of our efforts no similar work was found in literature.

The CPS problem has become noticeable in this decade and several works were already developed, trying to solve it in an acceptable way. Many of them use MCDA methods and the PIs belonging to the SMI as base indicators, especially the total cost/price involved (SUNDARESWARAN; SQUICCIARIN; LIN, 2012; REHMAN; HUSSAIN; HUSSAIN, 2012; GARG; VERSTEEG; BUYYA, 2013; KARIM; DING; MIRI, 2013; MACEDO; SILVA; FERREIRA, 2013; BARANWAL; VIDYARTHI, 2014; ACHAR; THILAGAM, 2014; WAGLE et al., 2015; SOUIDI et al., 2015; JAISWAL; MISHRA, 2017). Moreover, several surveys and reviews comprising the CPS problem have been published too (ALABOOL; MAHMOOD, 2013; WHAIDUZZAMAN et al., 2014; ALMANEA, 2014; DHIVYA; DEVI; SHANMUGALAKSHMI, 2016). Although there are a significant effort comprising the CPS solution, these works present disconnected problem descriptions and modelling (problem background), thus generating several methods and approaches lacking of an integrative and unified model (e.g., architecture) about the problem. This integration is important because it will allow the construction of an extensive framework using these methods, with standardized inputs and outputs, allowing

a fair comparison between them and also serving as a guideline for other authors for the construction of new methods in the future. In addition, most of these methods are not easily scalable for larger instances of this problem, i.e., with several CPs and PIs.

Therefore, this work proposes a general architecture for solving the CPS problem that is based on PIs of candidate CPs and customer requests, which uses several approaches and selection methods (e.g., deterministic and metaheuristic ones). There is no similar architecture found in the studied literature, at the best of our efforts. Particularly, in this work, the deterministic methods modelled, implemented and tested are: CPS-Matching and CPS-DEA. The metaheuristic ones are: CPS-SA, CPS-GA, CPS-BDE and CPS-DDE. Finally, the hybrid ones are: Matching-GA, Matching-BDE, Matching-DDE and Matching-SA. Several experiments with the developed methods were performed with simulated and real data. Results of these experiments are reported and compared, showing the method's answer suitability to this problem. Comprising the experiments, CP PI databases and customer requests were created since the CPS problem lacks well-accepted benchmarks.

Thus, taking into account the developed work, contributions can be highlighted. Moreover, comprising these contributions a set of scientific publications can be related as well. The following subsections list these achievements.

## 1.1 MAIN CONTRIBUTIONS

As aforementioned, regarding the general contributions of this work, it is possible to highlight:

1. The CPS problem solving modelling as a general software architecture using CPs' PIs as easy data inputs and outputs;
2. An architectural design contributing to a potential standardization on CPS problem solving;
3. The modelling of either deterministic, metaheuristic and hybrid selection methods for the CPS problem;
4. The individual (a CPS problem possible solution) and the suitable fitness/energy function modelling used by the developed metaheuristic algorithms and their hybrid counterparts;
5. The mathematical modeling leading to a set of equations about how to transform the CPS problem input data, i.e., CP database's PIs and user requests, into inputs and outputs to feed the DEA method used in one of our deterministic methods developed;

6. A general qualitative data <sup>1</sup> ontology (e.g., HT, LT, HLT PI characteristics) development that enables the proposed architecture to deal with qualitative beyond quantitative PIs;
7. A particular tolerance mechanism applied to tolerate small differences between user requested PIs and CPs offering;
8. The bibliographic review and a comparative related works table providing insights to approaching the CPSs problem; and
9. The lists of PIs for CPs, derived from bibliographic research (Appendix A).

## 1.2 PUBLICATIONS

By the time this work was written, this work has generated the following publications:

- **Published:**

1. MORAES, Lucas B.; FIORESE, Adriano; MATOS, Fernando. A multi-criteria scoring method based on performance indicators for cloud computing provider selection. In: 19th International Conference on Enterprise Information Systems (ICEIS 2017). Porto, Portugal, 2017. v. 2, p. 588–599.
2. MORAES, Lucas B.; FIORESE, Adriano. A scoring method based on criteria matching for cloud computing provider ranking and selection. In: S. SMIALEK M., C. O. F. J. H. (Ed.). Enterprise Information Systems. Porto, Portugal: Springer, 2018. v. 321, cap. Software Agents and Internet Computing, p. 339–365.
3. MORAES, Lucas B.; FIORESE, Adriano; PARPINELLI, Rafael. An evolutive scoring method for cloud computing provider selection based on performance indicators. In: Proceedings of the 16th Mexican International Conference on Artificial Intelligence (MICAI 2017). Baja California, Mexico, 2017. p. 1–12.
4. MORAES, Lucas B. et al. An efficiency frontier based model for cloud computing provider selection and ranking. In: Proceedings of the 20th International Conference on Enterprise Information Systems (ICEIS 2018). Madeira, Portugal, 2018. p. 543–554.
5. MORAES, Lucas B.; FIORESE, Adriano; PARPINELLI, Rafael. An Evolutive Hybrid Approach to Cloud Computing Provider Selection. In: IEEE Congress

---

<sup>1</sup> criteria/attributes/performance indicators, etc.

- on Evolutionary Computation (IEEE CEC 2018), Rio de Janeiro, Brazil, 2018. p. 1564–1571.
6. MORAES, Lucas B.; FIORESE, Adriano; PARPINELLI, Rafael. Exploring Evolutive Methods for Cloud Provider Selection based on Performance Indicators. In: 7th Brazilian Conference on Intelligent Systems (BRACIS 2018), São Paulo, Brazil, 2018. p. 157-162.
  7. MORAES, Lucas B.; FIORESE, Adriano. Um Método de Seleção de Provedores de Computação em Nuvem Baseado em Indicadores de Desempenho. In: Proceedings of the 15th Workshop de Computação em Clouds e Aplicações (WCGA 2017). Belém, Pará, Brazil, 2017. p. 44-57.
  8. MORAES, Lucas B.; BERTHELSEN, Robson; FIORESE, Adriano. Selecionando Provedores de Computação em Nuvem usando Indicadores de Desempenho. In: Proceedings of the 17th Escola Regional de Alto Desempenho do Rio Grande do Sul (ERAD-RS 2017), Ijuí, RS, Brazil, 2017. p. 367-370.
  9. MORAES, Lucas B.; FIORESE, Adriano; PARPINELLI, Rafael. Selecionando Provedores de Computação em Nuvem via um Algoritmo Genético e baseado em Indicadores de Desempenho. In: Proceedings of the 18th Escola Regional de Alto Desempenho do Rio Grande do Sul (ERAD-RS 2018), Porto Alegre, RS, Brazil, 2018. p. 269-270.

• **Submitted:**

1. MORAES, Lucas B.; FIORESE, Adriano; PARPINELLI, Rafael. A Cloud Provider Selection Architecture Composed of Deterministic, Stochastic and Hybrid Methods. In: Journal of Applied Soft Computing. Elsevier, 2018.

### 1.3 OBJECTIVES

The general objective of this work is to propose a new general architecture for solving the CPS problem, based on PIs of the various candidate CPs and customer requests. This architecture can use several selection methods modelling for the CPS problem, both deterministic and metaheuristic ones. Also, this approach can consider more than one CP to maximize the attendance of all the demanded resources needed by the cloud customer. Finally, this architecture was built to be easily improved and expanded in the future.

The specific objectives of this work are as follow:

1. To review concepts of: Cloud Computing, Performance Indicators and selection methods – MCDM, like AHP (Judgement Matrix) and DEA, SA and EA methods, like GA, BDE and DDE;
2. To list some notable PIs for CPs;
3. To present a bibliographical review to identify the state of the art for the CPS problem;
4. To specify each component of the proposed CPS architecture using PIs;
5. To specify the modelling developed for each CPSs methods – Deterministics (e.g., CPS-Matching and CPS-DEA), metaheuristics (e.g., fitness and energy function modelling, used by CPS-SA, CPS-GA, CPS-BDE and CPS-DDE) and hybrid ones (CPS-Matching combined with each one of the metaheuristics);
6. To implement the proposed CPSs architecture and all the selection methods described;
7. To test this implementation at feasible use cases; and
8. To compare results and performance of all the selection methods at each use case.

#### 1.4 RESEARCH METHODOLOGY

For the classification of this research there are considered the proposals in: Eden (2007), Gil (2008), Gerhardt e Silveira (2013) and Wazlawick (2014). Thus, this research is classified comprising to the method (bibliographic research, interview, questionnaires, etc.), nature (applied or pure), approach (quantitative or qualitative), objectives (exploratory, descriptive and explanatory), scientific method (dialectical, deductive, inductive, hypothetical-deductive), level of maturity (1–5) and paradigms of science (rationalist, technocratic and naturalist).

Regarding the method or procedure of research, this research is mainly classified as a bibliographical research, due to its elaboration being carried out from materials available in the specialized literature (books, articles, conference papers, journals, etc.), and also experimental, since this research suggests the application of the modelled methods to a set of tests with quantitative data, i.e, the solution must be proven through the application and analysis of some experiment.

As for nature, this research is categorized as applied, since it uses a set of specific theoretical foundations to generate knowledge for the practical application and to address the solution for solve real problems like the automated choice of a cloud

provider through a mathematical method, allowing customers to select real providers, based on a planned decision with a theoretical basis.

Regarding the approach, this research is categorized as quantitative, since it requires the use of resources and statistical techniques and mathematical calculations for the providers selection, adding a numerical value (ranking) to each one. It is worked predominantly with measurable quantities regarding cloud computing providers, possible to be treated algebraically, as amount of memory, response time, availability, etc. The generated results can be analysed statistically and through the computational experiments carried out, considered the viability of them.

According to the objectives or purpose, it can be said that this research is classified as exploratory, because through a bibliographical survey, it aims to carry out a preliminary overview of the study area, identifying research opportunities and also outlines some interpretations and solutions about the object of research.

As for the scientific method, the inductive method is predominantly adopted in this research, since the process of reasoning starts from the knowledge of specific phenomena of the research object for the formulation of the theory, that is, from the particular to the general. It is also a bit of deductive, fundamental for the elaboration of new mathematical methods.

Comprising the research maturity, according to the scale proposed in Wazlawick (2014), this research can be classified as level 2, that is, the presentation of something different, not necessarily the best one. This work was classified in this level of maturity due to the fact that there is no universal benchmark for selection of cloud providers, the tests are conducted by the author himself with his own data set in a controlled environment and a comparison is made between the developed methods.

Finally, as regards to the paradigm of science (EDEN, 2007), it is classified as predominantly naturalistic (followed by rationalist and technocratic), because the work is strongly based on mathematics (rationalist), involving mathematical decision-making methods, as well as evolutionary algorithms, which are a mixture of theoretical knowledge a priori and a posteriori (naturalist). The selection methods will have much prior knowledge of the providers (database).

## 1.5 WORK STRUCTURE

This work is divided into six chapters, and it is organized as follows. Chapter 2 presents the bibliographic review (theoretical foundation or general background) comprising the basic concepts related to the development of this work: Cloud Computing, Performance Indicators and selection methods.



Next, Chapter 3 presents some related works to this research. They involve methods, frameworks or architectures for CPs selection and ranking in general using a single method or a set of them, PIs-based or not. At the end, a comparative table with these works and the proposed selection architecture is presented. The objective is situating the proposed work in the context of its state of the art.

Chapter 4 is the central part of this work presenting in details the proposed general architecture for solving the CPS problem. It discusses the architecture main elements, concepts and selection methods' specifications using PIs.

Chapter 5 elaborates on a varied set of experiments to test and compare each employed selection method as well as to validate their developed modelling regarding the CPS problem. Databases and requests, parameters used, results obtained and their analyses are described here.

Finally, Chapter 6 presents the final considerations and future work.

## 2 GENERAL BACKGROUND

The purpose of this chapter is to present the main concepts necessary for the development and understanding of the proposed architecture and selection methods modelling, using PIs, for solving the CPS problem. This chapter is divided into three main sections. Section 2.1 discusses Cloud Computing, involving issues such as concept, key characteristics and reference architecture. Section 2.2 aims to present an introduction about Performance Indicators explaining the concept, characteristics and classification, also introducing the SMI and presenting a list of PIs for CPs. Section 2.3 introduces key concepts of the operation of some notable selection methods, being two deterministic: AHP and DEA and four metaheuristics: an energy function algorithm (SA) and the evolutionary algorithms (GA, BDE and DDE). The concepts about PIs and selection methods are the key elements for the understanding of the architecture formulation presented in Chapter 4.

### 2.1 CLOUD COMPUTING

Cloud Computing (CC) emerged as a new paradigm for hosting and distributing computing services using the Internet (ZHANG; CHENG; BOUTABA, 2010; HOGAN et al., 2013). CC is a model that abstracts from the user (e.g., service customer) the complex internal infrastructure and architecture of that service, not requiring the user to perform installations, configurations, software updates or purchase of additional hardware to use it (MIERS et al., 2014). In addition, it is a convenient service for its users, easily accessible via the network and priced only for the time and computing power demanded ("Pay-as-you-go" practice) (ZHANG; CHENG; BOUTABA, 2010).

National Institute of Standards and Technology (NIST) (HOGAN et al., 2013) defines CC as a service model that at any time must make available to its customer a set of physical or virtual computing resources, accessible from any device with access to the Internet, allowing the release or provisioning resources on demand. In this model all the computational resources and processing that the customer needs can be managed by the Cloud Provider (CP) himself (ZHANG; CHENG; BOUTABA, 2010). To accomplish that, the CP maintains the necessary resources for the effective creation, availability and maintenance of the cloud in superstructures called data centers. Data centers provide infrastructure with power, cooling, connectivity, properly trained professionals and high data availability (redundancy, scalability and security) (ZHANG; CHENG; BOUTABA, 2010; MIERS et al., 2014). CC is composed of five essential characteristics, three service models and four deployment models (HOGAN et al., 2013).

The five essential characteristics, according (HOGAN et al., 2013), are: (i) **On-demand self-service**: The customer can automatically define the type and amount of resources (processing power, storage capacity, bandwidth and virtual machines) of the cloud as needed, with or without minimal human interaction with the CP; (ii) **Broad network access**: The customer data is continuously distributed on the network and can be accessed from anywhere at any time on various platforms (smartphones, laptops, desktops, tables, etc.); (iii) **Resource pooling**: The CP resources are grouped to serve multiple customers and can be dynamically allocated according with the demand. Moreover, customers have no control or knowledge of the exact location of those resources; (iv) **Rapid elasticity**: Resources can be quickly allocated and moved between customers to ensure the scalability of services, giving the impression of unlimited resources and that they can be purchased in any quantity and at any time; and (v) **Measured service**: The cloud automatically controls and optimizes the use of its resources and promotes monitoring, metering, and billing according with the type of service, so that both provider and customer can monitor and control the use of the resources, transparently.

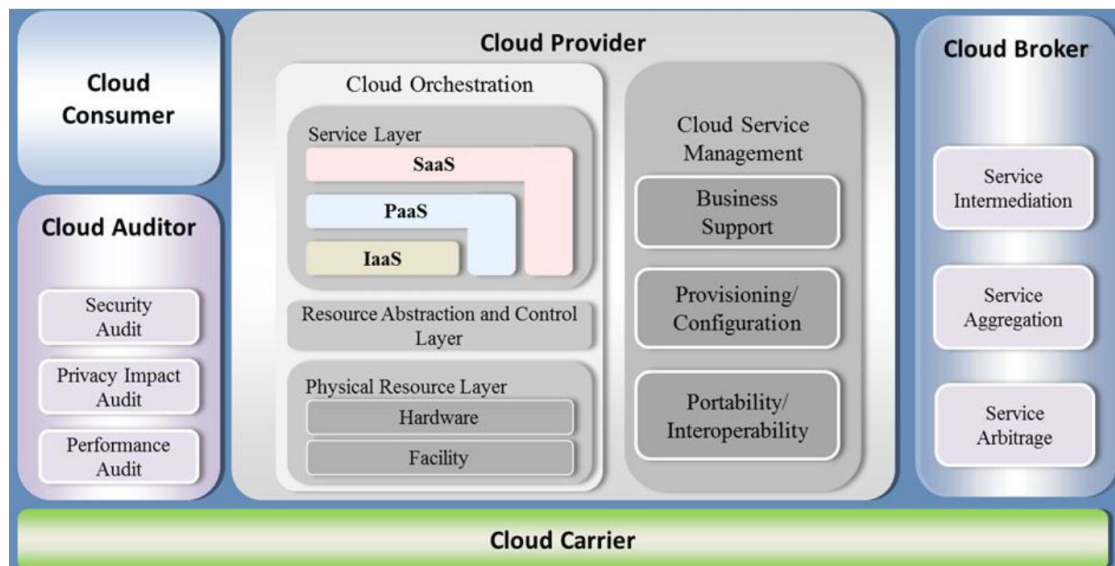
The service model specifies how the cloud features set can be made available to the end user (HOGAN et al., 2013). This classification is simply called SPI (Software-as-a-Service or SaaS, Platform-as-a-Service or PaaS, Infrastructure-as-a-Service or IaaS), characterizing what can be offered in each layer of cloud services abstraction (HOGAN et al., 2013): (i) **SaaS**: Cloud provider offers applications that are hosted on a cloud infrastructure, accessible to any device through an interface, so that the customer does not manage or control the cloud infrastructure, but only specific settings for the user in the application itself; (ii) **PaaS**: The customer can host, implement, test and make available their own applications through an online platform, using different programming languages, libraries and tools supported by the CP, reducing costs and complexity of installation and management; and (iii) **IaaS**: Delivery of computing services in the most fundamental form (e.g., processing, storage capacity, virtual machines, etc.), so the customer can use these resources in their own interest, as long as they do not control or manage the basic infrastructure of the cloud, since this is the CP's task.

The deployment models present how the cloud can be integrated into the organization environment. They are classified according to the location of the cloud infrastructure and how users access it (HOGAN et al., 2013): (i) **Public**: The cloud infrastructure is maintained and owned by the CP itself and it is made available to the general public and can be accessed by any user over the Internet; (ii) **Private**: This computational infrastructure is in general owned, maintained and used exclusively by an organization, constituting itself in a local or remote cloud where only the members

of this organization have exclusive access of the computational resources, obtaining greater trustworthiness and confidentiality on the information; (iii) **Community**: There is a sharing of cloud infrastructure across multiple organizations and/or communities, either locally or remotely, that can be managed by some community or third party company; and (iv) **Hybrid**: It is a combination of two or more of any of the cloud models described above.

NIST also offers a high-level reference architecture (Figure 1) that is a source of information relevant to the discussion of the key requirements, structures, and operations that CC should have (HOGAN et al., 2013). This architecture aims to facilitate understanding of the operational complexities of CC and it is relevant in identifying and delineating the responsibilities of each actor in the cloud.

Figure 1 – NIST reference architecture based on cloud agents combined



Source: Adapted from Hogan et al. (2013)

Figure 1 presents the reference architecture, identifying the five main actors by NIST. Each actor is an entity (person or organization) that participates in a transaction or process and performs tasks in the cloud (HOGAN et al., 2013):

- **Consumer**: Person or organization that maintains a business relationship and uses cloud services. He can request services according to his needs, choosing among the service models offered (SPI);
- **Provider**: Entity responsible for delivering a service to stakeholders. Provides the construction of the platform software or infrastructure demanded, abstracting the physical resources layer (hardware and facilities) and managing the cloud service (offering support, configurations, portability and interoperability), leaving the system with data security and privacy;

- **Auditor:** Person or group that can conduct an independent audit of the cloud services, collecting information about the system, then evaluating the quality of services provided in terms of security controls, impact on privacy and performance. It basically provides three types of audits: security, privacy and performance auditing;
- **Broker:** Entity that manages the use, performance and delivery of cloud services, and negotiates the relationship between providers and consumers. It basically provides three types of brokerage services: intermediation, aggregation and service arbitrage; and
- **Carrier:** Intermediary that provides connectivity between the provider and the consumer, allowing access to the cloud services through some data communication infrastructure.

The success of the CC paradigm has encouraged its adoption in practically every major IT company known (e.g., Google, Amazon, Microsoft, Salesforce, etc.) and has become a good source of investment in both academia and industry (HÖFER; KARAGIANNIS, 2011). Due to this success, a large number of new companies were created, competitively, as providers of cloud computing services.

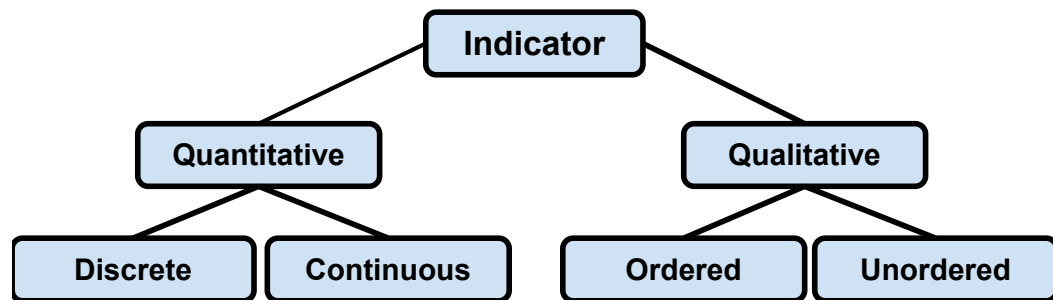
This new cloud companies creation growth has been making increasingly difficult for the end user (cloud consumer) to decide which CP is best suited to fit their needs. This happens because the differences between CPs are becoming more difficult to identify as well as the amount of CPs makes the manual process of choice inefficient. Therefore, the need for methods that assess the quality of CPs, based on customer demand (e.g., Performance Indicators), have become essential to make a comparative choice between them.

## 2.2 PERFORMANCE INDICATORS

An indicator is a tool that allows a synthesized collection (containing only the essential data) of information relating to a certain aspect of reality (POLLOCK, 2007). The choice of indicators to be used to measure the efficiency of a particular process in the organization depends on many factors, linked both to the interests of the customers and to the needs of the organization itself. For an indicator to be effective, it must be able to measure the relevant aspects of the organization, be easy to measure and have measures consistent with the organization's objectives. Indicators also should be measured with a certain frequency, to not become outdated (POLLOCK, 2007).

It is possible to classify indicators into two categories (JAIN, 1991). Figure 2 illustrates this classification.

Figure 2 – Categories of indicators



Source: Adapted from Jain (1991)

- **Quantitative:** They are those states, levels or categories that can be expressed numerically and can be worked algebraically. Assigned numeric values can be discrete or continuous. Regarding discrete, all values can be counted and matched one by one with the subset of all positive (natural) integers, that is, an integer value between two consecutive integer values can not be found, e.g.: number of processors, amount of RAM, size of disk blocks, etc. The continuous are capable of assuming countless distinct values within a given range, and so they are comprised by the set of real numbers (floating point number), e.g.: response time, weight, length of an object, area of a terrain, volume of a water box, etc.
- **Qualitative:** Also called categorical. They can represent distinct states, levels, or categories that are defined by an exhaustive and mutually exclusive set of subclasses, which may or may not be ordered. The ordered subclasses have a perceptible logical graduation among their subclasses, giving an idea of a progression, e.g.: security level (low, medium, high), categories of computers (microcomputer, minicomputer, supercomputer) public service (never, rarely, sometimes, usually, always), level of trust (distrust, trust little, trust partially, trust fully), etc. Unordered subclasses do not have this idea of progression. They are subclasses that are independent and eventually exclusive, e.g.: type of computational service (processing, storage, connectivity), research purpose (scientific, engineering, educational), etc.

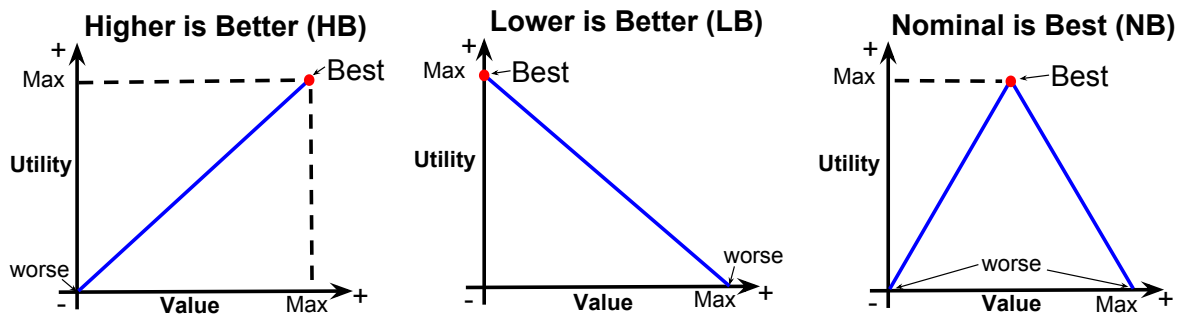
The quantitative indicators can also be classified according to the behavior of their utility function, that is, how good (useful, beneficial) the indicator actually becomes when its numerical value changes (increases or decreases) (JAIN, 1991). According to Jain (1991), there are three possible classifications:

- **Higher is Better (HB):** Users and/or system managers prefer always the highest possible values for this indicator. Examples: system throughput, amount of resources (money, memory, materials, etc.), availability of a service, etc.

- **Lower is Better (LB):** Users and/or system managers prefer always the lowest possible values for this indicator. Examples: response time (delay), costs, etc.
- **Nominal is Best (NB):** Users and/or system managers prefer specific values (higher and lower values are undesired). A particular value is considered the best. Usage is an example of this feature. Very high usage is considered poor by users because it generates high response times. Too low usage is considered by system administrators to be poor since system resources are not being used.

Figure 3 presents this classification and the utility function variation ( $utility = f(value)$ ) according to its numerical value in the system. It is always desired to maximize utility.

Figure 3 – Utility variation of a quantitative indicator according to its numerical value



Source: Adapted from Jain (1991)

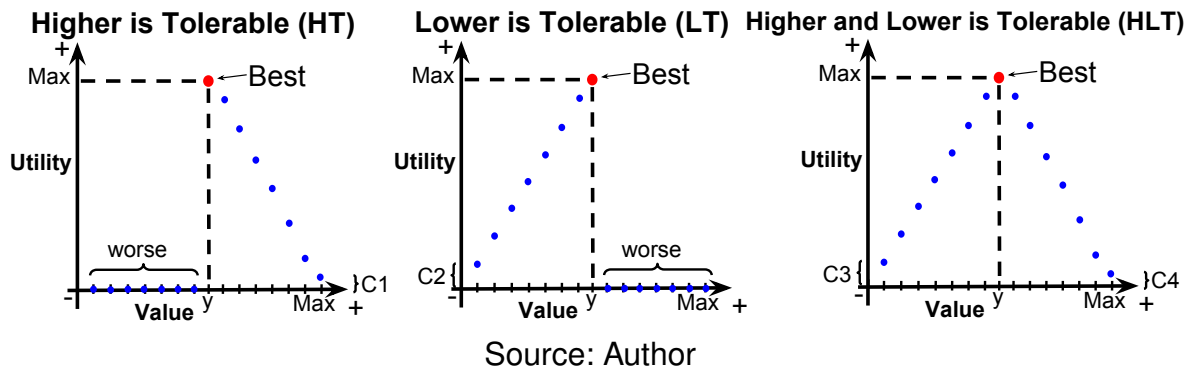
The utility function of qualitative indicators is harder to map using numerical functions. A trivial mapping is the binary one, i.e., the category or subclass (qualitative indicator “value”) chosen by the customer has maximum utility and all the others has minimal utility. This is a similar (but not the same) behaviour of the NB utility function for quantitative indicators. This is a useful mapping process for unordered qualitative indicators. However, the ordered ones have a perceptible logical graduation among their category and consequently a more intelligent mapping can be done. In this case, numbers representing levels can be associated with each different category, in ascending order. Thus, smaller level numbers are assigned for categories that express lower values (e.g., “low”, “little”, “never”) and larger numbers for categories that express higher values (e.g., “high”, “many”, “always”). Thus, this mapping can be useful when customer desires a specific category (“value”), but higher and/or lower level categories than the desired one can also help/satisfy the customer. For example, the qualitative indicators security and quality of service with values: “low”, “medium” and “high” (SUNDARESWARAN; SQUICCIARIN; LIN, 2012). If the indicator is quality of service and the customer requires value “medium”, the CP value “low” would not be appropriate, but the value “high” would be equally good, or even better. For the PI security level this

is not always true, because a very high degree of security can be harder to work and this may impair the customer's work. Thus, higher and/or lower level categories than the desired category can also satisfy the customer. Therefore, to solve this problem, comprising this work, an ontology, similar to the one in Jain (1991), was created in order to indicate if an ordered qualitative indicator has tolerances for categories below and/or above the level of the desired category (MORAES; FIORESE; MATOS, 2017):

- **Higher is Tolerable (HT):** Categories above of the desired one are tolerable.
- **Lower is Tolerable (LT):** Categories below of the desired one are tolerable.
- **Higher and Lower are Tolerable (HLT):** Categories above and below of the desired one are both tolerable.

Figure 4 shows this classification and an example of the expected behaviour of the utility function variation according to categorical values (always discrete) in the system. The categorical value  $y$  is the customer's desired one and it is the optimal one, with the maximum utility. All other values have a lower utility, inversely proportional to the distance to the category  $y$ . Also, note that if a category is in the tolerable side, but is the farthest from  $y$ , its utility should not be zero.

Figure 4 – Utility variation of a qualitative indicator according to its categorical value



Independently of the type, i.e., quantitative or qualitative, indicators are needed because measuring the performance of a system (enterprise, computational system, service providers, industry, etc.) is not always a simple or trivial task. A well-known performance measurement strategy is carried out by the systematic and structured analysis of the organization's Performance Indicators (PIs) (LOHMANN; FORTUIN; WOUTERS, 2004), reaching a certain value or ranking. The organization that presents the best ranking according to its indicators will theoretically have the highest chance of success (the best performance). The decision-making process should not only consider a single indicator (e.g., lower cost), but a set of PIs (SARI; SEN; KILIC, 2008).



In order to standardize the specification of PIs for the evaluation and comparison of CPs, the Cloud Service Measurement Index Consortium (CSMIC) was formed. The Service Measurement Index (SMI), proposed by CSMIC, represents a widely accepted set of PIs that can be used to measure service performance provided by a CP and also used to compare different services, regardless of whether the service is provided internally or from an outside company (CSMIC, 2014). SMI is a hierarchical structure whose upper level divides the measurement space into seven major categories and each major category is optimized by four or more attributes (subcategories) (CSMIC, 2014). The seven major categories are: accountability, agility, assurance, financial, performance, security, privacy and usability. Each category contains a set of attributes, where each attribute has its own definition and collection form (how to obtain the value) (SIEGEL; PERDUE, 2012). Agility, for example, is composed by the attributes of capacity (memory, processing and disk) and elasticity of services/resources. The detailed explanation of each major category and its attributes can be found in Siegel e Perdue (2012) or in CSMIC (2014).

SMI is an extensive and complete PIs framework. Even though, a bibliographical research was carried out on the most used PIs for the CPS problem (Subsection 2.2.1).

After having access to all the CP's PIs, it is possible to use these PIs to quantify the overall CP's quality/performance using one or more decision-making selection methods. These methods must have the different CP's PIs as input and generate a coherent value as output to the performance associated with that CP (i.e., creating a logical ranking). Thus, Section 2.3 will present some of classical decision-making methods.

These methods are divided into two basic categories: 1) deterministic (AHP and DEA, both from the MCDA area), which generate a ranking (the top ranked CP is elected the best performance one), and 2) metaheuristic (GA, BDE and DDE, both from EAs area). These methods have as input a database with the CPs candidates and their PIs, and the customer request, with the desired values and their respective customer established importance. The proposed methods evaluate the performance of each CP separately (deterministic) or each set of CPs (metaheuristic) and associates them with a value (score or fitness) for each different customer request.

### **2.2.1 Performance Indicators for Cloud Providers**

This subsection aims to expose and clarify some PIs that are used to evaluate different CPs. The PIs can be quantitative or qualitative, as discussed in Subsection 2.2. For CC, this same classification occurs. In this case, PIs usually are amounts of resources, transfer rates or delays, costs/prices over a period of time, percentages

(availability, accuracy, etc.), service time or response time, service names provided, operating system names supported, security, usability, reliability, popularity, etc. The main PIs found in the literature studied are listed in Appendix A, divided into quantitative and qualitative lists. These lists represent an initial PI set which covers the most requested aspects desired by customers and available by CPs. Each PI has a name, an explanation, the expected data type, and the bibliographic references in which they are found. The expected data type refers to the nature of the data or type of information measured/stored for that particular PI. This information can be an integer (discrete quantitative) or real (continuous quantitative), a category (simple qualitative), a set of them (qualitative compound – unordered qualitative PI that can have more than one characteristic simultaneously, e.g., type of service provided) or a boolean (unordered qualitative with only two possible values or categories or subclasses, true or false).

## 2.3 SELECTION METHODS

This section aims to present a description of some useful decision-making auxiliary methods, for customer selection assistance. These methods are indispensable for a careful CP selection based on PIs. Modelling and adaptation of these methods for the CPS problem are specified in Chapter 4.

These selection methods are divided into two main approaches: deterministic (Subsection 2.3.1) and metaheuristic (Subsection 2.3.2). The basic characteristic that differs deterministic methods from probabilistic ones (e.g., metaheuristic methods) is the fact that the same inputs always produce the same outputs, if the parameters are kept the same, in the deterministic methods (also called exact methods). A trivial example of an exact method is the exhaustive search, where all the points of the problem search space are visited and tested by the method. But, exhaustive search can easily become impracticable for problems where their search space grows exponentially.

Both approaches have pros and cons. Deterministic methods only need to be executed once and have more reliable answers, but develop an efficient exact method is not a trivial task and it is not always possible. On the other hand, metaheuristic methods need to be executed several times for more reliable answers, but they are much more time efficient (especially in relation to the exhaustive search) for problems with huge search spaces (complex optimization problems).

### 2.3.1 Deterministic Approaches

The deterministic selection methods are the most frequently found in the studied literature. They are the first attempt to solve the CPS problem. They solve a simplified instance of this problem, where a single provider (best one) is a good enough

answer. Each CP is basically evaluated separately from the other, based on the value of their criteria and the customer desired criteria. Thus, the CPS problem become a MCDA problem (WHAIDUZZAMAN et al., 2014), i.e., where CPs are alternatives to be chosen and CP's information, e.g., metrics, become the MCDA methods criteria (REHMAN; HUSSAIN; HUSSAIN, 2011). At the end, a final ranking with these CPs is built. In this sense, it is important to note that Multicriteria Decision Analysis (MCDA) area aims to develop mathematical and computational tools to select the best alternative among several ones using criteria, and Multicriteria Decision Method (MCDM) is a set of methodologies to compare, classify and select multiple alternatives, each one with multiple attributes (criteria) (ISHIZAKA; NEMERY, 2013). The methods detailed in this subsection are two most classical MCDM: AHP (1980) and DEA (1978).

#### *2.3.1.1 Analytic Hierarchy Process*

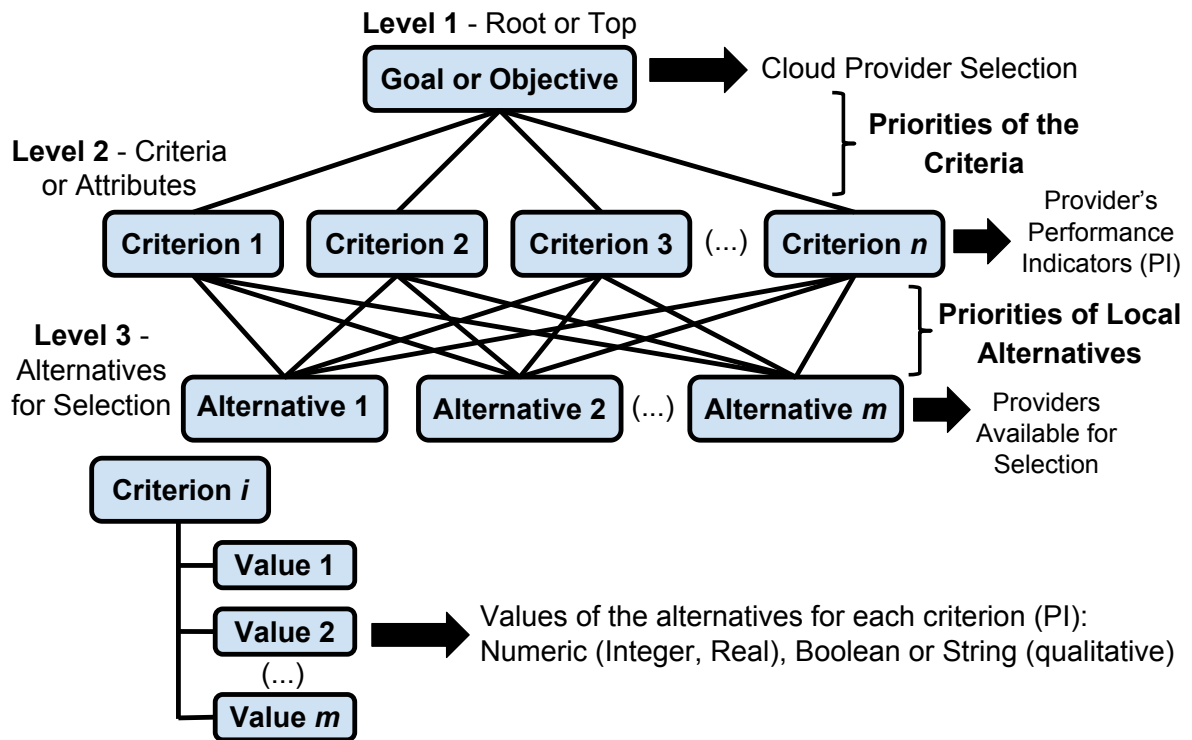
Analytic Hierarchy Process (AHP) is a mathematical multi-criteria decision-making method developed by Thomas L. Saaty from 1977 to 1980 (SAATY, 1990), and based on a list of scales (degrees of importance) to analyze the various decision problems (ISHIZAKA; NEMERY, 2013). The author argues that the method facilitates decision making by perceptions, feelings of organization, judgments and memories in a framework that exhibits the forces that influence the decision (SAATY, 2004).

In this case, in general, problems are arranged in hierarchical levels, from the most general and least controllable level (problem objective or goal) to more specific and controllable levels (attributes or criteria and alternatives). The alternatives are candidate entities to be chosen by the method's user for the problem objective. Each alternative has a key set of associated characteristics directly related to the problem objective in question, the called attributes. Some alternatives reach the goal more optimally than others, because of the values of their attributes. Thus, the purpose of the method is to choose the alternative that best optimize the user's goal. AHP is a "Divide and Conquer" method, decomposing the initial problem into several subproblems, making several comparisons, according to each criterion, in each step, so a ranking of alternatives can be built, in order to select one of the them (ISHIZAKA; NEMERY, 2013).

The first basic steps of AHP are: Definition of the general objective and definition of the structured hierarchy (SARI; SEN; KILIC, 2008). The definition of the hierarchy is made up of criteria (e.g., PIs) and their associated values, connected to each different available alternatives (e.g., CPs). This division allows the visualization of the system as a whole and its components, as well as the interactions of these components and the impacts they exert on the system. The simplest hierarchy have only three basic levels (objective, criteria and alternatives). In more complex and detailed hierarchies,

criteria can be subdivided into subcriteria (adding a new level) and these subcriteria can also be subdivided, and so on (ISHIZAKA; NEMERY, 2013). Figure 5 illustrates a simple (three levels) possible modelling of the AHP hierarchy for selecting CPs based on its PIs.

Figure 5 – Example of the basic AHP hierarchy construction to select CPs using PIs as criteria



Source: Author

The priority or influence weight is a score that classifies the importance of the alternative or criterion in the decision, after the problem structuring phase (hierarchy structuring). There are three types of priorities (ISHIZAKA; NEMERY, 2013):

1. **Priorities of the criteria:** Importance of each criterion to the main objective or goal. Assigned by the method user;
2. **Priorities of local alternatives:** Importance of an alternative value to a specific criteria. Assigned by the method user. It is done for each different criteria; and
3. **Priorities of global alternatives:** Priorities of the criteria and priorities of the local alternatives are intermediate results used to calculate the priorities of the global alternatives. The priorities of the global alternatives classify the alternatives with respect to all the criteria and, consequently, with the main objective. This is calculated automatically by the AHP method.

AHP uses a standard set of integer values (1-9) to describe the importance of the relationship between the hierarchical levels. In other words, these values are used in comparison or judgements matrices (between criteria and objective and between criteria and alternatives), and basically represent the degree of importance that one item has in relation to the other. Table 1 present values, meaning and explanation of these values, also known as Saaty scale (SAATY, 2004).

Table 1 – Standard scale values used to construct Judgement Matrix in AHP

Importance Degree	Meaning	Explanation
1	Both elements are of equal importance	Both elements contribute equally to achieving the goal
3	Moderate importance of one element over the other	Experience and opinion favor one element over the other
5	Strong importance of one element over the other	Experience and opinion strongly favour one element over the other
7	Very strong importance of one element over the other	Practice shows that one element is very strongly favoured over the other
9	Extreme importance of one element over the other	Evidence favours that one element is in the highest possible order of importance over the other
2, 4, 6, 8	Intermediate values between adjacent opinions	Consensus values between opinions, used when it is desired to have greater commitment to the measurements
Reciprocal	If the activity $i$ has one of the values above when compared to activity $j$ , then $j$ has the reciprocal value when compared with $i$ (the inverse of the designated value)	
Rational	Ratio resulting from the scale	If the consistency has to be forced, obtaining numerical values to complete the judgements matrix

Source: Adapted from Saaty (2004)

According to Table 1, the degrees of importance vary from 1 to 9 and represent dominate (number greater than 1) or be dominated (fractional number between 0 and 1, e.g., reciprocals) of one criterion over the other. Thus, the value 9 represents the largest possible discrepancy between the criteria, and 1 represents the lowest, that is, the attributes are equally important between each other.

Thus, in order to execute the AHP method, the first step is to calculate the priority of each criterion to the main objective by constructing a Judgement Matrix of dimension  $n$ , where  $n$  is the number of criteria that influence the objective. In the matrix, there must be the degrees of importance of each criterion to all others. Such degree of importance is defined by the method's user, according to his judgement and also according to Table 1, generally assuming integer values from 1 to 9. Table 2 illustrates this process. Each  $a_{ij}$  represents the degree of importance of the criterion in line  $i$  to the criterion in column  $j$ , for all  $i, j$  belonging to the set of natural numbers. Note that

the main diagonal of the matrix (where  $i = j$ ) must assume the value 1, obviously, since a criterion always has the same importance to itself. It is important to realize that each  $b_{ij}$  in the matrix represents the reciprocal of each  $a_{ji}$ . This is a basic rule of maintaining the logical consistency of the data in Judgement Matrix. The last line of Table 2 contains the sum of each column  $j$  of the matrix, from 1 to  $n$ , represented by  $s_j$ , with  $j$  belonging to the set of natural numbers. This sum is used to normalize the Judgement Matrix, which is the next step.

Table 2 – AHP Judgement Matrix: Modelling of the priorities of each criterion for the main objective

Obj.	Crit. 1	Crit. 2	Crit. 3	Crit. 4	...	Crit. n
<b>Crit. 1</b>	1	$a_{12}$	$a_{13}$	$a_{14}$	...	$a_{1n}$
<b>Crit. 2</b>	$b_{21} = \frac{1}{a_{12}}$	1	$a_{23}$	$a_{24}$	...	$a_{2n}$
<b>Crit. 3</b>	$b_{31} = \frac{1}{a_{13}}$	$b_{32} = \frac{1}{a_{23}}$	1	$a_{34}$	...	$a_{3n}$
<b>Crit. 4</b>	$b_{41} = \frac{1}{a_{14}}$	$b_{42} = \frac{1}{a_{24}}$	$b_{43} = \frac{1}{a_{34}}$	1	...	$a_{4n}$
...	...	...	...	...	...	$a_{in}$
<b>Crit. n</b>	$b_{n1} = \frac{1}{a_{1n}}$	$b_{n2} = \frac{1}{a_{2n}}$	$b_{n3} = \frac{1}{a_{3n}}$	$b_{n4} = \frac{1}{a_{4n}}$	...	1
<b>Sum Col.</b>	$s_1 = \sum a_{i1}$	$s_2 = \sum a_{i2}$	$s_3 = \sum a_{i3}$	$s_4 = \sum a_{i4}$	...	$s_n = \sum a_{in}$

Source: Author

Next, it is necessary to normalize this matrix in order to calculate the final weights of priority for each criterion regarding to the objective. This generates a score among the criteria. To do this, each element of the matrix is divided by the sum of the elements in its respective column. Table 3 illustrates this normalization process and calculates the final score (priority weight), for each criterion  $i$ , which corresponds to the average of all elements given in line  $L_i$ . The score as values, given by each  $C_i$ , are calculated in such a way that  $C_1 + C_2 + C_3 + \dots + C_n = 1$ .

Table 3 – Normalization of the Judgement Matrix and calculation of each criterion score to the objective

Obj.	Crit. 1	Crit. 2	Crit. 3	Crit. 4	...	Crit. n	Score
<b>Crit. 1</b>	$\frac{1}{s_1}$	$\frac{a_{12}}{s_2}$	$\frac{a_{13}}{s_3}$	$\frac{a_{14}}{s_4}$	...	$\frac{a_{1n}}{s_n}$	$C_1 = \text{average}(L_1)$
<b>Crit. 2</b>	$\frac{b_{21}}{s_1}$	$\frac{1}{s_2}$	$\frac{a_{23}}{s_3}$	$\frac{a_{24}}{s_4}$	...	$\frac{a_{2n}}{s_n}$	$C_2 = \text{average}(L_2)$
<b>Crit. 3</b>	$\frac{b_{31}}{s_1}$	$\frac{b_{32}}{s_2}$	$\frac{1}{s_3}$	$\frac{a_{34}}{s_4}$	...	$\frac{a_{3n}}{s_n}$	$C_3 = \text{average}(L_3)$
<b>Crit. 4</b>	$\frac{b_{41}}{s_1}$	$\frac{b_{42}}{s_2}$	$\frac{b_{43}}{s_3}$	$\frac{1}{s_4}$	...	$\frac{a_{4n}}{s_n}$	$C_4 = \text{average}(L_4)$
...	...	...	...	...	...	...	$C_i = \text{average}(L_i)$
<b>Crit. n</b>	$\frac{b_{n1}}{s_1}$	$\frac{b_{n2}}{s_2}$	$\frac{b_{n3}}{s_3}$	$\frac{b_{n4}}{s_4}$	$\frac{b_{nj}}{s_j}$	$\frac{1}{s_n}$	$C_n = \text{average}(L_n)$

Source: Author

The priority values of each local alternative are calculated for their respective criterion, i.e., how important a certain value is to its criterion. This is done for all  $n$  criteria, following the same rules for the Judgement Matrix assembling and consistency rules already mentioned. Table 4 illustrates the Judgement Matrix formed by the priorities of each value (one per alternative) for the respective criterion 1.

Table 4 – AHP Judgement Matrix: Modelling of the priorities of each value for the respective criterion 1

Crit. 1	Val. 1	Val. 2	Val. 3	Val. 4	...	Val. m
Val. 1	1	$a_{12}$	$a_{13}$	$a_{14}$	...	$a_{1m}$
Val. 2	$b_{21} = \frac{1}{a_{12}}$	1	$a_{23}$	$a_{24}$	...	$a_{2m}$
Val. 3	$b_{31} = \frac{1}{a_{13}}$	$b_{32} = \frac{1}{a_{23}}$	1	$a_{34}$	...	$a_{3m}$
Val. 4	$b_{41} = \frac{1}{a_{14}}$	$b_{42} = \frac{1}{a_{24}}$	$b_{43} = \frac{1}{a_{34}}$	1	...	$a_{4m}$
...	...	...	...	...	...	$a_{im}$
Val. m	$b_{m1} = \frac{1}{a_{1m}}$	$b_{m2} = \frac{1}{a_{2m}}$	$b_{m3} = \frac{1}{a_{3m}}$	$b_{m4} = \frac{1}{a_{4m}}$	...	1
Sum Col.	$s_1 = \sum a_{i1}$	$s_2 = \sum a_{i2}$	$s_3 = \sum a_{i3}$	$s_4 = \sum a_{i4}$	...	$s_m = \sum a_{im}$

Source: Author

Table 5 illustrates the normalization of Table 4 and the calculation of the score for each of the  $m$  values (equal the number of alternatives) available for the first criterion, so that:  $V_{11} + V_{12} + V_{13} + \dots + V_{1m} = 1$ .

Table 5 – Normalization of the Judgement Matrix and calculation of each score value to the criterion 1

Crit. 1	Val. 1	Val. 2	Val. 3	Val. 4	...	Val. m	Score
Val. 1	$\frac{1}{s_1}$	$\frac{a_{12}}{s_2}$	$\frac{a_{13}}{s_3}$	$\frac{a_{14}}{s_4}$	...	$\frac{a_{1m}}{s_m}$	$V_{11} = average(L_1)$
Val. 2	$\frac{b_{21}}{s_1}$	$\frac{1}{s_2}$	$\frac{a_{23}}{s_3}$	$\frac{a_{24}}{s_4}$	...	$\frac{a_{2m}}{s_m}$	$V_{12} = average(L_2)$
Val. 3	$\frac{b_{31}}{s_1}$	$\frac{b_{32}}{s_2}$	$\frac{1}{s_3}$	$\frac{a_{34}}{s_4}$	...	$\frac{a_{3m}}{s_m}$	$V_{13} = average(L_3)$
Val. 4	$\frac{b_{41}}{s_1}$	$\frac{b_{42}}{s_2}$	$\frac{b_{43}}{s_3}$	$\frac{1}{s_4}$	...	$\frac{a_{4m}}{s_m}$	$V_{14} = average(L_4)$
...	...	...	...	...	...	...	$V_{1i} = average(L_i)$
Val. m	$\frac{b_{m1}}{s_1}$	$\frac{b_{m2}}{s_2}$	$\frac{b_{m3}}{s_3}$	$\frac{b_{m4}}{s_4}$	$\frac{b_{mj}}{s_j}$	$\frac{1}{s_m}$	$V_{1m} = average(L_m)$

Source: Author

This process must be done for each of the  $n$  criteria. Finishing this step, it is calculated the global score (priorities of global alternatives) of each alternative related with the main objective. The global score for each alternative  $i$ , based on each criterion  $j$ , is calculated according Equation 2.1, if each value associated to the alternative  $i$  are also in position  $i$ , for all criterion  $j$ , with  $i = 1, 2, 3, \dots, m$  and  $j = 1, 2, 3, \dots, n$ .

$$Score(Alternative_i) = \sum_{j=1}^n C_j * V_{ji} \quad (2.1)$$

This global score is a normalized floating point number between 0 and 1. Thus, at the end of the method, each alternative has an associated global score value. The alternative with the largest score will be chosen by the AHP method.

Note that the amount of comparisons required for each generic Judgement Matrix is given by:  $\frac{(n^2-n)}{2}$ , where  $n$  represents the number of criteria belonging to this matrix (ISHIZAKA; NEMERY, 2013). The magnitude  $n^2$  represents the amount of comparisons that can be written in a square matrix of dimension  $n$ , and  $n$  of these comparisons represent the comparison with the alternative of  $s_i$  (matrix main diagonal), whose value is always 1, and should be unaccounted. As the matrix is reciprocal, only half of

the comparisons are required, the other half are automatically obtained from the first half (ISHIZAKA; NEMERY, 2013). The mechanisms required for complete consistency verification of a Judgement Matrix, as well as an example of Judgement Matrix solving with generic numbers are in Appendix B.

### *2.3.1.2 Data Envelopment Analysis*

Data Envelopment Analysis (DEA) is a well-known method for decision-making support, introduced first by Charnes et al. in 1978 (CHARNES; COOPER; RHODES, 1978). DEA has innumerable practical applications for measuring performance of a set of similar units called Decision-Making Units (DMUs), which are entities that can be manufacturing units, departments of big organizations such as banks, hospitals, schools, universities, police stations, prisons, etc. (RAMANATHAN, 2003). Each DMU consumes a variety of identical inputs and produces a variety of identical outputs. Summing up, DEA is a mathematical programming technique belonging to the MCDA area (ISHIZAKA; NEMERY, 2013), where basically, each solution alternative is called DMU and each criterion used to support the decision is an input or an output to the method itself.

Thus, performance of each DMU is assessed by DEA using the concept of efficiency or productivity, which is the ratio of total outputs regarding total inputs (RAMANATHAN, 2003). So, outputs are criteria to be maximized and inputs are criteria to be minimized for better efficiency. Thus, this technique aims to measure how efficiently a DMU uses the available resources to generate a set of outputs. Efficiency is a relative concept that compares what has been produced with what could have been produced. Ultimately, it can be understood as a comparison between observed productivities among competing DMUs (ISHIZAKA; NEMERY, 2013). Thus, efficiencies estimated using DEA are relative too comprising or compared to the best performing DMU, i.e., with the highest calculated efficiency value among the DMUs (RAMANATHAN, 2003). The efficiency vary between 0 and 100% (or from 0 to 1).

DEA modelling transforms this efficiency definition into a set of equations of Linear Programming (LP) to be solved for a LP algorithmic (e.g., Simplex). For such, multiple inputs and outputs are linearly aggregated using weights. This comes up with the concept of virtual inputs and virtual outputs. Virtual inputs of a DMU are obtained as the linear weighted sum of all its inputs. Similarly, virtual outputs are obtained as the linear weighted sum of all DMU outputs. Thus, the efficiency is defined as the ratio of the sum of its weighted outputs to the sum of its weighted inputs (RAMANATHAN, 2003), as shown in Equation 2.2, which is subject to the constraint modelled in Equa-



tion 2.3.

$$Eff_{DMU} = \frac{VirtualOutputs_{DMU}}{VirtualInputs_{DMU}} = \frac{\sum_{k=1}^s u_k * O_{k(DMU)}}{\sum_{l=1}^r v_l * I_{l(DMU)}} \quad (2.2)$$

$$0 \leq \frac{\sum_{k=1}^s u_k * O_{ki}}{\sum_{l=1}^r v_l * I_{li}} \leq 1, \forall i \quad (2.3)$$

It is important to note that  $u_k, v_l \geq 0, \forall k, l \in \mathbb{N}; i = DMU_1, DMU_2, \dots, DMU_n; k = 1, 2, \dots, s$  and  $l = 1, 2, \dots, r$ ; where  $s$  is the total amount of outputs and  $r$  is the total amount of inputs;  $u_k$  is the weight associated with each output and  $v_l$  the weight associated with each input. Thus, the DMU that produces more outputs with less inputs than the others will be evaluated with better efficiency values. The weights assigned to outputs ( $u_k$ ) and inputs ( $v_l$ ) are not allocated by users, but automatically calculated by the method in a linear optimization procedure in order to allow the most beneficial results possible for each DMU (RAMANATHAN, 2003), respecting the restriction of Equation 2.3. The set of DMUs with efficiency of 100% form DEA's efficiency frontier. Thus, it is important to note that the number of DMUs must be a reasonable quantity, otherwise, maybe the method can not satisfactorily calculate the efficiency and then all DMUs fall on the efficiency frontier, i.e., they are classified as 100% efficient by the method. Thus, it is expected that the number of DMUs is greater than the product between the number of inputs and outputs (RAMANATHAN, 2003). It is worth to note that the sample size should be at least two times larger than the sum of the number of inputs and outputs, for an appropriate efficiency calculation (RAMANATHAN, 2003).

Equation 2.2 restricted to Equation 2.3 is a fractional program (fractions), whose solution is more difficult than the solution by LP (RAMANATHAN, 2003). Thus, the model is converted to LP by normalizing the denominator (input orientation) or the numerator (output orientation) in the objective function. The normalization in this case becomes a constraint where the denominator or numerator is equal to 1 (constant) and the difference between outputs and inputs is less than or equal to 0. That is why DEA has two orientations (RAMANATHAN, 2003; ISHIZAKA; NEMERY, 2013):

- **Input orientation:** In this DEA orientation, the efficiency is the ratio of output to input. This orientation aims to model efficiency by decreasing inputs, i.e., it tries to model how to produce the same with less resources. In the model of the multipliers (the objective function uses multiplication of the weights by the output values), the maximization of the output is modelled, while in the dual model of the multiplier model, also known as the envelope model, it is tried to minimize the input.

- **Output orientation:** In this orientation, the efficiency is the ratio of input to output. This orientation aims to model efficiency by increasing outputs, i.e., it tries to produce more with the same resources. In the model of the multipliers, it is tried to minimize the input, while in the dual model, the envelope model, it is tried to maximize a multiplication factor for the outputs.

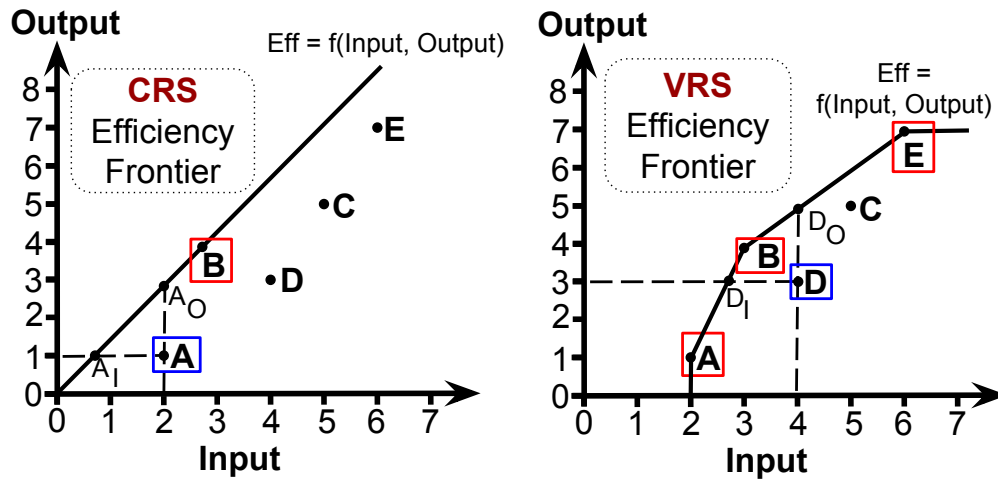
DEA also have two main models (RAMANATHAN, 2003; ISHIZAKA; NEMERY, 2013) that consider or not proportional scale returns. Scale returns are related to the proportion of the output produced relative to the input ratio consumed. These models also lead to the identification of two different efficiency frontiers. These models are:

- **Constant Returns to Scale (CRS):** Also known as Charnes, Cooper and Rhodes (CCR), it was developed in 1978. Assumes constant returns to scale, i.e., any input variation produces a proportional output variation. This is appropriate when all DMUs are operating at an optimal scale. However, note that this is quite an ambitious assumption. To operate at an optimal scale, DMUs should evolve in a perfectly competitive environment, which is seldom the case.
- **Variable Returns to Scale (VRS):** Also known as Banker, Charnes and Cooper (BCC), it was developed in 1984. It assumes variable returns to scale, i.e., variable scale and no proportionality among inputs and outputs. This is appropriate when DMUs are not operating at an optimal scale. This is usually the case when DMUs face imperfect competition, government regulations, etc.

In each model (CRS and VRS) DEA's user can choose between input or output orientations. The efficiency frontier form is different for each model. However, within each model, the efficiency frontier will not be affected by input or output orientation. DMUs located on the frontier when input orientation is used, also will be on the frontier when output orientation is used. The model's orientation should be chosen according to which variables (inputs or outputs) the decision maker has most control over (RAMANATHAN, 2003). Figure 6 presents the geometric interpretation of the efficiency frontier for a single input and output for CRS (left) and VRS (right) models with 5 hypothetical DMUs: "A", "B", "C", "D" and "E".

On the left graphic (CRS) the variation between inputs and outputs is always constant and proportional (linear), i.e., the efficiency function has a single slope (constant derivative). On the left graphic, only DMU "B" is efficient (fall in the efficiency frontier). The DMU "A" could be efficient if it can increase its outputs ( $A_0$ ), keeping the input of 2 (output orientation) or decreases its inputs ( $A_I$ ), keeping the output of 1 (inputs orientation). On the right graphic (VRS) the variation between inputs and outputs is not constant, i.e., the efficiency function has different slopes (different derivative, one

Figure 6 – Geometric interpretation of the efficiency frontier for CRS and VRS models



Source: Adapted from Ishizaka e Nemery (2013)

for each function slope). On the right graphic DMUs “A”, “B” and “E” are efficient. The DMU “D” could be efficient if it increases its outputs ( $D_0$ ), keeping the input of 4 (output orientation) or decreases its inputs ( $D_I$ ), keeping the output of 3 (inputs orientation).

The identification process of which inputs and outputs DEA have to use for calculating efficiency is an essential step for the correct use of this MCDA method, as well as choosing an appropriate model and orientation.

### 2.3.2 Metaheuristic Approaches

Metaheuristics methods are robust techniques to find satisfactory answers in an acceptable time for complex optimization problems. These techniques are probabilistic and use different background inspirations, like nature (physics, chemistry and biology). The most classical metaheuristic approaches are energy functions, e.g., Simulated Annealing (SA), and bio-inspired ones, e.g., Evolutionary Algorithms (EAs).

SA is inspired in the solids annealing technique and it use concepts of physics and thermodynamics like atomic structural energy, temperature, cooling and thermal equilibrium (TALBI, 2009). Each possible solid states correspond to possible solutions of the problem and the energy in each state corresponds to the value of the problem objective function. The minimum energy corresponds to the problem optimal solution. SA is the Metropolis algorithm used in a sequence of decreasing temperatures, where temperature is a control parameter that determines the probability of accepting non-improving solutions (escape from local optimum).

EAs use concepts like population of individuals (i.e., a set of candidate solutions), fitness (i.e., individual's ability to survive and reproduce), variability and heredity (i.e., new individuals possess many of the characteristics of their parents) and the use

of genetic operations (i.e., crossover and mutation, which directly influence genetic variability) and selection routines that guide the convergence process to the best solution (JONG, 2006). In this case, each possible solution to an optimization problem is modelled as an individual, which represents a point in the problem solution search space. Thus, each individual has an associated fitness (phenotype) indicating how well the solution it represents, encoded by its genotype, adequately solves the problem in question.

This section will present the fundamental theoretical base of operation of the SA and the three EAs - GA, BDE and DDE - for binary search spaces. The fitness and energy function modelling for these methods coping with the CPS problem are presented and discussed in Chapter 4.

### *2.3.2.1 Simulated Annealing*

Simulated Annealing (SA) is a very popular stochastic and physic-inspired algorithm proposed by Kirkpatrick, Gelatt e Vecchi (1983). This algorithm is classified as an iterative single-solution improvement based in a hill climbing process searching solutions that is commenced by a random move (TALBI, 2009).

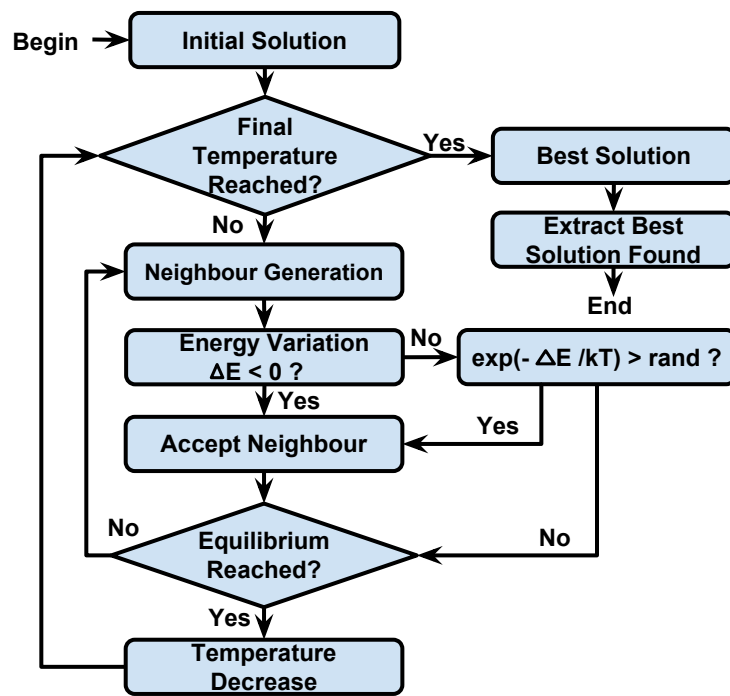
SA is motivated by an analogy to the statistical mechanics of solids annealing (e.g., metals and their alloys) (RUTENBAR, 1989). The annealing process consists of melting a material (allowing atomic rearrangements) and then cooling it very slowly until it solidify to obtain a strong crystalline structure (TALBI, 2009). The structure strength depends on the cooling rate. If the initial temperature is not sufficiently high or the cooling is not done slowly enough then the material will form a glass (crystal structures), where the structure has many imperfections (disordered atomic structure, meta stable state) (RUTENBAR, 1989). An ordered atomic structure (strong crystal) means that the material stored total energy lowers and it becomes more stable (low energy state) (RUTENBAR, 1989; TALBI, 2009).

Thus, the SA algorithm simulates the energy changes in a system subjected to a cooling process until it converges to an equilibrium state (parameter). Basically, the SA can be understood as the Metropolis algorithm (METROPOLIS et al., 1953) used in a sequence of decreasing temperatures (control parameter) to generate very optimized solutions (if the cooling is done correctly) of an optimization problem (KIRKPATRICK; GELATT; VECCHI, 1983). So, the control parameter “temperature” determines the probability of accepting non-improving solutions whose objective is to escape from local optima and so to delay the convergence (TALBI, 2009). Each possible states of a metal correspond to candidate solutions of the search space. The energy in each state corresponds to the value of the problem objective function. The decision variables associated with a problem solution are analogous to the metal molecular po-

sitions. The minimum energy corresponds to the value of an optimal problem solution. Finding a local minimum implies that a meta stable state has been reached. It is important to note that SA is a memoryless algorithm in the sense that the algorithm does not use any information gathered during the search.

Figure 7 presents an SA basic operation steps flowchart. It initializes with the random generation of the initial candidate solution with a high arbitrary temperature. From this initial candidate solution, SA proceeds with several iterations. The first loop check if the thermal equilibrium has been reached (final temperature usually close to zero). The second loop check if the atomic equilibrium state has been reached for that temperature, basically becoming the Metropolis algorithm. So, this parameter can be called Metropolis' constant, i.e., how many iterations are done for that temperature value.

Figure 7 – SA flowchart



Source: Author

The neighbour generation routine applies a small random perturbation in the current solution ( $s$ ) codification, modifying a single position in its encoding vector. This generated neighbour ( $s'$ ) is a new solution to be analysed. Thus, this new solution energy,  $f(s')$ , is calculated, as well as the energy difference,  $\Delta E$ , between it and the current solution ( $\Delta E = f(s') - f(s)$ ). So, if both energy values are positive ( $f(s'), f(s) > 0$ ) and  $\Delta E < 0$ , then  $f(s') < f(s)$ , so the new candidate solution has lower energy than the current solution. This means that this new candidate solution is better and it is immediately accepted as the new current solution. If  $f(s') \geq f(s)$  ( $\Delta E \geq 0$ ), then

the current neighbour is accepted with a given probability that follows the Boltzmann distribution (TALBI, 2009). Equation 2.4 present this neighbour acceptance probability formula, given between 0 and 1 (100% acceptance rate).

$$prob = \begin{cases} 1, & \text{if } f(s') < f(s) \\ \exp\left(\frac{-\Delta E}{k*T}\right), & \text{otherwise} \end{cases} \quad (2.4)$$

Where  $T$  is the current temperature and  $k$  is the Boltzmann's constant (KIRKPATRICK; GELATT; VECCHI, 1983). The Boltzmann distribution is a thermodynamic concept that describes the energy distribution in a system particle held at constant temperature. The default used value for the Boltzmann's constant is  $k = 1$  (RUTENBAR, 1989; TALBI, 2009). In practice, this probabilistic acceptance is achieved by generating a uniform random number  $x \in [0, 1]$  and comparing it with  $prob$ . So, only when  $x < prob$ , the neighbour is accepted. Also, note that, if  $\Delta E$  is kept constant. Higher temperature leads to a greater probability of a new solution  $s'$  to be accepted. So, at the SA execution start, the algorithm has a more random problem search space exploration behaviour. On the other hand, close to its end, the search will be more localized. This iterative process is performed until a predefined number of iterations is reached (equilibrium state, Metropolis' constant). Since this equilibrium state is reached, the temperature is gradually decreased according to a cooling schedule such that few non-improving solutions are accepted at the end of the search. Thus, the best solution found in the whole process ( $s^*$ ) is extracted and it is the SA answer.

Algorithm 1 presents the main SA parameters and its operation. Control parameters given by user are: Starting Temperature ( $T_0$ ), Final Temperature ( $T_f$ ), Temperature Decay function (TD) and Metropolis Constant (MC). So, the SA will stop after  $MC * NTD$  iterations, where  $NTD$  is the total number of temperature decays.

### 2.3.2.2 Genetic Algorithmic

Genetic Algorithms (GAs) are introduced by John Holland in the 1970s (HOLLAND, 1975) and popularized by David Goldberg in the 1980s (GOLDBERG, 1989). GA is the most popular bio-inspired search technique based on the theory of evolution of the species of Charles Darwin. It is based on the process of natural selection, where adaptability (fitness), heredity, and genetic mutation are able to influence the changes and selection of genetic material that will be passed on to future generations of a population of individuals of the same specie. GA has been used for real optimization problems of high complexity. Holland (1975) studied the natural evolution of the species and considered this a robust and powerful process that could be easily adapted to a computational algorithm capable of obtaining satisfactory solutions to optimization problems

**Algorithm 1** Simulated Annealing (SA) pseudocode

---

```

1: function SA( $T_f$ ,  $T_0$ ,  $TD$ ,  $MC$ )
2:    $s = s_0$  ▷ Initial solution
3:    $T = T_0$ 
4:   while  $T > T_f$  do ▷ Iterations until thermal equilibrium
5:     for  $ItT = 0$  to  $MC$  do ▷ Iterations for that temperature
6:        $s' = Neighbour(s)$  ▷ Generate new random neighbour
7:        $\Delta E = f(s') - f(s)$ 
8:       if  $\Delta E < 0$  then
9:          $s = s'$  ▷ Accept neighbour
10:        if  $f(s') < f(s^*)$  then ▷ Update the best solution
11:           $s^* = s'$ 
12:        end if
13:      else
14:         $p = exp(-\Delta E/T)$  ▷ With  $k = 1$ 
15:         $x = Random(0, 1)$ 
16:        if  $x < p$  then
17:           $s = s'$  ▷ Accept neighbour
18:        end if
19:      end if
20:    end for
21:     $T = TD(T)$  ▷ Temperature decrement
22:  end while
23:  Return  $s^*$ 
24: end function

```

---

Source: Adapted from Talbi (2009)

---

in an acceptable period of time.

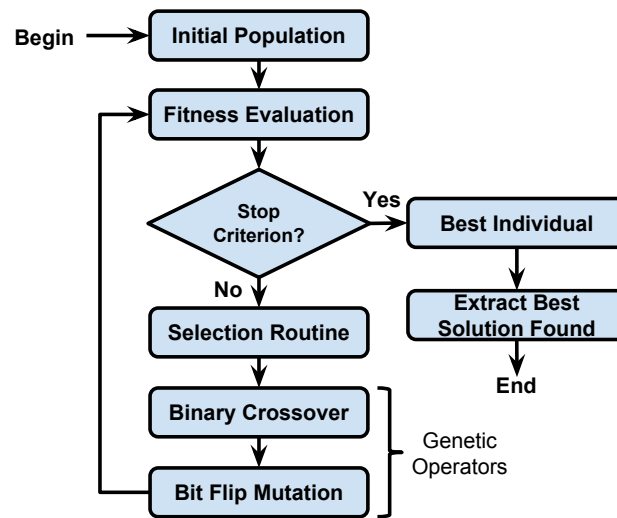
GAs are probabilistic and populational (GOLDBERG, 1989), i.e., work on a set of candidate solutions, named individuals, of a population in each generation (algorithm iteration). Population algorithms explore the search space in parallel with the individuals that make up the population. Each individual represents a point in the search space and it is composed by a genetic coding (genotype), implemented as a vector with different genes. Each gene is responsible for encoding one of the optimization problem variables. Each variable directly influences the quality of the problem solution and has a domain and a coding (e.g., binary, real, integer, combinatorial, etc.). All variables must have the same encoding. The GA looks for the best solution to a problem based on the adaptive value (fitness) of its individuals. The greater the fitness, the greater the chances of that individual passing on its genetic load (portions of its chromosomes/solutions) to future generations. The highest fitness individual in the population represents the solution to the problem in that current generation.

In a simple GA, candidate solutions are encoded in binary vectors of fixed size and emphasize crossover as the main genetic operator applying mutation with low probabilities (below 5%). Moreover, the selection routine is probabilistic and fitness proportional for each individual (GOLDBERG, 1989).

Figure 8 presents a flowchart of the simple GA basic operation steps. It initializes with the random generation of the initial population, followed by its fitness evaluation using the problem specific fitness function. At the end of each generation (after the

selection routine and genetic operators application) the stopping criteria is checked.

Figure 8 – Simple GA flowchart



Source: Author

The stop criterion defines when the algorithm stops (generation loop) and the best individual of the current population is selected and its genotype is decoded. Thus, the final solution is extracted and it is the GA answer at that execution. The stop criteria can be varied: a predefined number of generations, an arbitrary total fitness evaluation, maximum execution time, an minimum error (when the optimal answer is known), etc.

The selection routine is responsible to guide the evolution process over the generations, allowing the algorithm to converge to the best possible solutions. It is usually applied after the fitness evaluation of the new population (previous generation's offspring). Selection routines prioritizes the best adapted individuals (highest fitness) of the current population to generate offspring (through genetic operators) for a new population. Selection routines can be probabilistic (e.g., roulette wheel and stochastic tournament) or deterministic (e.g., greedy selection).

GA generally uses two genetic operators: crossover (for binary coding, e.g., one-point-crossover) and bit flip mutation. One-cut-point crossover, creates two new coding vectors (offsprings) by copying and exchanging two portions of the coding vectors of two individuals (parents) based on a single cut-off point, chosen randomly between positions 1 and "n", where "n" is the final position of the parents' coding vector. Bit flip is a mutation technique that simple changes a bit (with a specified ratio), e.g., from 0 to 1 , representing one variable in the vector coding an individual. An elitism routine can also be applied to help algorithm convergence to the optimal solution point and make an always upward fitness curve (best individual fitness in the current population). The elitism copy the best individual of the current population to next algorithm generation (ANDRÉ; PARPINELLI, 2015).



Algorithm 2 presents the main GA parameters and more details about its operation. Control parameters are: number of Dimensions (DIM), Population Size (POP), maximum number of Iterations/Generations (IT), Crossover Rate (CR), Mutation Rate (MR) and a flag indicating the occurrence of Elitism (ELI). A predefined number of generations is used as stopping criteria.

---

**Algorithm 2** Genetic Algorithm (GA) pseudocode
 

---

```

1: function GA(DIM, POP, IT, CR, MR, ELI)
2:   generation = 0
3:    $P = \text{RandomPop}(DIM, POP)$ , where  $P[i] = x_i$  and  $x_i \in \{0, 1\}^{DIM}$ 
4:   EvalFitness( $P$ )
5:   while generation < IT do
6:     for  $i = 1$  to POP step 2 do
7:        $x_i = \text{Select}(P)$  ▷ Selection Routine
8:        $x_{i+1} = \text{Select}(P)$ , with  $x_i \neq x_{i+1}$ 
9:       if  $\text{Random}(0, 100) < CR$  then ▷ Crossover
10:         $x_i, x_{i+1} = \text{Crossover}(x_i, x_{i+1})$ 
11:      end if
12:      for  $j = 1$  to DIM do ▷ Bit-Flip Mutation
13:        if  $\text{Random}(0, 100) < MR$  then
14:           $x'_i[j] = \text{not } x_i[j]$ 
15:        end if
16:        if  $\text{Random}(0, 100) < MR$  then
17:           $x'_{i+1}[j] = \text{not } x_{i+1}[j]$ 
18:        end if
19:      end for
20:       $\text{Add}(P, x'_i, x'_{i+1})$ 
21:    end for
22:    EvalFitness( $P$ )
23:     $x^* = \text{Best}(P)$ 
24:    if  $ELI == \text{True}$  then ▷ Elitism
25:       $\text{Add}(P, x^*)$ 
26:    end if
27:    generation = generation + 1
28:  end while
29:   $x^* = \text{Best}(P)$ 
30:  Return  $x^*$ 
31: end function

```

Source: Adapted from André e Parpinelli (2015)

---

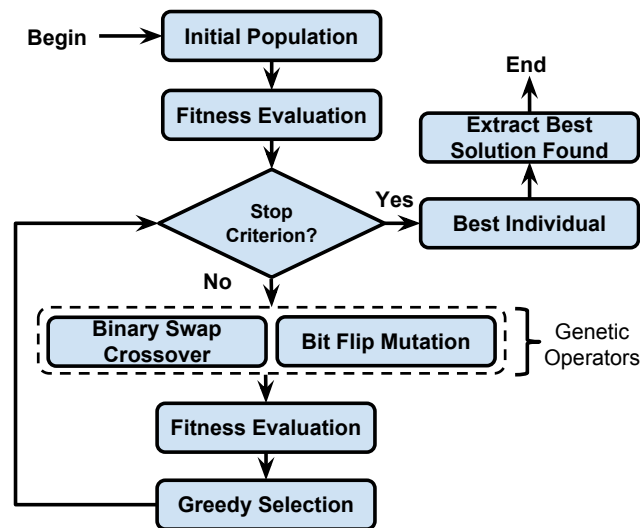
### 2.3.2.3 Binary Differential Evolution

Binary Differential Evolution (BDE) is a probabilistic and populational meta-heuristic algorithm inspired by the canonical Differential Evolution (DE) (STORN; PRICE, 1995). BDE is adapted to handle binary problem search spaces (KRAUSE; PARPINELLI; LOPES, 2012). Basically, it is a modification of the *DE/rand/1/bin* (ANDRÉ; PARPINELLI, 2015) variant, which combines each candidate solution of the current population with a randomly chosen solution through the crossover operator. However, the main modification, besides the binary representation, is the insertion of a bit-flip mutation operator, inspired on the GA. This modification adds to the algorithm the capacity to improve its global search ability, enabling diversity (ANDRÉ; PARPINELLI, 2015). Thus, the BDE consists in applying these simple operators (crossover and bit-flip mutation) in candidate solutions represented as binary strings. The adaptation starts on the initialization

of the population. In this case, instead of random continuous values (DE) the individuals are initiated with random binary ones. The original mutation is replaced by a random bit inversion and the perturbation rate is a new parameter, inserted to establish how many individuals of the population will pass through the mutation and crossover processes. At least one individual will be mutated and the DE crossover is kept as the original but with binary values. This process is activated during the perturbation procedure and after the mutation. In this case, the used selection routine is the greedy selection.

Figure 9 presents a flowchart of the BDE basic operation steps. It initializes with the random generation of the initial population, followed by its fitness evaluation using the fitness function. At the end of each generation (after the genetic operators and greedy selection application) the stopping criteria is checked.

Figure 9 – BDE flowchart



Source: Author

The stop criterion defines when the algorithm stops executing and the best individual of the current population is selected and its genotype is decoded, leading to the final solution extraction. BDE uses two genetic operators: crossover and bit flip mutation, like GA (ANDRÉ; PARPINELLI, 2015). In this case, crossover is a simple modification of the *DE/rand/1/bin* variant for binary coding and it uses only one additional random individual. Selection routine used in BDE is a simple greedy selection (KRAUSE; PARPINELLI; LOPES, 2012). Elitism routine is not necessary because of the greedy selection.

Algorithm 3 presents the main BDE parameters and more details about its operation. Control parameters are: number of Dimensions (DIM), Population Size (POP), maximum number of Iterations/Generations (IT), Perturbation Rate (PR) and Mutation Rate (MR). A predefined number of generations (IT) is used as stopping criteria.

---

**Algorithm 3** Binary Differential Evolution (BDE) pseudocode
 

---

```

1: function BDE(DIM, POP, IT, PR, MR)
2:   generation = 0
3:    $P = \text{RandomPop}(DIM, POP)$ , where  $P[i] = x_i$  and  $x_i \in \{0, 1\}^{DIM}$ 
4:    $\text{EvalFitness}(P)$ 
5:   while generation < IT do
6:     for  $i = 1$  to POP do
7:        $k = \text{Random}(1, POP)$ , with  $k \neq i$ 
8:        $z = P[k]$ 
9:        $j_{rand} = \text{Random}(1, DIM)$ 
10:       $y = x_i$  ▷ New individual
11:      for  $j = 1$  to DIM do
12:        if  $j == j_{rand}$  Or  $\text{Random}(0, 100) < PR$  then ▷ Perturbation
13:          if  $\text{Random}(0, 100) < MR$  then
14:             $y_j = \text{not } y_j$  ▷ Bit-Flip Mutation
15:          else
16:             $y_j = z_j$  ▷ Crossover
17:          end if
18:        end if
19:      end for
20:      if  $\text{EvalFitness}(y) > \text{EvalFitness}(x)$  then ▷ Greedy Selection
21:         $P[i] = y$ 
22:      end if
23:    end for
24:    generation = generation + 1
25:  end while
26:   $x^* = \text{Best}(P)$ 
27:  Return  $x^*$ 
28: end function

```

Source: Adapted from André e Parpinelli (2015)

---

#### 2.3.2.4 Discretized Differential Evolution

Discretized Differential Evolution (DDE) is a novel version of DE (STORN; PRICE, 1995), i.e., it is a probabilistic and populational EA adapted to handle binary problem search spaces (KRAUSE; LOPES, 2013). The main difference from BDE (Subsection 2.3.2.3) is that DDE encodes continuous variables (floating point numbers) and discretizes to binary only its solution vector (genotype) before the fitness evaluation, i.e., the genetic operators are used on real codification. Thus, the individual to be evolved by DDE is an  $n$  dimensional vector and each dimension is populated with a random floating point number between -1 and 1 (KRAUSE; LOPES, 2013). The basic DE operator is the *DE/rand/1/bin* strategy of mutation through the generations. Equation 2.5 present this strategy.

$$x_j = a_j + F * (b_j - c_j) \quad (2.5)$$

This mutation creates a new individual “x” using three different random individuals (“a”, “b”, and “c”) and a scale parameter  $F$  that ranges from 0 to 1 (KRAUSE; LOPES, 2013). The individual “x” genotype is perturbed in position  $j$  by receiving the random floating point value of “a”, plus the weighted difference variation between “b” and “c”, at that same position  $j$ . The parameter  $F$  is the weighting factor used to control the amplification of the differential variation. An alternative elitist strategy called

*DE/best/1/bin* uses the best individual of the current generation for “a”, instead of a random individual (KRAUSE; LOPES, 2013).

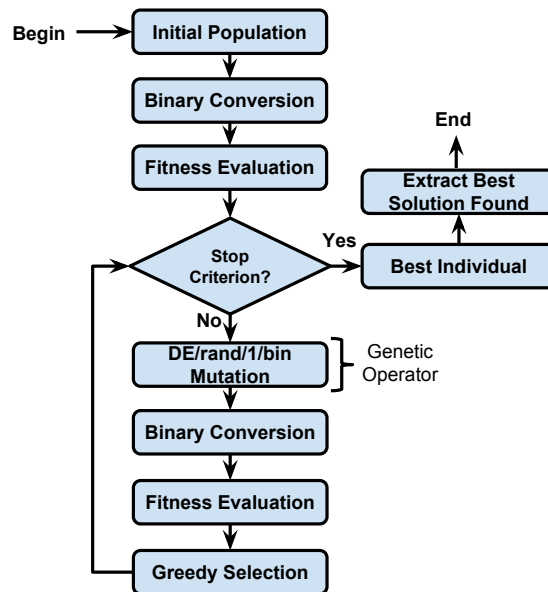
When the fitness must to be evaluated for an individual, DDE has to discretize (real codification,  $x_j$ , to binary codification,  $x'_j$ ) the genotype vector for that individual. This discretization is performed using the sigmoid function that allows each dimension to evolve gradually and individually. Equation 2.6 present the sigmoid function.

$$x'_j = \begin{cases} 1, & \text{if } \frac{2}{1+\exp(-2*x_j)} - 1 > 0 \\ 0, & \text{otherwise} \end{cases} \quad (2.6)$$

With the use of this sigmoid function, the evolved dimensions (coded variables) do not abruptly jump from 0 to 1, or vice-versa, in binary spaces. They evolve gradually around zero using weighted values to find the best continuous combination (KRAUSE; LOPES, 2013).

Figure 10 presents a flowchart of the DDE basic operation steps. It initializes with the random generation of the initial population (individuals with real codification between -1 and 1). Then, a binary conversion, using sigmoid function (Equation 2.6), is applied to each real coded variable in each individual on current population. This conversion is done using auxiliary data structures, like an additional population, keeping the original population intact. Therefore, fitness function is executed on binary values. At the end of each generation (after the genetic operators and greedy selection) the stopping criteria is checked.

Figure 10 – DDE flowchart



Source: Author

The stop criterion defines when the algorithm stops its execution. When this happens, the best individual of the current population is selected and its genotype is decoded, leading the final solution extraction. For DDE, a single genetic operator is enough to generate the necessary genetic variability: the  $DE/rand/1/bin$  mutation (Equation 2.5) (KRAUSE; LOPES, 2013). The selection routine used is the greedy selection (KRAUSE; LOPES, 2013). The elitism routine is not necessary because of the greedy selection.

Algorithm 4 presents the main DDE parameters and more details about its operation. Control parameters are: number of Dimensions (DIM), Population Size (POP), maximum number of Iterations/Generations (IT), Perturbation Rate (PR) and mutation weighting Factor (F). A predefined number of generations is used as stopping criteria.

---

**Algorithm 4** Discretized Differential Evolution (DDE) pseudocode

---

```

1: function DDE(DIM, POP, IT, PR, F)
2:   generation = 0
3:    $P_{real} = \text{RandomPop}(\text{DIM}, \text{POP})$ , where  $P_{real}[i] \in \{-1.0, 1.0\}^{\text{DIM}}$ 
4:    $P_{bin} = \text{Discretize}(P_{real}, \text{sigmoid}(x))$ 
5:   EvalFitness( $P_{bin}$ )
6:   while generation < IT do
7:     for i = 1 to POP do
8:       a = Random(1, POP)
9:       xa =  $P_{real}[a]$ 
10:      b = Random(1, POP), with  $b \neq a$ 
11:      xb =  $P_{real}[b]$ 
12:      c = Random(1, POP), with  $c \neq b$  &  $c \neq a$ 
13:      xc =  $P_{real}[c]$ 
14:      jrand = Random(1, DIM)
15:      yreal =  $P_{real}[i]$  ▷ New individual
16:      for j = 1 to DIM do
17:        if  $j == jrand$  Or  $\text{Random}(0, 100) < PR$  then
18:           $y_{real}[j] = xa_j + F * (xb_j - xc_j)$  ▷ Mutation
19:        end if
20:      end for
21:       $y_{bin} = \text{Discretize}(y_{real}, \text{sigmoid}(x))$ 
22:       $x_{bin} = \text{Discretize}(P_{real}[i], \text{sigmoid}(x))$ 
23:      if EvalFitness( $y_{bin}$ ) > EvalFitness( $x_{bin}$ ) then ▷ Greedy Selection
24:         $P_{real}[i] = y_{real}$ 
25:      end if
26:    end for
27:    generation = generation + 1
28:  end while
29:   $x^* = \text{Best}(P_{bin})$ 
30:  Return  $x^*$ 
31: end function

```

---

Source: Adapted from Krause e Lopes (2013)

---

## 2.4 PARTIAL CONSIDERATIONS

This chapter presented the main fundamental concepts related to this work. These concepts are necessary for the proper development and understanding of the proposed PI-based architecture and selection methods modelling for the CPS problem (main work objective). The main key concepts are about: Cloud Computing (CC), Performance Indicators (PIs) and selection methods.

CC paradigm is a convenient and efficient technique of hosting and distribution of computational resources and services to its consumers around the world. NIST presents CC as possessing five essential characteristics, three service model (SPI), four deployment models and the reference architecture with its five main actors: consumer, provider, auditor, broker and carrier.

PIs are metrics that allow a synthesized collection of information related to a certain organization. PIs quantify the objects to be measured. A good metric has the following characteristics: linearity, reliability, repeatability, ease of measurement, consistency and independence. It is possible to classify PIs in quantitative (discrete or continuous) or qualitative (ordered or unordered). According to the behaviour of their utility function, quantitative PIs are: HB, LB, NB. For ordered qualitative PIs it was created an auxiliary classification: HT, LT, HLT. Thus, to measure an organization performance, someone can systematically measure the quality of each of its PIs, reaching a certain value or ranking. The organization that presents the best ranking will theoretically present the highest performance. CC already has a widely accepted PI framework called SMI, developed by CSMIC. SMI represents a PI set that provides a standardized method for measuring and comparing Cloud Providers (CPs).

Selection methods are indispensable for a careful PI-based CP selection. Descriptions in this chapter are more general about the method itself, comprising their characteristics and their operation mode. These methods are divided into two basic categories: deterministic (e.g., AHP and DEA) and metaheuristics (e.g., SA, GA, BDE and DDE). The basic characteristic that differs deterministic methods from metaheuristic ones is that the same parameters and inputs always produce the same outputs. Although deterministic methods have more reliable answers, probabilistic ones are much more time efficient for complex problems. Therefore, the choice of which kind of method to be used to solve the CPS problem depends on several factors ranging from the user needs, quantity of available PIs and CPs, complexity of the need (user request), prior knowledge from the best answer, and so on.

### 3 RELATED WORK

This chapter presents relevant works, developed in the last decade, related to the problem of selecting and ranking CC services or CPs. Thus, this chapter intends to offer an overview of how this problem is being addressed and the proposals to solve it.

The selection process is done using several collected data from each CPs such as attributes, criteria, PIs, etc. Many works use as base criteria the SMI PIs (CSMIC, 2014). Also, many works use MCDA methods to select and rank CPs (WHAIDUZZA-MAN et al., 2014). This fact is so expressive that this chapter divides these works into two sections. Section 3.1 present works that does not utilize any MCDA methods and Section 3.2 present the works that utilize one or more MCDM, including hybrid versions. Finally, Section 3.3 present a comparative table between the works cited and the proposed architecture, summarizing and classifying them according to some important characteristics.

#### 3.1 CLOUD PROVIDER SELECTION USING NON MCDA METHODS

These works can select “the best” CPs by means the application of benchmarks to measure their performance, or executing their own methods/techniques or indexing operations and structures. Some works select CPs by measuring reliability/trust/reputation and compute risks for each interaction customer-CP using the SLA and complex mechanisms to estimate this qualitative measure.

“CloudCmp” (LI et al., 2010) is the first found systematic comparator for performance and cost of public CPs. This tool is developed to guide customers selecting the best cost-effective CP for their applications using performance benchmarks. It can measure features such as elastic computing, persistent storage and the networking services offered by four CPs: Amazon AWS, Microsoft Azure, Google AppEngine and Rackspace. That study was a start about the CPS problem though direct measurement of QoS metrics by benchmarks can be problematic and unstable. Also, this comparator takes into account only low-level performance metrics of Cloud services such as CPU utilization and network throughput, for example.

A brokerage approach using an indexing technique is used to create and index distinct CC services to assist the selection of CPs to users (SUNDARESWARAN; SQUICCIARIN; LIN, 2012). Cloud brokers are responsible for selecting the proper service for each client and have a contract with the CPs, collecting their properties (PIs), and with the consumers, collecting their service requirements. Brokers analyse and index service providers according to the similarity of their properties. Each property

(except service type) has a unique encoding. In this case, upon receiving a selection request, broker will search the index to identify an orderly list of candidate CPs based on how well they match the user needs. The generated index key is formed by the concatenation of the encoding type of service offered by the CP with a “Xor” operator among all the other encoded properties offered by such CP. The CPs are indexed in a tree structure called “B+-tree”. The approach is tested with a data set with six real CPs and nine PIs.

A framework for selecting the most trustworthy CC service providers in cloud environments using multiple qualitative recommendations is proposed by Bedi, Kaur e Gupta (2012). This model uses a multi agent system, based on cooperative social recommendation process (web of trust) where users select trustworthy CPs based on the recommendations given by their trustworthy acquaintances. All these recommendations are aggregated using weights and they build the reputation of each CP (given in the scale of 0-9, where 9 is the best value). The uncertainty present in these recommendations are treated using a Fuzzy Inference System (FIS), capable of inferring crisp output even when the inputs are imprecise or uncertain. Every agent in the system maintains a database with their personal experience about each CP and trust value on various trustworthy agents. The two main types of agents in this system are: user agents and service provider agents. The user agent receives cloud user requirements and becomes responsible for selecting a suitable CP that satisfy these requirements. This agent acts in community and help other agents at the time of request of any user’s requirements. The service provider agent communicates with the cloud owner in order to maintain a database with all corresponding resources available in this cloud owner data center. Experiments with real data considering 25 agents with a defined web of trust are presented. Also, six criteria are specified and a final trust ranking with eleven CPs was shown.

The evaluation model proposed by Wagle et al. (2015) verifies quality and status of CC services by an ordered heat map checking the Service Level Agreement (SLA) commitment. Data are obtained by cloud auditors and are viewed via a heat map ordered by the performance of each CP, showing them in descending order of overall provided QoS. This map represents a visual recommendation and support system to CC users and brokers. The main metrics are again based on the SMI and are: availability (divided in up-time, downtime and interruption frequency), reliability (load balancing, MTBF, recoverability), performance (latency, response time and throughput), cost (snapshot and storage cost) and security (authentication, encryption, and auditing). This selection approach is visual and apparently does not rely on an automated method for a final decision of which provider should be selected, making it complex to use if the number of CPs and QoS metrics grows. This problem is



solved by another CP ranking framework (WAGLE; GUZEK; BOUVRY, 2015) based on both CPs' service delivery measurements and cloud customer experiences (recommendation system). The CP recommendation is based on the entire SLA requirements as well as on the particular SLA parameters required by the customers. The framework decision making process combines the opinions of at least two cloud users and two auditors (internal and external). The internal auditor monitors CPs' performance data that could be used together with an application performance model to estimate its QoS. The external Auditor is a commercially available service on the Internet (e.g., Cloud-Harmony). This on-line CPs performance monitoring performed by auditors create additional degree of trust and reliability to the cloud users. If there are some CPs that may not allow to measure their service status, their confidence level will be lower, affecting its ranking. The ranking is performed by an intuitionist fuzzy decision making group, as it can include both measurable and non-measurable CP attributes. A simple example is performed with only three hypothetical CPs. The article is much simplified and there is a lack of data to conclude anything about its practical effectiveness.

The framework called "SelCSP" (GHOSH; GHOSH; DAS, 2015) can support customers identifying the most reliable cloud service provider by calculating the percentage of risk involved in each interaction customer-CP. Moreover, that work develops mathematical models for trust, reputation, competence, and risk estimation. The "SelCSP" acts as a third-party intermediary between customers and CPs and possess several modules. Each customer can provide trust ratings and feedbacks about its interactions with different CPs, creating a reputation data repository. Each registered CP needs to submit its SLA to the SLA manager module to the competence calculation for each CP. This is done by taking into account different recommendations/standards and controls which are supposed to be satisfied by the SLAs. The estimated risk of an interaction involves the computation of customer trustworthiness and CP competence values. Trustworthiness is computed from personal experiences built on direct interactions (trust) or from feedbacks related to CPs' reputations (relational risk). Competence is assessed based on transparency in CPs' SLA guarantees (performance risk). So, the total perceived interaction risk is the sum of the relational risk and the performance risk. The study case is performed for six hypothetical SaaS CPs, showing graphs with the obtained values for trustworthiness, competence and risks. The case where customers can act with malicious intents of submitting unfair ratings or wrong data was not treated by the framework yet.

An adaptive real time selection platform that plays the role of a broker between customers and IaaS CPs is proposed by (SOUIDI et al., 2015). This platform uses a set of benchmark tasks to evaluate the Quality of Experience (QoE) for each CP and it performs an adaptive selection in a dynamic network using the Page Hinkley algo-

rithm (AO; GAMA, 2009). This algorithm tracks the network changes and detects the break point of nonconformity between the customer needs and the specifications. The main criteria specified for the selection process are called Key Performance Indicators (KPIs). They are availability, security, reliability, network performance, scalability, instance performance and VM cost. So, the CP selection depends on CP performance and customer perception. The CP performance depends on three major parameters: location, time and VM performance. The end-user perception or QoE is a subjective measure of the adequacy of a service compared to the user expectation. The reliability is measured by a series of benchmarks executed in several VMs measuring data such as the processing power, the data I/O speed and memory accessing speed. The network performance is represented by measuring bandwidth and latency from the customer perspective, taking into account location and time parameters. The VM performance is a series of benchmark tests that measure the processing power, the input/output speed and the process creation speed. Each KPI also has different priority levels depending on the customer's preferences. This is modelled using weights. That work proposes two algorithms to perform selection: Simple CP Selection (SCPS) and Adaptive CP Selection (ACPS). SCPS uses LP techniques to select CPs according to the benchmark results and customer preferences. ACPS compares CPs performance to customer specifications and selects the best CP at each time. An experiment is performed using servers in different countries for the each one of the following CPs: Amazon, HP, Microsoft Azure and Numergy. The network dynamism supported by this platform and the developed benchmarks are important contribution of that work. However, its adoption can be too much time consuming if there are large amount of CPs involved on the selection process (horizontal scaling).

### 3.2 CLOUD PROVIDER SELECTION USING MCDA METHODS

These works can use one or more well established methods from the MCDA area. The main MCDMs used on the surveyed literature are: Analytic Hierarchy Process (AHP) (SAATY, 1990), Analytical Network Process (ANP) (SAATY, 1996), Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) (HWANG; YOON, 1981), fuzzy TOPSIS (KORE; RAVI; PATIL, 2017), Elimination et Choix Traduisant la Réalité (ELECTRE) (ROY, 1990), Preference Ranking Organization Method for Enriched Evaluation (PROMETHEE) (BRANS; VINCKE, 1985), Data Envelopment Analysis (DEA) (RAMANATHAN, 2003). These methods seem appropriate and intuitive to model and apply to the CPS problem. A more formal and general mathematical methodology is proposed in (REHMAN; HUSSAIN; HUSSAIN, 2011). It aims to transform CPS problem in an MCDA problem, including the customer's criteria request.

The work proposed in Rehman, Hussain e Hussain (2012) presents an appli-

cation example of several MCDMs for selecting IaaS cloud services. That work emphasizes the need of constant cloud monitoring tools and the need of development of new methodologies for CP selection. Also, it highlights the effectiveness of use of MCDM to this problem. The authors use seven notable MCDMs to select CPs: min-max and max-min methods, Compromise Programming, TOPSIS, ELECTRE, PROMETHEE and AHP. That is, they use both outranking and Multi Attribute Utility Theory (MAUT) methods. The study case contains thirteen CPs with five arbitrary performance measurements (two kinds of memory, cost, CPU and I/O) gathered from *cloudharmony.com*. The results are compared at end. Thus, seven methods did not select the same CP as the best one. The first CP was selected by three methods and the fourth CP was selected by two methods. No optimal CP was defined/identified. That work concludes that outranking methods and TOPSIS are more suitable to the selection, and if the number of CPs is too large, maybe TOPSIS can be the best choice. That work is interesting and looks promising, but it is very simplified, needs better explanations of the calculations made, the parameters/weights used, it is difficult to conclude anything with just that example. The case study is too much simplified, needs more data (e.g., the user's requirement criteria and its priority weights).

The framework called "SMICloud" (GARG; VERSTEEG; BUYYA, 2011; GARG; VERSTEEG; BUYYA, 2013) can rank CC services using indicators of the SMI. "SMI-Cloud" measures QoS of each CP and ranks them based on the calculated quality. For this, it uses the AHP method. Each indicator (AHP attribute/criteria) can be essential or non-essential and can be boolean, numeric, unordered set of classes or a range type. A small study case is provided at the end with three real CPs: Amazon EC2, Windows Azure and Rackspace. The main used SMI attributes are accountability level, agility (capacity and elasticity time), assurance (availability, stability, serviceability), cost (VM, storage), performance (response time) and security level. The unavailable data such as the security level are randomly assigned to each CP. The method is appropriate and logically plausible though it seems that the assembly of the hierarchy for a large number of CPs and PIs seems to be complex and tiresome. This method may also not be appropriate to treat qualitative PIs such as accountability and security levels. It also has implicit the subjectivity problem of the arbitrary choice of AHP weights.

A more detailed and structured AHP utilization for ranking CC services for end users is proposed by Karim, Ding e Miri (2013). This model maps users' QoS requirements (selection criteria/metrics/indicators) using the AHP hierarchy through different CC layers (e.g., SaaS and IaaS). In addition to mapping the relationships between features, AHP generates the QoS weights and computes the final QoS values for ranking the candidate services. The ten QoS criteria used are based on SMI. The model differential is that the requested QoS requirements are not end-to-end, but stratified in three

different layers: User Requirements, SaaS and IaaS. At user layer, the QoS criteria are divided into three main groups: customer-related (security, data control, reputation, cost and usability), performance-related (availability, reliability and response time), and system-related (elasticity and capacity). The final score of each one of these criteria involve sub-criteria of both SaaS and IaaS layer, except system-related ones, whose values are measured only at IaaS layer. At SaaS layer, there are classes and sub-classes that represent the specifications of each one of these criteria to measure QoS guarantees for an SaaS service. Analogously, at IaaS layer, there are classes and sub-classes that represent the specifications of each one of these criteria for IaaS services. Therefore, SaaS's QoS values are first computed using the metrics associated with SaaS services. Then, based on the used mapping rule, the IaaS values are calculated by computing the values of the IaaS's QoS metrics. The system aggregates both values using sequential composition. Basically, this mapping allows SaaS CPs to select the best IaaS CP aligned with the users' requirements to its own QoS specifications. So, both SaaS and IaaS CPs can provide the best service offer to end users through the mapping mechanism across these identified CC layers. At the end, a fictitious example is showed with five CPs with eight criteria and a user requirement example. The model seems appropriate, but maybe too complex to build for all SMI PI, making it hard to use for large instances with several CPs and more PIs.

An approach using the MCDM called ELECTRE or "Elimination and Choice Expressing Reality" for selecting and ranking IaaS CC services using attributes/criteria is also proposed in (MACEDO; SILVA; FERREIRA, 2013). ELECTRE is an outranking multi-criteria method, based on pairwise comparisons, that uses outranking relations to find a set of alternatives (CPs) that dominate over other alternatives (and not be not dominated by others), in a non-compensatory way, by calculating two main measures: the concordance and discordance indexes. This approach also specifies the format and structure of the user requirements (list of desired criteria, called selection policy) allowing the specification of user selection criteria in a format that can be used to feed a selection method. Each criterion has an importance weight, data type (numeric, boolean or string), constraint value, if it is mandatory (essential), etc. That approach has five steps: a) selection policy (user request) definition; b) obtainment of the services description (candidate services plus their criteria specified at the selection policy); c) application of constraints, eliminating any service that does not satisfy a constraint for a mandatory criterion; d) normalization of criteria values (between 0 and 1) of the services description using the selection policy, generating score values for each criterion of each service; e) application of a selection method using these normalized values. Two selection methods are used and compared: Elimination et Choix Traduisant la Réalité (ELECTRE) (non-compensatory technique) and a simple weighted additive function (compensatory technique). An example case with real data is presented with

32 services of the Amazon provider using eight criteria. No optimal answer was assumed. Thus, it is not possible to state that one method is better than another, but both methods choose the same services as the best and the worst solutions. The use of ELECTRE seems prominent, but more tests with larger databases and more diversified requests are needed for a better comparison with other MCDA methods.

The CP ranking voting model (BARANWAL; VIDYARTHI, 2014) can rank and select CC services based on users QoS expectation metrics. The base QoS metrics are again the SMI ones. The main actors of this model are the cloud exchange, cloud coordinator and cloud broker with a data directory that contains all information about CPs that are required when selecting the best one. The metrics are divided into two categories: application dependent and user dependent. Values of metrics can be of different types like numeric, boolean, range type, unordered set and data centre value. In a ranked voting system, each metric will act as a voter, and CPs are candidates for them. The proposed method to analyze ranked voting data was DEA (COOK; KRESS, 1990). A very similar proposal to this model is the CP measure index framework (SHIRUR; SWAMY, 2015), with the same QoS metrics (SMI) and same ranked voting method. The main actors of the system architecture are cloud swapping, cloud coordinator, cloud user and cloud mediate. The cloud index contains all information about service providers which are required to select the best CP. The description of each module or phase needs a better explanation and practical examples for better understanding of the method are also needed in both works.

A brokerage approach developed by Achar e Thilagam (2014) can rank IaaS CPs using a cloud broker measuring the QoS of each CP, prioritizing those that are most appropriate to the needs of each request. The key approach elements are: the broker, the requester of a CP and a CP list. Selection involves three steps: Identify which criteria are appropriate to the request by identifying the necessary PIs present in the SMI; assessing the weight of each of these criteria using the AHP method; and rank each CP using TOPSIS. TOPSIS is used to select the alternative that is closest to the ideal solution and further away from the ideal negative solution. The final example has six hypothetical providers with four PIs (availability, accountability, cost and security). The use of TOPSIS appears to be promising, but more analysis and examples are needed for further conclusions, with real data and users requests. Qualitative PIs need an appropriate modelling too.

TOPSIS and Fuzzy TOPSIS, with the conjugated use of AHP and ANP can also be used to the CPS problem (JAISWAL; MISHRA, 2017). In this approach, the purpose is to use TOPSIS and fuzzy TOPSIS to identify the most effective CC service according users' requirements. To assess criteria weights for each of these methods the AHP or ANP methods are used and the results are compared at the end. For perfor-

mance evaluation it is used an example with four hypothetical CPs with data gathered from *cloudharmony.com* and with eight arbitrary quantitative criteria. The presented examples are subjective and dependent on the weights assigned by the user, making the proposed method's efficiency questionable for large quantities of CPs. Also, the method does not seem to be able to handle qualitative criteria.

### 3.3 RELATED WORK ASSESSMENT

Table 6 presents a summarized overview of the main characteristics identified for comparison purposes between the related work discussed in this chapter and also with the proposed selection PI-based architecture, described at Chapter 4. The identified characteristics are:

- **Method:** The main mathematical methods/methodology/procedures used to select/rank CC providers/services.
- **Criteria:** The main CPs data used or identified as possible to be used by the authors. This data is used by the approaches/models/frameworks/methods to select/rank CC providers/services.
- **Service Model:** The kind of CC service model that the work is committed to handle and solve in its selection process. Three service models are possible: "SaaS", "PaaS" and "IaaS". Each model has different goals and characteristics, so the QoS criteria can change from one model to another and not all CPs offer all of them. In case the work scope is suitable to deal with all criteria, the word "All" is put in place.
- **QI:** Presence of qualitative attributes/criteria/Pis/data in the work modeling. In other words, the work uses or models or, at least, identifies, possible qualitative criteria for each CP in the selection process. It will be "Yes" in affirmative case and "No", otherwise.
- **Dim:** How scalable/dimensionable is the model presented, both vertically (increasing the number of criteria) and/or horizontally (increasing the number of providers)? In other words, how easy it is to handle cases with huge amounts of new criteria and CPs. Three types of responses are expected: "Low"; if the model/work is very specific, not very flexible and mutable, very difficult to scale vertically or horizontally; "Medium"; if the model/work can be expanded, but with a limit or with some difficulty; and "High", if the model/work can be easily expanded without a detectable limit that prevents its modelling or operation.

Table 6 – Overview and comparison between the related work and the proposed architecture

Work	Method	Criteria	Service Model	QI	Dim
Li et al. (2010)	Own method	Elastic computing, persistent storage and networking services	All	No	Low
Garg, Versteeg e Buyya (2011) and Garg, Versteeg e Buyya (2013)	AHP	SMI	All	Yes	Medium
Sundareswaran, Squicciarini e Lin (2012)	Own indexing method	Service type, security level, QoS level, measurement unit, instance sizes, OS and price	All	Yes	Medium
Bedi, Kaur e Gupta (2012)	Own method	Trust & Reputation	All	Yes	Low
Rehman, Hussain e Hussain (2012)	Min-max, max-min, Compromise Programming, TOPSIS, ELECTRE, PROMETHEE and AHP	Memory, cost, CPU and I/O	IaaS	No	High
Karim, Ding e Miri (2013)	AHP	A subset of SMI	IaaS and SaaS	Yes	Low
Macedo, Silva e Ferreira (2013)	ELECTRE	CPU, RAM, Platform, Storage, I/O Performance, EBS Available, Cost and OS	IaaS	Yes	High
Baranwal e Vidyarthi (2014)	Voting method using DEA	SMI	All	Yes	Medium
Achar e Thilagam (2014)	TOPSIS and AHP	SMI	All	No	Medium
Wagle et al. (2015)	Own method	A subset of SMI	All	Yes	Low
Wagle, Guzek e Bouvry (2015)	Own method	A subset of SMI	All	Yes	Medium
Ghosh, Ghosh e Das (2015)	Own method	Risk	All	Yes	Low
Soudi et al. (2015)	Own method	Availability, security, reliability, network performance, scalability, instance performance and cost	IaaS	Yes	Low
Shirur e Swamy (2015)	Voting method	SMI	All	Yes	Medium
Jaiswal e Mishra (2017)	TOPSIS, fuzzy TOPSIS, AHP and ANP	Cores, memory amount and performance, price, CPU performance, disk (input/output operations per second, consistency, performance)	IaaS	No	Medium
Proposed Architecture	Any	Any	All	Yes	High

Source: Author

After analysing Table 6 some conclusions can be highlighted:

- MCDA methods (especially AHP, followed by TOPSIS) and SMI PIs are well disseminated concepts among researchers for the CPS problem, starting with the work called “SMICloud” (GARG; VERSTEEG; BUYYA, 2011), which uses both concepts.
- Most of the selection criteria used are based on the entire SMI framework or in a part of it, either directly, by citing it explicitly, or indirectly. A notable and highly cited criterion is service cost/price, also present in the SMI framework as well.
- Apparently, most of these works try to solve the CPS problem for all possible cloud service delivery models (IaaS, PaaS and SaaS), i.e., their models are generic enough to handle different criteria sets from these three service models.
- Also, most models/works are concerned with the existence of qualitative criteria and with its modelling and treatment. This concern is justifiable, because this kind of criteria is abundant in the CPs’s measured characteristics (see SMI).
- However, most of these models/works does not have enough generality to be easily scalable for very large amount of CPs and criteria. Actually, all the models’ application examples presented have no more than fifty CPs with ten criteria. Basically, the two models with high scalability transform the CPS problem into an MCDA problem without a total dependence of arbitrary criteria or SMI criteria (but still dependent on MCDMs).
- Finally, the proposed architecture was developed to provide the greatest generality and scalability/dimensionality possible, becoming a powerful general tool that is able to handle and use all kinds of methods and criteria, independent whether the method is from the MCDA area or if the criteria is from the SMI. The service selection can be done for the all the three service delivery models.

In addition, it is important to state that all these works do not consider the cases of choosing the best set of CPs, i.e., when each CP can complement each other in order to maximize the attendance of the requested criteria. In other words, these related methods do not evaluate possible relationships between CPs with a proper utility function taking into account the whole customer request. As a matter of fact, these related works do not consider the case where a possible customer request can be attended by more than one CPs alone. Therefore, only exact methods (deterministic ones) are specified for these simple instances of the CPS problem.

However, the proposed architecture can even handle these more complex cases, involving metaheuristics for solve these instances. Moreover, these metaheuristic-



tic methods can be complemented using hybrid approaches, linking determinism and stochasticity, facilitating the search process of the optimal solution and making meta-heuristic methods more reliable for simple requests, regardless the CP database size. Moreover, even tolerance values can be specified at the request, giving the customer more control over the selection process. Thus, this architecture can solve harder instances of the CPS problem.

### 3.4 PARTIAL CONSIDERATIONS

This chapter presented fifteen relevant works for the CPS problem research area since the year 2010. The MCDA methods are widely recommended and used for this purpose (REHMAN; HUSSAIN; HUSSAIN, 2011; WHAIDUZZAMAN et al., 2014). So, these works can be divided between works that use one or more MCDM and those who do not use them.

In order to the selection process work, several data about each CP is gathered. These data are presented in form of several attributes, criteria or PIs and are used by the proposed selection method to select/rank CP. A notable PI set is the SMI (SIEGEL; PERDUE, 2012; CSMIC, 2014). The SMI framework is well-known and well accepted by researchers.

A brief summary for each work is presented as well as a comparative table with the main common characteristics identified in each work as well as in the proposed architecture in order to present its differential. Thus, the proposed work can be contextualized and inserted along with those already published works.

With the comparative table it is possible to state that the main MCDA methods used are AHP and TOPSIS and the main criteria are based on the SMI, either directly or indirectly. A notable and highly cited criterion found is the service cost or price. Most of these works try to solve the CP selection for all possible cloud service delivery models and they are concerned with the existence of qualitative criteria. However, most of these works does not have enough generality to be easily high scalable for very large amount of CPs and criteria. Furthermore, all these works do not consider the cases where choosing a set of CPs can be more beneficial in order maximize the requested criteria. Thus, in this context, the proposed architecture have the greatest generality and scalability needed to become a powerful tool able to handle all kind of methods, criteria and service models to solve the CPS problem.

## 4 SELECTING AND RANKING CLOUD PROVIDERS

This chapter presents the proposed PI-based architecture developed to solve the CPS problem. This architecture is composed of several components including a pool of developed CP selection methods. This chapter is divided into two sections. Section 4.1 presents the proposed modelled architecture components targeting the CPS problem: CP database, customer interface and customer request. Section 4.2 aims to specify some provider selection methods, either deterministic (CPS-Matching and CPS-DEA), metaheuristic (CPS-GA, CPS-BDE, CPS-DDE and CPS-SA supported by the individual fitness and energy functions modelling and implementation) or hybrid (CPS-Matching plus a metaheuristic in a sequential order), adapted specifically for the CPS problem solving.

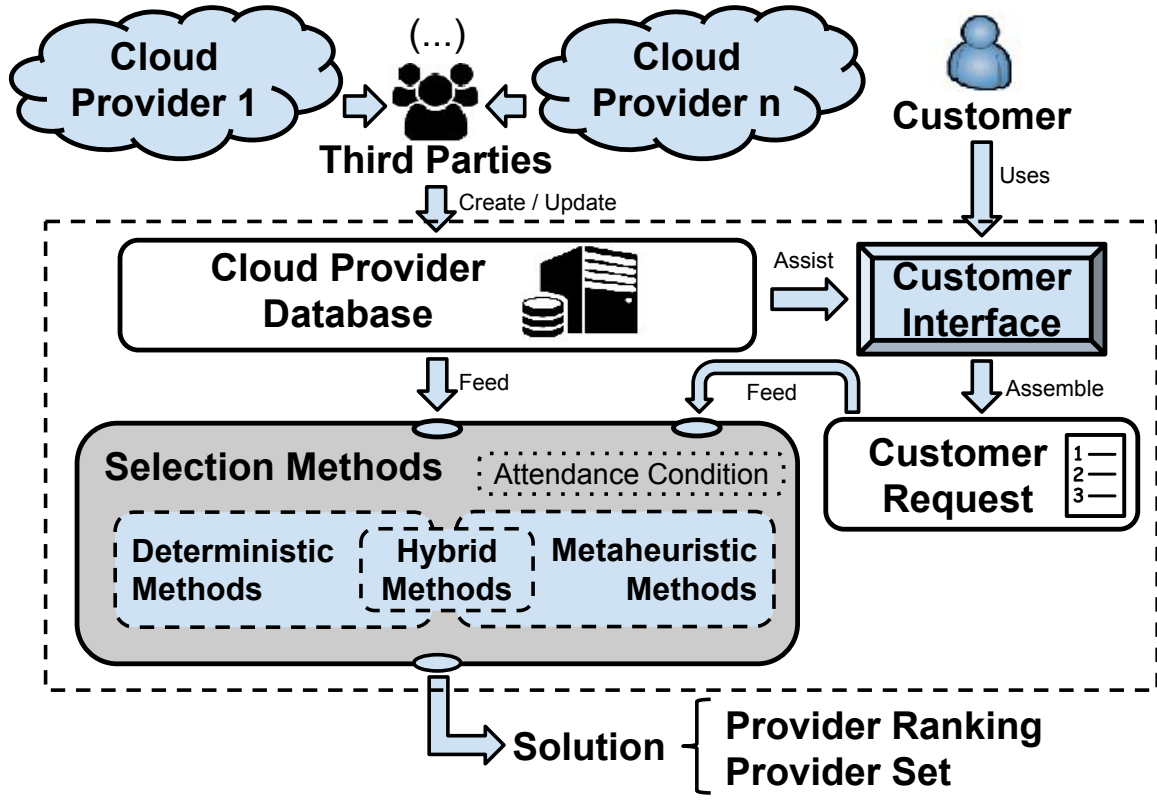
### 4.1 CLOUD PROVIDER SELECTION ARCHITECTURE

This section presents the proposed CPS PI-based architecture to solve CPS problem. Figure 11 presents a general overview of this architecture. This architecture is composed of three indispensable components: 1) CP database; 2) customer request and 3) selection methods, and an auxiliary one: customer interface.

This architectural modelling aims to define a generic and general approach regarding any kind of CP selection using PIs. This modelling assumes the existence of the basic initial database where the proposed selection methods (Section 4.2) can be applied. In practice, a third-party (e.g., server where the selection methods are hosted) must have an extensive database containing a list of CPs. Basically, each CP can measure and store its own PI set or contract an external cloud agent such as a Cloud Auditor or Cloud Broker (HOGAN et al., 2013) in order to make available these data to a cooperative and publicly accessible database. This database of candidate CPs and their PIs can be fed indirectly by online services such as “Cloud Harmony” (<cloudharmony.com>), “Cloudorado” (<www.cloudorado.com>) and “Cloudwards” (<www.cloudwards.net>), by means of CPs by their own (e.g: Amazon, Microsoft, etc.) or it can be consolidated by third parties.

In order to this architecture be used, a total of  $r$  PIs of interest should be chosen by the customer according to its goals comprising the CP selection request. The customer will have an informative support interface, that communicates directly with this database, informing the available PIs and able to collect the selected one’s desired value and importance weight, for instance. Thus, comprising this scenario, given a finite initial non-empty set  $P$  with  $n$  different available CPs, where each CP has  $m$  distinct

Figure 11 – Proposed architecture overview and plausible utilization scenario



Source: Author

PIs associated (plus CP price), and given a single customer request, the problem is to choose the smallest subset of CPs  $P' \subseteq P$ , that maximizes the attendance of the entire customer request (interest PIs list) with the lowest price involved.

In order to properly select these CPs, it is needed selection methods. These methods are divided into three main categories: deterministic, metaheuristic and hybrid. Each method loads a list with all the CPs from the database with their respective PIs and prices, along with a list with all PIs of consumer interest (consumer request). Thus, a specific selection method is applied on this initial CP list that can return two kind of responses: a CP ranking (highest scored CPs) in case of a deterministic method, or a CP set (CPs with highest fitness found in all method executions) in case the method used is a metaheuristic or hybrid one. Both responses can be suitable enough and can be returned to the customer as final solution for his request. All in all, the response with a CP set is expected for complex requests, where it offers more than one CP in order to attend all the customer needs.

#### 4.1.1 Candidate Providers' Database

The CP candidates database stores all available CPs with the name of all  $m$  registered PIs (plus price) with their respective values  $(x_{ij})$  and utility function type.

Each value  $x$  for the  $i$ th CP and  $j$ th PI can be quantitative (integer or real) or qualitative (boolean or string). As already mentioned, the utility function type available for quantitative PIs are: Higher is Better (HB), Lower is Better (LB) and Nominal is Best (NB). Qualitative PIs can be ordered, where the utility function can be Higher is Tolerable (HT), Lower is Tolerable (LT) and Higher and Lower are Tolerable (HLT). If the PI is unordered, the utility function is always NB. A relevant PI used for selecting CPs is price. Price is a PI very considered in the literature studied, even if it is not informed by the customer in the request, since it is a value that is always desired to minimize (LB).

Table 7 presents an example of a possible CP database. This database is agnostic and generic for all kinds of CPs and PIs. Thus, each CP has a list of associated PIs and each PI can be present in multiple CPs. The PI measurement unit must be the same for all CPs, e.g., if a PI represents RAM in gigabyte, all RAM values for the other CPs need to be stored in gigabytes too. This restriction makes the database more homogeneous, making the selection process easier by reducing the number of operations (unit conversions) at runtime, eliminating trivial errors.

Table 7 – Generic CP database

<b>CPs/PIs</b>	$PI_1$	$PI_2$	$PI_3$	...	$PI_m$	<b>Price</b>
<b>Type</b>	HB/LB/NB/...	HB/LB/NB/...	HB/LB/NB/...	...	HB/LB/NB/...	<b>LB</b>
<b>Group</b>	0/1,2,...	0/1,2,...	0/1,2,...	...	0/1,2,...	<b>0</b>
$CP_1$	$x_{11}$	$x_{12}$	$x_{13}$	...	$x_{1m}$	$y_1$
$CP_2$	$x_{21}$	$x_{22}$	$x_{23}$	...	$x_{2m}$	$y_2$
$CP_3$	$x_{31}$	$x_{32}$	$x_{33}$	...	$x_{3m}$	$y_3$
...	...	...	...	...	...	...
$CP_n$	$x_{n1}$	$x_{n2}$	$x_{n3}$	...	$x_{nm}$	$y_n$

Source: Adapted from Moraes et al. (2018)

PIs can also be identified by a group id (positive integer), where zero means that the PI is independent and is not involved with any group. PIs on a same group (equal group id) are inseparable from each other. This means they need to be all attended by the same CP and should not be attended separately by different CPs. This feature is useful for cases where there are interconnected characteristics or resources that does not make sense to belong to different CPs because they compose a single element (e.g., a VM is formed by RAM, disc memory, CPU, OS, etc.). Price is LB type and group 0, always.

If some values of Table 7 are not informed, default values are put in place. The default type for a PI is always NB (whether quantitative or qualitative) and default group id is 0, meaning it is independent/separable from the other PIs. Default PIs value for a CP varies according to the meaning of each PI and its type (quantitative or qualitative).

### 4.1.2 Customer Request

Customer request represents the customer computing needs to achieve its goals comprising CPs. The request must inform all the  $r$  customer's PIs of interest, which must be a subset of the database registered ones (i.e.,  $r \leq m$ ), with the respective desired value ( $X_j$ ). Thus, lets  $A$  be the set of PIs and their values in the customer request and  $B$  the set of all available PIs in the database, then  $A \subseteq B$ . Other features can be informed in the customer request, such as the importance weight of each PI of interest ( $w_j$ ), the tolerance value of the desired one ( $t_j$ ), eventually the PI utility function type and whether this PI is essential (must be attended) to the customer or not. Informing the PI type and/or group overwrites the CP database default one for only that request (assuming the customer knows what he is doing). An essential PI indicates that if that PI is not attended, the customer cannot achieve its goals, making the attendance of this PI mandatory. Price is always LB with group id of 0 and cannot be forced to other type.

Table 8 shows a generic customer request, with all fields that can be informed to the selection method. Important to mention that PI's name (identifier) and desired value (integer or floating point number) are mandatory information. The other ones are optional.

Table 8 – Generic customer request

Name	Value	Type	Group	Tolerance	Weight	Essential
$PI_1$	$X_1$	.	.	$t_1$	$w_1$	true
$PI_2$	$X_2$	NB	.	$t_2$	$w_2$	false
$PI_3$	$X_3$	.	0	$t_3$	$w_3$	false
...	...	...	...	...	...	false
$PI_r$	$X_r$	.	.	$t_r$	$w_r$	false

Source: Adapted from Moraes et al. (2018)

Request column "Type" forces the PI type overwriting the default type specified in the database (e.g.,  $PI_2$  will be NB). Similarly, column "Group" overwrites the PI group id in the database (e.g.,  $PI_3$  will be 0). The default PI tolerance value is zero. If the PI is qualitative, the tolerance is implicit (HT, LT and HLT – for ordered ones – or NB, otherwise). The default importance weight of each PI is 1. A PI is non-essential ("false") by default.

### 4.1.3 Customer Interface

Customer interface is an useful auxiliary component that interacts directly with the CP database component allowing cloud customer to assemble its request. To accomplish that, this component is responsible to present all PIs available in the database allowing customer to choose ones of its interest. Figure 12 presents an example of

a customer interface. Basically, this interface can be divided into four component: “Provider List”, “PI List”, “Information Frame” and “Assemble New Request”.

**Customer Interface**

**Provider List**  
 1. CP1   cp1@web.com  
 2. CP2   cp2@ex.com  
 ...   ...  
 n. CPn   cpn@site.com  

Choose this Provider

**Assemble New Request**  

PI	Value	Tol.	Weight	Type	Essential
PI1	<input type="text" value="x"/>	<input type="text" value="t"/>	<input type="text" value="w"/>	<input type="text" value="NB"/>	<input type="text" value="True"/>
				Group	<input type="text" value="0"/>

Add PI

Send Request

**PI List**  
 1. PI 1   Description 1  
 2. PI 2   Description 2  
 ...   ...  
 M. PI M   Description M

**Information Frame**  
 HB = Higher is Better  
 Ex: Memory, Cores,  
 Service Accuracy, ...  
 (...)

Exit

Figure 12 – Example of a customer interface frame

“Provider List” shows all stored CPs allowing the customer to check related information about each one (e.g., name, description, website, database localization, etc.) and their associated PIs (e.g., price). These information are showed at the “Information Frame”. With that, customer can also manually select a CP, if it wishes, without having to assemble a request. This manual selection is acceptable and understandable if the database has few registered CPs (20 or less) and/or if the customer needs involves only few PIs (one, two or three).

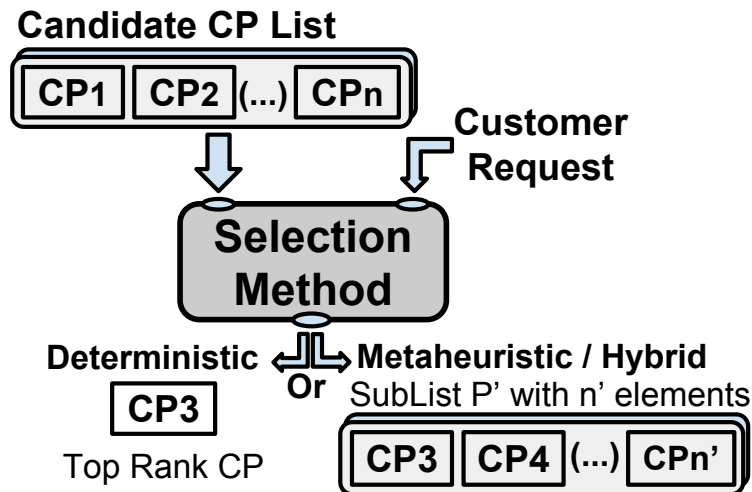
“PI List” shows all registered PIs in database. A PI has to be associated with at least one CPs to be listed there. The main purpose of this list is to inform the customer about what each PI means. Each PI has a name, a description, a type (quantitative or qualitative), a default utility function type (HB, LB, NB, HT, LT, HLT) and a group id at CP database. These information are showed at the “Information Frame”. Customer can use these information to decide what will be its PIs of interest on its future requests.

Finally, “Assemble New Request” component helps customer to create a new valid request that represent its needs, using the chosen PIs of interest. Each request is a list of PIs of interest. For each PIs, customer should provide: PI id/name (mandatory), desired value (mandatory), tolerance value (optional), weights (optional), arbitrary utility function type (optional) and whether that PI is essential (mandatory if it is, optional, otherwise). At the end, the request must be sent to the server where the selection methods component is hosted to be processed and the best CPs are selected.

## 4.2 SELECTION METHODS MODELLING

This section aims to specify some developed selection methods, particularly adapted for solving the CPS problem, present in our proposed CPS-Architecture. These methods are useful mathematical and/or nature-inspired techniques, that assist the customer decision-making process for the selection of the best CPs. These methods are indispensable to a careful PI-based CP selection and works on the initial scenario described at Section 4.1. Moreover, all these adapted selection methods uses the concept of PI attendance condition, concept presented in Subsection 4.2.1. These methods are divided into three categories: deterministic (Subsection 4.2.2), metaheuristic (Subsection 4.2.3) and hybrid (Subsection 4.2.4). Thus, Figure 13 presents an overview of the utilization and the different response types of the proposed selection methods.

Figure 13 – Selection methods utilization and its response types



Source: Adapted from Moraes, Fiorese e Parpinelli (2018b)

The method loads a list with all the CPs from the database with their PIs and prices, along with a list with all PIs of interest (customer request). Thus, a selection method is applied on this initial CP list that can return two kind of responses: a CP ranking (highest scored CPs) whether it is a deterministic method, or a CP set (highest fitness found in all method executions) in case it is a metaheuristic or a hybrid method. Both responses can be suitable enough and can be returned to the customer as final solution for his request. All in all, the response with a CP set is expected for complex requests, where the request is not fully attended at first. This mean the response offers more than one CP in order to attend the customer needs. Figure 13 shows  $CP_3$  as the CP with highest score, but not 100% suitable (i.e., it does not attend all requested PIs). Thus, depending on the minimal value of suitability established by the customer, only



$CP_3$  may not be enough, then a metaheuristic can be applied instead to find the final full set of CPs that can maximize the request attendance.

#### 4.2.1 The PI Attendance Condition Concept

The PI attendance condition concept is central for this work and it is used several times in all the selection methods addressed here. Basically, a PI value from a determined CP, attend or satisfy a certain customer desired value, if this value is the desired one or it is better than the desired one or, at least, it is in the tolerance range from the desired one. To make such evaluation, the type of each PI should be considered. The first step is to identify if the PI is quantitative or qualitative.

Thus, mathematically, a  $CP_i$  “Attend” certain quantitative  $PI_j$ , if its numeric value  $x_{ij}$  (present in the database), attends the customer desired value  $X_j$  (present in the request), with tolerance  $t_j$  (specified in the request, otherwise, tolerance is zero). Equation 4.1 presents this quantitative PI attendance condition function. Function “type” returns the utility function behavior of the PI.

$$AttendQT(PI_j, x_{ij}, X_j, t_j) = \begin{cases} x_{ij} \geq (X_j - t_j), & \text{if } type(PI_j) = HB \\ x_{ij} \leq (X_j + t_j), & \text{if } type(PI_j) = LB \\ x_{ij} \geq (X_j - t_j) \text{ and } x_{ij} \leq (X_j + t_j), & \text{if } type(PI_j) = NB \end{cases} \quad (4.1)$$

Meanwhile, if the  $PI_j$  is qualitative, it can be ordered or unordered (JAIN, 1991). If it is unordered, the rule is simple: if  $x_{ij}$  (category, present in the database) is the value that the customer desires ( $X_j$ , category, present in the request), then the  $PI_j$  is attended, otherwise it is not. However, if  $PI_j$  is ordered, then the discussion on Section 2.2 comes in place here, and the ontology HT, LT and HLT too. So, the attendance condition depends on its utility function and category levels of  $x_{ij}$  and  $X_j$ . Thus, Equation 4.2 presents this qualitative PI attendance condition function. Unordered PIs are always NB. Ordered ones can be one of four types (HT, LT, HLT or NB).

$$AttendQL(PI_j, x_{ij}, X_j) = \begin{cases} lev(x_{ij}) = lev(X_j), & \text{if } type(PI_j) = NB \\ lev(x_{ij}) \geq lev(X_j), & \text{if } type(PI_j) = HT \\ lev(x_{ij}) \leq lev(X_j), & \text{if } type(PI_j) = LT \\ true, & \text{if } type(PI_j) = HLT \end{cases} \quad (4.2)$$

Function “lev” returns the level (positive integer) associated to that categorical value for that  $PI_j$ . For example, if  $Cat = \{Cat_1, Cat_2, \dots, Cat_t\}$  is the ordered set with

all possible categories that  $x_{ij}$  and  $X_j$  can assume for  $PI_j$ , and  $Cat$  is sorted by increasing order of graduation between the categories (e.g.,  $\{low, medium, high\}$ ), then  $lev(Cat_k) = k$ , where  $k \in \mathbb{N}^*$ . Note that Equations 4.1 and 4.2 takes into account the utility function type of each PI and  $i = 1, 2, \dots, n$ , where  $n$  is the total number of CPs in database, and  $j = 1, 2, \dots, r$ , where  $r$  is the total number of PIs in the customer request. The default type of each  $PI_j$  is NB. Thus, in any case of  $PI_j$ , whether quantitative or qualitative, whose value  $x_{ij}$  does not satisfy Equation 4.1 or Equation 4.2, respectively, it is said that  $PI_j$  does not attend the desired value,  $X_j$ , specified by the customer.

#### 4.2.2 Deterministic Methods Modelling

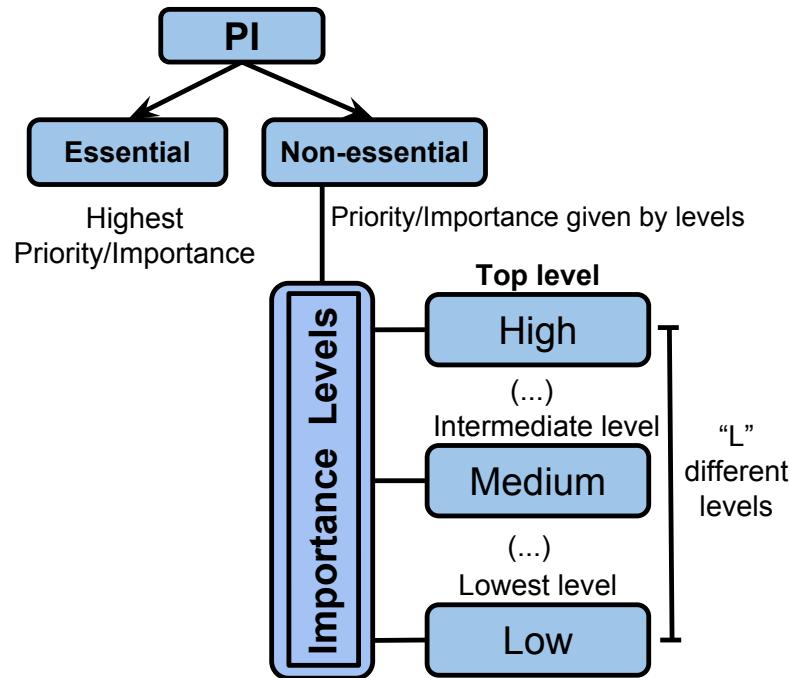
The deterministic selection methods solves a simplified instance of the CPS problem, where a single CP (the best one) is a good enough answer for the customer request. This approach simplifies the problem and drastically reduces the methods execution times. The basic idea is to evaluate each CP separately from each other. This evaluation is based on the value of their PIs and the customer desired PIs. Thus, each CP has a score value used to build a final CP ranking, where the top CP is basically considered the best one for the customer. A simple deterministic PI matching method (Subsection 4.2.2.1) can perform this work, as well as the MCDM, like DEA (Subsection 4.2.2.2). It is important to note that the PIs' grouping approach (Subsection 4.1.1) is not used by deterministic methods, since the dissociation between PIs and CPs does not exist in it. The PIs' grouping approach is only used by metaheuristic methods (fitness function).

##### 4.2.2.1 CPS-Matching

The Cloud Provider Selection using PI Matching (CPS-Matching) is a deterministic algorithm that can score and rank an extensive list of CPs based on value, type (quantitative or qualitative), nature (HB, LB, NB) and importance (essential or non-essential) of each PI requested by the user (cloud customer) (MORAES; FIORESE; MATOS, 2017; MORAES; FIORESE, 2018). The method is agnostic and generic for which PIs are used and can handle tolerances specified for each PI. A classification for the PIs' importance levels was created for this method and it is illustrated in Figure 14.

Customer can classify each PI as essential or non-essential. Essential PIs have the highest importance for the customer and consequently to the method. It indicates that if that PIs is not attended, the customer cannot achieve their goals. This makes the essential PI non attendance an elimination criterion. Thus, a CP that does not attend all essential PIs will be automatically deleted from the list of valid candidates because it is incompatible to that customer. Non-essential PIs have different priority levels, which can vary between levels "High" and "Low". If a non-essential PI is not

Figure 14 – Classification of priorities of the PIs by matching method



Source: Adapted from Moraes, Fiorese e Matos (2017)

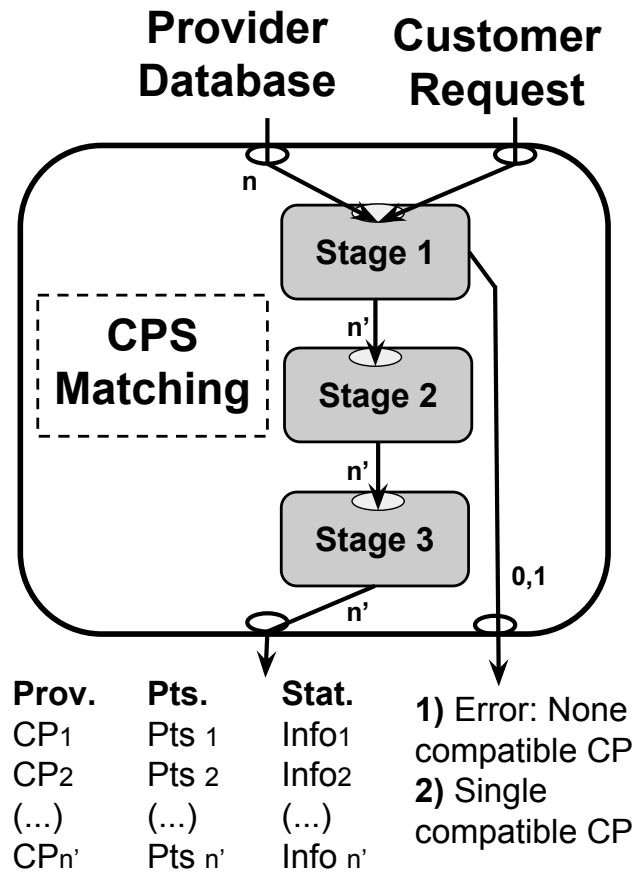
attended, it does not incapacitate the customer to achieve their goals, but it can jeopardize them. How much it could be jeopardized is directly related to the PI importance level, set by customer. Do not attending a very high level PI means that the user will be seriously impaired in achieving their goals. On the other hand, do not attending a minimum importance level PI implies that the negative impact will be virtually imperceptible. Optional PIs can be automatically classified with minimum importance level. Basically, the PI weight can define which is the importance level of each PI. The concept of importance level is used only by CPS-Matching.

Figure 15 presents the CPS-Matching method overview. Inputs corresponds to a list  $P$  with  $n$  different CPs candidates and the customer request, containing PI of interest and the importance level of each one. The expected output, except the special conditions, is a list with the CPs better scored by the method. The method is divided into three main stages (MORAES; FIORESE; MATOS, 2017):

1. Stage 1: Elimination of incompatible CPs;
2. Stage 2: Scoring quantitative and/or qualitative PIs by level of importance; and
3. Stage 3: Calculation of final score per CP, ranking them and returning results to the customer.

The initial CPs list  $P$  will be filtered, at the first method's stage, removing all

Figure 15 – CPS-Matching method overview



Source: Adapted from Moraes, Fiorese e Matos (2017)

incompatible CPs generating a new list  $P'$  with  $n'$  (with  $n \geq n'$ ) different CPs. Incompatible CPs are those that do not attend all the customer's essential PIs. If  $n' = 0$ , there are no available compatible CPs to the customer, so the method interrupts the process with an error message; if  $n' = 1$ , there is only a single compatible CP, which will be returned to the customer; however, if  $n' > 1$ , the method proceeds along to another stage to rank CPs.

The second stage aims to score each CP importance level individually, according to the utility (real benefit) of each one of its PIs. Higher utility means higher score. This stage will receive the list  $P'$ , with the  $n'$  filtered CPs from the previous stage. Basically, all PIs with the same weight are at the same importance level and the score of each importance level is the simple arithmetic average of the individual scores of each  $PI_j$  (quantitative or qualitative) within that level (MORAES; FIORESE; MATOS, 2017). This is done using the attendance condition (Subsection 4.2.1) for all PIs on the request. The score of a  $PI_j$  will be always 0 if its CP value does not attend the customer desired one. If the value is attended, it will be scored in proportion to how useful this value is compared to all other CPs available, multiplied by a constant.

The evaluation function of a quantitative  $PI_j$  of a  $CP_i$  is shown in Equation 4.3. There,  $x_{ij}$  is the PI value (numeric) in database,  $X_j$  is the desired value requested by the customer, with a maximum tolerance of  $t_j$  for that PI. This function always returns a normalized floating point number between 0 and 1 ( $\forall x_{ij}, X_j, X_{max}, X_{min} \geq 0$  and  $C_1, C_2, t > 0$ ). Function “AttendQT” is depicted in Equation 4.1. Function “type” is the same one of Subsection 4.2.1.

$$ScoreQT(PI_j) = \begin{cases} 0, & \text{if } AttendQT(PI_j, x_{ij}, X_j, t_j) = false \\ C_1 + C_2 * \frac{x_{ij} - X_j}{Max(x_{1j}, \dots, x_{nj}) - X_j}, & \text{if } type(PI_j) = HB \\ C_1 + C_2 * \frac{X_j - x_{ij}}{X_j - Min(x_{1j}, \dots, x_{nj})}, & \text{if } type(PI_j) = LB \\ C_1 + C_2 * \frac{t_j - |X_j - x_{ij}|}{t_j}, & \text{if } type(PI_j) = NB \end{cases} \quad (4.3)$$

The real constants (empirical parameters)  $C_1$  and  $C_2$  belong to the normalized open interval between  $]0, 1[$ , and  $C_1 + C_2 = 1$ , mandatorily. The number  $X_{max}$  is the highest value among all other  $n'$  CPs in the list  $P'$  for that  $PI_j$ ; as well as  $X_{min}$  is the lowest value and  $t_j$  is the maximum tolerated distance from the optimum point ( $X_j$ ) for an NB PI (since that PI attends  $X_j$ , i.e., belongs to the interval  $[X_j - t_j; X_j + t_j]$ ).

Constants  $C_1$  and  $C_2$  can be configured by the customer. They can be tuned according the customer understanding of how to weight the PI attending and how proportionally attended is the PI regarding the same PI on other CPs. The constant  $C_1$  weights the score given to the desired minimum match (including the tolerances associated with the PI, if any) between the CP value ( $x_{ij}$ ) and the value that the customer desires ( $X_j$ ), based on the PI type under analysis (HB, LB or NB). The constant  $C_2$  weighs the score given to how much this PI value excels the desired minimum, i.e., the value  $x_{ij}$  is, in practice, better than the desired value  $X_j$ . It is noteworthy that the first coefficient ( $C_1$ ) must be greater than the second one ( $C_2$ ), because it is not interesting to weight more how better a PI is comprising its value in other CPs, in prejudice of attending the user desired value. Also, it is essential that  $C_1 + C_2 = 1$  (normalization purpose).

The evaluation function of a qualitative  $PI_j$  of a  $CP_i$  is shown in Equation 4.4. Where  $x_{ij}$  is the PI value (category or subclass) in database,  $X_j$  is the desired value requested by the customer and  $K_1, K_2$  are the total number of tolerable categories higher and lower, respectively, to  $X_j$ , and  $K_3 = K_1 + K_2$ . This function always returns a normalized floating point number between 0 and 1 ( $\forall K_1, K_2, K_3 \geq 0$  and  $0 < C_3 < 1$ ). Function “AttendQL” is depicted in Equation 4.2. Functions “type” and “lev” are the

same ones specified at Subsection 4.2.1.

$$ScoreQL(PI_j) = \begin{cases} 0, & \text{if } AttendQL(PI_j, x_{ij}, X_j) = false \\ 1, & \text{if } x_{ij} = X_j \\ C_3 * \frac{K_1 - |lev(x_{ij}) - lev(X_j)| + 1}{K_1}, & \text{if } type(PI_j) = HT \\ C_3 * \frac{K_2 - |lev(x_{ij}) - lev(X_j)| + 1}{K_2}, & \text{if } type(PI_j) = LT \\ C_3 * \frac{K_3 - |lev(x_{ij}) - lev(X_j)| + 1}{K_3}, & \text{if } type(PI_j) = HLT \end{cases} \quad (4.4)$$

Note that the score of tolerable categories (only of ordered PIs with tolerances, i.e., HT, LT or HLT) will be directly influenced by the distance between the category specified by the customer ( $X_j$ ) and that offered by the CP ( $x_{ij}$ ). This means the greater the distance of the category in question to the desired one (both given by integers), the lower the score for that PI. The distance between the categories  $x_{ij}$  and  $X_j$  is the difference between their levels (positive integer, from 1 to the total of categories available in increasing order of graduation). Therefore, in case of perfect match ( $x_{ij} = X_j$ ) the maximum score 1 is applied. In this case, the desired neighbouring categories (above and below) will score  $C_3$ .

The real constant  $C_3$  represents the maximum score that a tolerable category (another category different of the desired  $X_j$ , but within the tolerances associated with that particular PI) can assume. Therefore, the smaller the value of  $C_3$ , the more aggressive is the penalty (loss of score) applied to any and every  $PI_j$ , whose category diverges from the desired optimal value. If  $C_3 = 0$ , then the method punctuates with zero any value different than  $X_j$ , whether the PI is ordered or not. This is an undesired behaviour, since it depreciates sub-optimal, and can penalize excessively CPs that are also appropriate for the user. If  $C_3 = 1$ , the method depreciates the importance of reaching the optimal point for a qualitative PI, assigning too much punctuation to sub-optimal ones, encouraging the wrong choice of the best CP.

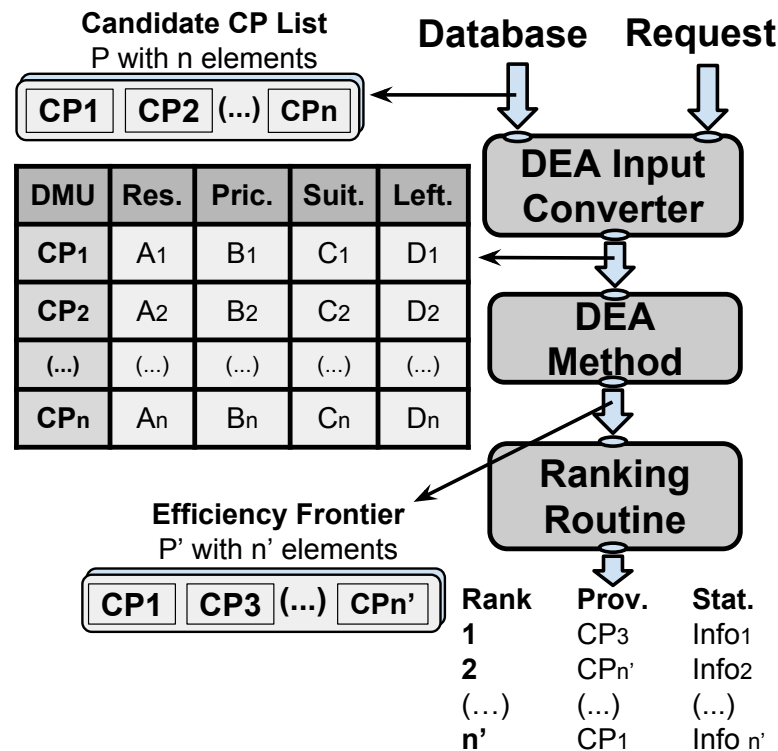
Finally, the third stage aims to calculate the final score for each CP. The final score is the weighted average of the score of each importance level available to that CP. The weights used in this average can be calculated using the importance weights values in customer request and the AHP's Judgement Matrix, as shown in Appendix B, or just be specified by the customer with floating point numbers. The score is calculated individually for each CP and varies in the range from 0 to 1. The closer to 1, the more adequate is that CP to satisfy that request. Finally, the CPS-Matching method returns a list with the highest-ranked CPs, containing their names and suitability. This suitability is the average of all requested PIs attended by that CP, pondered by their weight value, given in percentage.

#### 4.2.2.2 CPS-DEA

The Cloud Provider Selection using Data Envelopment Analysis (CPS-DEA) is a PI-based modelling of the classical method called DEA (Subsection 2.3.1.2) for the CPS problem (MORAES et al., 2018). Thus, in this case, each solution alternative is called a DMU (e.g., CPs) and each criterion is an input or an output related to the customer requested PIs values and CPs values. CPS-DEA selects CPs calculating the efficiency of each one. The efficiency is the ratio of the sum of its weighted outputs (to be maximized) to the sum of its weighted inputs (to be minimized). CPs with max efficiency (value 1, which means 100%) form a set called efficiency frontier. Therefore, identify which inputs and outputs CPS-DEA has to use for calculating efficiency is an essential step for the correct use of this method.

Figure 16 presents the CPS-DEA method overview. This method is divided into three main stages (MORAES et al., 2018): DEA input converter, DEA method application and the final ranking routine.

Figure 16 – CPS-DEA method overview



Source: Adapted from Moraes et al. (2018)

DEA input converter generate appropriate values of inputs and outputs for each DMU/CP through aggregation functions of their PIs. Thus, database and request data are converted to a format that DEA can work to properly calculate the efficiency of each CP (MORAES et al., 2018). Each CP has a constant pre-defined number of inputs and

outputs regardless of the database and request sizes. Therefore, regarding the CPS problem, each CP has two inputs and two outputs identified:

- **Inputs (criteria to be minimized):**

1. **Resources (“Res.”):** Is the weighted average of the normalized values present in the database. The used weights are the PI’s weight present in the customer request. Resources are basically inversely proportional to the sum of all the gross resources (represented by each interest PI) of each CP.
2. **Prices (“Pric.”):** It is a ratio representing the CP price divided by the price of the most expensive CP in database.

- **Outputs (criteria to be maximized):**

1. **Suitability (“Suit.”):** Is the weighted average of the attending condition of each CP PI, according to the customer request (except price). Basically, it indicates how appropriate the CP is to the request.
2. **Leftovers (“Left.”):** It is the weighted average of all the resources that had left in the CP, after attending the request (for quantitative interest PIs HB and LB, only). That is, if the CP offers more resources than the request asks for.

Moreover, in order to convert PI’s values into the identified suitable variables it is necessary to take into consideration the normalization aspect. To accomplish that, formulation used take into consideration maximum or minimum values of a particular PI comprising all the CP being analysed. Equations 4.5 and 4.8 show how the inputs “Resources” and “Prices” are converted for the provider  $CP_i$ , respectively, and Equations 4.9 and 4.10 calculate the outputs “Suitability” and “Leftovers”, respectively. All these inputs and outputs are normalized between 0 and 1. It is important to note that  $n$  is the total number of CP in database and  $r$  is the total number of PIs in request.



$$Rqt = \begin{cases} 1 - \frac{x_{ij}}{\text{Max}(x_{1j}, \dots, x_{nj}, X_j)} & , \text{ if } \text{type}(PI_j) = HB \\ \frac{x_{ij}}{\text{Max}(x_{1j}, \dots, x_{nj}, X_j)} & , \text{ if } \text{type}(PI_j) = LB \\ \frac{|x_{ij} - X_j|}{\text{Max}(x_{1j}, \dots, x_{nj}, X_j) - \text{Min}(x_{1j}, \dots, x_{nj}, X_j)} & , \text{ if } \text{type}(PI_j) = NB \end{cases} \quad (4.5)$$

$$Rql = \begin{cases} 0, \text{ if } \text{lev}(x_{ij}) = \text{lev}(X_j) \\ 1 - C * \frac{K_1 - |\text{lev}(x_{ij}) - \text{lev}(X_j)| + 1}{K_1}, \text{ if } \text{lev}(x_{ij}) > \text{lev}(X_j) \text{ and } \text{type}(PI_j) = HT \\ 1 - C * \frac{K_2 - |\text{lev}(x_{ij}) - \text{lev}(X_j)| + 1}{K_2}, \text{ if } \text{lev}(x_{ij}) < \text{lev}(X_j) \text{ and } \text{type}(PI_j) = LT \\ 1 - C * \frac{\text{Max}(K_1, K_2) - |\text{lev}(x_{ij}) - \text{lev}(X_j)| + 1}{\text{Max}(K_1, K_2)}, \text{ if } \text{type}(PI_j) = HLT \\ 1, \text{ otherwise} \end{cases} \quad (4.6)$$

$$\text{Res}(CP_i) = \left( \frac{\sum_{j=1}^r w_j * (Rqt_{[PI_j^i \in Qt]} | Rql_{[PI_j^i \in Ql]})}{\sum_{j=1}^r w_j} \right)^\alpha \quad (4.7)$$

$$\text{Prices}(CP_i) = \left( \frac{y_i}{\text{Max}(y_1, y_2, \dots, y_n)} \right)^\beta \quad (4.8)$$

$$\text{Suitability}(CP_i) = \left( \frac{\sum_{j=1}^r w_j * \text{Attend}(\dots)}{\sum_{j=1}^r w_j} \right)^\gamma \quad (4.9)$$

$$\text{Left}(CP_i) = \left( \frac{\sum_{j=1}^{r_{HB}+r_{LB}} w_j * \begin{cases} \frac{x_{ij}-X_j}{\text{Max}(x_{1j}, \dots, x_{nj}, X_j)-X_j}, \text{ if } \text{condition}_1 \\ 1 - \frac{x_{ij}-X_j}{\text{Max}(x_{1j}, \dots, x_{nj}, X_j)-X_j}, \text{ if } \text{condition}_2 \\ 0, \text{ otherwise} \end{cases}}{\sum_{j=1}^{r_{HB}+r_{LB}} w_j} \right)^\delta \quad (4.10)$$

Where  $i = 1, 2, \dots, n; j = 1, 2, \dots, r; \alpha, \beta, \gamma, \delta = 1, 2, 3, \dots; r_{HB}, r_{LB}, r_{NB} = 0, 1, 2, \dots$ ; representing the a amount of PIs with each one of those utility functions, subject to  $r_{HB} + r_{LB} + r_{NB} = r$ ; function  $\text{Attend}(\dots)$  is Equation 4.1 if  $PI_j$  is quantitative or Equation 4.2 if  $PI_j$  is qualitative; “Qt” and “Ql” are the sets of quantitative and qualitative PIs, respectively. Functions “type” and “lev” are the same ones specified at Subsection 4.2.1. Moreover,  $\text{condition}_1 = \text{AttendQT}(PI_j, x_{ij}, X_j, 0)$  and  $\text{type}(PI_j) = HB$ ;  $\text{condition}_2 = \text{AttendQT}(PI_j, x_{ij}, X_j, 0)$  and  $\text{type}(PI_j) = LB$ . In addition, in case

$X_j \geq \text{Max}(x_{1j}, \dots, x_{nj})$  happens, then, the “Leftovers” value for that  $PI_j$  is always zero  $\forall P_i, x_{ij}$ . Also, if a single PI specified as essential at the customer request is not attended by a CP, its “Suitability” value is always 0.

It is important to emphasize that qualitative PIs only affect “Suitability” and “Resources” equations. The “Resources” values for a qualitative  $PI_j$  of the  $CP_i$  ( $R_{ql}$ ) are scored similarly to how qualitative PIs are scored at CPS-Matching, but inversely proportional, because “Resources” is a criterion to be minimized. Variables  $K_1, K_2$  are the total number of tolerable categories higher and lower the desirable one ( $X_j$ ), respectively, and  $C$  is a constant value at the open interval  $]0, 1[$ , equivalent to constant  $C_3$  at CPS-Matching method to be compared.

The variable factors  $\alpha, \beta, \gamma, \delta$  are responsible for transforming all the inputs/outputs “Resources”, “Prices”, “Suitability” and “Leftovers”, respectively, into functions with a variation (increase/decrease), e.g., linear (1) is default, quadratic (2), cubic (3), etc. That is, the higher the value of such factors, the more significantly numerically these inputs/outputs will be affected for each change made. The higher the factor, the greater the loss of punctuation for every CP that does not reach 1 (efficient), since the numbers are always between 0 and 1. The farther from 1 and closer to 0, the more score the CPs will lose. For more critical input/output, it is appropriate to increase its associated factor. Bearing this in mind, the considered most critical feature is “Suitability”. Therefore, its factor will be 2 ( $\gamma = 2$ ), and for the others will be 1 (linear variation).

At the end of this first stage, an efficiency frontier is generated. This efficiency frontier frames the set of CPs that compared with others are 100% efficient, so any CP outside this frontier will have a lower value of efficiency. The second stage start after these inputs and outputs calculation for each candidate CP. Thus, it is possible to use this information as an input file of a program that implements the DEA method, such as Integrated System for Decision Support (ISYDS) (MEZA et al., 2005).

ISYDS is capable of dealing with 150 DMUs, 20 variables (inputs and outputs), and it works with six decimals accuracy (MEZA et al., 2005). ISYDS implements the two classic DEA models: CRS or CCR and VRS or BCC (explained in Subsection 2.3.1.2). In addition to the classical models implemented (CCR and BCC), user can choose between input or output orientation (Subsection 2.3.1.2). It solves DEA LP equations using Simplex algorithm using the multiplier model (MEZA et al., 2005).

For the CPS problem, the DEA output orientation (tries to maximize outputs) seems to be more appropriate because the “Suitability” is an output and it is the most important characteristic to be observed to select CPs. Comprising the DEA model to be chosen, VRS seems more realistic for the scope of the problem, since the variations of inputs and output are not proportional, especially taking into account that “Suitability”

and “Leftovers” (outputs) are largely dependent of customer request and the database, while “Resources” and “Prices” (inputs) are practically only dependent on the database. If the request keeps unchanged and database changes, i.e., the CP conditions improve, the inputs, most likely, will change as well, but “Suitability” can keep constant.

The third and final stage, uses each CP’s inputs and outputs values for ranking all the CPs in the DEA efficiency frontier. This stage is necessary because DEA method frequently returns more than one CP as efficient, composing the efficiency frontier. Thus, it would be more appropriate, for the customer decision-making (who wants the best CP and not a large set of them), the return of a ranking of these chosen CPs, whose efficiency is given by DEA.

Thus, each CP that belongs to the efficiency frontier will be ranked first by their “Suitability” value, followed by the value of “Prices”, then “Leftovers” and finally by “Resources” value in a non-compensatory way (*Suitability* > *Prices* > *Leftovers* > *Resources*, always). The CP with the highest value of “Suitability” will be the top of ranking. In case of a tie, CPs with lowest “Prices” will be first in the ranking. If two or more CPs have the same “Suitability” and “Prices”, then the highest value of “Leftovers” will be considered as a tiebreaker, and, for last case, the lowest value of “Resources” will be used. If one or more CPs tie in the four features, they will have the same rank number, sorted alphabetically. The final rank is immediately returned to the customer, with the CP name, suitability value (in %), estimated price and other available information as CP’s website, for example.

#### 4.2.3 Metaheuristic Methods Modelling

The metaheuristic approaches used in this work are Simulated Annealing (SA) and the Evolutionary Algorithms (EAs). SA is a thermodynamic-inspired algorithm of an iterative single-solution improvement based on the metal annealing process. EA is a bio-inspired technique that uses concepts like populations of individuals, fitness, variability, heredity, genetic operations and selection routines. The answers obtained by these methods are a non-empty CP set, where together they can maximizes request attendance with the smallest possible CP set and with the lowest possible price. This fact makes the CPS problem much more complex because it creates a combination of possible different solution sets. Therefore, the complexity for  $n$  CPs is  $2^n$  (search space grows exponentially), characterizing a complex optimization problem, justifying the use of metaheuristics.

Metaheuristics are probabilistic and must be executed several times for more reliable answers. Actually, there is no guarantee that a metaheuristic will find a suitable solution, but exhaustive experiments can minimize this problem as well as the perform of an appropriate statistical test on these experiments for better statistical con-

fidence. This section presents the individual (Subsection 4.2.3.1) and fitness modelling (Subsection 4.2.3.2), as well as the energy modelling (Subsection 4.2.3.3) to the CPS problem. This modelling is essential for the correct use of the SA (energy based) and three EAs (fitness) methods presented in Subsection 2.3.2: GA, BDE and DDE.

In addition to presented methods, an exhaustive fitness search can also be performed. The exhaustive fitness search method searches for the best fitness in all possible combinations of CPs solution sets using the fitness function described on Subsection 4.2.3.2. The fitness calculation uses the customer request and it is performed for each possible combination with all available CPs in the database. All CPs that possess the highest possible fitness value are returned as response to the customer. The exhaustive search is an exact method, always finding the optimal answer to the problem with a single execution, but possesses exponential complexity time of order  $2^n$ , where  $n$  is the number of CPs in database, making it feasible only for small databases. It was implemented to compare and justify the need of the other methods.

#### 4.2.3.1 Individual Modelling

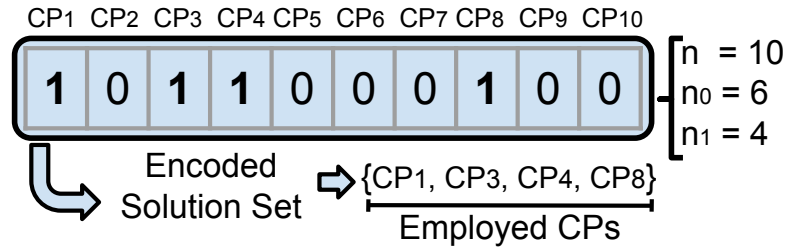
The individual modelling is strictly related to the problem being solved and it is where the encoding type is specified. The individual modelling is used by EAs. As already stated, the individual represents a candidate solution to the problem. So, for the proposed CPS problem modelling, the individual is coded as a vector using a binary encoding, where the number of coded variables is equal to the total number of CPs registered in the database. Each variable uses a position in the binary vector and it is represented by a single bit, indicating whether that CP belongs to the solution set (bit 1) or not (bit 0) (MORAES; FIORESE; PARPINELLI, 2017). All CPs with bit 1 will be called “employed CPs” for that individual, i.e., CP that will be effectively used in the solution set. This information is used in the fitness function to generate a fitness value to each individual.

Figure 17 illustrates this modelling with an example comprising 10 hypothetical CPs and the resulting solution set that this individual encodes. Thus, in this case, the employed CPs are P1, P3, P4 and P8.

Thus, let be  $I = (bit(0|1)_{CP_1}, bit(0|1)_{CP_2}, \dots, bit(0|1)_{CP_n})$  the tuple representing an individual as a single binary vector with  $n$  bits, where bit 1 in the position  $i$  of that vector indicates that CP  $P_i$  belongs to the solution set coded by that individual. Similarly, bit 0 informs that the  $i$ th CP in  $P$  does not belong to the solution set. So, the search space is discrete, multi-modal (multiple optimal) and its size is  $2^n - 1$ , since the whole code filled with 0 is not a valid solution (it indicates that none CP is part of the solution).

Thus, the CPS problem can be reduced to a specific instance of the clas-

Figure 17 – Example of a possible solution encoded in a binary vector



Source: Adapted from Moraes, Fiorese e Parpinelli (2018a)

sic combinatorial optimization problem called Knapsack Problem (MARTELLO; TOTH, 1990), that consists of organizing  $n$  items of different weights and profits inside a backpack, aiming at maximizing the profit resulting from the  $n$  items loaded and respecting the maximum capacity of the backpack. Each candidate CP is an item to the backpack. The weight of each item is influenced by the price of that item and the amount of other items already in the backpack. The profit of each item is directly influenced by the maximum number of PIs of each request that item (CP) is able to attend. Thus, it is need to maximize that profit with the minimal possible weight.

#### 4.2.3.2 Fitness Modelling

This subsection aims to specify the fitness function used by the EAs (GA, BDE and DDE) presented in Subsection 2.3.2. The first version of this fitness function was proposed by Moraes, Fiorese e Parpinelli (2017), but it can be improved to handle cases where some PIs need to belong to the same CP (MORAES; FIORESE; PARPINELLI, 2018a). This is done using the PI's grouping approach (Subsection 4.1.1).

The fitness of the individual is proportionally linked to the minimization of three factors: i) the total amount of employed CPs (encoded with bit 1) in the individual codification vector; ii) the total price of the individual; and iii) the difference between the requested PI values (customer request) and the PI values from the individual's employed CPs. Thus, if the objective function of the problem is the combination of these factors, the fitness function comprises the minimization of this function, plus the application of penalties if the individual is invalid (does not attend 100% of the requested PI values). Thus, Equation 4.11 presents the fitness function of the  $i$ th individual of the population ( $Ind$ ), where  $w_{pen}$  is a constant value for the penalty weight applied to  $Ind$ , whose default value is 1.

$$Fitness(Ind) = 1 - Obj(Ind) - w_{pen} * penalty(Ind) \quad (4.11)$$

Equation 4.12 presents the objective function ( $Obj$ ) to  $Ind$ . This objective func-

tion is the weighted average of the scores corresponding to the number of employed CPs ( $ecp$ ) and the total price ( $price$ ) of  $Ind$ , both normalized between 0 and 1. The constants weight values to the amount of CPs ( $w_n$ ) and to price value ( $w_p$ ) ponder the importance of the least amount of CPs and the lowest price desired for the final solution, respectively.

$$Obj(Ind) = \frac{w_n * ecp(Ind) + w_p * price(Ind)}{w_n + w_p} \quad (4.12)$$

So, let  $P = \{CP_1, CP_2, \dots, CP_n\}$  the set containing the  $n$  different candidate CPs available for selection in the database. Therefore,  $Ind$  has an encoding vector of size  $n$ . Thus, if  $n0_{Ind}$  represents the count of all bits 0 of the encoding vector of  $Ind$  and  $n1_{Ind}$  is the count of all bits 1, the score given to  $Ind$  according to its number of employed CPs is according to Equation 4.13.

$$ecp(Ind) = \begin{cases} 1, & \text{if } n1_{Ind} = 0 \\ \frac{|n1_{Ind} - minCP|}{n - minCP}, & \text{otherwise} \end{cases} \quad (4.13)$$

Where  $minCP$  is a constant integer that represents the minimal possible number of CPs estimated to maximize the attendance for the current request. Default is  $minCP = 1$ . Note that the extreme cases (encoding vector filled with all bits 0 or bits 1) are penalized to the maximum (value 1), because they are totally undesired to the CPS problem solution. The minimal CP amount ( $n1_{Ind} = minCP$ ) would be the ideal quantity.

Now, let  $Y = (y_1, y_2, y_3, \dots, y_n)$  be the respective prices associated with each of the  $n$  different CPs of  $P$ . The total price ( $tprice$ ) of  $Ind$  is the sum of the prices of all employed CPs encoded by  $Ind$ , i.e.,  $tprice_{Ind} = \sum_{k=1}^n y_k [Ind_k = \text{bit}(1)]$ , where  $Ind_k$  represents the  $k$ th bit of  $Ind$ 's encoding vector. Thus, if a CP does not belong (bit 0, i.e., non-employed CP) to its encoded solution, its price will be zero. In order to be usable to the fitness function (Equation 4.11), the total price of  $Ind$  must be normalized according Equation 4.14, where is the sum of all prices of  $Y$ , i.e.,  $maxPrice = \sum_{j=1}^n y_j$ , with  $j = 1, 2, \dots, n$ .

$$price(Ind) = \frac{|tprice_{Ind} - minPrice|}{maxPrice - minPrice} \quad (4.14)$$

The constant floating point number  $minPrice$  represents the minimal possible price value estimated to maximize the attendance of the current request. It is basically the total sum of the  $minCP$  lower prices in the database. Default is  $minPrice = min(Y)$ . Price Zero would be resulting of a CP that makes available its resources for free.

Penalties are applied to the fitness computation, according to Equation 4.11. Penalties can be applied to each invalid individual that can be generated in the EAs. These penalties are a way to handle constrained optimization problems by significantly decreasing the fitness value of the individual if it presents an inadequate solution to the problem (YENIAY, 2005). This work uses static penalties for undesirable solutions (individuals). Penalty is calculated proportionally as a function of how much the solution infringes the problem constraint. For the CPS problem there are two constraints (MORAES; FIORESE; PARPINELLI, 2017): the individual must not be all encoded with bit 0; and the individual employed CPs has to maximize attendance of all customer requested PIs. The first constraint prevents functions that generate the initial random population and the genetic operators (modify individual's codification vector) of generating an encoding vector filled with bit 0 for any individual. The second constraint will proportionally penalize every individual that does not attend all PIs of the customer request. So, Equation 4.15 shows penalty calculation where the ordered list  $w$  contains the weights of each  $PI_j$  and  $r$  is the number of PIs from the customer request. This penalty is normalized between 0 and 1.

$$\begin{aligned}
 Attend(PI_{ij}) &= \begin{cases} 0, & \text{if } AttendQT \text{ OR } AttendQL \\ 1, & \text{otherwise} \end{cases} \\
 pen(PI_{ij}) &= \begin{cases} w_j * Attend(PI_{ij}), & \text{if } groupID(PI_{ij}) = 0 \\ \frac{\sum w_j * Attend(PI_{ij})}{\sum w_{group(PI_{ij})}}, & \text{otherwise} \end{cases} \\
 penalty(Ind) &= \frac{\sum_{j=1}^r pen(PI_{ij})}{\sum_{j=1}^r w_j} \quad (4.15)
 \end{aligned}$$

The  $AttendQT$  function is Equation 4.1 (if PI is quantitative) and  $AttendQL$  function is Equation 4.2 (for qualitative PIs). A PI of the request is fully attended if at least one CP encoded in the individual attend it. Another point to be emphasized is that if those PIs are arranged into groups with sizes greater than 1, then, that group is fully attended if at least one CP encoded in the individual attends all the PIs on that group, otherwise, this individual will be proportionally penalized to how many PIs on that group are not attended by the encoded CP that better attend that group. If there is a single essential PI (specified at the request) that is not fully attended by a CP, the penalty applied for that individual is 1 (maximum).

The fitness calculation is applied to every individual in the population for each algorithm generation, up to a predefined number of generations. At the end of the generations, the best fitness individual is sought and returned as answer. If it has not been penalized, the method return the CPs encoded by the individual as the response to the problem, otherwise an error returns.

#### 4.2.3.3 Energy Modelling

The concept of energy is used by the SA method and has the same purpose of the fitness: to evaluate how good each generated solution is for the problem solution. But instead of maximizing its value, the SA tries to minimize it, by definition (TALBI, 2009). Each SA solution is encoded exactly the same as the individual modelling presented at Subsection 4.2.3.1. So, the function to generate a new neighbour candidate of the current solution is a simple bit-flip in the binary encoded vector (without generating the entire vector filled with zeros). Finally, Equation 4.16 presents the energy function for a solution  $s$ .

$$Energy(s) = \frac{\frac{w_n * ecp(s) + w_p * price(s)}{w_n + w_p} + w_{pen} * penalty(s)}{1 + w_{pen}} \quad (4.16)$$

Basically, it is the fitness function modelling presented at Subsection 4.2.3.2. However, to the fitness it is desired the highest values, to the energy it is desired the lowest ones. The fitness was normalized between -1 and 1. But energy is normalized between 0 and 1, because SA uses only positive energy values. The factor *ecp* is calculated by Equation 4.13, *price* is calculated by Equation 4.14 and *penalty* by Equation 4.15.

#### 4.2.4 Hybrid Methods Modelling

Hybrid methods are a combination of one or more deterministic methods with one or more metaheuristic methods. A trivial combination is the use of the simplest deterministic method (CPS-Matching) with one metaheuristic method (e.g., CPS-GA), in pipeline (MORAES; FIORESE; PARPINELLI, 2018a). Thus, they combine the qualities of deterministic (quick answers - Single CP) and metaheuristic (complex answers - CP set) approaches. For the hybrid methods, the customer must choose the minimum desired percentage of suitability for its request (until 100%). The CPS-Matching is firstly applied on the initial CP list and the suitability of highest-rated CPs are calculated. If the CP with better score has equal or greater suitability (given by CPS-Matching) than the desired value, the method returns that single CP to the customer. However, as result, if there are more than one CP with equal suitability value, the lowest price CP is returned. If the suitability of all CPs is lesser than the desired, then a metaheuristic algorithm is started. The final CP set with the highest fitness (EAs) or lowest energy (SAs) found in all executions is returned to the customer as final solution to its request. The four hybrid methods employed in this work are: Matching-GA (CPS-Matching plus CPS-GA), Matching-BDE (CPS-Matching plus CPS-BDE), Matching-DDE (CPS-Matching plus CPS-DDE) and Matching-SA (CPS-Matching plus CPS-SA).



### 4.3 PARTIAL CONSIDERATIONS

This chapter presented the concepts related to the proposed PI-based architecture for solving the CPS problem. This architecture is formed by the proposed scenario and its selection methods modelling. This architecture can handle several selection methods modelling for the CPS problem (both deterministic, metaheuristic and hybrid), being easily improved and expanded in the future.

The proposed modelling copes with any kind of PI-based CPS. Thus, given a finite initial non-empty set with different CPs, where each CP has distinct PIs associated, plus price, and given a customer request, the problem is to choose the smallest subset of CP, in order to maximize the request attendance with the lowest price involved. The attendance condition is used by all the selection methods addressed here, and basically, indicates if a PI value (quantitative or qualitative) from a determined CP, attend the customer desired value, i.e., if this value is the desired one or it is better than the desired one or, at least, it is in the tolerance range from the desired one.

The proposed selection methods are useful mathematical and/or nature-inspired techniques, that assist the customer's decision-making process. The selection methods are divided into three categories: deterministic, metaheuristic and hybrid. Methods proposed load a list with all the CPs candidates from the database with their PIs and prices, along with a list with all customer's PIs of interest. Thus, a selection method is applied on the initial CP list and can return two kind of responses: a CP ranking (deterministic method), or a CP set (metaheuristic or hybrid method). Both responses can be suitable enough and can be returned to the customer as final solution for his request.

The deterministic methods solve a simplified instance of the CPS problem, where a single CP (top ranked CP) is enough for the customer request. This evaluation assesses each CP separately. Three methods are specified: CPS-Matching, CPS-DEA and exhaustive fitness searching. The CPS-Matching method is a mathematical algorithm that can score and rank an extensive CPs list based on value, type, nature and importance of each requested PI. The method is divided into three main stages. All PIs with same weight are basically at the same importance level and the score of each importance level is the simple arithmetic average of the particular scores of each PI within that level. Finally, the final score is the weighted average of the score of each importance level available for that CP. The score ranges from 0 to 1. The closer to 1, the more adequate is that CP to satisfy that request. The CPS-DEA method is a mathematical model for selecting and ranking CPs using the MCDM called DEA. In this case, each CPs is called DMU and their PIs can be transformed/aggregated to be an input or an output. DEA uses inputs and outputs criteria to calculate efficiency and use it to selects DMUs. DMUs with max efficiency form a DMU set called efficiency

frontier. In this work, the input criteria identified are “Resources” and “Prices” to be minimized. The outputs criteria are “Suitability” and “Leftovers”. All these criteria are normalized between 0 and 1. The model is divided into three main stages: DEA Input Converter, DEA Method and Ranking Routine. The final ranking also use inputs and outputs. The most important criteria is “Suitability”, followed by “Prices”, “Leftovers” and “Resources”. The exhaustive fitness search method looks for the highest fitness in all possible combinations of CPs solution sets using the proposed fitness function. This method is exact, but possesses exponential complexity time, making it feasible only for small databases. It was implemented to compare and justify the need of the other methods.

The metaheuristic methods used are the EAs and SA. The answer obtained by these methods (i.e., GA, BDE and DDE) is a non-empty CP set that can maximize the attendance of all customer requested PIs. The correct use of EAs demands an individual and its fitness function modelling. In this case, the individual is modelled using binary encoding, where the number of coded variables is equal to the total number of CP in the database. Each variable is represented by a single bit, indicating whether that CP belongs to the solution set (bit 1) or not (bit 0). The search space is discrete, multimodal and its size is  $2^n - 1$ . The total price of the individual is equal to the sum of the prices of all its encoded CPs. The fitness of the individual is proportional to the minimization of three factors: distance from the CPs’ PI values encoded in the individual to the request PI values; the total price of the individual; and the amount of CPs encoded by the individual. Thus, if the objective function of the problem is the combination of these components, the fitness function is the minimization of this objective function, plus the application of penalties if the individual is invalid, i.e., it does not fully attend the request. For the use of SA algorithm an energy function was specified. The encoded solution is the same of the individual encoding vector and the energy is basically the minimization of the fitness function, normalized between 0 and 1. The function to generate a new neighbour candidate of the current solution is a simple bit-flip in the encoding vector.

The hybrid methods combine the CPS-Matching method with one metaheuristic method in pipeline. For these methods, the customer must choose the minimum desired percentage of suitability for its request. CPS-Matching is firstly applied on the initial CP list. If the CP with better score has equal or greater suitability than the desired value, the method returns that single CP to the customer. If the suitability of all CPs is lesser than the desired, then the metaheuristic algorithm is started. The final CP set with the highest fitness found in all executions is returned to the customer. The four hybrid methods employed in this work are: Matching-GA, Matching-BDE, Matching-DDE and Matching-SA.

## 5 EXPERIMENTS AND RESULTS

After elaborating on the selection architecture and its methods, it is fundamental to expose some examples of feasible scenarios involving a possible CP selection with simulated and/or real data in order to show the architecture application, operation and results. Thus, this chapter describes the experimentation protocol and algorithms' configuration, problem instances and the customer's requests as well as the results obtained followed by their analysis. All algorithms were implemented in Java (JDK 1.8), in 64 bits Windows 10 OS, and they are executed in a host with 8 gigabytes of memory RAM and an Intel Core i5, 3.0 Ghz CPU. Main CP selection methods evaluated are: CPS-Matching ("Mtc"), CPS-DEA ("DEA"), CPS-GA ("GA"), CPS-BDE ("BDE"), CPS-DDE ("DDE") and CPS-SA ("SA") algorithms. The hybrid approaches evaluated are: Matching-GA ("Mtc-GA"), Matching-BDE ("Mtc-BDE"), Matching-DDE ("Mtc-DDE") and Matching-SA ("Mtc-SA"). The exhaustive fitness search was also implemented, but just to justify the need of other methods for larger databases instances. The source code, the CPs database and the requests used are available at the Github repository <sup>1</sup>.

Three use case examples are presented in this chapter. The first two cases use simulated databases and the last one use real data, i.e., real criteria/PIs and real CC providers/services that actually exist in the real world. The simulated databases are simple, feasible and controlled, where the global optima for each request is previously known. These databases are essential to validate the modelling and to adjust the parameters of each proposed method. Thus, any major problem with these models can be identified and corrected in advance, before applying them to real and more complex data, where the ideal optima can be hard to determinate/arbitrate. Thus, initially, Section 5.1 presents the main control parameters configuration, the search space exploration and selection routines used by the metaheuristic methods as well as the naming the statistical tests used to evaluate results. The next three sections describe each use case developed and experimented along with their results and result analyses.

### 5.1 EXPERIMENTS SETUP

The minimal suitability value required for the hybrid methods ("Mtc-GA", "Mtc-BDE", "Mtc-DDE" and "Mtc-SA") used in experiments is 100%. The constants used by the CPS-Matching method, and its hybrid counterparts too, are:  $C_1 = 0.999$  and  $C_2 = 0.001$ , so, prioritizing much more the attendance of the minimum necessary (see

<sup>1</sup> Accessible at link: <https://github.com/LBMbr/SelectionMethodsProject/tree/master>.

Subsection 4.2.2.1). Moreover, constants used for qualitative PIs evaluation are  $C = C_3 = 0.7$ , on CPS-Matching and CPS-DEA methods, respectively. Regarding CPS-DEA method, factors used are:  $\alpha, \beta, \delta = 1$  and  $\gamma = 2$ , VRS model with output orientation (Subsection 4.2.2.2). Moreover, the weights for price ( $w_p$ ), quantity of CPs ( $w_n$ ) and penalty ( $w_{pen}$ ) on the fitness (Equation 4.11) and energy function (Equation 4.16) have the same importance and will be constant and unitary ( $w_p = w_n = w_{pen} = 1$ ).

Table 9 presents main parameters and their values, which are defined empirically, applied to the EA methods/algorithms: CPS-GA, CPS-BDE, CPS-DDE and their hybrid counterparts. These parameters were already discussed when these methods were presented in Subsections 2.3.2.2, 2.3.2.3 and 2.3.2.4 and represent the main control parameters of these algorithms. The stop condition of these EAs are 2.000 generations (number of iterations –  $IT$ ). Size of the population ( $POP$ ) is 50 individuals, thus generating a total of 100.000 fitness evaluations ( $POP * IT$ ). The number of dimensions/variables ( $DIM$ ) is always equal to the number of CPs ( $n$ ) registered in the CP database for all the algorithms, including SA and Matching-SA. A total of 30 executions were performed for each problem instance (i.e., database plus customer request) for each EA and also SA.

Table 9 – Parameters used by GA, BDE and DDE algorithms

Parameter	Acronym	GA	BDE	DDE
Population size	$POP$	50	50	50
Number of generations	$IT$	2000	2000	2000
Crossover Rate	$CR$	95%	–	–
Mutation Rate	$MR$	1%	5%	–
Probability of perturbation	$PR$	–	50%	50%
Stochastic tournament size	$k$	5	–	–
Mutation weighting factor	$F$	–	–	0.5

Source: Author

For the SA algorithm, the starting temperature ( $T0$ ) is 1 and final ( $Tf$ ) is 0. In that way, both the temperature and the energy variation ( $\Delta E$ ) values, used in acceptance probability given by Equation 2.4, are always normalized between 0 and 1. The Metropolis' constant ( $MC$ ) is 500. The temperature decay (cooling) function for each iteration  $i$  is:  $TD(i) = T0 + \frac{A}{i+1} - A$ , where  $A = \frac{(T0-Tf)*(N+1)}{N}$  and  $N$  is the total amount of temperature decreases. This cooling function is chosen because it drastically lowers the initial temperature, quickly increasing the process of intensification of local search, which is interesting to this problem. The low mutation rate in GA and BDE algorithms does exactly the same. Finally, to be fair to the EAs, that performed 100.000 fitness evaluations, if  $MC = 500$ , then  $N = 200$ , so the total energy evaluation ( $MC * N$ ) performed are 100.000 too. Both the constant  $MC$  and the function  $TD(i)$  were obtained empirically. The constants  $T0$ ,  $Tf$ ,  $MC$  and the function  $TD(i)$  are the SA's control

parameters, see Subsection 2.3.2.1.

The selection routine used by GA is called stochastic tournament and presents a parameter  $k$  that represents the size of the group that will compete in each GA iteration. To the experiments,  $k$  is the constant value of 5. BDE and DDE uses greedy selection. Greedy selection is applied between two individuals (parent and offspring). This selection always choose the individual with the highest fitness.

GA uses the one-cut-point crossover and bit-flip mutation (Subsection 2.3.2.2). BDE has its own specific crossover (Subsection 2.3.2.3) as well as bit-flip mutation too. DDE uses only its own specific mutation routine according Equation 2.5, with a mutation weighting factor of 0.5. An elitism routine is used only by GA. It is applied after the fitness evaluation of the new population generated. This routine replaces the individual with the lower fitness on the current generation by the one with the best fitness of the previous generation.

To perform a statistical analysis of the results obtained in each case, the Shapiro-Wilk (WILK; SHAPIRO, 1968) normality test is employed with 5% significance level, i.e.,  $\alpha = 0.05$ . Based on the results obtained by the Shapiro-Wilk normality test, the null-hypothesis of normality ( $H_0$ : The data follow a normal distribution) was rejected with  $p$ -values smaller than 0.05 for all use cases. Therefore, the data do not follow the normal distribution. Hence, the non-parametric Dunn's test is employed also with 5% significance level, i.e.,  $\alpha = 0.05$  (DUNN, 1964). Due to the multiple comparisons involved, the Bonferroni  $\alpha$  correction is used.

## 5.2 USE CASE – QUANTITATIVE PIs

The first controlled scenario uses only quantitative PIs that are characteristic of the IaaS model. It was initially proposed in Moraes, Fiorese e Parpinelli (2018a).

Table 10 informs the basic CP candidates database used in these experiments. This database is simulated and involves ten CPs and six quantitative PIs that could occur in actual scenarios. The used PIs are: total amount of "RAM" available (Gb); maximum "HD Memory" for storage (Gb); maximum amount of "CPU Power" (e.g., CPU frequency times the number of CPU cores) usable; average CP resources "Availability" accessible via the Internet (% of accessible resources per year); estimated CP level of information "Security" (levels 1-5); and, final "Price" of all that resources available for each CP (US\$ per month of use). Each PI has one type of associated behaviour (HB, LB or NB), one group id ("ID" zero means that PI is not involved in any group) and one value for each CP. PI "Security" was handled as a quantitative NB. PIs "RAM", "HD Memory" and "CPU Power" form a VM specification and must be all attended by a same CP.

Table 10 – E.g.1: Simulated CP database with their quantitative PIs

<b>CPs/PIs</b>	<b>RAM</b>	<b>HD Memory</b>	<b>CPU Power</b>	<b>Availability</b>	<b>Security</b>	<b>Price</b>
<b>Type</b>	<b>HB</b>	<b>HB</b>	<b>HB</b>	<b>HB</b>	<b>NB</b>	<b>LB</b>
<b>ID</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>
CP1	1	5	2.0	99.99	1	0.50
CP2	2	20	2.5	95.2	2	1.25
CP3	4	40	3.5	98.3	2	2.50
CP4	8	25	4.0	96.8	3	3.80
CP5	16	50	5.5	97.9	3	4.50
CP6	32	100	6.0	98.9	5	7.50
CP7	64	200	10.0	99.3	4	10.0
CP8	32	150	8.0	98.5	4	8.25
CP9	16	20	7.0	96.4	3	7.80
CP10	8	10	5.0	97.5	4	6.50

Source: Adapted from Moraes, Fiorese e Parpinelli (2018a)

On the other hand, Table 11 informs eight possible cloud customer requests considering the CP database shown in Table 10. Note that not necessarily all database stored PIs are needed to the customer. Request 4, for example, ignores PI “Security”, that is, its value does not matter to the customer and it can assume any value. In the same way, request 5 only cares about the main resources to form a customer virtual machine, regardless of their “Availability” and “Security” level. Request 6 does not care much about CPU processing and Request 7 just concerns about storage with medium “Availability” and “Security”. Finally, customer of request 8 just needs that its service has the highest possible “Availability” and “Security”, an uncommon need, but possible.

Price is a special PI, which is not stated explicitly in the customer request though it is always taken into account in the fitness/energy function and in CPS-DEA method. However, in the canonical version of the CPS-Matching method (MORAES; FIORESE; MATOS, 2017), price must be explicitly specified in each request, otherwise, it will be ignored. Therefore, for the sake of comparison, to the experiments with the CPS-Matching, the score associated with the price of a CP will be:  $1 - \frac{price_{CP}}{MaxPrice}$ , i.e., inversely proportional to the ratio of its price to the maximum price at the database. This is similar to what is done in CPS-DEA (Equation 4.8).

In this example, each PI has the same importance weight (1.0) as all others. Last line of Table 11 presents the known answers (optimal) for each request. The easiest requests are 1, 5, 6 and 7, because a single CP, of the 10, is already able to attend them completely. The most complex request to be answered in this scenario is the third request since the answer demands the highest values of resources and criteria levels being composed by three CPs.

Note that request 4 can be fully attended by CP7 on its own, but the final price (10) is much higher than the set CP1 & CP2, which together also fully attend request

Table 11 – E.g.1: Simulated customer requests

<b>Pls/Reqs</b>	<b>Req. 1</b>	<b>Req. 2</b>	<b>Req. 3</b>	<b>Req. 4</b>
RAM	8	16	64	2
HD Memory	20	50	200	20
CPU Power	4.0	8.0	10.0	2.0
Availability	98.0	98.0	99.9	99.0
Security	4	5	5	–
<b>Optimal</b>	CP8	CP6 & CP8	CP1 & CP6 & CP7	CP1 & CP2
<b>Pls/Reqs</b>	<b>Req. 5</b>	<b>Req. 6</b>	<b>Req. 7</b>	<b>Req. 8</b>
RAM	16	4	–	–
HD Memory	100	40	50	–
CPU Power	5.0	–	–	–
Availability	–	98.0	97.0	99.99
Security	–	2	3	5
<b>Optimal</b>	CP6	CP3	CP5	CP1 & CP6

Source: Adapted from Moraes, Fiorese e Parpinelli (2018a)

4, with lower price ( $0.5 + 1.25 = 1.75$ ). Therefore, as the weight for price and quantity of CPs are the same, the set CP1 and CP2 is better than only CP7. For the sake of anticipation, it is important to state that the CPS-Matching and CPS-DEA methods can not identify this possible CP grouping.

### 5.2.1 Results and Analysis – Quantitative Pls

Tests with all methods discussed were performed for 10, 20, 30, 50, 100 and 200 CPs using Table 10 data. To accomplish that, those data were replicated 2, 3, 5, 10 and 20 times, respectively, to make new databases with 20, 30, 50, 100 and 200 CPs (vertical scalability). Thus, data replicated to 20 CPs will generate CP11 with the same original data of the CP1; CP12 tied to CP2, CP13 tied to CP3, and so on for each replication. This data replication was performed because it is a simple way to increase the database size without modifying the known optimum. The requests are the same for all these databases.

It is important to note that the search space of possible solutions containing all possible combinations of  $n$  CPs is equal to  $2^n - 1$ . Thus, the search space for 10, 20, 30, 50, 100 and 200 CPs is 1023, 1048575, 1073741823,  $1.1259 * 10^{15}$ ,  $1.2676 * 10^{30}$  and  $1.6069 * 10^{60}$  combinations, respectively. The exhaustive fitness search method is deterministic and possess the most reliable answers (hits percentage is always 100%), but its execution time is exponential, making it ideal only for small CP databases. The average time to complete a single fitness assessment for 200 CPs was approximately 0.002 milliseconds in the hardware architecture employed.

Thus, the average execution time of the exhaustive search for 10, 20 and 30 CPs for these experiments are approximately 2 milliseconds, 2.1 seconds and 35.8

minutes, respectively. For 50 CPs, the estimated execution time is about 71 years (not parallelized), making its use inappropriate for databases with more than 30 CPs. That is why other methods are needed for larger databases with 50, 100 and 200 CPs. Thus, next result tables will present only the results for these larger databases.

Table 12 presents the results obtained when applying the CPS-Matching (“Mtc”), CPS-DEA (“DEA”), CPS-GA (“GA”), CPS-BDE (“BDE”), CPS-DDE (“DDE”) and CPS-SA (“SA”) methods on the database of Table 10 replicated for 200 CPs, taking as input each one of the eight requisitions. For all methods, the hits percentage (based on the known answer – last line of Table 11) has been calculated. For the metaheuristic methods, average and standard deviation of the fitness (to be maximized “↑”) and energy (for “SA”, to be minimized “↓”) has been calculated as well. The initial “SA” solution and EAs’ population are completely random. The results for databases with less CPs (50 and 100 CPs) were omitted because they all reached 100% hits for all the methods and requests. “Mtc” and “DEA” methods did not use the fitness function (Equation 4.11), thus, they have different answers for each request that can be attended by more than one CP. The top scored CP for each request according to “Mtc” and “DEA” methods are: CP8 (Req. 1), CP6 (Req. 2), CP7 (Req. 3), CP7 (Req. 4), CP6 (Req. 5), CP3 (Req. 6), CP5 (Req. 7), CP1 (Req. 8). From all metaheuristic methods, “SA” appears to be the less efficient. The optimal energy values for that requests who did not get 100% hits in “SA” are: 0.0018 (Req. 1), 0.0035 (Req. 2) and 0.0017 (Req. 5).

Table 12 – E.g.1: Results for 200 CPs database with 5 quantitative PIs plus the price

Request	Hits (%)						Fitness ↑			Energy ↓	
	Mtc	DEA	GA	BDE	DDE	SA	GA	BDE	DDE	SA	SA
<b>Req. 1</b>	100%	100%	100%	100%	100%	63.33%	0.9963 ± 0.0	0.9963 ± 0.0	0.9963 ± 0.0	0.0026 ± 0.0013	0.0026 ± 0.0013
<b>Req. 2</b>	0.0%	0.0%	100%	100%	100%	80%	0.9930 ± 0.0	0.9930 ± 0.0	0.9930 ± 0.0	0.0036 ± 0.0002	0.0036 ± 0.0002
<b>Req. 3</b>	0.0%	0.0%	100%	100%	100%	100%	0.9921 ± 0.0	0.9921 ± 0.0	0.9921 ± 0.0	0.0039 ± 0.0	0.0039 ± 0.0
<b>Req. 4</b>	0.0%	0.0%	100%	100%	100%	100%	0.9969 ± 0.0	0.9969 ± 0.0	0.9969 ± 0.0	0.0016 ± 0.0	0.0016 ± 0.0
<b>Req. 5</b>	100%	100%	100%	100%	100%	60%	0.9967 ± 0.0	0.9967 ± 0.0	0.9967 ± 0.0	0.0018 ± 0.0002	0.0018 ± 0.0002
<b>Req. 6</b>	100%	100%	100%	100%	100%	100%	0.9990 ± 0.0	0.9990 ± 0.0	0.9990 ± 0.0	0.0005 ± 0.0	0.0005 ± 0.0
<b>Req. 7</b>	100%	100%	100%	100%	100%	100%	0.9981 ± 0.0	0.9981 ± 0.0	0.9981 ± 0.0	0.0010 ± 0.0	0.0010 ± 0.0
<b>Req. 8</b>	0.0%	0.0%	100%	100%	100%	100%	0.9967 ± 0.0	0.9967 ± 0.0	0.9967 ± 0.0	0.0017 ± 0.0	0.0017 ± 0.0
<b>Average</b>	50%	50%	100%	100%	100%	88%	–	–	–	–	–

Source: Author



According to Table 12 it is possible to notice that the performance of deterministic methods (“Mtc” and “DEA”) reached the expected limits by attending half of the requests. As a matter of fact, they are optimal for simple requests but unsuitable attending the most complex ones. Performance of each metaheuristic was surprisingly exceptional (except “SA” for request 1 and 5), because each of the 30 executions finds an optimum CP set, counting each as one hit. It is important to remember that there are several optimal CP sets because of the multimodality problem instances created by the several original database replications. For example, for 50, 100 and 200 CPs, there are multiple optimal possible solutions, because the data was replicated every 10 CPs. Request 1, for example, for 50 CPs has as optimal answers CP8 or CP18 or CP28 or CP38 or CP48, request 2, has as optimal answers CP6 and CP8 or CP16 and CP18 or CP26 and CP28 or any combination between one of these two CPs ended at 6 and 8. Each convergence to one of any of these combinations has the maximum possible fitness (or minimum energy) and counts as a hit, justifying the exceptional performance for all requests observed at Table 12.

So, at using the aforementioned replicated CP database, the greater the number of CPs, the more optimal combinations exist and even more if the response set is larger (e.g., request 3). This fact increases the probability of the metaheuristic methods converging to any of these optimal. Moreover, in order to explore a search scenario without these multiple optimal, the price of the first 10 CPs that composes each solution of the current request was arbitrarily lowered, generating the results present at Table 13. These instances configure a harder search space to be solved by any of these metaheuristic methods. Column named “Best Fitness” presents the best possible (highest) fitness value, i.e., the fitness of the known optimal answer encoded. Similarly, column “Best Energy” presents the best possible energy (lowest) found for “SA” method. The results with 50 and 100 CPs were omitted because they reached practically 100% of hits in all the requisitions with price changes. However, “SA” with 100 CPs presents low average hit of only 35%. Therefore, in this harder case (200 CPs) “SA” performed very poorly (average hit percentage of 8.33%) compared to the EAs. This bad performance is probably due to the SA simple exploration routine (neighbour generation) where only a single bit in the codification is changed in each iteration.

Table 13 – E.g.1: Results for 200 CPs database with arbitrary price decrease (single global optimal)

Request	Hits (%)				
	Mtc	DEA	GA	BDE	SA
Req. 1	100%	100%	56.67%	100%	83.33%
Req. 2	0.0%	0.0%	40%	36.67%	76.67%
Req. 3	0.0%	0.0%	26.67%	6.67%	53.33%
Req. 4	0.0%	0.0%	83.33%	50%	86.67%
Req. 5	100%	100%	73.33%	100%	73.33%
Req. 6	100%	100%	80%	100%	90%
Req. 7	100%	100%	73.33%	100%	90%
Req. 8	0.0%	0.0%	43.33%	53.33%	83.33%
Average	50%	50%	59.58%	68.33%	79.58%
Fitness ↑					Energy ↓
Request	Best Fitness			Best Energy	
	GA	BDE	DDE	SA	
Req. 1	0.9966 ± 0.0002	0.9968 ± 0.0	0.9967 ± 0.0002	0.0031 ± 0.0014	0.0016
Req. 2	0.9935 ± 0.0004	0.9936 ± 0.0002	0.9938 ± 0.0002	0.0036 ± 0.0002	0.0030
Req. 3	0.9926 ± 0.0004	0.9926 ± 0.0003	0.9929 ± 0.0003	0.0039 ± 0.0002	0.0035
Req. 4	0.9973 ± 0.0001	0.9972 ± 0.0001	0.9973 ± 0.0001	0.0016 ± 0.0001	0.0013
Req. 5	0.9970 ± 0.0002	0.9971 ± 0.0	0.9970 ± 0.0002	0.0018 ± 0.0003	0.0014
Req. 6	0.9994 ± 0.0002	0.9995 ± 0.0	0.9994 ± 0.0002	0.0005 ± 0.0006	0.0002
Req. 7	0.9984 ± 0.0002	0.9986 ± 0.0	0.9985 ± 0.0002	0.0011 ± 0.0007	0.0007
Req. 8	0.9984 ± 0.0003	0.9970 ± 0.0002	0.9970 ± 0.0002	0.0017 ± 0.0001	0.0014

Source: Author

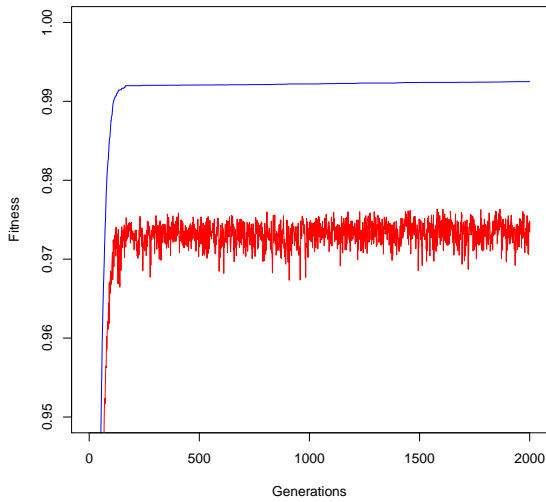
Therefore, according to Table 13, EAs present consistent results and have a satisfactory average assertiveness (above 50%), specially “DDE” (almost 80%). The fitness standard deviation is in the order of  $10^{-4}$  showing EA’s robustness for this problem instance. “Mtc” and “DEA” present the same top ranking CPs, so they have the same hits. Values in bold are those hits percentage considered statistically different for all the other metaheuristic methods for that request, according to the Dunn’s test. The complete p-values’ table (Table 29) of this statistical test and its discussion is in Appendix C. Basically, it can be concluded that “DDE” is statistically better than “GA”, “BDE” and “SA” in 2 out of 8 requests (i.e., requests 2 and 3, the most complex ones in this use case) and statistically equal at the other ones. On the other hand, “SA” is statistically worse in 6 out of 8 requests (1, 4, 5, 6, 7 and 8).

Figure 18 presents an average convergence plot regarding 30 executions for the “GA”, “BDE”, “DDE” and “SA” in the hardest problem instance, which is request 3 with 200 CPs and price decrease. The other requests have similar convergence behaviour than request 3, for each one of these methods. The y-axis shows fitness or energy values and the x-axis shows generations or iterations. The y-axis scale starts at 0.95 for the EAs and from 0 to 0.3 for “SA” in order to allow a better view and analysis. The upper curve represents the fitness of the best individual in the population and the bottom curve represents the average fitness of the entire population. “SA” presents only a single energy curve, corresponding to the one of the current solution candidate. The best fitness value (GA, BDE and DDE) is 0.9931 and the best energy (SA) is 0.0035.

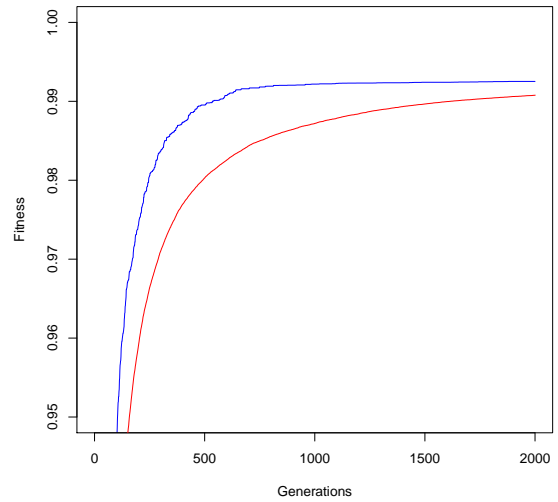
It is possible to note that the amount of generations is sufficient for a good convergence of “GA” and “DDE”. The fitness improvement is very noticeable up to 500 generations. After that, small improvements still occur, and are essential for convergence to the optimal solution, resulting in a hit. “BDE” has a smoother convergence curve and would probably benefit from more generations. “SA” definitively needs more iterations to converge since the energy curve only starts to stabilize at the very end (iteration 95.000). Thus, “DDE” has the best performance with these parameter values. The curve shape for average fitness in “GA” is different from the others because of the selection routine adopted, which is the stochastic tournament, a non-greedy procedure that allows the survival of worse individuals than the ones in previous generation (smaller fitness), thus allowing the average fitness of the entire population to vary (for worse or for better) between generations. This generates the shape of the average fitness seen in Figure 18d (bottom curve).

Table 14 presents the hit percentages obtained via experiments with the hardest instances (requests with a 200 CPs database, with arbitrary price decrease) for the four hybrid methods: “Mtc-GA”, “Mtc-BDE”, “Mtc-DDE” and “Mtc-SA”. Although simple

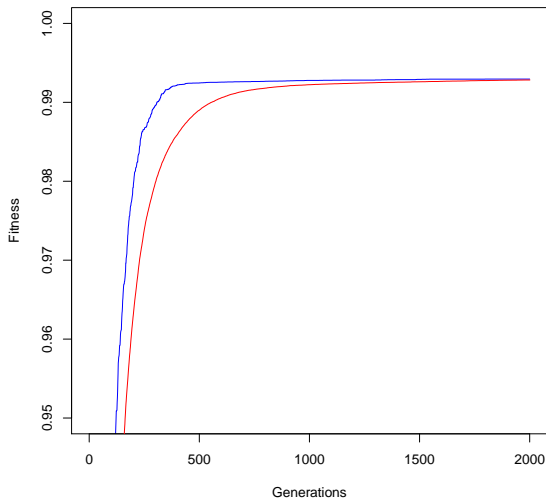
Figure 18 – E.g.1: Convergence plots for request 3 with 200 CPs and price decrease



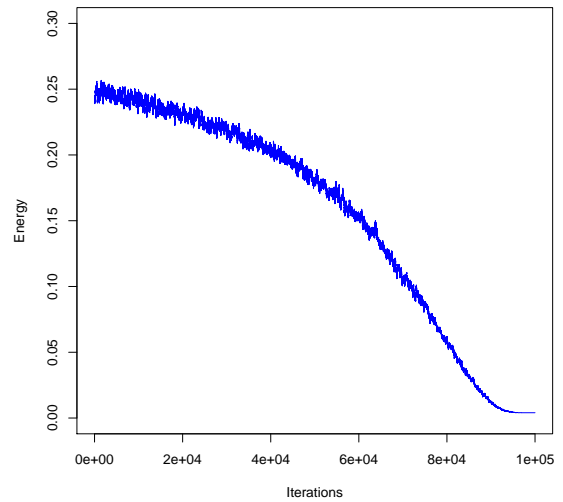
(a) GA convergence curve



(b) BDE convergence curve



(c) DDE convergence curve



(d) SA convergence curve

Source: Author

implementation, this kind of hybridization can be very useful to improve the hit percentages of these metaheuristics and also to reduce execution time for simpler requests, because these hybrid methods combines both the qualities of “Mtc” and metaheuristic methods. The average hits percentage of “Mtc-GA”, “Mtc-DDE” and “Mtc-SA” methods increased 14.59%, 7.92% and 44.17% compared to the single version of “GA”, “DDE” and “SA”, respectively. The “Mtc-BDE” performance remained the same as the single

version of “BDE” because of its almost 100% performance for simpler requests.

Table 14 – E.g.1: Hit percentages of the hybrid methods for 200 CPs with arbitrary price decrease

Request	Mtc-GA	Mtc-BDE	Mtc-DDE	Mtc-SA
<b>Req. 1</b>	100%	100%	100%	100%
<b>Req. 2</b>	40%	36.67%	<b>76.67%</b>	6.67%
<b>Req. 3</b>	26.67%	6.67%	<b>53.33%</b>	3.33%
<b>Req. 4</b>	83.33%	50%	86.67%	<b>6.67%</b>
<b>Req. 5</b>	100%	100%	100%	100%
<b>Req. 6</b>	100%	100%	100%	100%
<b>Req. 7</b>	100%	100%	100%	100%
<b>Req. 8</b>	43.33%	53.33%	83.33%	<b>3.33%</b>
<b>Average</b>	74.17%	68.33%	87.5%	52.5%

Source: Author

The values in bold are those hits percentage considered statistically different for all the other hybrid methods for that request using the Dunn’s test. The resulting table with p-values (Table 30) can be found in Appendix C. Requests 1, 5, 6 and 7 were not statistically tested because their hits percentage are the same for all methods (i.e., same performance) reaching 100%. Analyzing these results it is possible to notice that, in requests 2 and 3, “Mtc-DDE” is statistically better than the other ones and “Mtc-SA” is statistically the worst at request 4 and 8.

Finally, Table 15 shows the average execution time for all main selection algorithms, in milliseconds (ms), per execution (metaheuristics were executed 30 times). Thus, the execution time ranking, in ascending order, are: “Mtc”, “DEA”, “Mtc-SA”, “SA”, “Mtc-BDE”, “Mtc-GA”, “BDE”, “GA”, “Mtc-DDE” and “DDE”.

Thus, according to Tables 12, 13, 14 and 15, it is possible to state that all EAs are able to find very satisfactory answers, even for the most difficult cases (e.g., request 3 with 200 CPs) in an acceptable execution time. Deterministic methods like CPS-Matching and CPS-DEA presents the minimal execution time and can find optimal solutions for the simpler requests (1, 5, 6 and 7) with 100% of request attendance, but not very satisfactory for more complex (requests 2, 3, 4 and 8) cases where more than one CPs is required to fully attend each request. “SA” shows a poor performance compared with the others EAs. The hybrid methods presents the most optimized answers, especially “Mtc-DDE”, and the performed statistical analysis proves that “Mtc-DDE” method achieved the best overall performance.

### 5.3 USE CASE 2 – QUALITATIVE PIs

The second simulated scenario uses only qualitative PIs plus price. These PIs are not specific of any CC service models. So, this example aims to explore the use of qualitative criteria and different PI weights in the proposed architecture.

Table 15 – E.g.1: Execution time for the developed experiments with 200 CPs, in milliseconds

Request	Mtc	DEA	GA	BDE	DDE	SA	Mtc-GA	Mtc-BDE	Mtc-DDE	Mtc-SA
Req. 1	15	134	588	477	1611	260	15	15	15	15
Req. 2	10	184	592	487	1635	259	681	576	1724	265
Req. 3	5	154	593	488	1621	259	663	558	1691	264
Req. 4	14	118	580	465	1598	239	665	550	1683	236
Req. 5	9	92	573	458	1588	226	9	9	9	9
Req. 6	9	121	584	471	1602	234	9	9	9	9
Req. 7	7	84	580	464	1610	210	7	7	7	7
Req. 8	1	43	561	441	1575	130	606	486	1620	130
Average	9	116	581	469	1605	227	332	276	845	117
<b>With arbitrary decrease in price (single global optimal)</b>										
Request	Mtc	DEA	GA	BDE	DDE	SA	Mtc-GA	Mtc-BDE	Mtc-DDE	Mtc-SA
Req. 1	16	124	584	476	1603	250	16	16	16	16
Req. 2	10	172	587	487	1610	276	675	575	1698	266
Req. 3	3	162	585	489	1614	259	660	564	1689	258
Req. 4	14	109	576	465	1591	233	670	559	1685	234
Req. 5	7	98	571	457	1587	226	7	7	7	7
Req. 6	10	117	578	471	1596	236	10	10	10	10
Req. 7	7	72	575	465	1588	215	7	7	7	7
Req. 8	1	32	554	440	1567	126	600	486	1613	137
Average	8	111	576	469	1594	228	331	278	841	117

Source: Author

Table 16 present the CP database used in the experiments. This database is simulated and involves ten CPs and four qualitative PIs that could occur in actual scenarios plus CP price, in US\$ per month of use. PI “TS” is the type(s) of service(s) made available to customers. It is a qualitative compound PI, i.e., always NB behaviour type, with four basic categories: A (Storage), P (Processing), VM (Virtual Machines) and S (Software). The other PIs are qualitative ordered. The PI “Security” is the estimated degree of information security and privacy for that CP. In this case, “Security” is treated as a qualitative HT PI, with three categories: L (Low), M (Medium) and H (High). Next, “Usability” means the average level of general CP usability attributed by its users, given by four categories: B (Bad), F (Fair), G (Good) and VG (Very Good). Finally, PI “Support” indicates the overall quality of customer support offered for free to all its users. The support can be: P (Poor), R (Regular), G (Good) and E (Excellent). So, it is logic that the PIs “Usability” and “Support” should be HT, by their definitions. In this case, the PIs “TS” and “Security” are directly related and influence each other, so they are in the same group and must be attended simultaneously by a CP if requested by customer.

Table 17 informs five customer requests considering the database shown in Table 16. Note that in this scenario some PI are considered more important than others. For example, in request 1, the PI “TS” is three times more important than PI “Security” and nine times more important than PI “Usability”; and “Security” is three times more important than “Usability”. “Support” is not a highly demanded customer PI and it can assume any value. Using these importance rules in the AHP’s Judgement Matrix (see Subsection 2.3.1.1 and Appendix B) it is generated the observed PI weight values in

Table 16 – E.g.2: Simulated CP database with their qualitative PIs

<b>CPs/PIs</b>	<b>TS</b>	<b>Security</b>	<b>Usability</b>	<b>Support</b>	<b>Price</b>
<b>Type ID</b>	<b>NB</b>	<b>HT</b>	<b>HT</b>	<b>HT</b>	<b>LB</b>
	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>
CP1	A	L	G	R	1.00
CP2	P	M	F	G	1.50
CP3	S	M	G	R	2.50
CP4	VM	H	B	P	3.00
CP5	A,P	H	F	G	3.50
CP6	A,VM	L	G	E	5.00
CP7	A,P,S	M	G	R	4.50
CP8	P,VM	L	VG	P	4.00
CP9	A,P,VM	M	G	R	6.00
CP10	A,VM,S	H	F	E	9.00

Source: Author

Table 17 for request 1. In request 2, “TS” is three times more important than “Usability”. For request 3, “TS” is three times more important than “Security” and “Support”; and “Security” are equally important as “Support”. Next, in request 4, “TS” is two times more important than “Security”, four times “Usability” and eight times “Support”; “Security” is two times “Usability” and four times “Support”; and “Usability” is two times more important than “Support”. At last, all PIs have the same importance for the customer of request 5. Thus, in this use case, the type of service can be considered the most important and critical PI for the customers, followed by the security factor, which often makes sense in the real world, considering these PIs.

Table 17 – E.g.2: Simulated customer requests

<b>Pis/Reqs</b>	<b>Request 1</b>		<b>Request 2</b>		<b>Request 3</b>		<b>Request 4</b>		<b>Request 5</b>	
<b>PI Name</b>	<b>Value</b>	<b>Weight</b>	<b>Value</b>	<b>Weight</b>	<b>Value</b>	<b>Weight</b>	<b>Value</b>	<b>Weight</b>	<b>Value</b>	<b>Weight</b>
TS	A	0.69	P	0.75	VM	0.6	S	0.53	A,P	1
Security	H	0.23	–	–	M	0.2	M	0.27	H	1
Usability	F	0.08	VG	0.25	–	–	G	0.13	VG	1
Support	–	–	–	–	G	0.2	G	0.07	E	1
<b>Optimal</b>	CP5		CP8		CP2 & CP4		CP2 & CP3		CP5&CP6&CP8	

Source: Author

Note that Request 1 demands storage with the highest data security possible, with a bit of usability in the service offered. Request 2 only cares about processing with the highest usability available, so even support is not needed. Request 3 just asks for VMs with decent amount of security and support quality, so usability is not important. Request 4 is satisfied with a software service with the medium values for each PI available: security, usability and support. Finally, Request 5 demands a CP that offer the services of storage and processing with the highest quality possible in this scenario. Last line of Table 17 presents the known answers (optimal) for each request. The easiest requests are 1 and 2, because a single CP is enough to attend

them completely. The most complex request is definitely the fifth one, since it requires the highest possible values for each PI, demanding three CPs to attend it entirely. Also, note that request 3 can be fully attended by CP10 on its own, but the final price (9) is much higher than the set CP2 & CP4, which together also fully attend request 3, with lower price ( $1.5 + 3.0 = 4.5$ ). Therefore, as the weight for price and quantity of CPs are the same, the set CP2 and CP4 is better than only CP10. For the sake of anticipation, it is important to state that the CPS-Matching and CPS-DEA methods can not identify this possible CP grouping and can elect CP10 as the best one.

### 5.3.1 Results and Analysis – Qualitative PIs

Tests with all methods discussed were performed for 10, 20, 30, 50, 100 and 200 CPs using Table 16 data. To accomplish that, the same CP PI data replication and methodology was used. Also in this use case, the exhaustive fitness search method could solve the CPS problem but since it goes through all search space, time needed is not feasible beyond a database with 30 CPs. As in the previous use case, next result tables present only the results for the larger databases. The requests are the same for all databases.

Table 18 presents the results obtained when applying the CPS-Matching (“Mtc”), CPS-DEA (“DEA”), CPS-GA (“GA”), CPS-BDE (“BDE”), CPS-DDE (“DDE”) and CPS-SA (“SA”) methods on the database of Table 16 replicated for 200 CPs, taking as input each one of the five requests. For all methods, the hits percentage has been calculated, as well as the average and standard deviation of the fitness (to be maximized “↑”) and energy (to be minimized “↓”) for the metaheuristic methods. Results for databases with less than 200 CPs

Table 18 – E.g.2: Results for 200 CPs database with 4 qualitative PIs plus the price

Request	Hits (%)						Fitness ↑		Energy ↓	
	Mtc	DEA	GA	BDE	DDE	SA	GA	BDE	DDE	SA
Req. 1	100%	100%	100%	100%	100%	100%	0.9984 ± 0.0	0.9984 ± 0.0	0.9984 ± 0.0	0.0008 ± 0.0
Req. 2	100%	100%	100%	100%	100%	100%	0.9981 ± 0.0	0.9981 ± 0.0	0.9981 ± 0.0	0.0009 ± 0.0
Req. 3	0.0%	0.0%	100%	100%	100%	83.33%	0.9953 ± 0.0	0.9953 ± 0.0	0.9953 ± 0.0	0.0025 ± 0.0003
Req. 4	0.0%	0.0%	100%	100%	100%	73.33%	0.9987 ± 0.0	0.9987 ± 0.0	0.9987 ± 0.0	0.0008 ± 0.0003
Req. 5	0.0%	0.0%	100%	100%	100%	100%	0.9940 ± 0.0	0.9940 ± 0.0	0.9940 ± 0.0	0.0030 ± 0.0
Average	40%	40%	100%	100%	100%	91.33%	–	–	–	–

Source: Author



were omitted because they all reached 100% hits percentage for all the methods and requests. The top scored CP to “Mtc” and “DEA” methods are: CP5 (Req. 1), CP8 (Req. 2), CP10 (Req. 3), CP3 (Req. 4) and CP6 (Req. 5). Again, from all metaheuristic methods, SA appears to be the less efficient. The optimal energy values for requests that did not get 100% hits in SA are: 0.0024 (Req. 3) and 0.0006 (Req. 4).

Table 18 proves the expected performance of the deterministic methods: excellent for requests 1 and 2 (simpler requests), but unsuitable for requests 3, 4 and 5 (more complex ones). The performance of each metaheuristic method was exceptional (except “SA” for request 3 and 4). Of course, this use case has multiple optimal, as in the previous one. So, to explore this multimodal search scenario without these multiple optimal, the price of the first 10 CPs that compose each solution of the current request was arbitrarily lowered, generating the results present in Table 19 and configuring a harder search space to be solved by these metaheuristic methods. The column named “Best Fitness” presents the highest possible fitness values and the column “Best Energy” presents the lowest possible energy values. The instances involving less than 200 CPs were omitted because their results reached practically 100% for all the requests, including the ones with the price decrease. The only exception is “SA” performing with 100 CPs whose average hit percentage is 52%. So, in the case with 200 CPs, “SA” presents a very poor performance again in comparison with the EAs. In this case, “SA” average hit percentage is only 7.33% .

Therefore, according to Table 19, EAs present consistent results and have a satisfactory average assertiveness (above 50%), specially “DDE”. The fitness standard deviation is in the order of  $10^{-4}$  showing EA’s robustness for this problem instance. “Mtc” and “DEA” presents the same top ranking CPs, so they have the same hits. Moreover, the top CP for request 3, with price decrease, is CP4, instead CP10 (without price decrease) for both “Mtc” and “DEA” methods. Hit percentage values in bold represent the ones whose Dunn’s test considers statistically different from all other method’s results. The complete p-values’ table (Table 31) of this statistical test is in Appendix C. Analyzing Table 31, it can be concluded that “DDE” (best average result) is statistically more significant than the other metaheuristic methods only at request 5 (the hardest one) and equal for all another requests (i.e., requests 1 to 4). On the other hand, “SA” obtain the worst statistical performance, except on request 5, compared to “GA” and “BDE”.

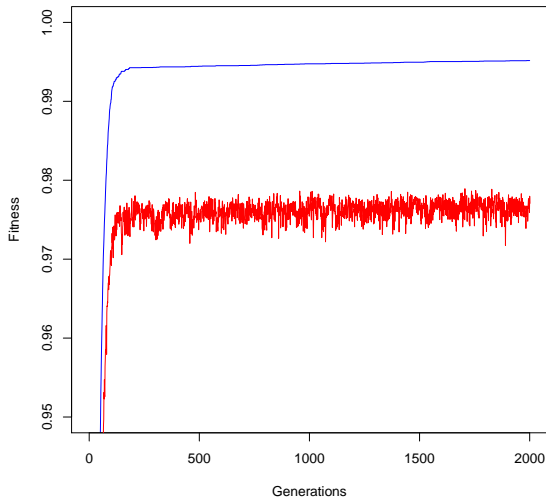
As with previous use case, a convergence plot for the hardest problem instance, which is request 5 with 200 CPs with price decrease, is presented in Figure 19 for each metaheuristic method, but for the qualitative PIs, according to Table 19. The curves behavior and conclusions are very similar. The best fitness value (GA, BDE and DDE) is 0.9959 and the best energy (SA) is 0.0020.

Table 19 – E.g.2: Results for 200 CPs database with arbitrary price decrease (single global optimal)

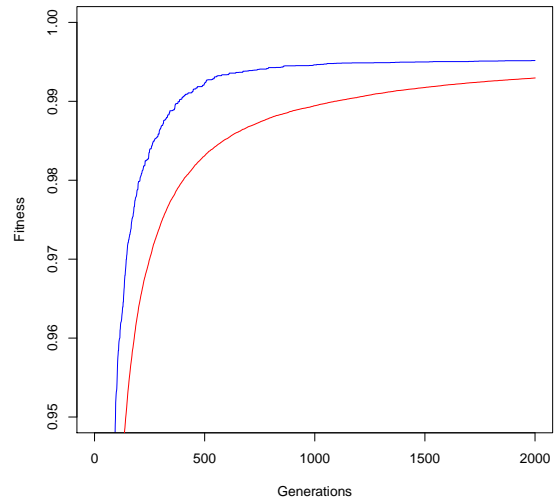
Request	Hits (%)				
	Mtc	DEA	GA	BDE	SA
Req. 1	100%	100%	73.33%	100%	80%
Req. 2	100%	100%	60%	100%	93.33%
Req. 3	0.0%	0.0%	50%	56.67%	80%
Req. 4	0.0%	0.0%	76.67%	50%	80%
Req. 5	0.0%	0.0%	30%	16.67%	66.67%
Average	40%	40%	58%	64.67%	80%
Request	Fitness ↑			Energy ↓	
	GA	BDE	DDE	Best Fitness	SA
Req. 1	0.9989 ± 0.0003	0.9991 ± 0.0	0.9989 ± 0.0006	0.9991	0.0007 ± 0.0001
Req. 2	0.9985 ± 0.0003	0.9987 ± 0.0	0.9986 ± 0.0002	0.9987	0.0009 ± 0.0001
Req. 3	0.9959 ± 0.0004	0.9960 ± 0.0003	0.9961 ± 0.0003	0.9962	0.0024 ± 0.0003
Req. 4	0.9995 ± 0.0003	0.9994 ± 0.0003	0.9996 ± 0.0003	0.9997	0.0008 ± 0.0003
Req. 5	0.9953 ± 0.0005	0.9952 ± 0.0004	0.9957 ± 0.0004	0.9959	0.0028 ± 0.0002

Source: Author

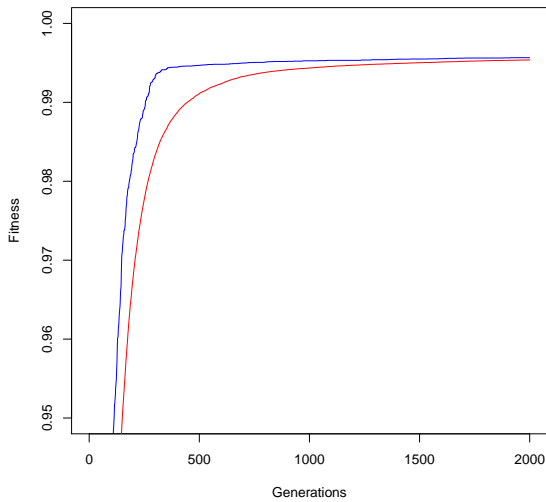
Figure 19 – E.g.2: Convergence plots for request 5 with 200 CPs and price decrease



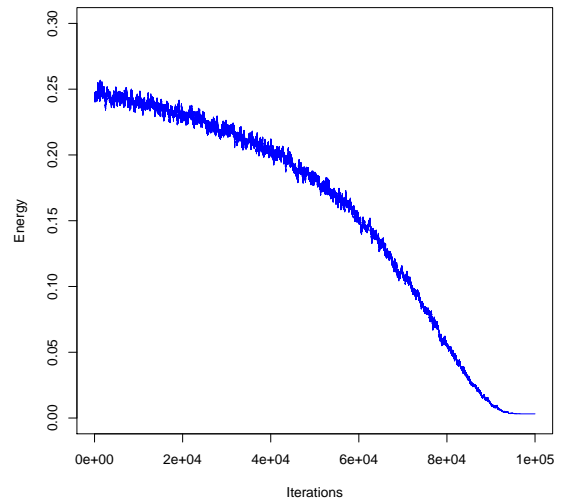
(a) GA convergence curve



(b) BDE convergence curve



(c) DDE convergence curve



(d) SA convergence curve

Source: Author

Table 20 presents hit percentages obtained via experiments with the hardest instances (Table 19) for the four hybrid methods: “Mtc-GA”, “Mtc-BDE”, “Mtc-DDE” and “Mtc-SA”. The average hits percentage of ‘Mtc-GA’, “Mtc-DDE” and “Mtc-SA” methods increased 13.33%, 5.33% and 36% compared to “GA”, “DDE” and “SA”, respectively. The “Mtc-BDE” performance remained the same as the “BDE”.

The values in bold are those hits percentage considered statistically different

Table 20 – E.g.2: Hit percentages of the hybrid methods for 200 CPs with arbitrary price decrease

Request	Mtc-GA	Mtc-BDE	Mtc-DDE	Mtc-SA
<b>Req. 1</b>	100%	100%	100%	100%
<b>Req. 2</b>	100%	100%	100%	100%
<b>Req. 3</b>	50%	56.67%	80%	<b>6.67%</b>
<b>Req. 4</b>	76.67%	50%	80%	<b>6.67%</b>
<b>Req. 5</b>	30%	16.67%	<b>66.67%</b>	3.33%
<b>Average</b>	71.33%	64.67%	85.33%	43.33%

Source: Author

for all the other hybrid methods for requests 3, 4 and 5 using Dunn's test. The table with its p-values (Table 32) can be find in Appendix C. Requests 1 and 2 were not tested because their hits percentage are the same for all methods (same performance) reaching 100%. Thus, it can be concluded that "Mtc-DDE" is statistically better than any other methods for the hardest request (i.e., 5) and statistically equal for the others, while "Mtc-SA" is statistically worse at request 3 and 4.

Table 21 shows the average execution time for all main selection algorithms, in milliseconds (ms) per execution. The execution speed ranking is the same as the first use case discussed.

Table 21 – E.g.2: Execution time for the developed experiments with 200 CPs, in milliseconds

Request	Mtc	DEA	GA	BDE	DDE	SA	Mtc-GA	Mtc-BDE	Mtc-DDE	Mtc-SA
<b>Req. 1</b>	6	112	529	445	1582	224	6	6	6	6
<b>Req. 2</b>	2	93	523	446	1589	207	2	2	2	2
<b>Req. 3</b>	2	101	526	455	1598	218	528	457	1600	220
<b>Req. 4</b>	4	132	531	456	1595	238	535	460	1599	240
<b>Req. 5</b>	3	125	540	465	1611	275	543	468	1614	278
<b>Average</b>	3	113	530	454	1595	232	323	279	964	149
<b>With arbitrary decrease in price (single global optimal)</b>										
Request	Mtc	DEA	GA	BDE	DDE	SA	Mtc-GA	Mtc-BDE	Mtc-DDE	Mtc-SA
<b>Req. 1</b>	5	110	524	447	1584	212	5	5	5	5
<b>Req. 2</b>	2	95	519	442	1575	194	2	2	2	2
<b>Req. 3</b>	2	112	519	451	1581	206	521	453	1583	208
<b>Req. 4</b>	3	126	521	457	1584	223	524	460	1587	226
<b>Req. 5</b>	3	131	532	469	1601	275	535	472	1604	278
<b>Average</b>	3	115	523	453	1585	222	317	278	956	144

Source: Author

Finally, with Tables 18, 19, 20 and 21, it is possible to state that all EAs are able to find very satisfactory answers, even for the most difficult problem instances in an acceptable execution time. Deterministic methods ("Mtc" and "DEA") present the minimal execution time and can find optimal solutions for the simpler requests (1 and 2) with 100% of request attendance, but they are not completely satisfactory for more complex (requests 3, 4 and 5). Again, hybrid methods present the most optimized answers and "Mtc-DDE" has the best performance to solve the problem instances of this use case.

## 5.4 USE CASE – REAL PIs

The first real data scenario was created by collecting and consolidating data available at the Cloudwards website ([www.cloudwards.net](http://www.cloudwards.net)) in December 11, 2018. Cloudwards was chosen, because in addition to comparing and measuring several CPs, it informs the price associated to each CP. Price is a fundamental PI for the proposed architecture. Initially, only CPs that offer specialized storage services were considered (SaaS). The main PIs identified in the Cloudwards' Cloud Storage Comparison table are all quantitative. This use case is intended to prove that the proposed architecture and its methods are feasible and suitable in the real world and can be applied at any time to those types of databases that will generate adequate solutions. In addition, it shows how a CP database can be built in practice and also reinforces that the two previous simulated cases have data consistent with reality.

Table 22 presents the CP database built with the comparative data of Cloudwards website. All CPs are for storage service. A total of 15 different CPs companies are listed: "Sync.com", "pCloud", "Tresorit", "OneDrive", "DropBox", "GoogleDrive", "SugarSync", "AmazonDrive", "Mega", "OpenDrive", "iCloud", "MediaFire", "Bitcasa", "Hubic", "JustCloud.com". However, each CP company have several payment plans, with different values, payment types (monthly, annual or single purchase) and number of users (personal, one user, or business/enterprises plans for several ones). For simplicity purpose, only personal monthly ones are considered in this use case. Thus, each payment plan can be considered an independent CP in the proposed architecture, making a total of 61 CPs in the database. Four important PIs were identified at Cloudwards. The first, of course, is "Storage", that means the maximum amount of memory (Gb) for customer's storage purpose at CPs servers. Basically it is the most important PI because the service type needed is storage. "Storage" is clearly a HB PI type. The second PI, always present and LB, is the service cost/price, given in US\$ per month. The PIs "Ease of Use", "Security" and "Support" (called quality of service PIs) are also measured by Cloudwards for each one of the CPs companies and they are given quantitatively in percentage with an HB behaviour type. The PI "Ease of Use" has a similar concept to "Usability" (use case 2) and it means the customer easiness to access and use the service, via Internet. PI "Security" stands for the CP service quality comprising information security and privacy policy offered to its customers, such as encryption, compliance with security standards, etc.. Finally, "Support" indicates the quality of the setup guides, tutorials and contact available for customer teaching and support. In this case, all PIs were considered independent and thus they do not need to be attended by the same CP. Thus, no groups are needed and "ID" field is zero for all of them.

Table 22 – E.g.3: CPs database built with data from Cloudwards

<b>CPs and Pls</b>	<b>Storage</b>	<b>Ease of Use</b>	<b>Security</b>	<b>Support</b>	<b>Price</b>
<b>Type</b>	<b>HB</b>	<b>HB</b>	<b>HB</b>	<b>HB</b>	<b>LB</b>
<b>ID</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>
1) SyncStarter	5	95	100	90	0
2) SyncPro1	500	95	100	90	4.08
3) SyncPro2	2000	95	100	90	8
4) pCloudFree	10	95	87	75	0
5) pCloudPremium	500	95	87	75	4.99
6) pCloudPremiumPlus	2000	95	87	75	9.99
7) TresoritBasic	3	75	100	75	0
8) TresoritBasic	200	75	100	75	12.5
9) TresoritSolo	2000	75	100	75	30
10) OneDriveFree	5	80	55	80	0
11) OneDrive50	50	80	55	80	1.99
12) OneDrive1T	1000	80	55	80	6.99
13) OneDrive5T	5000	80	55	80	9.99
14) DropBoxPlus	1000	90	65	65	9.99
15) DropBoxBusiness	2048	90	65	65	15
16) GoogleDriveFree	15	75	60	88	0
17) GoogleDrive100	100	75	60	88	1.99
18) GoogleDrive1T	1000	75	60	88	9.99
19) GoogleDrive2T	2000	75	60	88	19.99
20) GoogleDrive10T	10000	75	60	88	99.99
21) GoogleDrive20T	20000	75	60	88	199.99
22) GoogleDrive30T	30000	75	60	88	299.99
23) SugarSync100	100	75	75	56	7.49
24) SugarSync250	250	75	75	56	9.99
25) SugarSync500	500	75	75	56	18.95
26) AmazonDrive5	5	80	50	60	0
27) AmazonDrive100	100	80	50	60	1
28) AmazonDrive1T	1000	80	50	60	5
29) AmazonDrive2T	2000	80	50	60	10
30) AmazonDrive3T	3000	80	50	60	15
31) AmazonDrive4T	4000	80	50	60	20
32) AmazonDrive5T	5000	80	50	60	25
33) AmazonDrive6T	6000	80	50	60	30
34) AmazonDrive7T	7000	80	50	60	35

**Table 22 – Continuation of the database**

<b>CPs and PIs</b>	<b>Storage</b>	<b>Ease of Use</b>	<b>Security</b>	<b>Support</b>	<b>Price</b>
35) AmazonDrive8T	8000	80	50	60	40
36) AmazonDrive9T	9000	80	50	60	45
37) AmazonDrive10T	10000	80	50	60	50
38) AmazonDrive20T	20000	80	50	60	100
39) AmazonDrive30T	30000	80	50	60	150
40) MegaFree	15	78	79	50	0
41) MegaProLite	200	78	79	50	5.79
42) MegaPro1	1000	78	79	50	11.59
43) MegaPro2	4000	78	79	50	23.19
44) MegaPro3	8000	78	79	50	34.8
45) OpenDriveFree	5	68	80	65	0
46) OpenDrivePersonal	500	68	80	65	5
47) iCloudFree	5	70	80	85	0
48) iCloud50	50	70	80	85	0.99
49) iCloud200	200	70	80	85	2.99
50) iCloud2T	2000	70	80	85	9.99
51) MediaFireFree	10	70	45	75	0
52) MediaFirePro	1000	70	45	75	5
53) BitcasaFree	5	0	0	0	0
54) BitcasaPremium	1000	0	0	0	10
55) BitcasaPro	10000	0	0	0	99
56) HubicFree	25	65	20	10	0
57) Hubic100	100	65	20	10	1.12
58) Hubic10T	10000	65	20	10	5.6
59) JustCloudHome	75	35	20	10	10.69
60) JustCloudPremium	250	35	20	10	11.94
61) JustCloudUltimate	1000	35	20	10	14.44

Source: Author

Some interesting observations and considerations can be made with data of Table 22. This scenario has a total of 12 CPs that offer up to 25 Gb of storage for free. These CP are interesting for further test the proposed methods and the behaviour of the fitness/energy function. From the paid CPs, if only the PIs “Storage” and “Price” are considered, the CP with the best price-storage ratio is “Hubic10T” (CP58), that offer 10 Tb of memory for only US\$ 5.6 per month. The CP company with the highest quality of service PIs values is Sync.com. Actually, Sync.com is the best evaluated company on Cloudwards and it is considered by them the most reliable and safest CP

for storage services in the year 2018, followed by “pCloud”. Both these companies have free plans, but with a very limited amount of storage. The CP company worst evaluated is “JustCloud”. The PIs quality of service from company “Bitcasa” is not evaluated at Cloudwards, so the value zero is put in place. The only companies that offer memory over 10 Tb are “GoogleDrive” and “AmazonDrive”. Amazon is cheaper and easier, but Google has more security and support quality. Summing up, the optimal CP depends on the importance weigh assigned to each PI.

Table 23 informs five customer requests considering the database shown in Table 22. This use case/scenario also uses PIs with different weighs generated by importance rules according to the AHP’ Judgement Matrix. Taking this into account, in request 1 PI “Storage” is three times more important than PI “Security”. “Ease of Use” and “Support” are not PIs of customer’s great interest and can assume any value. This request represent just a small storage need (15 Gb) with a reasonable good security quality (70%). In request 2, “Storage” is seven times more important than “Security”, “Ease of Use” and “Support”; that are equally important among them. This request demands a good amount of storage memory (1 Tb) and all the other PIs. In request 3, “Storage” is four times more important than “Ease of Use” and two times more important than “Security”; and “Security” is two times more important than “Ease of Use”. Request 3 only cares about a high storage memory (5 Tb) with a good (80%) “Ease of Use” and a very good (90%) “Security” quality, so “Support” is not needed. In request 4, “Storage” is six times more important than “Ease of Use” and “Support”; and three times more important than “Security”. By its turn, “Security” is two times more important than “Ease of Use” and “Support”; and “Ease of Use” is equally important as “Support”. The customer raising this request just needs a very high storage memory (10 Tb), the other PIs can assume at least a medium quality (50%). Finally, in request 5, “Storage” is four times more important than “Ease of Use” and “Support” and two times more important than “Security”; “Security” is two times more important than “Ease of Use” and “Support”; and “Ease of Use” is equally important as “Support”. This request represents the most difficult demand possible in this scenario, where the extreme values are desired for each PI.

Table 23 – E.g.3: Proposed customer requests

PIs/Reqs	Request 1		Request 2		Request 3		Request 4		Request 5	
PI Name	Value	Weight	Value	Weight	Value	Weight	Value	Weight	Value	Weight
Storage	15	0.75	1000	0.7	5000	0.57	10000	0.6	30000	0.5
Ease of Use	—	—	80	0.1	80	0.14	50	0.1	95	0.125
Security	70	0.25	80	0.1	90	0.29	50	0.2	100	0.25
Support	—	—	80	0.1	—	—	50	0.1	90	0.125
Optimal	CP40		CP3		CP1 & CP58		CP37 or (...)		CP1 & CP39	
(…) = CP58 & (CP1 or CP4 or CP7 or CP10 or CP16 or CP26 or CP40 or CP45 or CP47)										

Source: Author

For this case, the storage memory can be considered the most important and



critical PI for the customers, followed by the security quality. Moreover, the last lines of Table 23 presents the possible optima for each request. The easiest requests are 1 (simpler, attended only by CP40) and possibly request 3, because it has several optima, including a single CP answer. Theoretically, the most complex request is the fifth one, since it demands the highest possible values for each PI.

It is important to note that finding the ideal optima for request 2, 3 and 4 is not visually trivial or as easy as in use case 1 and 2, with a simulated controlled database. Request 2 needs a CP that offer the double amount of the necessary storage memory, even though it is more expensive, because of the other PIs demanded values. Request 3 can almost be attended only by CP13, except by its “Security”. On other hand, CP1 and CP58 can also be a good answer, CP58 is the best price-storage trade and CP1 is free and has the best values available for all the PIs quality of service. Request 4 is even harder to arbitrate a single optima set, from the 61 CPs, 50 CPs have at least 50% for each quality of service PIs and eight CPs offer 10 Tb or more for storage. If only a single CP is desired, CP37 is the best one and it can even attend all the request. However, it costs US\$ 50 per month, while CP58 offer the same amount for US\$ 5.6 per month (almost nine times lower) but its “Security” and “Support” PIs are very bad. So, to attend these two PIs another free CP can be added to the solution set, with the condition of presenting “Security” and “Support” values equal or superior to 50%. The CPs that meet this condition are: 1, 4, 7, 10, 16, 26, 40, 45 and 47. So, both CP37 and a set with CP58 plus one for these cited CPs will be considered the optimal. Only CP22 and CP39 can offer the huge amount of memory demanded by request 5, but CP39 is two times cheaper than CP22 and both have decent values (but insufficient for the request) for their quality of service PIs. Thus, in this case, CP39 will be prioritized. Moreover, CPs 1, 2 and 3 have the maximum values for the quality of service PIs. Which one to choose? The free one (CP1). Therefore, a set with CP1 and CP39 should be the best answer for request 5.

#### 5.4.1 Results and Analysis – Real PIs

Tests with all methods were performed for the 61 CPs using the proposed database showed in Table 22 and customer requests shown in Table 23. The problem instances’ search space size is  $2.31 * 10^{18}$ , approximately. The average time to complete a fitness assessment for 61 CPs was approximately 0.005 milliseconds. So, an exhaustive fitness search will take millennia and it is impractical.

Thus, Table 24 presents the results obtained when applying the CPS-Matching (“Mtc”), CPS-DEA (“DEA”), CPS-GA (“GA”), CPS-BDE (“BDE”), CPS-DDE (“DDE”) and CPS-SA (“SA”) methods on the database of Table 22 taking as input each one of the five requests. For all methods, the hits percentage has been calculated. These hits

are based on the known optima. For the metaheuristic methods, average and standard deviation of the fitness (energy for SA) has been calculated as well.

All metaheuristic methods are very robust and efficient in this scenario. None request was really hard to find the answers. Thus, these results validate the model for real data too. The better scored CP for each request according to “Mtc” and “DEA” methods are: CP40 (Req. 1), CP3 (Req. 2), CP13 (Req. 3) and CP37 (Req. 4). For request 5, “Mtc” selects CP39 and “DEA” selects CP1 as the best. If the hybrid methods are applied (Mtc-GA, Mtc-BDE, Mtc-DDE and Mtc-SA), 100% of hits is obtained in each request for each method. Dunn’s statistical test was applied with 5% significance level for requests 1, 2 and 3. Test shows that there are no statistical difference between the methods for these requests. Thus, the results indicate that metaheuristic methods present high potential for solving the CPS problem. Nevertheless, more experiments need to be performed in order to get more conclusive data. Moreover, at this use case, any hybrid

Table 24 – E.g.3: Results for the real CPs database and from Cloudwards

Request	Hits (%)					Energy ↓			
	Mtc	DEA	GA	BDE	DDE	SA	Best Fitness	SA	Best Energy
Req. 1	100%	100%	100%	100%	96.67%	100%	1.0	0.0 ± 0.0	0.0
Req. 2	100%	100%	96.67%	100%	96.67%	100%	0.9974	0.0013 ± 0.0	0.0013
Req. 3	0.0%	0.0%	100%	93.33%	100%	90%	0.9982	0.0010 ± 0.0002	0.0009
Req. 4	100%	100%	100%	100%	100%	100%	0.9899	0.0051 ± 0.0	0.0051
Req. 5	0.0%	0.0%	100%	100%	100%	100%	0.9519	0.0240 ± 0.0	0.0240
Average	60%	60%	99.33%	98.67%	98.67%	98%			
Request	Fitness ↑					Energy ↓			
	GA	BDE	DDE	Best Fitness	SA	Best Energy	SA	Best Energy	Best Energy
Req. 1	1.0 ± 0.0	1.0 ± 0.0	0.9999 ± 0.0001	1.0	0.0 ± 0.0	0.0			
Req. 2	0.9972 ± 0.0014	0.9974 ± 0.0	0.9972 ± 0.0014	0.9974	0.0013 ± 0.0	0.0013			
Req. 3	0.9982 ± 0.0	0.9981 ± 0.0004	0.9982 ± 0.0	0.9982	0.0010 ± 0.0002	0.0009			
Req. 4	0.9899 ± 0.0	0.9899 ± 0.0	0.9899 ± 0.0	0.9899	0.0051 ± 0.0	0.0051			
Req. 5	0.9519 ± 0.0	0.9519 ± 0.0	0.9515 ± 0.0	0.9519	0.0240 ± 0.0	0.0240			

Source: Author

method has assertiveness of almost 100%. Hybrid methods were promising in all

tested use cases.

At the end, Table 25 shows the average execution time for all selection methods, in milliseconds (ms), per execution. Note that the hybridization significantly decreases the execution time. In request 4, “Mtc” returns CP37, which is also an acceptable optima (100% request attendance), so the hybrid did not run its metaheuristic component.

Table 25 – E.g.3: Execution time for the real use case, in milliseconds

<b>Request</b>	<b>Mtc</b>	<b>DEA</b>	<b>GA</b>	<b>BDE</b>	<b>DDE</b>	<b>SA</b>	<b>Mtc-GA</b>	<b>Mtc-BDE</b>	<b>Mtc-DDE</b>	<b>Mtc-SA</b>
<b>Req. 1</b>	2	52	214	166	605	67	2	2	2	2
<b>Req. 2</b>	3	74	207	159	576	68	3	3	3	3
<b>Req. 3</b>	2	65	203	163	615	73	205	165	617	75
<b>Req. 4</b>	3	82	215	181	598	81	3	3	3	3
<b>Req. 5</b>	2	86	215	172	613	77	217	174	615	79
<b>Average</b>	2	72	211	168	601	73	86	69	248	32

Source: Author

## 5.5 PARTIAL CONSIDERATIONS

This chapter aims to validate the proposed architecture and its methods performing several experiment instances (CP database plus request). To achieve this, three examples of practical use cases involving CP selection are presented here. The first two cases use simulated databases and last one uses real data. Simulated databases are simple, feasible and controlled, where the global optima of each request can be easily determined. This is an essential step to test and validate the modelling and adjustment of parameters of each previously proposed method, before applying them to real and more complex data, where the ideal optima is hard to determinate.

Each use case describes the experimentation protocol and algorithms' parameters, the problem database and the customer's requests as well as the results obtained followed by their analysis. Moreover, each use case tests different aspects of the proposed architecture and its methods. Methods evaluated in this chapter are: CPS-Matching ("Mtc"), CPS-DEA ("DEA"), CPS-GA ("GA"), CPS-BDE ("BDE"), CPS-DDE ("DDE") and CPS-SA ("SA"), Matching-GA ("Mtc-GA"), Matching-BDE ("Mtc-BDE"), Matching-DDE ("Mtc-DDE") and Matching-SA ("Mtc-SA").

The first case uses the scenario published at Moraes, Fiorese e Parpinelli (2018a). This database has only quantitative PIs that are characteristic of the laaS model. It is a practical start point that uses the proposed PI group and a replication approach of every ten CPs to create a bigger database for further stress each method.

The second case uses only qualitative PIs plus price. These PIs are not specific of any CC service model. So, this example allows to explore qualitative criteria and also uses different weights for each PI in the request. This database also can be replicated to stress each method with that parameters. A limit with 200 CP with price decrease (to eliminate multiple optima, due to replication) are found for cases 1 and 2.

Finally, the third case uses a real database, created by collecting and consolidating the information about CPs specialized in storage services at the Cloudwards website. There are a total of 61 CPs with five quantitative PIs. Different PI weights are used in the requests too. This use case aims to prove the suitability of the proposed architecture and its methods for data from the real world and also to prove the similarity of the simulated databases (cases 1 and 2) with the reality. This use case has less CPs, but it is easy to understand and shows how a database can be built in practice.

In general, deterministic methods are more stable (same type of response when it is used the same arguments each run regardless of database size). They present minimal execution time and can find optimal solutions for simpler requests, i.e., only need a single CP to attend entirely the request. The metaheuristic methods, specially EAs, can find very satisfactory answers even for the most complex request,

i.e., requests that need several CPs to maximize their attendance, in an acceptable execution time. This fact ensures the validity of the proposed fitness/energy function for the problem, which is one of the main contributions of this work. Finally, the hybrid methods presents the most optimized answers for larger databases, since they combine both the qualities of deterministic (quick answers, stable, single run, efficient finding of the single best CP) and metaheuristics (find the best CP set for complex requests), making it an appropriate choice for real applications. The hybrid method CPS-DDE presents the most positive statistically significant performance and has the highest average hits percentage for all the tested use cases, becoming the most prominent method implemented for solving complex CPS requests.

## 6 FINAL CONSIDERATIONS

The Cloud Computing (CC) service model allows a large and on demand hosting and distribution of computational resources around the world using computer networks. CC can optimize the use of computational resources and it is a service easily accessible via Internet. The success and popularization of computing clouds inspired the emergence of a large number of new companies providing CC services, the so called Cloud Providers (CPs). This rapid growth in the quantity of CPs does not mean a guaranteed CC quality service and can difficult a precise CP choice by customers. Thus, selecting which CPs are the most suitable to each customer has become a complex problem. Thus, the need for an automated selection method became necessary. In order to quantify the quality of each available CP, selection methods need metrics, like PI. PIs are tools for systematic information collection about crucial aspects of an organization. PIs are responsible for quantifying the studied objects and can be quantitative (discrete or continuous) or qualitative (ordered or unordered).

The Cloud Provider Selection (CPS) problem has become noticeable in the 2010 decade and several works already has been developed in this area. However, many of those works are disconnected and should start from a common background (integrative problem modelling, that allows to build an extensible framework using several selection methods). Thus, the central idea adopted for the solution of the CPS problem is to find out the smallest CPs set that maximizes customer satisfaction with the lowest possible price. In this case, maximizing customer satisfaction means to maximize the attendance of the customer request. The request represents customer's computational needs comprising the CPs services necessary to achieve its goals.

The main objective of this work was achieved. In this sense, it was stated to propose a more general architecture to solve the CPS problem. The proposed architecture is composed by three modules: 1) CP database, 2) customer request and interface, and 3) a pool of selection methods. The database should present all information needed comprising the CPs' PIs. It can be built and consolidated by specialized third parties. The customer request represent all its computational needs regarding the CPs represented as a PI list. This list can be built with the aid of an auxiliary interface with direct communication with the CPs database. Finally, the selection methods are responsible for selecting the most appropriate CP to the customer. To accomplish that, they use customer request and the PIs data available in the CPs database.

This architectural solution is based on PIs of the several and various candidate CPs and the customer requests. In addition, the proposed architecture provides

the respective modelling and implementation of particular selection methods to solve the CPS problem. In order to accomplish this objective, this work reviewed basic concepts like CC, PI and selection methods; delimited some PI for the CPS problem; presented a bibliographical review comparing related works with the proposed one; and proposed and implemented several diverse selection methods to deal with the problem, providing a pool of selection methods. These methods are classified in deterministic (CPS-Matching and CPS-DEA), metaheuristic (CPS-GA, CPS-BDE, CPS-DDE and CPS-SA) and hybrid ones (CPS-Matching plus a metaheuristic method, in pipeline).

Several experiments are performed with different simulated and real databases to prove these methods suitability for the CPS problem. Three use case examples are tested. The first two cases use simulated CP PI-databases and the last one uses real data. The first use case only process quantitative PIs and the second one uses mainly qualitative ones. The simulated databases are simple, feasible and represent a controlled scenario with a well-know optima for each request. These databases are used to validate the modelling and to adjust parameters for each implemented selection method. With these parameters adjusted, the real PIs use case could be solved with almost 100% of accuracy. Results are computed and analysed and methods compared to each other, showing their qualities. The analysis of the results lead to the following observations. Deterministic methods have stable responses regardless of the request and/or database size, and they present the lowest execution time. Overall, they are very good for simpler requests with 100% of customer request attendance. Metaheuristic methods are good for complex requests, because they can find the best CP set that maximize the request attendance. The hybrid methods present the most optimized answers for larger databases, since they combine both the qualities of deterministic and metaheuristics ones, making them appropriate choices for real applications. These conclusions are supported by the Shapiro-Wilk and Dunn test with 5% of significance level, which are performed to evaluate statistical significance among the results. This result analysis uncovered the observation that CPS-DDE presents the most positive statistically significant performance and its hybrid counterpart, Matching-DDE, has the highest average hits percentage for all the tested use cases and it is probably the most reliable and prominent hybrid method implemented for solve complex requests.

This developed architecture is easily expandable and it is a valuable support tool for the customer decision-making process. This work provides several contributions. The main ones comprise the CPS problem solving modelling as a general software architecture using CP's PIs as easy data inputs and outputs. In addition, it is highlighted the modelling information representing an individual and the suitable fitness and energy functions used by the metaheuristic and hybrid methods. Also, the mathematical modeling leading to a set of equations comprising how to transform the

problem input data into inputs and outputs to the CPS-DEA method. Moreover, a general and agnostic ontology enabling the proposed architecture to deal with qualitative data as well as a tolerance mechanism coping with small differences between customer requested PIs and CP's offering. At the end, it is also presented a comparative table of related works and a list of PIs for CPs derived from bibliographic research.

Finally, taking into account contributions provided, it is noteworthy that the proposed architecture is a solid base for customer's better understanding of the CPS problem reasoning the importance and the impact of a correct CP choice. At last, with this architecture, the customers can become more judgmental in their choices and help leveraging healthy competition between providers for the improvement of their own services and consequently their PIs making them be selected more often.

## 6.1 RECOMMENDATIONS FOR FUTURE WORK

Taking into consideration this work improvement, some recommendations and further research work can be stated. In this sense, more experiments and tests are needed, especially with real cases, with larger databases, i.e., more CPs providing vertical scalability and more PIs also providing results for horizontal scalability. This is important for further stress the algorithms, aiming to know the parameter thresholds in order to use them as arguments for other strategies and maybe to find specific heuristics related to the problem itself. These heuristics are strategies related to the CPS problem nature and are used to make several cuts in the search space. The implementation of a simple local search routine (deterministic or stochastic) in order to improve the final response given by the metaheuristic/hybrid methods can further increase the overall assertiveness of each method.

Moreover, in addition, these tests can be performed with different weights to the PIs, as well as for the fitness parameters for price ( $w_p$ ), quantity of providers ( $w_n$ ) and penalty ( $w_{pen}$ ). These parameters can also be adjusted using strategies of online control parameters of the methods provided by the architecture.

A multi-objective approach applied to the fitness and energy function modeling is also a very promising idea. So, the weights  $w_p$ ,  $w_n$  and  $w_{pen}$  become unnecessary. This eliminates the need to adjust those parameters making the study of the influence of the three main components that affects the fitness/energy function more transparent and controllable.

The proposed CPS solving architecture can be aggregated in a single model with standardized inputs and outputs, creating a CPS framework. The idea is to make this framework easily expandable for future uses and improvements. To accomplish that, additional different selection method modellings, like TOPSIS and/or combina-



tions of other MCDA methods adapted to the CPS problem, can be developed and incorporated, constantly. Also, it can be improved by the development of intelligent routines for choosing the most appropriate methods to perform the CP selection, using the characteristics of the current CP database, such as the total amount of CPs and PIs. Moreover, other enhancement could be making available to the customer the choice of which methods it wants to test, comparing results, increasing its reliance in the final choice.

A useful auxiliary work could better specify how to effectively build and maintain a large CP database with several providers and their PIs. This database must be reliable, representing the truth, consistent, updated, secure and always available. In practice, a strong external entity (third party), independent of the CPs whose PIs are being measured, would be needed to act as an intermediary and monitor.

Also, a possible future work includes the implementation of this selection framework for real-world applications, in large-scale utilization, i.e., making its functionality available as a service through an online platform. Finally, this framework can be used as base to other selection purposes, i.e., not only for selecting CPs, using different criteria beyond PIs.

## BIBLIOGRAPHY

- ACHAR, R.; THILAGAM, P. A broker based approach for cloud provider selection. In: **2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI 2014)**. Delhi, India: IEEE, 2014. p. 1252–1257.
- ALABOOL, H. M.; MAHMOOD, A. K. Review on cloud service evaluation and selection methods. In: IEEE. **Proceedings of the 6th International Conference on Research and Innovation in Information Systems (ICRIIS 2013)**. Salerno, Italy, 2013. p. 61–66.
- ALMANEA, M. I. M. A survey and evaluation of the existing tools that support adoption of cloud computing and selection of trustworthy and transparent cloud providers. In: IEEE. **2014 International Conference on Intelligent Networking and Collaborative Systems (INCoS 2014)**. Malaysia: IEEE SMCS, 2014. p. 628–634.
- ANDRÉ, L.; PARPINELLI, R. S. The multiple knapsack problem approached by a binary differential evolution algorithm with adaptive parameters. **Polibits**, scielomx, p. 47–54, 06 2015.
- AO, R. S.; GAMA, J. A study on change detection methods. In: **New Trends in Artificial Intelligence. 14th Portuguese Conference on Artificial Intelligence (EPIA)**. Aveiro, Portugal: Springer, 2009. p. 353–364.
- BARANWAL, G.; VIDYARTHI, D. P. A framework for selection of best cloud service provider using ranked voting method. In: **2014 IEEE International Advance Computing Conference (IACC 2014)**. Gurgaon, India: IEEE, 2014. p. 831–837.
- BEDI, P.; KAUR, H.; GUPTA, B. Trustworthy service provider selection in cloud computing environment. In: **2012 International Conference on Communication Systems and Network Technologies**. Rajkot, Gujarat, India: IEEE, 2012. p. 714–719.
- BHATTACHARYA, S.; MUKHERJEE, T.; DASGUPTA, K. Cloudrank: A statistical modelling framework for characterizing user behaviour towards targeted cloud management. In: **14th IEEE/IFIP Network Operations and Management Symposium (NOMS 2014)**. Krakow, Poland: IEEE, 2014. p. 1–8.
- BRANS, J.; VINCKE, P. A preference ranking organization method: The promethee method for multiple criteria decision-making. **Management Science**, INFORMS, v. 31, n. 8, p. 647–656, June 1985.
- CHAN, H.; CHIEU, T. Ranking and mapping of applications to cloud computing services by svd. In: **12th IEEE/IFIP Network Operations and Management Symposium Workshops (NOMS 2010)**. Osaka, Japan: IEEE, 2010. p. 362–369.
- CHARNES, A.; COOPER, W.; RHODES, E. Measuring the efficiency of decision making units. **European Journal Of Operational Research**, v. 2, n. 6, p. 429–444, November 1978.
- COOK, W.; KRESS, M. A data envelopment model for aggregating preference rankings. **Management Science**, INFORMS, v. 36, n. 11, p. 1302–1310, 1990.

CSMIC. **Service Measurement Index Framework**. 2.1. ed. Moffett Field, California, 2014.

DHIVYA, R.; DEVI, R.; SHANMUGALAKSHMI, R. Parameters and methods used to evaluate cloud service providers: A survey. In: IEEE. **2016 International Conference on Computer Communication and Informatics (ICCCI 2016)**. Coimbatore, India: IEEE, 2016. p. 1–5.

DUNN, O. J. Multiple comparisons using rank sums. **Technometrics**, Taylor & Francis, v. 6, n. 3, p. 241 – 252, 1964.

EDEN, A. H. Three paradigms of computer science. **Minds Machines**, Kluwer Academic Publishers, Hingham, MA, USA, v. 17, n. 2, p. 135–167, July 2007.

GARG, S. K.; VERSTEEG, S.; BUYYA, R. Smicloud: A framework for comparing and ranking cloud services. In: **Proceedings of the 4th IEEE International Conference on Utility and Cloud Computing (UCC 2011)**. Melbourne, Australia: IEEE, 2011. p. 210–218.

GARG, S. K.; VERSTEEG, S.; BUYYA, R. A framework for ranking of cloud computing services. **Future Generation Computer Systems**, v. 29, p. 1012–1023, June 2013.

GERHARDT, T. E.; SILVEIRA, D. T. **Métodos de pesquisa**. Porto Alegre, Brazil: Plageder, 2013.

GHOSH, N.; GHOSH, S. K.; DAS, S. K. Selcsp: A framework to facilitate selection of cloud service providers. **IEEE transactions on cloud computing**, IEEE, v. 3, n. 1, p. 66–79, 2015.

GIL, A. C. **Como elaborar projetos de pesquisa**. Brazil: Atlas S.A., 2008.

GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization, and Machine Learning**. University of Michigan: Addison-Wesley, 1989.

HÖFER, C. N.; KARAGIANNIS, G. Cloud computing services: taxonomy and comparison. **Journal of Internet Services and Applications**, v. 2, p. 81–94, September 2011.

HOGAN, M. D. et al. **Nist Cloud Computing Standards Roadmap**. USA: NIST Special Publication 500 Series, 2013.

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence**. Bradford Books: MIT Press, 1975.

HWANG, C.-L.; YOON, K. Methods for multiple attribute decision making. In: **Multiple attribute decision making**. USA: Springer, 1981. p. 58–191.

ISHIZAKA, A.; NEMERY, P. **Multi-Criteria Decision Analysis: Methods and Software**. United Kingdom: John Wiley & Sons, Ltd, 2013.

JAIN, R. **The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling**. Littleton, Massachusetts: John Wiley & Sons, 1991.

JAISWAL, A.; MISHRA, R. Cloud service selection using topsis and fuzzy topsis with ahp and anp. In: ACM. **Proceedings of the 2017 International Conference on Machine Learning and Soft Computing**. Ho Chi Minh City, Vietnam (ICMLSC 2017): ACM, 2017. p. 136–142.

JONG, K. A. D. **Evolutionary computation: A unified approach**. Cambridge, MA: MIT Press, 2006.

KARIM, R.; DING, C.; MIRI, A. An end-to-end qos mapping approach for cloud service selection. In: IEEE. **2013 IEEE Ninth World Congress on Services**. Santa Clara, California, USA: IEEE, 2013. p. 341–348.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. **Science**, American Association for the Advancement of Science, v. 220, n. 4598, p. 671–680, 1983.

KORE, N. B.; RAVI, K.; PATIL, S. B. A simplified description of fuzzy topsis method for multi criteria decision making. **International Research Journal of Engineering and Technology (IRJET)**, v. 4, p. 2047–2050, May 2017.

KRAUSE, J.; LOPES, H. S. A comparison of differential evolution algorithm with binary and continuous encoding for the mkp. In: **2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence**. Recife, Brazil: IEEE, 2013. p. 381–387.

KRAUSE, J.; PARPINELLI, R.; LOPES, H. Proposta de um algoritmo inspirado em evolucao diferencial aplicado ao problema multidimensional da mochila. Curitiba, PR:SBC, oct 2012.

LI, A. et al. Cloudcmp: comparing public cloud providers. In: ACM. **Proceedings of the 10th ACM SIGCOMM conference on Internet measurement**. Melbourne, Australia: ACM, 2010. p. 1–14.

LOHMANI, C.; FORTUIN, L.; WOUTERS, M. Designing a performance measurement system: A case study. **European Journal of Operational Research**, v. 156, p. 267–286, July 2004.

MACEDO, H.; SILVA, C. da; FERREIRA, L. A comparative analysis of multicriteria approaches for cloud services selection. In: **XLV Brazilian Symposium of Operational Research**. Natal, RN, Brazil: [s.n.], 2013.

MARTELLO, S.; TOTH, P. **Knapsack problems – Algorithms and Computer Implementation**. New York, USA: John Wiley & Sons, 1990.

METROPOLIS, N. et al. Equation of state calculations by fast computing machines. **Chemical Physics**, AIP Publishing LLC, v. 21, n. 6, p. 1087—1092, 1953.

MEZA, L. A. et al. Isyds– integrated system for decision support (siad – sistema integrado de apoio a decisao): a software package for data envelopment analysis model. **Pesquisa Operacional**, v. 25, n. 3, p. 493–503, 2005.

MIERS, C. et al. Análise de segurança para soluções de computação em nuvem. In: **SBRC 2014 Minicursos**. Florianópolis, Brazil: Sociedade Brasileira de Computação, 2014. v. 1.

MORAES, L.; FIORESE, A. A scoring method based on criteria matching for cloud computing provider ranking and selection. In: S. SMIALEK M., C. O. F. J. H. (Ed.). **Enterprise Information Systems**. Porto, Portugal: Springer, Cham, 2018. v. 321, cap. Software Agents and Internet Computing, p. 339–365.

MORAES, L.; FIORESE, A.; MATOS, F. A multi-criteria scoring method based on performance indicators for cloud computing provider selection. In: **Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017)**. Porto, Portugal: INSTICC, 2017. v. 2, p. 588–599.

MORAES, L.; FIORESE, A.; PARPINELLI, R. S. An evolutive scoring method for cloud computing provider selection based on performance indicators. In: **Proceedings of the 16th Mexican International Conference on Artificial Intelligence (MICAI 2017)**. Baja California, Mexico: Springer LNAI, 2017. p. 1–12.

MORAES, L.; FIORESE, A.; PARPINELLI, R. S. An evolutive hybrid approach to cloud computing provider selection. In: **Proceedings of IEEE Congress on Evolutionary Computation (CEC)**. Rio de Janeiro, Brazil: IEEE, 2018. p. 1564–1571.

MORAES, L.; FIORESE, A.; PARPINELLI, R. S. Exploring evolutive methods for cloud provider selection based on performance indicators. In: **Proceedings of the 7th Brazilian Conference on Intelligent Systems (BRACIS)**. São Paulo, Brazil: SBC, 2018. p. 157–162.

MORAES, L. B. de et al. An efficiency frontier based model for cloud computing provider selection and ranking. In: **Proceedings of the 20th International Conference on Enterprise Information Systems (ICEIS 2018)**. Madeira, Portugal: INSTICC, 2018. p. 543–554.

POLLOCK, W. K. Using key performance indicators (kpis) to measure and track the success of your services operation. **AFSMI's Sbusiness**, p. 1131–1152, September 2007.

RAMANATHAN, R. **An Introduction to Data Envelopment Analysis: A Tool for Performance Measurement**. London: Sage Publications, Ltd, 2003.

REHMAN, Z.; HUSSAIN, F.; HUSSAIN, O. Towards multi-criteria cloud service selection. In: **Proceedings of the 5th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing**. Seoul, South Korea: IEEE, 2011. p. 44–48.

REHMAN, Z.; HUSSAIN, F.; HUSSAIN, O. IaaS cloud selection using mcdm methods. In: **Proceedings of the 9th IEEE International Conference on e-Business Engineering**. Hangzhou, China: IEEE, 2012. p. 246–251.

ROY, B. The outranking approach and the foundations of electre methods. In: COSTA, C. A. Bana e (Ed.). **Readings in Multiple Criteria Decision Aid**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990. p. 155–183.

RUTENBAR, R. A. Simulated annealing algorithms: An overview. **Circuits and Devices Magazine**, IEEE, v. 5, n. 1, p. 19–26, January 1989.

SAATY, T. L. How to make a decision: The analytic hierarchy process. **European Journal of Operational Research**, v. 48, p. 9–26, September 1990.

SAATY, T. L. **Decision Making with Dependence and Feedback: The Analytic Network Process**. Pittsburgh, Pennsylvania: RWS Publications, 1996.

SAATY, T. L. Decision making – the analytic hierarchy and network processes (ahp/anp). **Journal of Systems Science and Systems Engineering**, v. 13, p. 1–35, March 2004.

SARI, B.; SEN, T.; KILIC, S. E. Ahp model for the selection of partner companies in virtual enterprises. **The International Journal of Advanced Manufacturing Technology**, v. 38, p. 367–376, August 2008.

SHIRUR, S.; SWAMY, A. A cloud service measure index framework to evaluate efficient candidate with ranked technology. **International Journal of Science and Research**, v. 4, March 2015.

SIEGEL, J.; PERDUE, J. Cloud services measures for global use: The service measurement index (smi). In: **Annual SRII Global Conference 2012**. San Jose, California, USA: SRII/IEEE, 2012. p. 411–415.

SQUIDDI, M. et al. An adaptive real time mechanism for iaas cloud provider selection based on qoe aspects. In: **Communications (ICC), 2015 IEEE International Conference on**. London, UK: IEEE, 2015. p. 6809–6814.

STORN, R.; PRICE, K. **Differential Evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces**. CA, USA, 1995.

SUNDARESWARAN, S.; SQUICCIARIN, A.; LIN, D. A brokerage-based approach for cloud service selection. In: **2012 IEEE 5th International Conference on Cloud Computing**. Honolulu, HI, USA: IEEE, 2012. p. 558–565.

TALBI, E.-G. **Metaheuristics: From Design to Implementation**. Hoboken, New Jersey: John Wiley & Sons, 2009.

WAGLE, S.; GUZEK, M.; BOUVRY, P. Cloud service providers ranking based on service delivery and consumer experience. In: **4th International Conference on Cloud Networking (CloudNet)**. Niagara Falls, ON, Canada: IEEE, 2015. p. 209–212.

WAGLE, S. et al. An evaluation model for selecting cloud services from commercially available cloud providers. In: **7th International Conference on Cloud Computing Technology and Science (CloudCom)**. Vancouver, BC, Canada: IEEE, 2015. p. 107–114.

WAZLAWICK, R. S. **Metodologia de pesquisa para ciência da computação**. Rio de Janeiro, Brazil: Elsevier, 2014.

WHADUZZAMAN, M. et al. Cloud service selection using multicriteria decision analysis. **The Scientific World Journal**, Hindawi Publishing Corporation, v. 2014, 2014.

WILK, M. B.; SHAPIRO, S. The joint assessment of normality of several independent samples. **Technometrics**, Taylor & Francis, v. 10, n. 4, p. 825–839, 1968.

YENIAY, Ö. Penalty function methods for constrained optimization with genetic algorithms. **Mathematical and Computational Applications**, Multidisciplinary Digital Publishing Institute, v. 10, n. 1, p. 45–56, 2005.

ZHANG, Q.; CHENG, L.; BOUTABA, R. Cloud computing: state-of-the-art and research challenges. **Journal of Internet Services and Applications**, v. 1, p. 7–18, April 2010.

## APPENDIX A – PERFORMANCE INDICATORS LISTS FOR CLOUD PROVIDERS

This appendix presents two lists that delimit the most common PIs used by other authors to evaluate different CPs. First list presents quantitative PIs and second one shows qualitative ones. These lists constitute an important general background for the construction of the related work comparative table as well as to delineate the most notable PI types, aiding the architecture to correct evaluate their utility based on type. Moreover, it was found that the PI cost/price is one of the most notable ones and it is very used, even on the SMI framework, directly influencing the CP evaluation in the architecture. Finally, all the PIs chosen for each use case with simulated data are derived from these lists, as well as assisted the identification of others in the real PI use case.

Thus, the quantitative PIs used for solving the CPSs problem can be an integer (discrete) or real (continuous). The surveyed PIs classified as predominantly quantitative are:

1. **Number of instances offered:** Total quantity of different possible instances to be created (distinct flavors – Refers to the amount of resources to be used in the user's Virtual Machine (VM)). **Expected data type:** Positive integer number (quantity). **References:** Sundareswaran, Squicciarini e Lin (2012).
2. **Cost:** The price paid for the service used, such as: dollars per hour (or day or month) of computing, per Gigabyte used, per Gigabyte transferred, per used VM, etc. **Expected data type:** Real number. **References:** Chan e Chieu (2010); Garg, Versteeg e Buyya (2011); Sundareswaran, Squicciarini e Lin (2012); Garg, Versteeg e Buyya (2013); Baranwal e Vidyarthi (2014); Bhattacharya, Mukherjee e Dasgupta (2014); Wagle et al. (2015); and Shirur e Swamy (2015).
3. **Number of supported Operational Systems (OSs):** Number of different OSs supported by the CP in question. **Expected data type:** Positive integer number (quantity). **References:** Sundareswaran, Squicciarini e Lin (2012).
4. **Number of supported platforms:** Refers to how many different platforms (Java, PHP, WinDev, etc.) are supported by the CP in question. **Expected data type:** Positive integer number (quantity). **References:** Baranwal e Vidyarthi (2014); and Shirur e Swamy (2015).
5. **Response time:** It measures how fast the CP service can be made available for use. It is usually an average time given in milliseconds. **Expected data type:**



Real number (unit of time). **References:** Garg, Versteeg e Buyya (2011); Garg, Versteeg e Buyya (2013); Baranwal e Vidyarthi (2014); Shirur e Swamy (2015); and Wagle et al. (2015).

6. **Availability:** It is the estimated percentage of time that a customer can access the service via network, i.e., how long the CP service remains available online in a given period of time (usually years or months). **Expected data type:** Real number (%). **References:** Chan e Chieu (2010); Garg, Versteeg e Buyya (2011); Garg, Versteeg e Buyya (2013); Baranwal e Vidyarthi (2014); Shirur e Swamy (2015); and Wagle et al. (2015).
7. **Average repair time:** It measures the average time it takes for the CP to be recoverable from a crash or from an incident that adversely affected the availability of the CP. **Expected data type:** Real number (unit of time). **References:** Chan e Chieu (2010); and Wagle et al. (2015).
8. **Sustainability:** Indicates the degree of environmental impact when using the service. It can be measured against the average carbon emitted or the energy efficiency of the cloud service. **Expected data type:** Real number (amount of carbon emitted or % energy efficiency). **References:** Garg, Versteeg e Buyya (2011); Garg, Versteeg e Buyya (2013); Baranwal e Vidyarthi (2014); and Shirur e Swamy (2015).
9. **Accuracy:** It measures the degree of closeness between the actual values used (resources) and the values expected or promised to the customer by the CP. **Expected data type:** Real number (%). **References:** Garg, Versteeg e Buyya (2011); and Garg, Versteeg e Buyya (2013).
10. **Reliability:** Reflects how a service operates flawlessly during a given time and condition. It is defined based on the amount of average failure time promised by the CP and what the customer experienced in practice. That is, the probability that a CP will operate satisfactorily under the conditions specified/promised by himself in a defined period of time. **Expected data type:** Usually a real number (% or unit of time), but can also be given by fixed categories such as: unreliable, few reliable, regular, reliable, very reliable. **References:** Garg, Versteeg e Buyya (2011); Garg, Versteeg e Buyya (2013); Baranwal e Vidyarthi (2014); Shirur e Swamy (2015); and Wagle et al. (2015).
11. **Stability:** Indicates the variability in the performance of a CP service. For storage, it is the variance of the average time to read and write data in the cloud. For computational resources, it is the measured performance deviation regarding the

specified in the Service Level Agreement (SLA). **Expected data type:** Real number (% or unit of time). **References:** Garg, Versteeg e Buyya (2011); Garg, Versteeg e Buyya (2013); Baranwal e Vidyarthi (2014); and Shirur e Swamy (2015).

12. **Adaptability:** It is the ability of the CP to adjust the changes in services based on the customer's request. It may be the time needed to change or evolve the service to the next level. **Expected data type:** Real number (unit of time). **References:** Garg, Versteeg e Buyya (2011); and Garg, Versteeg e Buyya (2013).
13. **Elasticity:** Express how much the CP can be escalated during peak times. It may be the time required to expand or contract the service attendance capacity, the maximum service capacity and maximum number of computing units provided at peak times. **Expected data type:** Real number (unit of time, capacity). **References:** Garg, Versteeg e Buyya (2011); Garg, Versteeg e Buyya (2013); Baranwal e Vidyarthi (2014); and Shirur e Swamy (2015).
14. **Throughput and efficiency:** Throughput is the number of tasks completed by the CP per unit of time. Efficiency is related to efficient use of contracted services (higher values indicate a lower overhead in the system). **Expected data type:** Real number (task per second or %). **References:** Garg, Versteeg e Buyya (2013); Baranwal e Vidyarthi (2014); Shirur e Swamy (2015); and Wagle et al. (2015).
15. **Scalability:** Indicates whether a system can handle a large number of application requests simultaneously. It has two dimensions: horizontal ("scale out") and vertical ("scale up"). Horizontal scalability means raising similar resources in the cloud (initializing more VMs of the same type) during the peak periods. Vertical scalability is the ability to extend the capacity of cloud services (e.g., increase the physical memory or CPU speed or network bandwidth of a VM). **Expected data type:** Real number (proportion or %). **References:** Garg, Versteeg e Buyya (2013); Baranwal e Vidyarthi (2014); and Shirur e Swamy (2015).
16. **Capacity:** Means the maximum amount of resources that a CP can provide at peak times. These capabilities can be quantified by: CPU capacity (multiplying the number of CPUs cores by their frequency), memory capacity (GB), storage capacity (GB) and network capacity (measured by bandwidth). **Expected data type:** Integer or real numbers. **References:** Baranwal e Vidyarthi (2014).

The qualitative or categorical PIs used for the CPSs problem can be ordered or unordered (simple, compound or boolean). The surveyed PIs classified as predominantly qualitative are:

1. **Service type:** Indicates the nature of the on-demand service offered, such as: processing, storage, connectivity, software, IPs, etc. **Expected data type:** A set of categories (qualitative unordered compound). **References:** Sundareswaran, Squicciarini e Lin (2012).
2. **Security level:** Denotes the level of security and/or privacy of the CP. It can be represented by three basic categories: high, medium and low. It can be divided into more specific subcriteria, such as: authentication, encryption, auditability, physical security of the installation, etc. **Expected data type:** A category (qualitative ordered). **References:** Chan e Chieu (2010); Sundareswaran, Squicciarini e Lin (2012); Baranwal e Vidyarthi (2014); Bhattacharya, Mukherjee e Dasgupta (2014); Shirur e Swamy (2015); and Wagle et al. (2015).
3. **Types of OSs supported:** Refers to the names of the different OSs available or supported to the customer instances, such as: MS-Windows, GNU/Linux, etc. **Expected data type:** A set of categories (qualitative unordered compound). **References:** Baranwal e Vidyarthi (2014); and Shirur e Swamy (2015).
4. **Types of supported platforms:** Refers to the names of the different platforms (Java, PHP, WinDev, etc.) supported by the CP. **Expected data type:** Set of categories (qualitative unordered compound). **References:** Baranwal e Vidyarthi (2014); and Shirur e Swamy (2015).
5. **Quality of Service (QoS):** Determined by cloud brokers, responsible for analyzing and collecting, over time, information about the service provider, evaluating and comparing with other providers. It can be represented by three basic categories: low, medium, high. **Expected data type:** A category (qualitative ordered). **References:** Sundareswaran, Squicciarini e Lin (2012).
6. **Transparency:** Indicates the extension of how much the service usability/performance is affected during some change in the CP. It can be the time when the customer's application was affected during the change or calculated in terms of the frequency of such effects. **Expected data type:** qualitative ordered (low, medium, high), but can also be given in percentages (real number). **References:** Garg, Versteeg e Buyya (2011); and Garg, Versteeg e Buyya (2013).
7. **Interoperability:** It measures the ability of a service to interact with other services on both the same CP and other CPs. Can be defined by customer experience. **Expected data type:** Usually, it is qualitative ordered (low, medium, high) or a percentage. **References:** Garg, Versteeg e Buyya (2011); and Garg, Versteeg e Buyya (2013).

8. **Usability:** Indicates the ease of using a cloud service (operability, ability to learn, installability and intelligibility). **Expected data type:** A category (qualitative ordered). **References:** Garg, Versteeg e Buyya (2011); and Garg, Versteeg e Buyya (2013).
9. **Customer interface:** Informs the presence or absence of an interface to serve the customer, such as: website, Application Programming Interface (API), by a management console, etc. **Expected data type:** Boolean. **References:** Baranwal e Vidyarthi (2014); and Shirur e Swamy (2015).
10. **Free trial:** Informs if the CP in question offers a free trial of their services. It can be very useful for customers, since it is possible to test the services before they are deployed and without paying. **Expected data type:** Boolean. **References:** Baranwal e Vidyarthi (2014); and Shirur e Swamy (2015).
11. **User experience:** It represents the level of user experience and satisfaction with the service, based on different users opinions for the same CP. **Expected data type:** Qualitative ordered (expressed by a name or number). **References:** Baranwal e Vidyarthi (2014); and Shirur e Swamy (2015).
12. **Virtualization technique:** Informs what types of virtualizations and hypervisors are supported by the CP (Hyper-V, VMware, Xen, Vsphere, etc.). Generally, a CP only supports one virtualization platform (each one with its pros and cons). The choice depends on the requirements of the applications. **Expected data type:** A category (qualitative unordered simple). **References:** Baranwal e Vidyarthi (2014).
13. **Customer support:** Indicates if the CP offers some support to its customers, paid or free, and other data, such as the type of support offered, the average response time spent in this process and the burden of effort deposited on the customer. **Expected data type:** A category (qualitative ordered). **References:** Baranwal e Vidyarthi (2014).
14. **Monitoring:** Indicates the degree of the CP commitment with the provision of monitoring resources to its customers. Whether by the use of measuring tools or indicators and alerts. It has three levels: poor, medium and extensive. **Expected data type:** A category (qualitative ordered). **References:** Baranwal e Vidyarthi (2014).

## APPENDIX B – CONSISTENCY VERIFICATION OF A JUDGEMENT MATRIX

This appendix presents the mechanisms required for consistency verification of an AHP Judgement Matrix, as well as an example of AHP Judgement Matrix solving with generic PI "level" numbers.

It is necessary to keep in mind that, it is important to maintain/verify the consistency of the Judgement Matrix for the results to be viable. Therefore, a consistency check can be performed to detect possible contradictions and inconsistencies in the matrix inputs. This check is done right after a Judgement Matrix is filled. This must be done because when several comparisons of successive pairs are presented, they may contradict, for several reasons, such as: loosely defined problems, insufficient information, uncertain information, or lack of concentration (ISHIZAKA; NEMERY, 2013). As human nature is often inconsistent, AHP allows an inconsistency of up to 10% compared to the mean inconsistency of 500 randomly matched matrices (calculation made via supporting software), indicating whether the matrix needs to be reconsidered or not (ISHIZAKA; NEMERY, 2013), since small inconsistencies are common and do not entail major problems (SARI; SEN; KILIC, 2008).

The basic rules for maintaining data consistency in Judgement Matrix are (ISHIZAKA; NEMERY, 2013):

1.  $a_{ij} > 0 \longrightarrow$  Positive numbers for the priorities/importances;
2.  $a_{ii} = a_{jj} = 1 \longrightarrow$  Main diagonal is unitary;
3.  $a_{ij} = \frac{1}{a_{ji}} \longrightarrow$  Reciprocity; and
4.  $a_{ij} = a_{ik} * a_{kj} \longrightarrow$  Transitivity.

For all  $i, j, k = 1, 2, 3, 4, \dots, n$ , where  $n$  is the dimension of the matrix. For such verification, we first need to calculate the Consistency Index (CI) (Equation B.1) followed by the Consistency Ratio (CR) of the square matrix with dimension  $n$ , according to Equation B.2.

$$CI = \frac{(\lambda_{max} - n)}{(n - 1)} \quad (B.1)$$

$$CR = \frac{CI}{RI} \quad (B.2)$$

Where  $\lambda_{max}$  represents the maximum eigenvalue of the matrix and Random Index (RI) the average CI of 500 randomly filled matrices. In linear algebra, a real number  $\lambda$  is an eigenvalue of an matrix  $A_{n \times n}$ , if there is a non-zero vector  $x_n$  so that:

$A*x = \lambda*x$ . For the variable RI, there are some recommendations of values, according to Saaty (2004), as the Judgement Matrix, under analysis, with dimension  $n$ , varies. These values are listed in Table 26.

Table 26 – Recommended RI values for a Judgement Matrix with dimension  $n$

<b>n</b>	1	2	3	4	5	6	7	8	9	10
<b>RI</b>	0	0	0,52	0,89	1,11	1,25	1,35	1,40	1,45	1,49

Source: Adapted from Saaty (2004)

Thus, using the values in Table 26 and calculating CI via Equation B.1, it is possible to calculate the value of CR via Equation B.2, to a Judgement Matrix with dimension  $n = 3, 4, \dots, 10$ . For the Judgement Matrix to be considered consistent, the CR value must be less than 0,10, meaning it has less than 10 % of inconsistencies.

## B.1 SOLVING AN AHP JUDGEMENT MATRIX

As already stated, Judgement Matrix is an efficient technique for calculating weights or importance levels for each PI (MORAES; FIORESE; MATOS, 2017). Therefore, the AHP Judgement Matrix is a matrix with dimension  $n$ , where each row and each column represents a different weight/level, arranged in descending order of importance (from top to bottom – lines, from left to right – columns). Table 27 presents an example of possible Judgement Matrix. This matrix models relations of importance between each different PI level using Saaty scale and following the consistency rules already mentioned.

Table 27 – Judgement Matrix: Relations of importance between each different PI level

<b>Levels</b>	Essential	High	Medium	Low
Essential	1	2	4	9
High	1/2	1	2	6
Medium	1/4	1/2	1	3
Low	1/9	1/6	1/3	1
Sum Col.	1,8611	3,6667	7,3333	19,00

Following the Judgement Matrix technique, Judgement Matrix normalization takes place. This process takes each column element divided by its “Sum Col.” position, according Table 27. Results can be seen in Table 28, which represents Table 27 normalized.

Finally, the weights for each level are calculated by the average of each row of the normalized matrix. The sum of the four weights is always 1.

The consistency check can be performed in Table 28 to detect possible contradictions and inconsistencies. Thus, the maximum matrix eigenvalue  $\lambda_{max} = 4,01$ .

Table 28 – Normalized Judgement Matrix for each importance level

<b>Levels</b>	Essential	High	Medium	Low	<b>Weights</b>
Essential	0,5373	0,5455	0,5455	0,4737	0,5255
High	0,2687	0,2727	0,2727	0,3158	0,2825
Medium	0,1343	0,1364	0,1364	0,1579	0,1413
Low	0,0597	0,0455	0,0455	0,0526	0,0508

Therefore,  $CI = 0,003$  according to Equation B.1. Taking  $RI = 0,89$  (because matrix dimension  $n = 4$ ) and Equation B.2, then  $CR = 0,004$ , approximately, indicating a consistent Judgement Matrix ( $CR < 0,1$ ).

## APPENDIX C – STATISTICAL EVALUATION OF EXPERIMENTAL RESULTS

This appendix presents all the statistical result tables for Dunn's test with 5% significance level ( $\alpha = 0.05$ ), with Bonferroni  $\alpha$  correction, applied after Dunn's test. This correction inflates Type I error (incorrectly rejecting a null hypothesis, making a false positive, and it is more likely when conducting multiple analyses on the same dependent variable) by testing each individual hypothesis at a significance level of  $\frac{\alpha}{m}$ , where  $\alpha$  is the desired significance level and  $m$  is the number of hypotheses, which in this case is  $m = 2$ .

Dunn's tests are performed using as input the outputs (hit or miss) of each execution for each metaheuristic method and consequently in their hybrid counterparts too. Deterministic methods are not included because they do not have stochastic behavior, so the execution output always will be the same value when keeping the same arguments regardless how many iterations are executed. Each table shows p-values associated with developed methods hits in their respective rows and columns for each different request. These values range from 0 (completely different) to 1 (completely equal). Values in bold are from those methods considered statistically different (p-value  $< 0.05$ ) according to the Dunn's test and Bonferroni alpha correction. To that matter, two methods are statistically different if the difference between their results is strong enough that it is very unlikely to be obtained due to random factors such as experimental errors and lack of control of the variables involved. That is what Dunn's test is want to calculate.

Therefore, Table 29 presents Dunn's test results for the hit values presented in Table 13, corresponding to the Use Case 1, for each request and each metaheuristic method. Therefore, it can be concluded that CPS-DDE is statistically different from all the other methods in 2 out of 8 requests (2 and 3) and statistically equal for the other requests. Moreover, CPS-SA is statistically different in 6 out of 8 requests (1, 4, 5, 6, 7 and 8).

Table 30 presents Dunn's test results for hit values presented in Table 14, corresponding to use case 1, for requests 2, 3, 4 and 8 and each hybrid method. Requests 1, 5, 6 and 7 were not tested because their hits percentage are the same for all methods, thus having the same performance. Thus, analyzing Table 30 it is possible to state that in 2 out of 4 requests (i.e., requests 2 and 3) the "Mtc-DDE" is statistically better than "Mtc-GA", "Mtc-BDE" and "Mtc-SA". "Mtc-DDE" achieved statistically the same results than "Mtc-GA" for request 4 and the same results than "Mtc-BDE" for request 8. Finally, "Mtc-SA" is statistically different from all the other of requests 4 and 8.



Table 29 – Ex.1: Dunn test with Bonferroni correction for the hits values at Table 13.

<b>Req. 1</b>	GA	BDE	DDE	<b>Req. 2</b>	GA	BDE	DDE
BDE	<b>0.0018</b>	–	–	BDE	1.0	–	–
DDE	0.1032	0.5584	–	DDE	<b>0.0117</b>	<b>0.0049</b>	–
SA	<b>0.0002</b>	<b>0.0</b>	<b>0.0</b>	SA	<b>0.0261</b>	0.0546	<b>0.0</b>
<b>Req. 3</b>	GA	BDE	DDE	<b>Req. 4</b>	GA	BDE	DDE
BDE	0.1941	–	–	BDE	<b>0.0284</b>	–	–
DDE	<b>0.0413</b>	<b>0.0</b>	–	DDE	1.0	<b>0.0130</b>	–
SA	0.0935	1.0	<b>0.0</b>	SA	<b>0.0</b>	<b>0.0022</b>	<b>0.0</b>
<b>Req. 5</b>	GA	BDE	DDE	<b>Req. 6</b>	GA	BDE	DDE
BDE	0.0959	–	–	BDE	0.2522	–	–
DDE	1.0	0.0959	–	DDE	1.0	1.0	–
SA	<b>0.0</b>	<b>0.0</b>	<b>0.0</b>	SA	<b>0.0</b>	<b>0.0</b>	<b>0.0</b>
<b>Req. 7</b>	GA	BDE	DDE	<b>Req. 8</b>	GA	BDE	DDE
BDE	0.0811	–	–	BDE	1.0	–	–
DDE	0.5011	1.0	–	DDE	<b>0.0059</b>	0.0607	–
SA	<b>0.0</b>	<b>0.0</b>	<b>0.0</b>	SA	<b>0.0059</b>	<b>0.0003</b>	<b>0.0</b>

Source: Author

Table 30 – Ex.1: Dunn's test with Bonferroni correction for the hits values at Table 14.

<b>Req. 2</b>	Mtc-GA	Mtc-BDE	Mtc-DDE	<b>Req. 3</b>	Mtc-GA	Mtc-BDE	Mtc-DDE
Mtc-BDE	1.0	–	–	Mtc-BDE	0.1941	–	–
Mtc-DDE	<b>0.0117</b>	<b>0.0049</b>	–	Mtc-DDE	<b>0.0413</b>	<b>0.0</b>	–
Mtc-SA	<b>0.0261</b>	0.0546	<b>0.0</b>	Mtc-SA	0.0935	1	<b>0.0</b>
<b>Req. 4</b>	Mtc-GA	Mtc-BDE	Mtc-DDE	<b>Req. 8</b>	Mtc-GA	Mtc-BDE	Mtc-DDE
Mtc-BDE	<b>0.0284</b>	–	–	Mtc-BDE	1.0	–	–
Mtc-DDE	1.0	<b>0.0130</b>	–	Mtc-DDE	<b>0.0059</b>	0.0607	–
Mtc-SA	<b>0.0</b>	<b>0.0022</b>	<b>0.0</b>	Mtc-SA	<b>0.0059</b>	<b>0.0003</b>	<b>0.0</b>

Source: Author

Table 31 presents Dunn's test results for the hits percentage presented in Table 19 (use case 2) for each request and each metaheuristic method. Analyzing Table 31, it can be concluded that CPS-DDE is statistically better than the other metaheuristic methods only at request 5. On the other hand, the CPS-GA is statistical different from all the other methods at request 2 and CPS-SA is different for all requests, except request 5 for CPS-GA and CPS-BDE.

Table 32 presents Dunn's test results for the hit values presented in Table 20 (use case 2) for requests 3, 4 and 5 and each hybrid method. Requests 1 and 2 were not tested because their hits percentage are the same for all methods, thus having the same performance. Analyzing Table 32, it can be concluded that "Mtc-DDE" is statistically different than any other methods for request 5 and statistically equal for the others, while "Mtc-SA" is statistically different from the other ones at requests 3 and 4.

Table 31 – Ex.2: Dunn's test with Bonferroni correction for the hits values at Table 19.

<b>Req. 1</b>	GA	BDE	DDE	<b>Req. 2</b>	GA	BDE	DDE
BDE	0.0874	–	–	BDE	<b>0.0037</b>	–	–
DDE	1.0	0.3053	–	DDE	<b>0.0211</b>	1.0	–
SA	<b>0.0</b>	<b>0.0</b>	<b>0.0</b>	SA	<b>0.0</b>	<b>0.0</b>	<b>0.0</b>
<b>Req. 3</b>	GA	BDE	DDE	<b>Req. 4</b>	GA	BDE	DDE
BDE	1.0	–	–	BDE	0.1178	–	–
DDE	0.0618	0.2152	–	DDE	1.0	0.0611	–
SA	<b>0.0025</b>	<b>0.0003</b>	<b>0.0</b>	SA	<b>0.0</b>	<b>0.0024</b>	<b>0.0</b>
<b>Req. 5</b>	GA	BDE	DDE	–	–	–	–
BDE	0.7737	–	–	–	–	–	–
DDE	<b>0.0056</b>	<b>0.0001</b>	–	–	–	–	–
SA	0.0710	0.7737	<b>0.0</b>	–	–	–	–

Source: Author

Table 32 – Ex.2: Dunn test with Bonferroni correction for the hits values at Table 20.

<b>Req. 3</b>	Mtc-GA	Mtc-BDE	Mtc-DDE	<b>Req. 4</b>	Mtc-GA	Mtc-BDE	Mtc-DDE
Mtc-BDE	1.0	–	–	Mtc-BDE	0.1178	–	–
Mtc-DDE	0.0618	0.2152	–	Mtc-DDE	1.0	0.0611	–
Mtc-SA	<b>0.0025</b>	<b>0.0003</b>	<b>0.0</b>	Mtc-SA	<b>0.0</b>	<b>0.0024</b>	<b>0.0</b>
<b>Req. 5</b>	Mtc-GA	Mtc-BDE	Mtc-DDE	–	–	–	–
Mtc-BDE	0.7737	–	–	–	–	–	–
Mtc-DDE	<b>0.0056</b>	<b>0.0001</b>	–	–	–	–	–
Mtc-SA	0.0710	0.7737	<b>0.0</b>	–	–	–	–

Source: Author