Democratic and Popular Republic of Algeria
الجمهورية الجزائرية الديمقراطية الشعبية
Ministry of Higher Education and Scientific Research
وزارة التعليم العالي و البحث العلمي
Abbes Laghrour University -Khenchela-
جامعة عباس لغرور خنشلة

Faculty of Science and Technology
Department of Mathematics and Computer Science

Thesis

To obtain the degree of DOCTOR IN COMPUTER SCIENCE

Option : Security and Web Technologies

Learning a multi-criteria decision model for cost and performance optimization in the Cloud

Presented by :
Youcef BEZZA

Presented publically in :

Dr. Jamel NESSAH	Abbes Laghrour University Khenchela	President
Pr. Ouassila HIOUAL	Abbes Laghrour University Khenchela	Supervisor
Dr. Ouided HIOUAL	Abbes Laghrour University Khenchela	Co-supervisor
Dr. Hichem RAHAB	Abbes Laghrour University Khenchela	Examiner
Pr. Saber BENHARZALLAH	Mostafa Benboulaid University Batna2	Examiner
Dr. Toufik MARIR	Larbi Ben Mehidi University Oum El Bouaghi	Examiner

Defended : 2024

Acknowledgements

Before inviting you to the presentation of this work, I have the opportunity to express my gratitude to all the people who, through their help and encouragement have enabled me to complete this dissertation.

First of all, I would like to thank Pr. Ouassila HIOUAL and Dr. Ouided HIOUAL for the trust they have kindly granted me, for Their precious help, their advice, as well as their availability.

My thanks go to the honourable president of the jury for agreeing to examine my work.

May the members of this prestigious and distinguished jury be assured of my gratitude for having done me the honour of evaluating my work.

Dedication :

I dedicate this modest work to : MY DEAR PARENTS

No dedication can express my respect, eternal love and consideration

Eternal love and consideration for the sacrifices you have made for my education and well-being.

I thank you for all the support and love you have shown me

since I was a child, and I hope that your blessing will always be with me.

May this modest work be the fulfillment of your long-held wishes

the fruit of your countless sacrifices, though I can never repay you enough.

my brother Houssem-eddine, and my sisters Ikram and Nihal.

Also all my friends and dear ones that stoud by my side in the toughest moments.

For their sincere friendship and trust, and to whom I owe my gratitude and attachment.

To all those who are dear and close to me, To my colleagues.

Abstract

Cloud computing has emerged as a transformative force in modern computing, offering ubiquitous access to on-demand services that cater to diverse client needs. As this paradigm continues to permeate everyday life, the optimization of cloud provider characteristics has become paramount. Combinatorial optimization, drawing upon discrete mathematics and computer science techniques, serves as a vital mechanism to address real-life challenges in this domain. However, the proliferation of numerous cloud providers, coupled with a myriad of selection criteria, has rendered decision-making increasingly complex.With the growing array of cloud providers, it becomes increasingly important to optimize the performance of these services to effectively meet the demands of users and enterprises alike. This thesis presents novel methodologies aimed at enhancing cost-efficiency and performance within cloud computing environments, harnessing sophisticated multicriteria decision support systems. Indeed, we introduce two distinct contributions to address the challenges associated with performance optimization.

The first contribution of this thesis addresses the pressing need for robust decision-making frameworks in cloud service selection. By proposing a multicriteria decision support system model, we introduce a novel approach that integrates a modified neural network architecture with two hidden layers. In this model, the first layer employs the ELECTRE technique to assess cloud services based on multiple criteria, while the second layer utilizes a modified Tabu search algorithm, leveraging a weighted sum approach for efficient decision-making. Notably, the input for our modified neural network is cloud performance, allowing stakeholders to navigate the complex trade-offs between cost and performance effectively. Through this contribution, we aim to provide decision-makers with a comprehensive framework to optimize costs and performances in cloud computing, thus enhancing the overall efficiency and effectiveness of cloud service selection processes.

Building upon this foundation, the second contribution of this thesis introduces a pioneering approach termed Multi-Objective Optimization Approach for Cloud Services Finding (MOOA-CSF). By harnessing supervised learning and multicriteria decision techniques, MOOA-CSF aims to optimize both price and performance in cloud computing environments. Through the utilization of an artificial neural network (ANN) to classify cloud services based on their features, coupled with the application of the ELECTRE method to order these services, MOOA-CSF offers a systematic framework to identify optimal cloud solutions tailored to the specific needs of clients and systems. Furthermore, by incorporating a modified genetic algorithm to generate hybrid cloud services, MOOA-CSF extends the boundaries of traditional optimization techniques, offering innovative solutions to complex optimization problems in cloud computing. Through rigorous evaluation using diverse scenarios, simulation results demonstrate the efficiency and effectiveness of our approach, underscoring its potential to enhance cloud computing performance optimization in real-world settings significantly.

Keywords: Multi-criteria decision-making, cloud computing, Multi-objective optimization, artificial neural networks, Tabu Search, genetic algorithm, similarity measures, supervised learning, Performance.

Résumé

L'informatique en nuage s'est imposée comme une force de transformation de l'informatique moderne, offrant un accès omniprésent à des services à la demande qui répondent aux divers besoins des clients. Alors que ce paradigme continue d'imprégner la vie quotidienne, l'optimisation des caractéristiques des fournisseurs d'informatique en nuage est devenue primordiale. L'optimisation combinatoire, qui s'appuie sur les mathématiques discrètes et les techniques informatiques, est un mécanisme essentiel pour relever les défis de la vie réelle dans ce domaine. Cependant, la prolifération de nombreux fournisseurs de services en nuage, associée à une myriade de critères de sélection, a rendu la prise de décision de plus en plus complexe. Avec l'éventail croissant de fournisseurs de services en nuage, il devient de plus en plus important d'optimiser la performance de ces services pour répondre efficacement aux demandes des utilisateurs et des entreprises. Cette thèse présente de nouvelles méthodologies visant à améliorer la rentabilité et la performance dans les environnements d'informatique en nuage, en exploitant des systèmes d'aide à la décision multicritères sophistiqués. En effet, nous introduisons deux contributions distinctes pour relever les défis associés à l'optimisation des performances.

La première contribution de cette thèse répond au besoin pressant de cadres décisionnels robustes dans la sélection de services en nuage. En proposant un modèle de système d'aide à la décision multicritère, nous introduisons une nouvelle approche qui intègre une architecture de réseau neuronal modifiée avec deux couches cachées. Dans ce modèle, la première couche utilise la technique ELECTRE pour évaluer les services en nuage sur la base de critères multiples, tandis que la seconde couche utilise un algorithme de recherche Tabu modifié, en s'appuyant sur une approche de somme pondérée pour une prise de décision efficace. Notamment, l'entrée de notre réseau neuronal modifié est la performance du nuage, ce qui permet aux parties prenantes de naviguer efficacement dans les compromis complexes entre le coût et la performance. Grâce à cette contribution, nous visons à fournir aux décideurs un cadre complet pour optimiser les coûts et les performances de l'informatique en nuage, améliorant ainsi l'efficacité et l'efficience globales des processus de sélection des services en nuage.

Sur cette base, la deuxième contribution de cette thèse introduit une approche pionnière appelée Approche d'Optimisation Multi-Objective pour la Recherche de Services dans le Nuage (MOOA-CSF). En exploitant l'apprentissage supervisé et les techniques de décision multicritères, MOOA-CSF vise à optimiser à la fois le prix et la performance dans les environnements d'informatique en nuage. Grâce à l'utilisation d'un réseau neuronal artificiel (RNA) pour classer les services d'informatique en nuage en fonction de leurs caractéristiques, associée à l'application de la méthode ELECTRE pour ordonner ces services, MOOA-CSF offre un cadre systématique pour identifier les solutions optimales d'informatique en nuage adaptées aux besoins spécifiques des clients et des systèmes. De plus, en incorporant un algorithme génétique modifié pour générer des services hybrides, MOOA-CSF repousse les limites des techniques d'optimisation traditionnelles, offrant des solutions innovantes à des problèmes d'optimisation complexes dans le domaine de l'informatique en nuage. Grâce à une évaluation rigoureuse utilisant divers scénarios, les résultats des simulations démontrent l'efficacité et l'efficience de notre approche, soulignant son potentiel pour améliorer de manière significative l'optimisation des performances de l'informatique en nuage dans le monde réel.

Mots clé : Prise de décision multicritère, cloud computing, optimisation multi-objectif, réseaux neuronaux artificiels, Tabu Search, algorithme génétique, mesures de similarité, apprentissage supervisé, performance.

ملخص

برزت الحوسبة السحابية كقوة بارزة، بتوفيرها للوصول إلى الخدمات مِن أي مكان حسب الطلب، وَذلك تَلبيَّةً لاحتياجات المستخدمين. بينما يستمر هذا النموذج في أخذ دور أكثر أهمية في الحياة اليومية، أصبح تحسين مواصفات مزودي الخدمات ضروريا للغاية. يُعتَبر التحسين الجماعي بالاعتماد على تقنيات الرياضيات وَعلوم الكمبيوتر عاملا حيويًا لِبلوغ تحديات الحياة الواقعية في هذا الميدان. غير أنَّ الانتشار المتواصل لِمزودي الخدمات وَالذي يرتبط بدورهِ بمعايير انتقاءٍ متنوعة، قد جَعل صنع القرار معقدًا أكثر. ردًّا على هذه التَّحديات، تقدم هذه الأطروحة منهجيات جديدة تهدف إلى تعزيز كفاءة التكلفة والأداء في بيئات الحوسبة السحابية، وتسخير أنظمة دعم القرار المتطورة متعددة المعايير. في الواقع، نقدم مساهمتين متميزتين لمعالجة التحديات المرتبطة بتحسين الأداء .

تَهدف المساهمة الأولى مِن هذه الأطروحة إلى الردِّ على التحديات الخاصة بالحاجة المُلِحَّة إلى أطُر عملٍ قويَّةٍ لِصُنعِ القرار في انتقاء الخدمات السحابية. مِن خلال اقتراح نموذج اتخاذ القرارات متعدد المعايير المتقدمة ، نُقَدِّمُ طريقة جديدة مبنية على شبكة عصبية خاصَّة تتكون مِن طبقتين داخليتين. في هذا النموذج، الطبقة الأولى تستخدم تقنية ELECTRE لإثبات صِحَّة الخوارزمية، وَيتم ذلك عبر الاستفادة مِن طريقة المجموع المرجح مِن أجل اتخاذ القرار بكفاءة. مِن الجدير بالذكر، أنَّ مُدخلات هذه الشبكة العصبية هي آداءات الحوسبة السحابية ، وَذلك مِن أجل السماح لِلأطراف المعنية بِأخذِ القرار الصحيح وَالفعَّال بشأن صفقات التكلفة وَالآداء. يهدِفُ هذا النموذج إلى تزويد أصحاب القرار بإطار عَملٍ مُناسب مِن أجل تحسين التكلفة والآداء في سِيَّاقِ الحوسبة السَّحابية، وَالذي يعني بدورهِ تعزيز الكفاءة وَالفعالية في اختيار الخدمة السَّحابية المُناسِبة .

بِنَاءً على هذا الأساس، تُقَدِّمُ المساهمة الثَّانية أسلوبًا فعَّالا تحت اسم .(MOOA-CSF) مِن خلال استخدام التَّعَلُّمِ الخاضِعِ للإشراف وَتقنيات اتخاذ القرارات متعدد المعايير المتقدمة ، يهدِف MOOA-CSF لتحسين التَّكلُفة والآداء في بيئات الحوسبة السَّحابية. مِن خلال استخدام شبكة عصبية اصطناعية (ANN) لِتصنيف الخدمات السَّحابية عن طريق التركيز على مميزاتِها، مَع تطبيق طريقة ELECTRE بِهدَفِ ترتيبِها. تُوَفِّرُ هذه المساهمة إطار عملٍ لتحديد أفضل الخدمات السحابية عبر تحديد متطلبات العملاء وَالنُّظُم. بالإضافة إلى ذلك، وَمِن خلال استخدام خوارزمية جينية معدلة لإنشاء خدمات سحابية هجينة، يتجاوز MOOA-CSF حدود تقنيات التَّحسِين التَّقليدية، مقدمًا حلولًا مبتكرة لِمشاكل التَّحسين المُعقَّدة في الحوسبة السَّحابية. مِن خلال تقييمٍ دقيقٍ بِاستخدام سيناريوهات متنوعة، تُظهِر نتائج المُحاكاة كفاءة وَفعالية هذه الطَّريقة، مِمَّا يُؤَكِّد إمكانية تعزيز آداء الحوسبة السَّحابية بشكل كبير على أرض الواقِع.

الكلمات المفتاحية : أنظمة دعم القرار المتطورة متعددة المعايير ، الحوسبة السحابية، التحسين متعدد الأهداف، الشبكات العصبية الاصطناعية، خوارزمية البحث في التابو، الخوارزمية الجينية، مقاييس التشابه، التعلّم الخاضع للإشراف، الأداء.

Contents

List of Figures **vi**

List of Tables **vii**

List of Algorithms **viii**

List of acronyms **xi**

Introduction **1**

.1 Context 1
.2 Problem Statement 2
.3 Contributions 2
.4 Organization of the thesis 3

I Cloud Computing **7**

I.1 Introduction 7
I.2 Definitions 7
I.3 Essential Characteristics 8
I.4 Deployment Models 8
I.5 Service Models 10
I.6 Cloud Actors 13
I.7 Cloud Data Centers 14
I.8 Virtualization 15
I.9 Cloud Computing Performances 16
I.10 Service Level Agreement 16
I.11 Performance Management in Cloud Computing 16
I.12 Optimization Problems 17
I.13 Conclusion 17

II Multi Objective Optimization **19**

II.1 Introduction 19
II.2 Overview 19
II.3 Local Search 20

II.3.1 Basic Local Search . . . 20
II.3.2 Hill Climbing (HC) . . . 21
II.3.3 Simulated Annealing . . . 21
II.4 Swarm Intelligence Algorithms . . . 22
II.4.1 Particle Swarm Optimization Algorithms . . . 22
II.4.2 Ant Colony Optimization Algorithms . . . 23
II.4.3 Artificial Bee Colony (ABC) . . . 24
II.5 Evolutionary Algorithms . . . 25
II.5.1 Genetic Algorithms . . . 26
II.5.1.1 Selection . . . 27
II.5.1.2 Crossover . . . 27
II.5.1.3 Mutation . . . 28
II.6 Tabu Search . . . 28
II.6.1 Overview . . . 28
II.6.2 Basic Concepts . . . 28
II.6.3 Adaptive Memory . . . 28
II.6.4 Algorithm . . . 28
II.7 Conclusion . . . 29

III Multi-criteria Decision Making Methods: Overview 30
III.1 Introduction . . . 30
III.2 Multi-criteria Decision Making . . . 30
III.3 MCDM Key Concepts . . . 32
III.4 The decision-making process . . . 32
III.5 MCDA Approaches Categories . . . 33
III.5.1 Single synthesizing criteria . . . 33
III.5.1.1 Analytic Hierarchy Process (AHP) . . . 34
III.5.1.2 Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) . . . 35
III.5.2 Outranking approaches . . . 37
III.5.2.1 PROMETHEE method . . . 37
III.5.2.2 ELECTRE . . . 38
III.6 Synthesis of Multi-Criteria Decision-Making Method Families . . . 41
III.7 Conclusion . . . 42

IV Related Work 44
IV.1 Introduction . . . 44
IV.2 Related work classification based on parameters . . . 44
IV.3 Related work classification and analysis based on the nature of the used algorithms . . . 48
IV.3.1 Heuristic Algorithms . . . 48
IV.3.2 Non-heuristic Algorithms . . . 57

IV.4 Discussion and critical analysis 63
IV.5 Conclusion 64

V Proposed Models **65**
V.1 Introduction 65
V.2 Tabu-OptiNimbus 65
V.2.1 Functionality of the Proposed Model 66
V.2.1.1 ELECTRE Tri 67
V.2.1.2 Calculation of the Concordance Matrix 67
V.2.1.3 Calculation of the Discordance Matrix 68
V.2.1.4 Modified Tabu Search 69
V.2.2 Practical Application Analysis 71
V.3 MOOA-CSF: A Multi-objective Optimization Approach for Cloud Services Finding 72
V.3.1 Classification Step 73
V.3.1.1 Activation Function 74
V.3.1.2 ANN training 75
V.3.2 Sorting and Elimination Step 76
V.4 Optimization Step 76
V.4.1 Similarity Step 78
V.4.2 Case study 79
V.5 Conclusion 84

VI Experimental results and evaluation **85**
VI.1 Introduction 85
VI.2 Experimental Environment 86
VI.3 Experimental part of Tabu-OptiNimbus 86
VI.3.1 Experiment 1: Hypervolume Indicator (HV) 86
VI.3.2 Experiment 2: MakeSpan 86
VI.3.3 Experiment 3:Response Time 88
VI.4 Experimental part of MOOA-CSF 88
VI.4.1 Experiment 1 88
VI.4.2 Experiment 2 89
VI.4.3 Experiment 3 89
VI.4.4 Experiment 4 89
VI.4.5 Experiment 5 90
VI.4.6 Experiment 6 90
VI.4.7 Experiment 7 92
VI.5 Comparative Analysis: MOOA-CSF vs. Tabu-OptiNimbus Approaches . . 92
VI.5.1 The comparison between the HV indicators 92
VI.5.2 Comparative Analysis of MakeSpan 93
VI.5.3 Comparative Analysis of Response Time 93

VI.6 Analysis and discussion 95
VI.7 Conclusion 96

Conclusion and Perspectives **97**

List of Figures

I.1 Cloud Computing essential characteristics . . . 9
I.2 Cloud Computing deployment models . . . 9
I.3 Private Cloud . . . 9
I.4 Public cloud . . . 10
I.5 Community Cloud . . . 10
I.6 Cloud Computing service models management . . . 11
I.7 Infrastructure-as-a-Service . . . 12
I.8 Platfrom-as-a-Service . . . 12
I.9 Software-as-a-Service . . . 13
I.10 Three-tier network architecture . . . 14

II.1 Flowchart of EA's steps . . . 26

III.1 The Decision-Making Process . . . 33
III.2 Structure of AHP . . . 35
III.3 TOPSIS process . . . 36
III.4 The steps of PROMETHEE. . . . 38

IV.1 Related work classification based on parameters . . . 45
IV.2 The contextual situation of our research work . . . 64

V.1 An overview of the general architecture . . . 66
V.2 An overview of the proposed general architecture . . . 73
V.3 Sequence Diagram of the general architecture . . . 74
V.4 The layers of the CC-ANN component . . . 74
V.5 Activity diagram of the CC-ANN component . . . 75
V.6 Initial populations of CP, SP and CS classes . . . 77
V.7 Illustration of the crossover operation results . . . 78
V.8 Classification process of the CC-ANN component . . . 80
V.9 The outrank relation between services relating to the illustrative example 83
V.10 The similarity component functioning . . . 83

VI.1 An analysis of the hypervolume metric. 87
VI.2 The outcomes regarding the MakeSpan concerning the quantity of services 87
VI.3 Impact of service quantity on response time. 88
VI.4 average of services in CPC before and after optimization 89
VI.5 Average of services in SPC before and after optimization 89
VI.6 Average of services in CSC before and after optimization 90
VI.7 Impact of the services number on the optimization rate 91
VI.8 The results of the MakeSpan on the number of services 91
VI.9 Effect of the number of service on Response time 92
VI.10 Comparison of the hypervolume indicator 93
VI.11 Comparison of the hypervolume of MOOA-CSF and Tabu-OptiNimbus . . 94
VI.12 Comparative Analysis of MakeSpan 94
VI.13 Comparative Analysis of Response time 95

List of Tables

III.1 Decision Matrix MCDM classification by Guitouni and Martel (1998) . . . 34
III.2 ELECTRE methods and handled criteria type 40
III.3 Synthetic presentation of the advantages and weaknesses of families of multi-criteria decision aid methods 41

IV.1 A comparative summary of some previous studies on performance optimization. 48
IV.2 A comparative summary of previous studies utilizing Heuristic algorithms 54
IV.3 A comparative summary of previous studies utilizing Non-heuristic algorithms 61

V.1 Decision matrix illustration 67
V.2 Performance table 71
V.3 Performances weight table 72
V.4 Initialization of the five criteria weights 80
V.5 Performance matrix of the illustrative example 81
V.6 The concordance matrix relating to the illustrative example 81
V.7 The discordance matrix relating to the illustrative example 82
V.8 The dominant matrix relating to the illustrative example 82

List of Algorithms

1 Pseudo code of Local Search 21
2 Pseudo code of Hill Climbing 21
3 Pseudo code of Simulated Annealing 22
4 Pseudo code of Particle Swarm Optimization 24
5 Pseudo code of Ant Colony Optimization Algorithm 25
6 Pseudo code of Artificial Bee Colony Algorithm 25
7 Pseudo code of Tabu Search 29
8 Basic Tabu Search Algorithm 69
9 Modified Tabu Search Algorithm 70

Acronyms

ABC Artificial Bee Colony

ACO Ant Colony Optimization

AGSA Adaptive Gravity Search Algorithm

AHP Analytic Hierarchy Process

AI Artificial Intelligence

ANN Artificial Neural Network

APIs Application Program Interfaces

AWS Amazon Web Services

BAT bandwidth-aware divisible task

BG background

BPaaS Business Process-as-a-Service

CBA cost-benefit analysis

CC Cloud Computing

CC-ANN classification component using neural network

CH cluster head

CPC client preference class

CSC common service class

CSPs Cloud Service Providers

DAABA Dynamic Artificial Bee Colony Algorithm

DaaS Database-as-a-Service

DC Data Centers

DM Decision Makers

DMOPs Dynamic Multi-objective Optimization Problems

EA Evolutionary Algorithms

EC Edge computing

ELECTRE ÉLimination Et Choix Traduisant la REalité

FACO Fuzzy Ant Colony Optimization

FG foreground

GA Genetic Algorithms

GAIA Geometrical Analysis for Interactive Aid

GOA Grasshopper Optimization Algorithm

HaaS Hardware-as-a-Service

HC Hill Climbing

HV hypervolume

IaaS Infrastructure-as-a-Service

IOPS Inputs/Outputs Per Second

IoT Internet of Things

IPMaaS Identity and Policy Management-as-a-Service

IT Information Technology

KPIs Key Performance Indicators

LoRaWAN long-range wide area networks

MCDA Multi-Criteria Decision Analysis

MCDM Multi-criteria decision making

MFOSF Multifaceted Optimization Scheduling Framework

ML machine learning

MMHT Modified Merkle Hash Tree

MODM multi-objective decision making

MOF Modified Firefly Optimization Algorithm

MOML Minimum Overload and Minimum Lease

MOOA-CSF Multi-Objective Optimization Approach for Cloud Services Finding

MWaaS Middleware-as-a-Service

NaaS Network-as-a-Service

NIST National Institute of Standards and Technology

OC-GA optimization component using genetic algorithm

OS Operating System

PaaS Platform-as-a-Service

PROMETHEE Preference Ranking Organization Method for Enrichment Evaluation

PSO Particle Swarm Optimization

QoS Quality of Service

RIM Reference Ideal Method

S2aaS Sensors-as-a-Service

SA Simulated Annealing

SaaS Software-as-a-Service

SC Storage Component

SC-EL sorting component using ELECTRE

SI Swarm Intelligence

SLA Service Level Agreement

SMC similarity component

SPC system preference class

TOPSIS Technique for Order of Preference by Similarity to Ideal Solution

ToR Top-of-Rack

TS Tabu Search

VMM Virtual Machine Manager

VMs Virtual Machines

VNS variable neighborhood search

WHO Wild Horse Optimization

WOA Whale Optimization Algorithm

WSM weighted sum models

Introduction

.1 Context

Cloud computing has emerged as a significant concept in both the technical and economic domains. It refers to the practice of storing and accessing data and programs over the Internet, rather than on local computer hard drives. This paradigm allows for the utilization of servers, networks, databases, software, and applications online, eliminating the need for organizations to own or manage these resources directly. Instead, they can be rented and accessed as needed. Cloud computing provides on-demand computing services that can be utilized whenever required. A cloud service denotes a service hosted on a remote cloud infrastructure, accessible to users via the Internet. This can be seen through various models like Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS).

Cloud Service Providers (CSPs) offers clients access to cloud computing resources and services. These resources are used flexibly according to the customers' needs based on a specific business model. Services spanning various sectors like business, education, and governance are accessible online to customers through a web browser over the Internet. Meanwhile, data and software programs are stored on cloud servers in data centers. Cloud computing has transformed how businesses of all sizes deliver and use Information Technology (IT) services by offering several advantages like decreased IT costs, on-demand services, access to extensive resources, and the ability to scale services quickly. Numerous cloud service providers operate in the market, including IBM, Oracle, and Google Cloud. While this abundance provides consumers with a wide range of options, improved services, and potential cost savings, it also presents some challenges. The extensive selection can make it notably challenging and complex for consumers to navigate and select the appropriate services to meet their specific requirements.

As the number of cloud providers increases, it's vital to ensure these services perform at their best to meet the needs of both users and businesses. Yet, challenges like varying workloads, resource conflicts, and security risks can hinder efforts to maintain consistent and effective performance. Consequently, cloud service providers and organizations must continuously innovate and adopt strategies to improve their cloud systems' performance, ensuring they remain reliable and responsive and meet technology and user demands.

.2 Problem Statement

Performance and costs are critical points for a data center manager and its infrastructure provider, but these are conflicting objectives. It is therefore necessary to optimize these costs, such as energy consumption, equipment repair and replacement, and system performance, such as load balancing or service performance, such as response time, throughput, cost, etc.
In a dynamic system such as the cloud, designing a resource management model that dynamically adapts the system to instant needs is essential. Such a model should enable resource adjustment on demand, decrease operational costs, and maintain an efficient system.

Our thesis's problem revolves around optimizing cloud service performance to elevate service quality while minimizing operational costs. The study develops a multi-objective optimization approach explicitly tailored for cloud services to achieve this. This approach categorizes cloud services according to client and system preferences, optimizing price and performance. The research endeavors to tackle the challenge of selecting the most suitable cloud services through a comprehensive methodology. By leveraging supervised learning, multi-criteria decision techniques, and modified genetic algorithms, the study aims to provide decision-makers with robust tools to navigate the complexities of cloud service selection, ensuring alignment with diverse organizational requirements and preferences. Through this integrated approach, the research endeavors to advance the field of cloud computing optimization, fostering enhanced service delivery and cost-efficiency in cloud-based.

.3 Contributions

This thesis put forward two significant contributions to advancing knowledge in optimizing the performance and cost of cloud services. The main contributions of this study are outlined below:

The first contribution introduces a novel approach to address the need for robust decision-making in cloud service selection. It proposes a multi-criteria decision support system model, integrating a modified neural network with two hidden layers. The first layer applies the ELECTRE technique to evaluate cloud services based on multiple criteria, while the second layer employs a modified Tabu search algorithm for efficient decision-making. The model's input is cloud performance, enabling stakeholders to balance cost and performance effectively. The goal is to provide decision-makers with a comprehensive framework to optimize costs and performance in cloud computing, enhancing the efficiency of service selection processes.

The second presents a pioneering approach, the Multi-Objective Optimization Approach for Cloud Services Finding (MOOA-CSF), aiming to optimize both price and performance in cloud computing. By leveraging supervised learning and multi-criteria decision techniques, MOOA-CSF classifies cloud services using an Artificial Neural Network (ANN) and orders them using the ELECTRE method. This systematic framework tailors optimal cloud

solutions to client needs. Additionally, by integrating a modified genetic algorithm to generate hybrid cloud services, MOOA-CSF extends traditional optimization techniques, providing innovative solutions to cloud computing challenges. Evaluation through diverse scenarios demonstrates the approach's efficiency and effectiveness, indicating its potential to optimize cloud computing performance in real-world applications.

.4 Organization of the thesis

Our thesis consists of two parts. The first part is a theoretical overview of the environments subject to our study (state of the art), while the second part focuses on the design of our work.

The chapters of this thesis are organized in a well-structured manner. It begins with the first chapter focusing on Cloud Computing and its various paradigms, followed by the second chapter exploring algorithms and optimization methods. The third chapter provides a general overview of MCDM. The fourth chapter consists of a study of related works, while the fifth proposes optimization approaches. The final chapter is the experimental part. So, the content of each chapter is as follows:

- **Chapter 01: Background of Cloud Computing**, we conduct a theoretical study of this field in this chapter. We begin by addressing Cloud Computing, presenting its definition, characteristics, and service and deployment model. We also explore virtualization and optimization issues in the cloud.
- **Chapter 02: Multi-Objective Optimization: Methods and Algorithms**, The chapter on multi-objective optimization delves into the study and application of methods aimed at optimizing multiple conflicting objectives simultaneously. The chapter explores different optimization algorithms and techniques for handling multiple objectives, such as evolutionary algorithms, genetic algorithms, and particle swarm optimization. This chapter is a comprehensive guide to understanding and implementing multi-objective optimization techniques to tackle multifaceted optimization problems across diverse domains.
- **Chapter 03: Multi-criteria Decision-Making Methods: Overview**, this chapter comprehensively examines various techniques and approaches used in multi-criteria decision-making (MCDM). It begins by introducing the concept of MCDM and its significance. The chapter explores a range of MCDM methods, including but not limited to the Analytic Hierarchy Process (AHP), Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS), and Preference Ranking Organization Method for Enrichment Evaluation (PROMETHEE) and ELECTRE (Elimination Et Choix Traduisant la Réalité). Each method is discussed in detail, highlighting its principles. Overall, this chapter serves as a comprehensive guide to understanding the principles and applications of MCDM methods, empowering decision-makers to make informed and effective decisions in complex and uncertain environments.

- **Chapter 04: Related Work**, this chapter aims to provide insights into the state-of-the-art approaches and emerging trends in optimizing the performance of cloud services. We classify the approaches into two classes and make a comparative table for each class. We conclude with a critical analysis where we discuss our position.

- **Chapter 05: Proposed Models**, this chapter introduces two contributions. The first one is Tabu-OptiNimbus which, utilizes a two-layer neural network to optimize decision-making in cloud service provider selection. The first layer integrates the ELECTRE technique for ranking alternatives, while the second employs a modified Tabu search algorithm to navigate complex search spaces. This dual-layer approach enhances decision accuracy and efficiency. The second contribution is MOOA-CSF. It uses supervised learning and multi-criteria decision techniques to optimize price and performance in cloud computing. It employs ANN for classification, ELECTRE for ordering, and a modified genetic algorithm for hybrid cloud service generation, showing potential for enhancing cloud performance optimization.

- **Chapter 06: Experimental results and evaluation**. This chapter outlines the research evaluation process, presents the results, and initiates discussions. We examine the outcomes of the conducted experiment and assess our proposed approach across various scenarios.

- **Conclusion and Perspectives**, in which we summarise the main ideas of our proposals. We highlight the main contributions and address some reflections to identify outstanding questions and future perspectives of this research.

Part One: Technical Background

Cloud computing refers to providing computing services over the Internet, including storage, processing power, and applications. This model eliminates the need for local servers and infrastructure, allowing users to access and utilize computing resources remotely. With its scalability, cost efficiency, and ability to adapt to changing workloads, cloud computing is becoming a critical technology in the digital transformation era. Optimization is key in cloud computing by addressing resource allocation challenges and performance improvement. Optimization aims to streamline operations, minimize costs and maximize efficiency. In cloud computing, optimization techniques are used to optimize resource allocation, efficiently manage workloads, and keep applications and services running smoothly. Whether it's optimizing the allocation of virtual machines, load balancing servers or fine-tuning algorithms, optimization is essential to realize the full potential of cloud infrastructure.

As cloud computing environments become increasingly complex, decision makers are faced with a multitude of criteria and objectives that must be considered simultaneously. Multi-criteria decision making (MCDM) is an area of research that addresses this complexity by providing a framework and methodology for making decisions when multiple and often conflicting criteria must be considered. In the context of cloud computing, MCDM can help select the best cloud service provider, determine the optimal resource mix and align IT strategy with organizational goals.

The integration of cloud computing, optimization and MCDM is a synergistic approach to meet the challenges of the digital age. Using optimization techniques and MCDM methodologies, companies can make informed decisions about resource allocation, cost management and overall system performance in the cloud environment. This integration not only improves the efficiency of cloud operations, but also contributes to strategic decision-making, ensuring that technology is seamlessly aligned with the organization's goals. Understanding the intersection of cloud computing, optimization and multi-criteria decision-making is essential for organizations looking to harness the full potential of their digital resources while expertly navigating the complexity of the modern technology ecosystem.

In the first part of this thesis, we present a background of Cloud computing, Optimization Algorithms and Multi-criteria decision making. Whereas the first chapter covers the

definition, essential characteristics, deployment, and service models of Cloud computing, the second chapter offers an overview of Multi-objective Optimization and the Genetic Algorithm. The third chapter discusses Multi-criteria Decision Making methods.

Chapter I

Cloud Computing

I.1 Introduction

In recent years, cloud computing has surged as a cutting-edge information technology, significantly impacting the entire information sector. This advancement presents numerous advantages for businesses and organizations by transforming IT services into readily accessible commodities. Companies can enhance their operational efficiency and effectiveness by engaging cloud services for software applications, data storage, and processing capabilities. Cloud computing acts as a facilitator, enabling users to deliver dependable, inventive, and prompt services. This chapter serves as a foundational guide to navigating the vast terrain of cloud-based solutions

I.2 Definitions

Cloud computing is increasingly becoming a popular method for providing computing services to users. The availability of cost-effective computers, servers, storage devices, and strong networks has motivated cloud providers to make excess resources accessible to consumers online. Their objective is to optimize service utilization while maintaining rigorous quality standards and guaranteeing customer satisfaction [1].

There is currently no standard definition of cloud computing, but several proposed definitions exist. As stated by the National Institute of Standards and Technology (NIST), Cloud Computing is “a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort” [2]. Different authors have defined cloud computing as a set of shared resources presented as a standardized computing utility running on a server, which caters to multiple groups of users through multitenancy[3].

Additionally, from a deployment perspective, cloud computing is defined as: "parallel and distributed system consisting of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider

and consumers" [4].

I.3 Essential Characteristics

Since systems can be built and deployed using a variety of architectures, technologies, designs, etc., it is essential to identify the characteristics of a system for a comprehensive understanding of the definition. The physical implementation behaviour is very beneficial, regardless of the behaviour. Defined by NIST, there are five essential characteristics of cloud computing as shown in figure (I.1) [5].

On-demand self-service: This implies that a consumer with an immediate requirement can effortlessly access computing resources like CPU, memory, storage, network, and software without manual intervention, ensuring a convenient and self-service experience[6].

Broad network access: This means that Cloud computing services and resources are provided and delivered to the different client platforms (e.g., mobile phones, tablets, laptops, PDAs, and workstations) over the network, specifically the Internet, and through specific mechanisms and protocols [6].

Resource pooling: A cloud provider's computing resources are pooled to provide services to multiple consumers using a multi-tenancy model, "with different physical and virtual resources that are dynamically assigned and allocated to the consumer according to its demand.". The primary motivation of the resource pooling model is to create a location independence mode in which the consumer will not have any control and knowledge of the location of the provided resources. Of course, the consumer will be able to determine the location of resources at a high level (e.g., country, state, or data centre) [6].

Rapid elasticity: This means that the provided resources are flexible and allocated or released according to the consumer's demand. This feature allows consumers to scale up the resources they demand whenever they want and release them once they finish scaling down. Also, from the consumer's perspective, resources can be provided unlimited, in any quantity, and at any time [6].

Measured Service: Multiple consumers can pool and share computing resources. In the meantime, the cloud providers can utilize appreciative mechanisms to calculate, monitor, and report the usage of computing resources for each consumer [6].

I.4 Deployment Models

Cloud Computing encompasses a wide range of cloud infrastructures, each of which can exhibit distinct characteristics. Figure (I.2) shows that these deployment models govern various aspects, including privacy levels, data storage, and access controls. The primary deployment models comprise Private Cloud, Public Cloud, Hybrid Cloud, and Community Cloud [7].

Private Cloud: The infrastructure figure(I.3) is provisioned by a single organization and comprises multiple consumers in the form of, e.g., business units. The infrastructure is

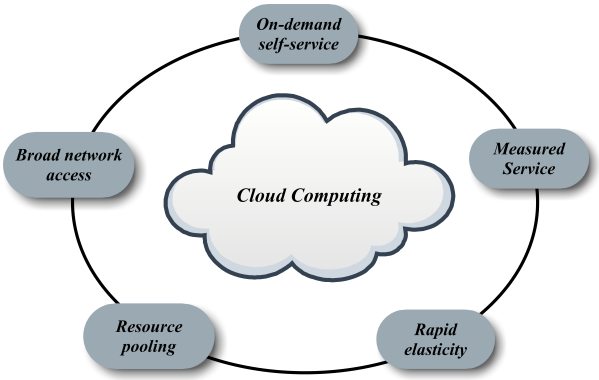


Figure I.1: Cloud Computing essential characteristics

Figure I.2: Cloud Computing deployment models

the collection of hardware and software that enables the benefits of CC. The private Cloud can be owned, managed, and operated by the organization in question, on or off-premise, but also by a professional cloud vendor, which allocates dedicated infrastructure to the organization [8].

Figure I.3: Private Cloud

Public Cloud: The infrastructure is provisioned for extensive use, and compared to the private cloud, it is open use for the general public as shown in figure (I.3). It can be

owned, managed, and operated by a business, academic, or government organization. The infrastructure is on the premises of the cloud provider [8].

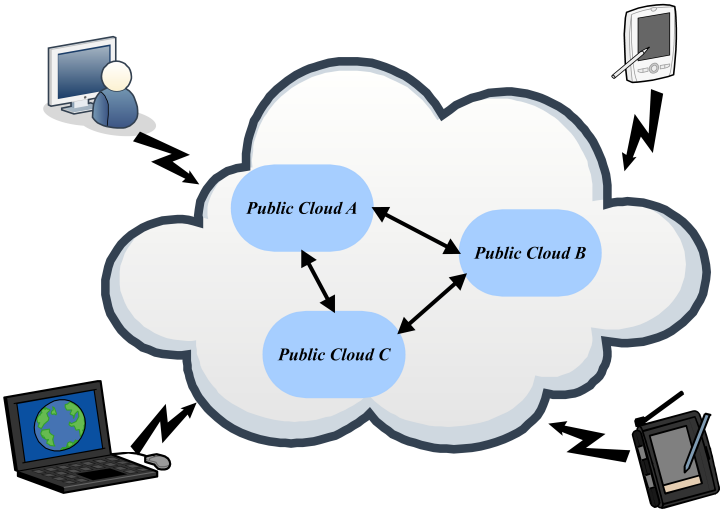


Figure I.4: Public cloud

Hybrid Cloud: The infrastructure is a compromised version of the two deployments above models. They are unique entities but bound together with standardized technology that enables data and application portability. One example is industries heavily regulated due to processing sensitive data, like the health sector. They can utilize a private cloud for sensitive data and a public cloud for the more common one [8].

Community Cloud: The infrastructure is provisioned from a specific community of consumers from different organizations that share the same concerns as shown in figure (I.4). This can be regarding the same security concerns, requirements, policies, compliance considerations, etc. It can be operated by a third party, one of the organizations, or a combination of them, on or off-premise [8].

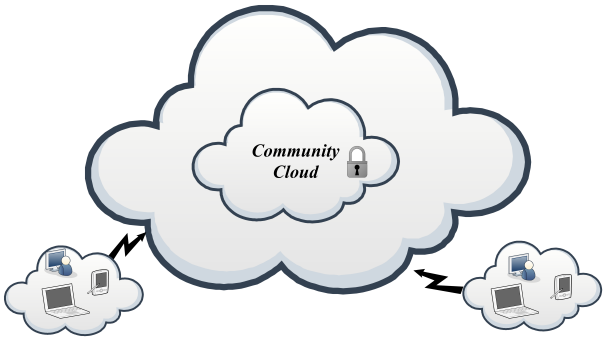


Figure I.5: Community Cloud

I.5 Service Models

Cloud computing mainly includes three types of services, as shown in Figure (I.6): infrastructure-as-a-service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service

(SaaS).

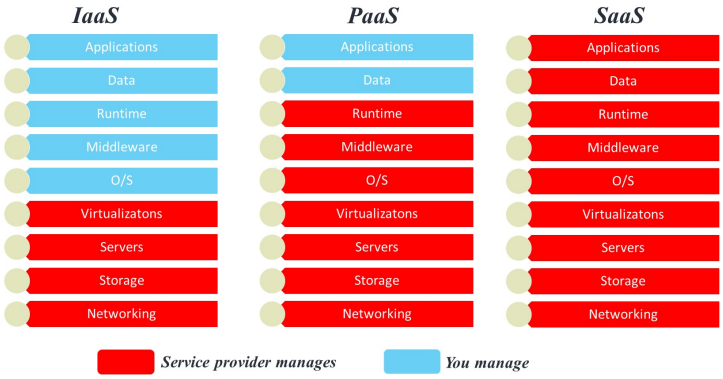


Figure I.6: Cloud Computing service models management

IaaS: In this model, computing or hardware resources such as networks, processing servers, and storage are provided to consumers to deploy and run software such as applications and operating systems [9, 10]. This type of service is similar to servers in the form of virtual machines in the cloud [11, 12]. Consumers cannot control the cloud infrastructure, but they can manage deployed applications and other provisioned resources [2] as in Figure(I.7). The functionality of IaaS is similar to that of a data centre where the provider manages and controls the data centre and the consumer deploys and manages their applications [13]. This capability allows individuals and organizations to rent these resources instead of spending money to purchase and manage the servers that provide the resources [14]. IaaS is often compared to hosting, but the user does not enter into a long-term agreement with the provider and resources are provided on demand [15]. Popular examples of IaaS are Amazon's S3 storage service, Rackspace Cloud server, OpenNebula, Joyent, and Terremark [9] [16, 17].

PaaS: is a model in which Application Program Interfaces (APIs) or development environments are provided in which consumers can build and deploy their applications in the cloud [9, 12, 18]. Consumers can manage their deployed applications and change some storage settings but cannot control the cloud infrastructure [2] as shown in Figure(I.8). Examples of PaaS include Google App Engine, Microsoft Azure Services Platform, Amazon Relational Database Service, Amazon Web Services (AWS), Salesforce's Force.com, Rackspace Cloud Sites, and International Business Machines (IBM) Cloudburst [17, 19]. Vendors may use different programming languages to create environments where consumers can deploy their applications [20]. For example, Google AppEngine used Java and Python, and Windows Azure used .Net [4, 21]. This is probably one of the challenges consumers may encounter when switching from one supplier to another [20, 22].

SaaS: This is the most famous cloud service model that allows consumers to use software applications from providers over the Internet [12, 13]. SaaS applications can be accessed anytime, anywhere using thin client interfaces such as web browsers or programming

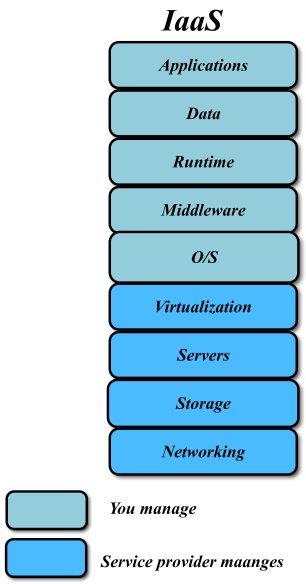


Figure I.7: Infrastructure-as-a-Service

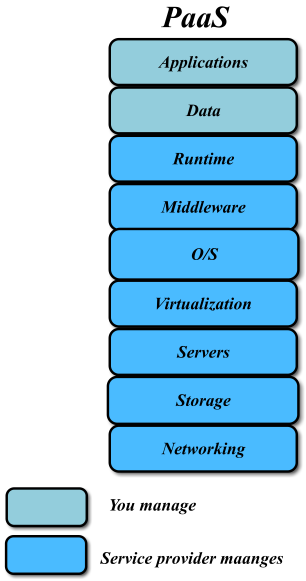


Figure I.8: Platform-as-a-Service

interfaces [23]. SaaS allows consumers to use various software when they need it without having to purchase and maintain that software or purchase and maintain servers [24]. In this case, consumers do not have control over cloud infrastructure and applications, as shown in Figure(I.9). Still, they may be allowed to configure and change basic user-specific settings [2]. SaaS is equivalent to renting software for a limited time instead of buying it, as the software will be available on demand, and consumers will only pay for what they use [25]. Examples of such services are Google Docs, Salesforce CRM, and Trend Micro [26].

In addition, other service models are considered particular types among the three well-known models presented [27]. They are : Database-as-a-Service (DaaS) to provide storage,

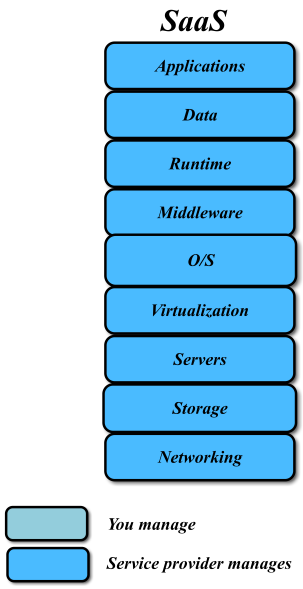


Figure I.9: Software-as-a-Service

Hardware-as-a-Service (HaaS) to provide hardware, Identity and Policy Management-as-a-Service (IPMaaS)) to manage identity management and consumer control policies, Network-as-a-Service (NaaS) to provide virtualized networks, Business Process-as-a-Service (BPaaS) to provide business process outsourcing, Database as a Service (DBaaS) for database outsourcing, Sensors-as-a-Service (S2aaS) for delivering sensor applications, Middleware-as-a-Service (MWaaS) for outsourcing solutions middleware such as application servers, databases, and Messaging. It can be noted that HaaS, DaaS, and NaaS are specific types of IaaS [17],[28–30]. Cloud service models are offered by cloud service providers, which are providers rent cloud services according to customer requirements [31].

I.6 Cloud Actors

Identifying the major players in Cloud Computing regardless of the service target and physical deployment is necessary. According to the NIST Cloud Computing Reference Architecture, there are four key elements in a cloud computing environment [32]:

Cloud Provider: Entities responsible for managing and operating the Cloud's physical infrastructure and the mechanisms that provide consumer services. Vendors are typically responsible for overall cloud management, including physical server maintenance, cooling systems, virtual machine scheduling, and resource management [5].

Cloud Auditor: Vendor-independent entities evaluate cloud services to determine whether they comply with standards such as system operations, performance, and infrastructure security controls. The auditor's goal is to ensure that these controls are implemented correctly and produce the desired results for the system [5].

Cloud Broker: Responsible for integrating Cloud services; Cloud brokers are con-

sidered consumers by cloud providers and providers when interacting with consumers. The three main activities of Cloud brokers operating on Cloud services can be divided into intermediation, aggregation, and arbitrage [5].

Cloud Carrier: Responsible for connecting, managing and transporting Cloud Computing services through networks, telecommunications and other devices between providers and consumers [5].

I.7 Cloud Data Centers

According to [33], Data Centers (DC) are organized in a multi-level network hierarchy. Figure (I.10) presents the general architecture of a DC internal network. The network architecture mainly consists of three layers, each of which plays a specific role in traffic management. Each physical server is connected to one or two access switches at the access level. At the aggregation level, each access switch is connected to aggregation switches. In the Core layer, each aggregate switch is connected to one or more core switches [34].

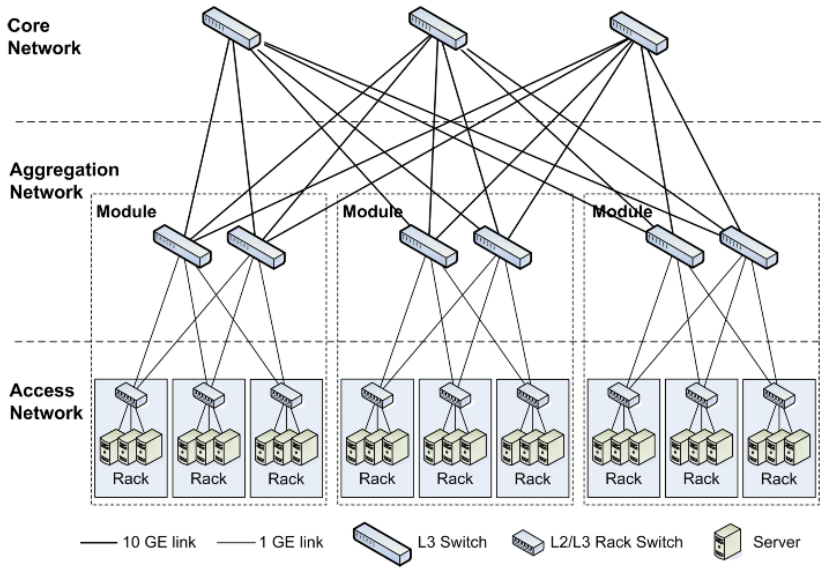


Figure I.10: Three-tier network architecture [35]

Indeed, access switches are responsible for connecting servers to higher levels. Aggregation switches connect access switches. Additionally, they allow traffic localization between servers. Finally, the core switches provide connectivity between the aggregation switches so that there is one connection between each pair of servers. It also includes ports for traffic to allow communication outside the DC [34].

In modern data centres, servers are often organized in racks, each rack having a Top-of-Rack (ToR) switch. The ToRs are connected to the aggregation switches, while the aggregation switches are connected to the upper-level core switches. In traditional DCs, tree network architecture has been widely used due to its simplicity in reducing costs [33].

However, this network architecture may encounter scaling and network congestion issues. To solve these problems, recent studies have proposed new network architectures, e.g., VL2 [36], BCube [37], Portland [38].

I.8 Virtualization

An important technology used in many cloud computing deployments is virtualization. Virtualization is defined as the "separation of a service requested from the physical delivery of that service.". It is specifically defined as "the creation of substitutes for real resources, that is, substitutes that have the same functions and external interfaces, but differ in attributes such as size, performance and cost" [5].
These alternative resources are called resources and can create multiple simulated computing environments defined as Virtual Machines (VMs) on a single physical server called a host. Virtualization consolidates hardware resources such as CPU, memory, disk, and network, helping reduce hardware costs. These virtual machines are controlled through the use of a Virtual Machine Manager (VMM) (also known as a hypervisor), which is responsible for creating, deleting, moving, and monitoring the health of virtual machines [39]. Interaction between the virtual machine and the underlying hardware is accomplished through the use of virtualization software using programs called system calls [5].
There are several widely used techniques for virtualization: Full Virtualization, Para-virtualization and Operating System-level Virtualization.
Full Virtualization provides complete abstraction of the underlying physical hardware resources to encapsulate virtual machines. Specifically, no modifications should be made to the Operating System (OS) and VM without knowing that it is a virtualized environment and shares the same physical infrastructure as other VMs. VMM provides access and resource allocation between virtual machines and physical hardware through classification and paging. Applications can be deployed within each VM's platform, running on VMM [5].
f para-virtualization This concept is similar to full virtualization. Still, the difference is that the operating system is modified to be aware that it is in a virtualized environment and is, therefore, aware of the required resource needs. Source from it. Server, resulting in fewer virtualization costs. However, because virtual machine operating systems require modifications, their compatibility and portability are limited as they require configuration and modifications to operate across different architectures. different server architecture [5].
Operating System-Level Virtualization This causes the host operating system to be virtualized instead of the hardware, creating multiple isolated user-space instances that share an identical operating system. This allows system administrators to allocate resources during VM creation and dynamically modify them at runtime. It is also known as Single Kernel Image or container-based virtualization. This approach provides minimal listening but requires all virtual machines on the host to share a consistent operating system [5].

I.9 Cloud Computing Performances

Cloud computing can reduce infrastructure costs, but when network resources such as data usage and bandwidth are heavily used, it increases data communication costs [17]. IT resource costs will be higher if more resources are used to exchange data between the vendor and the organization. For example, a hybrid cloud customer with organizational data distributed across different clouds will consume more computing resources than a private or public customer. Therefore, the saved infrastructure costs will now be spent on communications and data transfer [40]. Pricing models include pay-as-you-go and subscription pricing. The supplier's costing model and resulting trade-offs must be analyzed before implementation [12]. In this case, benefits must be weighed against costs to maximize benefits over costs by considering the desired level of security [40]. For organizations with limited budgets, a pay-as-you-go plan is recommended, while subscription pricing is best when there are long-term and clearly defined requirements. On the other hand, using a hybrid cloud provides a better return on investment[12, 40].

I.10 Service Level Agreement

Service Level Agreement (SLA) is a contract between a cloud service provider and a customer that states the services the provider will provide [41, 42]. Providers use SLAs to ensure efficient provisioning of cloud resources and services. This involves: Ensuring the quality, availability, reliability and performance of services [12, 17, 40]. SLA is considered one of the important considerations when choosing a suitable deployment model. Customers often negotiate SLAs with suppliers to ensure service delivery functions meet their expectations [12]. Some concerns related to SLA relate to the interpretation of terms, the omission of certain customer requirements, and the criteria for evaluating the agreement terms [40].

I.11 Performance Management in Cloud Computing

Performance management is perhaps one of the most important management tasks in the cloud. Although the term can be used broadly to refer to energy efficiency and Cloud ability concepts, we traditionally refer to application performance in the context of expected QoS [43]. QoS can be defined as SLA that is part of a service provider's obligations to its customers. SLAs include service-level objectives defined using metrics and Key Performance Indicators (KPIs) and specify desired values for these metrics. The selection of KPIs depends on the service under consideration [44]. For example, KPIs for a network service might include round-trip time, packet loss, and network bandwidth, while KPIs for a storage service might include average read/write speed, Inputs/Outputs Per Second (IOPS), and free disk space [43]. Examples of KPIs that apply to different applications include response time (the time it takes for a request to receive a response) and throughput (the amount of work completed per unit of time). KPIs can also refer to

infrastructure metrics such as number of virtual instances used, number of CPUs used, CPU usage, amount of provisioned memory, memory usage, the minimum number of VMs, etc. [43].
Applications in the cloud can be broadly divided into foreground (FG) and background (BG). His BG jobsCloudh, as workflows and batch jobs, are typically of finite duration and have static workload patterns. H. A constant load in which all work arrives at once [43]. QoS requirements are also typically more relaxed. Therefore, resource management for BG applications is less demanding than that of FG applications. For example, an instance associated with a BG application can be temporarily delayed, stopped, terminated, and restarted later. Performance is typically evaluated regarding makespan time, primarily focusing on throughput. In contrast, his FG services, such as search and social networking, are often online, interactive applications that typically run indefinitely. These are usually driven by dynamic workload patterns, with workloads that vary widely over time and depend on the number of users accessing the service. These applications have generally strict QoS requirements, such as very low latency [43].

I.12 Optimization Problems

Optimization is a complex tool capable of helping decision-makers solve complex problems that arise when available resources are limited and under various constraints. Indeed, modelling involves developing a simplified representation to solve a given problem. The optimization model includes determining the objective function, design (or decision) variables, and problem constraints. However, the choice between different representations dramatically influences the efficiency of the resulting solution. Optimization brings several advantages. Indeed, this makes it possible to explore unknown approaches and find the best approaches within some constraints. Additionally, through optimization, decisions will be automated and can be validated by further exploring scenarios and testing new alternatives [34].
Since cloud computing has two main players, there are two aspects of cost optimization: cost optimization performed by the provider and cost optimization performed by the user. Cost optimization performed by cloud providers mainly focuses on minimizing the cost of maintaining a physical data centre [34].

I.13 Conclusion

This chapter introduces a comprehensive overview of cloud computing, illuminating its significance and impact in today's digital era. We have explored the fundamental principles of cloud computing, including its essential characteristics, deployment models, and service models. By understanding these concepts, readers are equipped with the knowledge to navigate the dynamic landscape of cloud technology effectively.

Moreover, we have examined cloud actors, data centres, virtualization, and service level agreements. Thus, we have also discussed the optimization problems in Cloud computing.

With a solid understanding of cloud computing principles and best practices, businesses can harness the cloud's full potential to achieve strategic objectives and drive sustainable growth in the digital age.

Chapter II

Multi Objective Optimization

II.1 Introduction

Multi-objective optimization algorithms are computational techniques designed to address problems with multiple conflicting objectives. In contrast to traditional optimization methods focused on a single goal, these algorithms navigate the complex trade-offs inherent in systems with multiple, often competing, objectives. The primary goal is to identify solutions representing the best compromise among these objectives rather than a singular optimal solution. Employing strategies such as evolutionary algorithms, genetic algorithms, and swarm intelligence, these algorithms explore the solution space efficiently, aiming to converge to a set of solutions known as the Pareto front. The significance of multi-objective optimization algorithms lies in their ability to assist decision-makers in making informed choices by providing diverse trade-off solutions. As technological challenges become more intricate, these algorithms are crucial in addressing complex, real-world problems across various domains.

II.2 Overview

Optimization algorithms are advanced techniques for solving an optimization problem by determining its optimality. Mathematically, an algorithm is a technique used to produce an output for a given set of inputs under specific constraints [45]. Every optimisation method uses an objective function to evaluate the objectives under specific constraints. Optimization algorithms work by defining a search space and trying to maximize or minimize the objective function [46]. Once an optimization problem is formulated, the algorithm searches the space for optimal solutions using appropriate mathematical techniques. A choice can be made depending on the goal: find an optimal but time-consuming solution or a near-optimal solution that is less complicated in terms of time [47].
For distributed systems, many factors are needed to select an appropriate algorithm, and no effective algorithm can be used in all cases. Instead, optimization algorithms have specific goals and objectives. Optimization techniques can be classified into several types

based on different factors. A popular classification divides algorithms into deterministic and stochastic algorithms based on the nature of the algorithm and the method of finding solutions [46]. Deterministic algorithms follow a repeatable path and variables, while stochastic algorithms rely on the randomness of the path and variables. For example, in genetic algorithms, when searching for an optimal solution, the set of solutions will be different each time because it is based on chance. In contrast, there is no large difference in the results of the stochastic algorithms, even when the paths in each population are repeated. Some methods combine deterministic and stochastic algorithms to take advantage of the advantages of each algorithm while overcoming their limitations [47].
Stochastic algorithms themselves can be divided into two categories: heuristics and metaheuristics. Heuristic methods find optimal solutions with low computational cost but do not guarantee the optimal solution [48]. Metaheuristic algorithms often perform better than simple heuristics and use randomization with local search [46]. No specific definition defines the difference between heuristic and meta-heuristic algorithms. However, some researchers describe all stochastic algorithms with stochastic properties and global search as meta-heuristics [49]. Randomization in metaheuristic algorithms provides a method to move from local to global space when applying global optimization [45].
Meta-heuristic algorithms find near-optimal solutions within an acceptable time frame based on one or more objectives [49]. Compared with deterministic algorithms, metaheuristic algorithms give better results in terms of quality and computational time [50]. For these reasons, meta-heuristic algorithms will be used in this study to optimize cost and performance in a cloud computing environment.
Metaheuristic algorithms include several different algorithm groups classified according to algorithm behaviour and strategy. The following sections provide a general overview and description of these algorithms.

II.3 Local Search

The local search method is a classical method used in meta-heuristics; according to Johnson et al [51], it is considered the basic principle of optimization strategies. Local search methods attempt to find a local optimum by starting from an initial solution and gradually improving it at each iteration. A search process characterized by a trajectory in the search space is called a trajectory method [52]. The trajectory's characteristics provide the algorithm's behaviour and effectiveness; Bar-Yam [53] views the search process of the trajectory method as the evolution of a discrete dynamical system in (discrete) time. Their algorithm starts from an initial solution and describes the trajectory in state space. Here, we discuss some heuristic methods based on local search [52].

II.3.1 Basic Local Search

Basic local search is also known as iterative refinement. Small changes are made within the neighbourhood search to obtain a better solution. At each iteration, if the current solution is better than the previous one, the changes are kept; otherwise, the changes are modified to get a better solution. This process continues until no further improvements can be made. Below is a pseudo-code for a basic local search with solution s [52]. N(s) is the

Algorithm 1 Pseudo code of Local Search

```
1: Generate a random initial solution s;
2: repeat
3:     s ← ImproveN(s);
4: until no improvement is possible
```

neighbourhood of the solution. The ImproveN(s) function is an improvement function that obtains a new solution. The function can be the best improvement on a single instance or based on several improved instances until no further improvement is possible. The method searches N(s) neighbourhoods, returns one of the solutions with the better objective function value and then stops at a local optimum. Its performance is, therefore, highly dependent on the definition of N. The performance of iterative refinement procedures for combinatorial problems is often quite inefficient due to time constraints [52].

II.3.2 Hill Climbing (HC)

Hill Climbing (HC) logic is one of the simplest optimization methods for problems with a single objective function. The fitness of randomly selected solution candidates is evaluated, and a new solution candidate is generated using the existing solution. If the new solution candidate has a better fitness, it replaces the current solution candidate, otherwise the current solution candidate is retained. This procedure is looped and repeated until an optimal solution is found. Hill climbing uses the current best solution x^* to generate new solution candidates [54]. One of the main problems with hill-climbing logic is that the algorithm cannot escape from the local optimum, leading to premature convergence. To overcome the problem of premature convergence, it has been proposed to randomly restart the search process several times, also known as stochastic hill climbing [54].

Algorithm 2 Pseudo code of Hill Climbing

```
1: Choose an initial configuration;
2: while stopping condition is not reached do
3:     Generate new configuration, a perturbation of old configuration;
4:     if quality(new configuration) is better than quality(old configuration) then
5:         old configuration ← new configuration;
6:     end if
7: end while
```

II.3.3 Simulated Annealing

Simulated Annealing (SA) is one of the oldest meta-heuristic algorithms, introduced by Kirkpatrick et al. [55] and Cerny [56] as a search algorithm for combinatorial optimization problems. The basic idea is to escape from the local optimum solution and find the global optimum solution by allowing hill-climbing moves that result in a solution worse than the current solution. The pseudo-code of this algorithm is given below [57].

Algorithm 3 Pseudo code of Simulated Annealing

```
1: T ← T_0
2: s ← Starting_state
3: E ← C(s)
4: while not stopping_criteria do
5:     s' ← generate(s) with probability G(s, s')
6:     E' ← C(s')
7:     Δ ← E' − E
8:     if Δ ≤ 0 or random() < e^{−Δ/T} then
9:         s ← s'
10:        E ← E'
11:    end if
12:    T ← reduce_temperature(T)
13: end while
```

Line 1 sets the start temperature to T_0; the loop in lines 4 to 12 generates trial states-s and evaluates the change in cost $\wedge$. The next available state is selected, and the temperature is lowered until the stopping criterion is met. Line 8 shows how Stimulated Annealing accepts trial states [52]. The first term ($\wedge \leq 0$) represents greed. It always accepts low-cost trial states; the second term in line 8 (random() $< e^{-\wedge/T}$) represents the probability of accepting more costly trial states. When the stopping criterion is met, simulated annealing returns the current state s as a result [52]. At high temperatures, the second term in line 8 forces the algorithm to search the entire state space. As the temperature drops, it accepts almost all cost increases and explores large valleys, then smaller sub-valleys to reach the result. This allows it to avoid local minima [52].

II.4 Swarm Intelligence Algorithms

Swarm Intelligence (SI) algorithms describe several related approaches to solving problems that resemble biological swarms and are based on models of social behaviour in insects and animals [58]. The SI algorithm involves multiple swarms sharing information in a search space to find a solution. Specifically, SI is a meta-heuristic developed to solve problems by simulating the behaviour of real flocks or insects [59]. SI algorithms are used to optimize complex problems with distributed structures. It can also be applied to systems with elastic and flexible characteristics without affecting the overall structure. As a result, many SI techniques, such as ACO and PSO, have been applied to cloud computing to improve resource scheduling [60]. Next, we present the details of each of these algorithms.

II.4.1 Particle Swarm Optimization Algorithms

Particle Swarm Optimization (PSO) is a global search algorithm consisting of particles characterized by random velocity and position; each particle in PSO has a velocity that represents its movement within the search space and dynamically adjusts this based on past behaviour. Thus, particles tend to move toward better points in the search space and

explore the solution space by varying their position and velocity [61].
The main design philosophy of the PSO algorithm is closely related to two studies: one is an evolutionary algorithm, and like evolutionary algorithms, PSO uses a swarm mode that simultaneously searches a large region of the solution space for an optimized objective function. The other is artificial life, which studies artificial systems with the characteristics of life; Millonas, in studying the behaviour of social animals using artificial life theory, proposed five basic principles as a way to construct artificial life systems with cooperative behaviour on a computer [62]:

1. **Proximity**: the swarm should be able to carry out simple space and time computations.

2. **Quality**: the swarm should be able to sense the environmental quality change and respond to it.

3. **Diverse response**: the swarm should not limit its way to get the resources in a narrow scope.

4. **Stability**: the swarm should not change its behaviour mode with every environmental change.

5. **Adaptability**: the swarm should change its behaviour mode when this change is worthy.

The main steps of the standard PSO algorithm are summarized in Algorithm 4 [63]. The algorithm begins by initializing the particles with the available solutions. The fitness function for each particle is then computed according to a predefined objective to select the best fitness value (lines 4 and 5). The best value for each particle is then selected and compared to the values of pbest and gbest (lines 8-14). Finally, these steps are repeated until the stop condition is met.

II.4.2 Ant Colony Optimization Algorithms

Ant Colony Optimization (ACO) is a meta-heuristic algorithm that can solve complex optimization problems by simulating the behaviour of ants when searching for food. It uses a mechanism that simulates the behaviour of real ant colonies that cooperate using pheromone pathways. As ants move to find food, they follow a pathway and leave pheromones there. By sensing the pheromone, other ants can follow that path to reach the food source [64]. Most ants choose the shortest path because more pheromone is accumulated along this path [65, 66] .
The ACO algorithm has many advantages: adaptability, robustness, and redundancy. ACO

Algorithm 4 Pseudo code of Particle Swarm Optimization

```
1: INITIALIZE (S, V, P, pbest, gbest)
2: while stop criteria not satisfied do
3:     for each p ∈ S do
4:         f(p) ← ComputeFitness(p)
5:         UpdateVelocity(p)
6:         UpdatePosition(p)
7:         if f(p) < pbest(p) then
8:             pbest(p) ← f(p)
9:         end if
10:    end for
11:    if min_{p∈S} f(p) < gbest then
12:        gbest ← min_{p∈S} f(p)
13:    end if
14: end while
15: return gbest
```

methods are applied to solve discrete optimization problems by finding the shortest path to the goal. They have also been successfully used in other applications such as routing problems in dynamic networks, solving travelling salesman problems, multidimensional knapsack problems, job depot scheduling, and task scheduling in cloud environments [64]. The disadvantages of ACOs are overhead and convergence to locally optimal solutions. However, the main problem with ACOs is the slow convergence, and therefore, they are applied in a narrow spatial domain [67].
The steps of the ACO algorithm are summarized in Algorithm 5. Specifically, in the first step of the solution search, the pheromones and the optimal solution are initialized, and the ants are randomly distributed as shown in lines 1-3. The pheromones of all ants are identified and computed (line 6). The fitness function is then computed to determine the optimal path (line 8). The optimal value is updated if a shorter path is detected (lines 9-11). In lines 12-13, the local and global values are updated. These steps are repeated until the shortest path is selected.

II.4.3 Artificial Bee Colony (ABC)

Artificial Bee Colony (ABC) algorithm is a relatively new method proposed by Karaboga [68]. The foraging behaviour of a swarm of honeybees inspires ABC. In the ABC algorithm, a colony consists of three types of bees: employed bees, onlooker bees and scouts. The colony population is twice the size of the food source. The number of food sources represents the location of possible solutions to the optimization problem. In contrast, the amount of nectar in the food sources represents the quality (fitness) of the corresponding solution [69]. The employed bees are responsible for pioneering food sources and informing the onlookers about the nectar quality of their food sources. The number of onlookers is equal to the number of employed bees. The onlookers decide which food sources to exploit based on the information from the employed bees. Those who use depleted food sources

Algorithm 5 Pseudo code of Ant Colony Optimization Algorithm

```
1: INITIALIZE pheromone
2: Optimal ← null
3: INITIALIZE ants
4: while end condition not met do
5:     for each ant do
6:         for each path do
7:             detect the pheromone on each path
8:         end for
9:         Best ← ComputeFitness(ant)
10:        if Best < Optimal then
11:            Optimal ← Best
12:        end if
13:    end for
14:    Update local pheromone
15:    Update global pheromone
16: end while
17: return Optimal
```

become scouts and continue to search for new food sources randomly. As the amount of nectar increases, the probability of an onlooker bee choosing a particular food source increases [68, 70, 71]. The ABC algorithm is shown below.

Algorithm 6 Pseudo code of Artificial Bee Colony Algorithm

```
1: Initialize the food source positions.
2: Evaluate the food sources.
3: while not termination criteria do
4:     Produce new food sources (solutions) for the employed bees.
5:     Apply greedy selection.
6:     Calculate the fitness and probability values.
7:     Produce new food sources for onlookers.
8:     Apply greedy selection.
9:     Determine the food source to be abandoned and allocate its employed bee as a
       Scout for searching for new food sources.
10:    Memorize the best food source found.
11: end while
12: return the best food source found.
```

II.5 Evolutionary Algorithms

In the 19th century, J. Mendel was the first to clarify the basic lines of inheritance from parent to child. In 1859, C. Darwin published his theory of evolution in his famous book The Origin of Species [72]. In the 1980s, these theories about the creation and evolution of new species inspired computer scientists to design Evolutionary Algorithms (EA). In the last few decades, different schools of evolutionary algorithms have developed independently,

including Genetic Algorithms (GA), evolutionary strategies, evolutionary programming, and especially genetic programming developed by Holland [73].
Evolutionary algorithms (EA) are probabilistic population-based metaheuristics successfully applied to many realistic and complex problems (epistatic, multimodal, multiobjective and highly constrained problems). It is the most studied population-based algorithm. Its success in solving challenging optimization problems in various fields (continuous optimization, combinatorial optimization, system modelling and identification, planning and control, engineering design, data mining and machine learning, artificial life) has stimulated the field known as evolutionary computation [74].
EA is based on the evolution of a population. This population is usually randomly generated. All individuals in the population are encoded with a temporary solution. The objective function associates each individual with a fitness value that indicates their suitability for the problem. At each step, the parent individuals are selected according to a selection paradigm in which individuals with better fitness are selected with higher probability. The selected individuals are then reproduced using variation operators (e.g. crossover, mutation) to produce new offspring. Finally, a replacement scheme is applied to determine which individuals from the offspring and parents will survive. This iteration corresponds to one generation. This process is repeated until the stopping criterion is met see Figure II.1 [75].

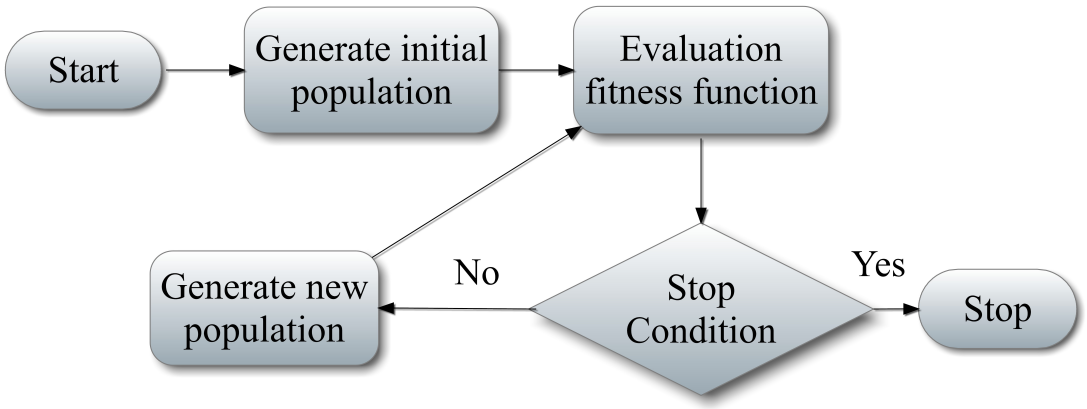


Figure II.1: Flowchart of EA's steps [76]

II.5.1 Genetic Algorithms

Genetic algorithms, one of the most common optimization methods, are stochastic global search methods that mimic the 'theory of natural evolution' proposed by Charles Darwin through three basic processes (selection, recombination and mutation) [77]. In accordance with the evolutionary theory of 'survival of the fittest', populations of candidate solutions are considered simultaneously to produce a better approximate solution in the next generation based on the goodness of fit of the objective function. Each generation creates a new set of individuals based on goodness-of-fit using natural operators such as crossover and mutation. Thus, more favourable traits are retained in individuals, and more competent individuals survive until a satisfactory result is obtained [78]. Genetic algorithms are

based on an initial set of random solutions known as a population. Chromosomes are individuals within a population. The solutions of a population are used to reproduce a new population. This is done with the expectation that the new population produced will be superior to the old population. Appropriate resources are used to reproduce new populations [79]: GA generates a set of populations and applies control operators such as mutation and crossover to find the best among them. At each step, individuals are randomly selected from the existing population [80, 81].

II.5.1.1 Selection

Selection is determining how many times a particular individual will participate in reproduction. It consists of two processes. In the first process, raw fitness values are converted into real-valued expectations of the probability of an individual being selected for reproduction. The second process, called 'sampling', probabilistically selects individuals for reproduction based on their relative fitness relative to others. The performance of the selection algorithm can be determined using three metrics: bias, spread and efficiency. Bias is the absolute value of the difference between the actual and expected probability of an individual being selected. The spread can be defined as the time interval over which an individual is likely to be selected. Efficiency then depends on the overall time complexity of the GA. Therefore, a selection algorithm is suitable if the bias is zero, the spread is minimal, and the contribution to the time complexity of the GA is minimal [78].

II.5.1.2 Crossover

GAs use 'crossover or breeding' operators to produce new offspring from a parental population that contains parts of both parents' chromosomes. Single-point crossover is the most common method in double-stranded chromosomes, where parents exchange parts of their chromosomes after a predetermined point. Other standard crossover methods include multi-point, uniform, mixed, surrogate, and intermediate recombination [78]. In multipoint crossover, several crossover points are randomly selected without overlapping and are arranged in ascending order. The bits between consecutive crossover points are then exchanged between the two parents, resulting in the production of new offspring from the parents, but the bits between the first allele position and the first crossover point are not exchanged [78].
Multipoint crossover is, therefore, suitable for various optimization problems. In uniform crossover, each trajectory can become a crossover point. The crossover mask is randomly generated, and depending on the value of a particular bit in the mask, it is determined which upper bits feed the lower bits at that position. Smooth traversal can be parameterized by applying probabilities to the displacement of bits. By controlling distortion during crossover, the bias against the length of the chromosome representation can be reduced. Another crossover method, shuffle crossover [82], shuffles bits before recombination at a crossover point and does not shuffle bits after recombination. In the reduced surrogate operator, recombination is performed only at points where gene values differ [83]. Intermediate and linear recombination are two other types of intersection operators.

II.5.1.3 Mutation

Mutation is a random process that replaces one gene allele with another to produce a new gene. In GA, mutation is performed with a very low probability. A mutation operator is mainly used for two purposes. One is to ensure that the probability of searching for any particular solution is never zero. At the same time, it may recover the good candidates if they are lost due to selection and crossover operations [78].

II.6 Tabu Search

II.6.1 Overview

Fred Glover first proposed the Tabu Search (TS) in 1986 to overcome the local optima faced by local search algorithms. Although many of the basic key features were proposed by Fred Glover, many were not used in the early days of tabu search research. This section describes the principles of tabu search as defined by Fred Glover [84, 85]. The tabu search algorithm is used to explore new regions of the search space. Tabu search uses 'intelligence' to guide the iterative search in a forward-looking direction. Therefore, the effectiveness of tabu search in problem-solving depends on the use of adaptive memory and responsive search [86].

II.6.2 Basic Concepts

Many researchers have improved the tabu search algorithm over the last decade to obtain a desired solution for a given problem. Despite the improvements, the main concepts of the tabu search algorithm have not changed. The widespread use of tabu search is due to its memory utilization and responsive search. These two features are the most important factors of tabu search in guiding the search process and finding a suitable solution for a given problem [84].

II.6.3 Adaptive Memory

To perform an intelligent search for a given problem, it is first necessary to have data on the movement of past processes. Tabu search, therefore, includes a memory to store the history of past actions performed during the search process. Flexible memory structures are used to store the history. The Tabu search algorithm faces storage space challenges by using memory to store the history [86].

II.6.4 Algorithm

Tabu search is the most widely used meta-heuristic for solving combinatorial optimization problems. It is an effective algorithm for solving difficult problems. It provides the ability to escape local optima while avoiding certain moves to avoid circularity. The pseudo-code of the algorithm is given in algorithm 7 [52].

TS uses short-term memory, called tabu lists, to save recently visited solutions and prohibit movement towards them, thus helping to escape local minima. Taboo lists prevent

Algorithm 7 Pseudo code of Tabu Search

```
1: Generate initial solution s;
2: Initialize Tabu lists (TL_1, TL_2, TL_3, ..., TL_n);
3: s' ← s;
4: repeat
5:     Find the best admissible solution s1;
6:     if f(s1) < f(s') then
7:         s' ← s1;
8:     else
9:         s ← s1;
10:        Update Tabu list TL;
11:    end if
12: until stopping criteria is met
```

circular motion and also prevent motion reversal. Therefore, the neighbourhood of the current solution is restricted to solutions that do not belong to the taboo list. These solutions are classified as admissible sets [52]. Each iteration selects the best solution from the allowed set as the new solution. This solution is added to the tabu list by removing the oldest solution already in the tabu list; the FIFO (first-in-first-out) technique is used to remove the oldest solution from the tabu list. The algorithm terminates when the allowed set is empty, or the termination condition is met [52].

The length of the tabu list, called the tabu tenure, controls the memory of the search process. A large tabu tenure forces the search over a large area. Conversely, a small tabu tenure forces the search in small areas. By changing the tab duration, TS can be made robust [52].

Due to the strong nature of taboos, reasonable solutions that have not been visited can be lost. The aspiration criterion is used to overcome this problem. The basic idea of the aspiration criterion is to allow a move even if it provides a solution with a better objective value than the best available solution [52].

II.7 Conclusion

In conclusion, we can mention that multi-objective optimization methods are indispensable for decision-making for complex problems with multiple conflicting objectives. Their ability to navigate and reveal trade-offs by exploring diverse solution sets, often depicted by the Pareto front, provides valuable insights for decision-makers. As technology advances and the intricacies of real-world challenges continue to grow, the significance of these algorithms remains paramount. Whether applied in engineering, finance, or operations research, the adaptability and efficiency of multi-objective optimization algorithms make them vital for tackling multifaceted issues. As the field evolves, ongoing advancements promise to enhance the capabilities of these algorithms, further solidifying their role in finding effective and balanced solutions to complex problems in various domains.

Chapter III

Multi-criteria Decision Making Methods: Overview

III.1 Introduction

Multi-criteria decision making (MCDM) methods are mathematical frameworks utilized to make decisions in situations where potential options are assessed across numerous competing criteria.
In our daily lives, countless decisions are made based on diverse criteria. To facilitate these decisions, weights are assigned to different criteria, typically derived from input provided by expert groups. Establishing the problem's structure and thoroughly assessing multiple criteria is crucial.

This chapter introduces the fundamental concepts of Multi-Criteria Decision-Making (MCDM). We explore the key concepts in section (III.3), and then the decision-making process in section (III.4). Section (III.5) reviews the categories of MCDA approaches. Section (III.7) concludes the chapter.

III.2 Multi-criteria Decision Making

Since the 1950s [48], Multi-Criteria Decision Making (MCDM) (also called Multi-Criteria Decision Analysis (MCDA)) methods have been an invaluable tool in decision-making processes, especially in complex and uncertain environments. Developed as a branch of operations research [87–89], an MCDA model is a set of available alternatives characterized by the levels of achievement of multiple and often conflicting sets of attributes. It specifies a method for evaluating, prioritizing, and selecting the most suitable alternative [90–92].

MCDM can be divided into two categories: multi-criteria decision Analysis (MCDA) and multi-objective decision making (MODM)[93–95]. MCDA assists decision-makers in assessing and comparing often constrained and conflicting decisions. It is used to help evaluate and compare a finite or discrete number of alternatives [96]. As Examples of MCDA methods, we can cite the weighted sum models (WSM), Analytic Hierarchy Process

(AHP), Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS), Preference Ranking Organization Method for Enrichment Evaluation (PROMETHEE), Exclusions and Transformation between Alternatives (ELECTRE), Reference Ideal Method (RIM) and others. Examples of MODMs include neural networks, goal programming, and genetic algorithms; [96]. As mentioned earlier, MCDAs are primarily associated with "selection processes," while MODMs tend to be more aligned with "design-like" algorithms..

Three problems are distinguished in MCDM: selection, ranking, and sorting problems. The objectives of Decision Makers (DM) are different for each type of problem [97, 98]:

- **Choice problems**: The objective is to assist decision-makers in choosing a subset of the most optimal solutions or alternatives. The final result is a decision or selection process.
- **Ranking problems**: Its purpose is to help decision-makers (DM) categorize the "most attractive" behaviour into equivalent classes. Ranking involves arranging a set of solutions in a specific order. The goal is to assess the quality of all alternatives, typically represented as a ranking from the best to the worst. These alternatives can be ranked fully or partially based on preferences. The outcome is a ranking process.
- **Sorting problems**: The goal is to know which alternatives belong to each class of a predefined set of ordered classes. The decision maker assigns each action to a category. The result is an assignment procedure.

As mentioned above, MCDM methods can be divided into multi-objective decision-making (MODM) and multi-criteria decision analysis (MCDA). The main difference between these two groups of methods is based on the identification of alternatives [99]. MODM has been widely studied in mathematical programming. The theoretical framework is well formulated as the method deals with an optimal design problem that achieves multiple objectives simultaneously [100].
The MCDA method evaluates a set of predetermined alternatives against criteria and selects the alternative with the highest score. The best alternative is usually chosen by comparing the options according to each attribute [101]. Since this study aims to rank cloud computing services, MCDA approaches and methods classified as ranking problems are considered. In the ranking problem, the DM wants to find a ranking structure of the alternatives. This ranking depends on the importance of each criterion and the performance of the other options on a given criterion [102].
Furthermore, the MCDM approach focuses on decision support and interacts with other disciplines, such as intelligent systems dealing with uncertainty. MCDM methods are concerned with the study of decision-making processes to support people. In many cases, such decision-making processes are based on data and information, cannot exclude subjectivity and certainty, and must manage uncertainty. Selecting the most appropriate option from the available (or considered) options can be an MCDM problem in which each option is evaluated against a set of criteria [102].

III.3 MCDM Key Concepts

The MCDM approach used in this thesis has several key concepts. The meaning of certain terms and key concepts can be briefly summarized as follows [102]:

- **Alternatives:**, also called actions, options, strategies, or plans, are possible solutions or potential actions to a decision-making problem. Alternatives are represented as follows: $A = \{a_1, a_2, ..., a_m\}$, A is the finite set of other options, and m is the number of alternatives in A.

- **Criteria:**, attributes or fundamental factors, are measured for each alternative to find a solution. Criteria are tools that describe the goodness and attractiveness of other options. For example, alternatives can be compared regarding their suitability for DM needs. Each criterion corresponds to a perspective that is considered in the decision-making process. Criteria are expressed as follows: $C = \{c_1, c_2, ..., c_n\}$, C is the finite set of criteria, and n is the number of criteria in C. Each $c_j(a)$ represents the performance value of alternative $a \in A$ on criterion $c_j \in C$, in which $j \in \{1, 2, 3, ..., n\}$.

- **Weights:** In MCDM, the interpretation of weights differs somewhat from other fields. For example, while the mathematical representation of the preference structure may look like a weighted sum, the interpretation of weights is more subjective than the "importance" that weights imply. Weights are, in fact, a subjective expression of trade-offs, roughly equivalent to an expression of importance relative to one another. The interpretation of the weights also depends on the methodology of the MCDM. It can be considered close to a trade-off in value measurement-based methods, while superiority ranking methods can be considered a strength of evidence. Weights are represented as $W = \{w_1, w_2, ..., w_n\}$,$W$ is the finite set of weights.

- **Performance Matrix:** The performance matrix M is built for $A \times C$, where $c_j(a)$ is the performance in row a and column j.

$$\mathrm{M}=\begin{pmatrix} & c_1 & c_2 & c_3 & ... & c_n \\ & w_1 & w_2 & w_3 & ... & w_n \\ a_1 & a_{11} & a_{12} & a_{13} & ... & a_{1n} \\ a_2 & a_{21} & a_{22} & a_{23} & ... & a_{2n} \\ ... & ... & ... & ... & ... & ... \\ a_m & a_{m1} & a_{m2} & a_{m3} & ... & a_{mn} \end{pmatrix}$$

III.4 The decision-making process

Figure III.1 illustrates the entire decision-making process, starting from defining the problem as an objective, identifying alternatives and developing criteria, selecting indicators and assigning weights, creating the evaluation matrix, applying suitable methods for alternative evaluation, and finally, choosing alternatives based on the nature of the problem. The selected alternatives are then implemented and evaluated in a specific

application [102].

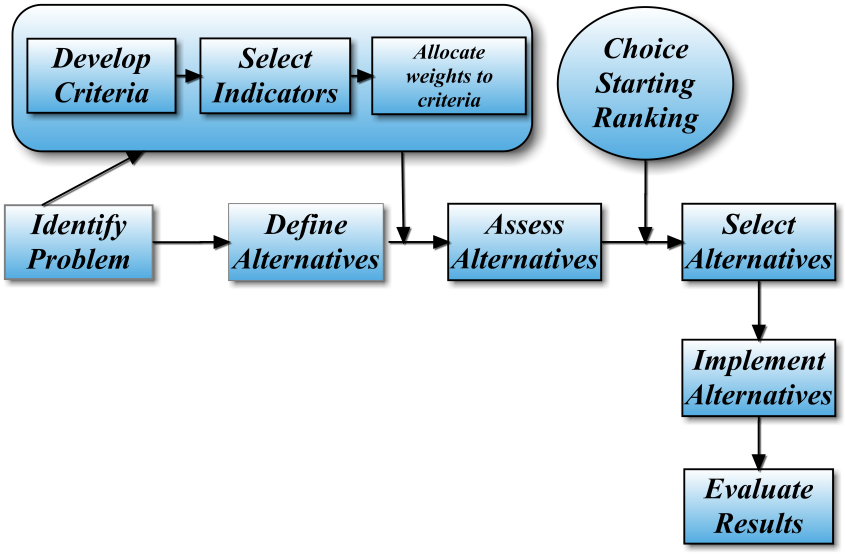


Figure III.1: The Decision-Making Process

III.5 MCDA Approaches Categories

Different combining procedures, each with its own required information and mathematical methods, lead to different multi-criteria decision analysis (MCDA) approaches [102]. There are two primary schools of research from which MCDA was first developed: The American school, known as more descriptive MCDA, such as MAUT, AHP, and TOPSIS, and the French school, known as Criteria Decision Support Methodology, which developed more constructive methods such as ELECTRE and PROMETHEE, commonly known as multi-criteria decision analysis. Different classifications of MCDA methods according to these two schools can be found in the literature [102].
Guitouni and Martel [103] proposed four categories: (i) basic methods, (ii) single synthetic criteria approaches, (iii) superior synthetic approaches, and (iv) mixed methods. Table III.1 enumerates the principal methods associated with each of these categories.

III.5.1 Single synthesizing criteria

MCDA methodologies employing a single-criterion approach aim to consolidate the impacts of various criteria into a singular criterion or attribute. This serves as the foundation for comparing different alternatives. An illustrative instance of a single-criterion approach is the cost-benefit analysis (CBA), where the Net Present Value is utilized as the criterion, necessitating the transformation of impacts into monetary values. However, CBA is not considered an MCDA method because MCDA methods evaluate each effect on its most appropriate scale, which can be either cardinal (monetary and non-monetary) or ordinal (qualitative) [104].

Category	Methods
Elementary methods	Weighted sum, Lexicographic method, Conjuctive methods, Disjunctive methods, Maximine methods.
Signle synthesizing criterion	TOPSIS,MAUT,MAVT,SMART, Utility Theory Additive(UTA),AHPEVAMIX, Fuzzy Weighted sum,Fuzzy maximin.
Outranking methods	ELECTRE,PROMETHEE,MELCHIOR, ORESTE,REGIME.
Mixed methods	QUALIFLEX,Martel & Zaras method, Fuzzy conjuctive/disjunctive method.

Table III.1: Decision Matrix MCDM classification by Guitouni and Martel (1998)

For MCDA methods employing a single synthesizing criterion, the combination of evaluations for each criterion results in a singular index associated with each alternative. This index reflects the alternative's performance across all evaluation criteria simultaneously, where higher values typically signify better ranks. The following sub-sections detail two methods that fall into this category: the Analytic Hierarchy Process (AHP) and TOPSIS. [104].

III.5.1.1 Analytic Hierarchy Process (AHP)

The Analytic Hierarchy Process (AHP) is a structured decision-making method for qualitative and quantitative analysis that decomposes decision-relevant elements into goals, guidelines, programs, and other levels. This method was introduced and subsequently widely applied in various decision-making contexts. Instead of making the right decision, the analytic hierarchy process (figure III.2) tries to find the optimal decision consistent with the decision maker's understanding. To use the analytic hierarchy process, the decision maker must decompose the decision-making problem into a series of independent sub-problems [105]. The decision-making process allows the decision-makers to participate by making their judgments. This means that the subjective judgment of the individual has a significant impact on the decision-making process [105].

The decision-making process of the analytical hierarchy process is as follows: [105]

1. Model the decision problem as a hierarchy. Define decision objectives, alternatives, and criteria;
2. A series of decisions, based on pairwise comparisons of elements, are used to determine priorities between elements of the hierarchy;
3. Synthesize these judgments to yield a set of overall priorities for the hierarchy;
4. Check the consistency of the judgments;
5. Make a conclusive decision based on the outcomes of this process.

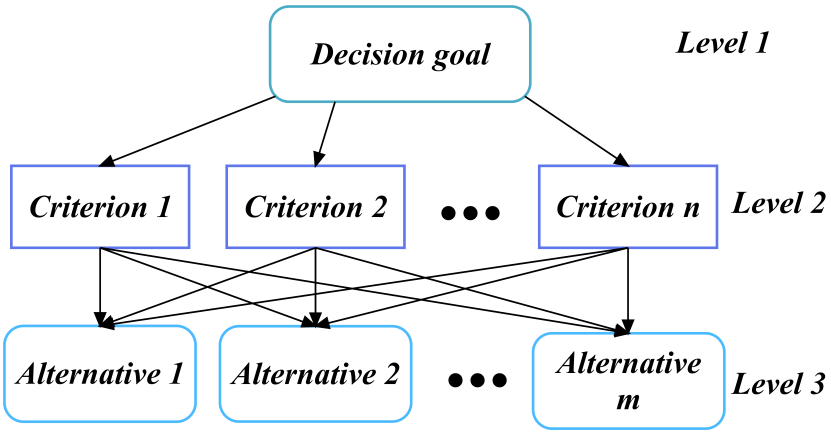


Figure III.2: Structure of AHP

The analytic hierarchy process can be beneficial for groups facing complex problems. They can successfully overcome decision-making problems, even if they overlook essential elements. The analytic hierarchy process is widely used in complex decision situations. It can be applied in the conditions mentioned by Liu [105]. Firstly, in the context of **selection**, it aids in choosing the best policy from a set of candidates. Secondly, it is employed in **ranking**, where all candidates are classified based on specific criteria. Lastly, in the context of **quality control**, the Analytic Hierarchy Process measures different aspects of quality [105].

III.5.1.2 Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS)

Hwang and Yoon developed TOPSIS in 1981 [106]. As mentioned by Zavadskas [107], the TOPSIS is recognized as one of the well-established classical MCDA methods and is considered the second most popular multi-criteria decision-making method. This method provides a cardinal ranking of alternatives and does not require independence of attribute preferences so that attribute information can be fully utilized [106, 108]. Therefore, the optimal alternative is the one with the shortest distance to the positive ideal solution and simultaneously the farthest distance from the negative ideal solution. [109]. Hence, TOPSIS simultaneously considers distances to positive and negative ideal solutions, setting priorities based on their relative proximity and a combination of these two distance measures[110]. The concept of distance measures, fundamental to the TOPSIS method and acknowledged as the simplest technique in multi-criteria decision-making, has positioned the method as a significant branch within the field of decision-making [111]. When dealing with a positive ideal solution, a "virtual alternative" is identified, possessing the highest ratings for all considered criteria. Conversely, in the case of a negative ideal solution, the "virtual alternative" is characterized by the worst reference values [109]. In addition, TOPSIS requires that all attribute values are numerically and monotonically increasing or decreasing and that all attributes have proportional units [112].

The steps of performing the TOPSIS procedure (Figure III.3), based on the methodology of [106], consist of constructing the decision matrix, normalizing the decision matrix, and

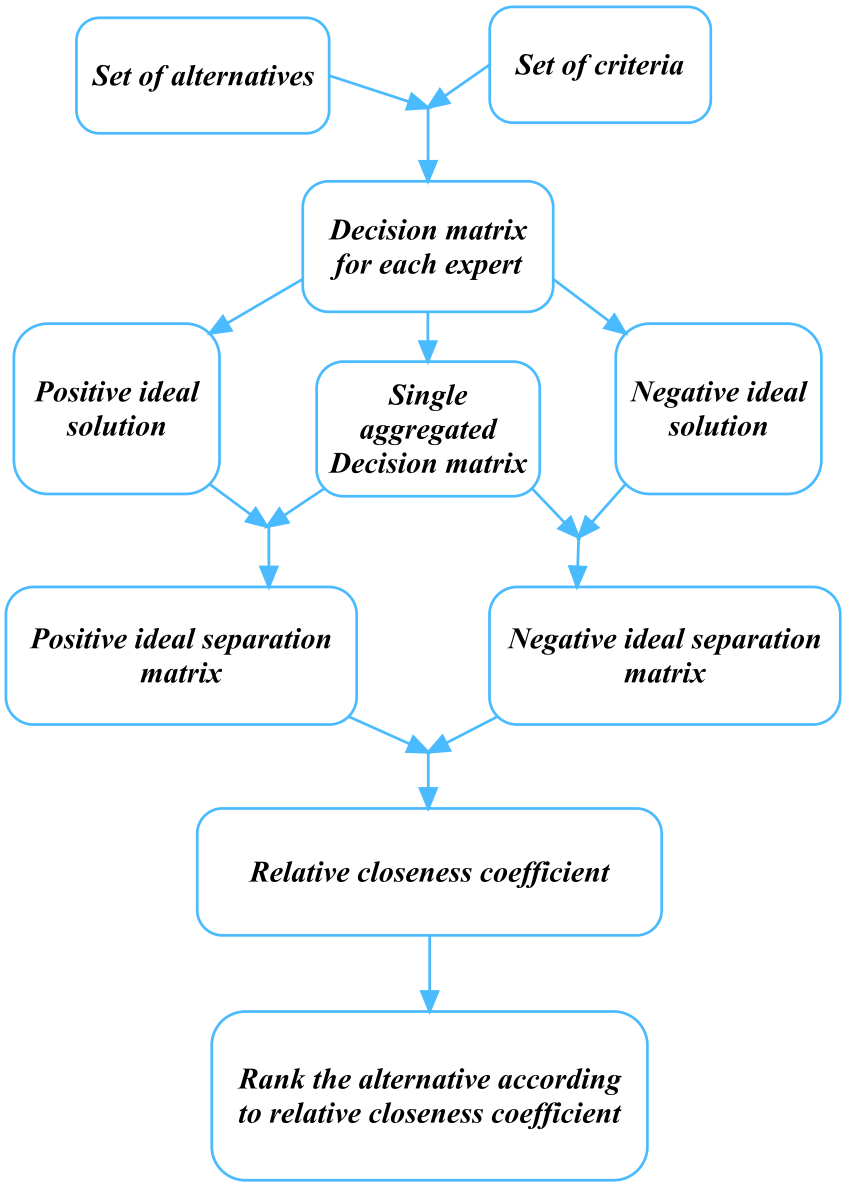


Figure III.3: TOPSIS process

the weighted normalized decision matrix. Then, positive and negative ideal solutions are calculated, and the separation measure for each alternative is determined. The last step involves computing the relative closeness coefficients and arranging the alternatives in descending order based on their corresponding closeness coefficient values. [112]. Several different developments, refinements, and extensions of the traditional TOPSIS method have been developed, and recently, many hybrid methods have been developed that integrate the TOPSIS method with other methods such as AHP, PROMETHEE, ELECTRE, and DEA [107].

On another side, **fuzzy extensions** of the TOPSIS method have been proposed by the researchers to capture and reduce the uncertainty inherent in rating values and human judgments [109]. [109]. The MCDA methods using a fuzzy decision-making approach, including Fuzzy TOPSIS, was proposed by a review on fuzzy multi-criteria decision-making

techniques and applications [89]. We notice that Fuzzy TOPSIS is widely used in research areas related to engineering and computer science [113]. It is also related to group decision-making and multi-objective decision-making. This leads to better choices [114].

The application of this method's basic or enhanced versions has proven successful across various scenarios, consistently yielding satisfactory results [112]. Examples of real-world applications include e-commerce [115], industry [116], health [117], solid waste management [118] and cloud computing [119].

III.5.2 Outranking approaches

The concept of outranking methods was first introduced by Roy [120] in the original ELECTRE method. According to Vincke [97], the fundamental idea behind incorporating outranking methods is to bypass overly stringent mathematical assumptions and the need for intricate information from the decision-maker (DM). In doing so, it is deemed preferable to accept results that may be less comprehensive than what the utility-based method offers [121].
An outranking method comprises two stages: 1) determining the outranking relationship and 2) utilizing the outranking relationship to offer recommendations to the decision-maker (DM) [122]. The next sub-sections present the most popular outranking methods, PROMETHEE and ELECTRE,

III.5.2.1 PROMETHEE method

The Preference Ranking Organization Method for Enrichment Evaluations (PROMETHEE), proposed by Brans [123], aims at identifying the pros and cons of alternatives and obtaining a ranking among them. It is a well-established decision support system that evaluates and selects other options based on several criteria. This simple ranking method is well adapted, both in concept and in practice, to problems where a limited number of alternatives are ranked considering several criteria [102].
PROMETHEE is a superiority ranking method that uses several iterations to obtain a ranking among a finite set of alternatives, as shown in figure (III.4). The ranking process (both in concept and in practice) is straightforward compared to other multi-criteria decision-making methods, which has led to an increasing number of decision-makers using this method to deal with complex problems over the years. Many researchers have also studied the sensitive aspects of this method [124]. The method allows the use of different preference functions according to the criteria characteristics used to eliminate the scaling effects of the requirements. This is the main advantage of the PROMETHEE method: The preference functions and thresholds used in the PROMETHEE process should be chosen according to the decision maker's perspective on each problem. Hence, the choice should be made according to the decision maker's point of view [125].

Within the PROMETHEE family of methods, various techniques exist. Initially, PROMETHEE I and II were created to handle partial and complete ranking. Following that, several additional versions were introduced. Notably, PROMETHEE III is employed

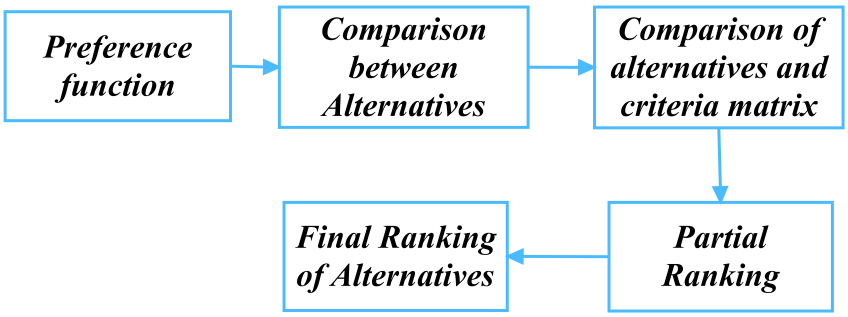


Figure III.4: The steps of PROMETHEE.

for interval-based ranking. At the same time, PROMETHEE IV is applied in scenarios where the decision maker confronts a continuous array of feasible solutions for both partial and full ranking. Furthermore, the PROMETHEE V method addresses problems involving partitioning constraints, and PROMETHEE VI is also available for specific applications. Moreover, GDSS and other variants like Geometrical Analysis for Interactive Aid (GAIA) serve multiple purposes, such as representing the human brain, facilitating group decision-making, and providing graphical representation. Specifically, GAIA functions as a visual interaction module, proving valuable when the decision-making problem is exceptionally complex. The latest PROMETHEE TRI and CLUSTER versions have also been developed for ranking problems and nominal classification. These versions are successful MCDM methods due to their user-friendly formats and mathematical properties [124].

III.5.2.2 ELECTRE

Bernard Roy introduced the ELECTRE (Elimination Et Choix Traduisant la REalité) method in France in the late 1960s. ELECTRE stands out as a widely recognized Multiple Criteria Decision Making (MCDM) method, finding extensive applications in diverse domains such as cloud services [126], decision-making for renewable energy policies [127], investments in small hydroelectric power plants [128], and various real-life scenarios. The main idea of ELECTRE is to utilize the "outrank relationship." It has been developed into ELECTRE IV, ELECTRE IS, ELECTRE TRI, ELECTRE TRI-C, ELECTRE TRI-NC, and many other developed into many other variants [129, 130]. Currently, the most widely used are ELECTRE I, ELECTRE II, and ELECTRE III [130].

The analysts should focus on key aspects of the ELECTRE method when applying it in real-life decision support scenarios. These include the method's handling of the four preference situations, preference modelling via dominance relations, consideration of agreement and disagreement concepts, understanding the method's structure, and being aware of the primary advantages and disadvantages associated with the ELECTRE method [122].

The ELECTRE method realistically represents four basic states of preference: indifference, weak preference, strong preference, and incomparability. Given two alternatives a and b, the four basic states are defined as follows [121]:

1. **Strict preference** $p(aPb)$**:** This corresponds to situations with clear and positive reasons favoring one of the two options.

2. **Weak preference** $p(aQb)$**:** This corresponds to a situation where there are clear and positive reasons to override a strict preference for one of the two options but not enough to infer a strict preference for the other option or indifference between the two options.

3. **Indifference** $p(aIb)$**:** This corresponds to a situation with clear and positive reasons justifying equivalence between two actions.

4. **Incomparability** $p(aRb)$**:** This corresponds to the absence of clear and positive reasons to justify any previous three relations. According to Roy [1991], there are several reasons for incommensurability:

    - Uncertainties, conflicts, and contradictions in the DM's mind.
    - Analysts building the model partly ignore how DMs compare the two options.
    - imprecision, uncertainty, and misspecification of the baseline performance map comparing a and b.

For selecting the most appropriate ELECTRE method, depending on the context of the decision aid, three types of criteria may be considered: true-criteria, quasi-criteria, and pseudo-criteria. We initially present the two discrimination thresholds linked to various criteria, which can be set values or contingent on the performance $g_j(a)$. The uncertainty of the DM preference model can be represented with the following intra-criteria parameters [121]:

- The indifference threshold $q_j[g_j(a)]$ is the threshold at which the DM is indifferent between two options according to the performance of criterion g_j.
- The preference threshold $p_j[g_j(a)]$ is a threshold indicating that the DM prefers, clearly and strictly, one option over the other concerning performance against the criterion g_j.

When comparing two alternatives $a, b \in A$ concerning the criterion $g_j \in G$, it is usually assumed that the threshold is a function of the worst performance of the two alternatives. For the sake of simplicity, the notation $q_j[g_j(a)]$, $p_j[g_j(a)]$ will be simplified to $q_j(a)$, $p_j(a)$ [121] in the rest of this thesis.

- **True-criteria:** This criterion model applies for $q_j(a), p_j(a) = 0$. Thus, indifference arises only when $g_j(a) = g_j(b)$.
- **Quasi-criteria:** This criterion model considers indifference between small differences such as $q_j(a) > 0$ and $q_j(a) = p_j(a)$.

- **Pseudo-criteria:** Modern ELECTRE methods model criteria as pseudo-criteria to handle the imprecision and uncertainty inherent in complex human evaluation processes [131]. As a result, the outranking relationship can be interpreted as a fuzzy relationship such that $q_j(a), p_j(a) > 0$ and $q_j(a) < p_j(a)$.

Table III.2 shows the criteria used in each EELECTRE method and specifies the corresponding decision problems they address. [121].

Criteria type	Choice	Ranking	Sorting
True-criteria	I	II	-
Quasi/Pseudo-criteria	IS	III,IV	TRI-B,TRI-C,TRI-NC

Table III.2: ELECTRE methods and handled criteria type

For all binary relations $(a, b) \in A \times A$ on the j_{th} criterion, three relations may be established: preference, weak preference, and indifference (i.e., P_j, Q_j, I_j). These can be grouped into partial outranking relations aS_jb, such that $aP_jb \vee aQ_jb \vee aI_jb = aS_jb$. can be grouped into partial outranking relations aS_jb. For instance, aS_jb means that a is at least as good as b in criterion j [121].
Given the possible binary relationships of $(a, b) \in A \times A$, four basic preference situations can occur:

1. aSb and $\neg(aSb)$: a is weakly or strictly preferred to b.
2. bSa and $\neg(aSb)$: b is weakly or strictly preferred to a.
3. aSb and bSa :a is indifferent to b.
4. $\neg(aSb)$ and $\neg(aSb)$: a is incomparable to b.

In the ELECTRE method, the concepts of 'majority opinion' and 'veto' define the agreement and disagreement indices. Two criteria parameters are needed to establish the superiority relationship: the relative importance of each criterion w_j and the veto threshold $v_j[g_j(a)]$.
For the simplest ELECTRE methods (i.e., ELECTRE-I and ELECTRE-II), the concordance and discordance index calculation considers actual criteria (with rejection thresholds), not pseudo-criteria. ELECTRE-I was created for choice questions, and ELECTRE-II is the result of the first modification to the original ELECTRE to rank alternatives [121]. The ELECTRE II method requires the definition of three concordance parameters (c-, c+), two discordance parameters (d-, d+), and a criterion weight vector (wj) considering j criteria. A reliability index is created from the obtained concordance and discordance matrices. From the reliability matrix, create ascending and descending distillations, each distillation producing a rank for an action. Thus, each action is classified into two different ranks. The unique step of the ELECTRE II method is to remove cycles between actions

to find a valid solution at this stage. A combination algorithm merges the ascending and descending ranks, thus obtaining a partial or complete pre-order. However, a complete pre-order can always be achieved if the midpoint of both ranks is used instead of the original combination algorithm [132].

A. The Concepts of Concordance and Discordance Outranking-based methods, in essence, rely on the fundamental concepts of fit and mismatch, forming the core principles underlying outranking.[122].

A1. Concordance Index Definition: To validate an outranking (aSa'), a sufficient majority of criteria in favour of this assertion must occur.
$C(a \text{ Rel } a')$ is the coalition of criteria in which relations Rel hold for a, a' [122]. The concordance index is used to model the concept of concordance. It permits to build an m $\times$ m concordance matrix C composed of elements $c(a, a')$, for all a, $a' \in A$ [122].

A2. Discordance Index Definition: The assertion aSa' cannot be validated if a minority of criteria is firmly against this assertion [122].

III.6 Synthesis of Multi-Criteria Decision-Making Method Families

Table (III.3) summarizes the various advantages and scopes of use of the families of multi-criteria decision-making methods presented earlier. This synthesis work can assist a researcher in choosing a type of method based on the characteristics of their problem. Through this brief overview of multi-criteria decision-making methods, we have seen that the choice of a method depends on the type of decision problem desired. The main obstacles to using all the methods mentioned are the need to systematically evaluate all alternatives to be compared with the same set of decision criteria and carefully interpret and formalize the decision maker's preferences. This preparatory work is often long and tedious.

Table III.3: Synthetic presentation of the advantages and weaknesses of families of multi-criteria decision aid methods

	Outranking	Synthesis criterion
Size of action sets to compare	Small to moderate	Small to very large
Conflict detection	Yes	No
Complexity of calculations	Moderate to significant	Low
Trade-off between criteria	Partial	Total
Automation of the process	Yes	Yes

III.7 Conclusion

This chapter has explored various methods and approaches in Multi-Criteria Decision Making (MCDM), providing insights into key concepts, categories, and methodologies employed in decision analysis. Through our examination of the Analytic Hierarchy Process (AHP), Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS), PROMETHEE method, and ELECTRE, we have shed light on the diverse range of tools available to decision-makers for addressing complex decision problems involving multiple criteria.
Through our discussion of these methodologies, we have underscored the importance of selecting appropriate MCDM techniques based on the specific characteristics of the decision problem and decision-maker preferences. Moreover, we have highlighted the need for rigorous analysis and sensitivity testing to ensure the reliability and validity of decision outcomes.

In conclusion, the diverse MCDM methodologies discussed in this chapter provide decision-makers valuable tools for addressing complex problems and navigating the trade-offs inherent in multi-criteria decision environments. By leveraging these methodologies effectively, decision-makers can enhance the transparency, efficiency, and effectiveness of decision-making processes across various domains and contexts. As the field of MCDM continues to evolve, ongoing research and innovation in methodology development promise to further advance our ability to tackle complex decision challenges and drive positive outcomes.

Part Two: Contributions and Experimental

Chapter IV

Related Work

IV.1 Introduction

With the rapid evolution of the digital world, optimizing the performance of cloud computing has emerged as a critical necessity. On the other hand, with the widespread adoption of cloud technologies across all sectors, the demand for efficient resource utilization and seamless service delivery has never been higher. Businesses rely on cloud computing to support their operations, from data storage to application deployment, making performance optimization a key priority. This entails leveraging various features of cloud platforms to ensure scalability, reliability, and cost-effectiveness. Optimizing cloud performance cannot be overstated as organizations strive to meet the dynamic demands of modern applications and services.

This chapter presents a review and analysis of related work, which includes academic research work, and commercial tools. This review positions our work within a broader context to show how it fits into current trends and recent developments in our domain of interest. Then, we describe the evaluation criteria adopted in different approaches in relation to our research context. We focus on two major aspects: a parameters-based classification and analysis and another based on the nature of the methods and algorithms used. Finally, we present a critical analysis that allows us to discuss our position according to other work.

IV.2 Related work classification based on parameters

The performance optimization of cloud computing, based on different features, has become increasingly important in today's world. For this reason, many studies have been developed in the literature. In this section, we will focus on some of them and suggest a comparative table (see Table 1) to introduce an analysis of these studies based on different parameters (features).

In [133], Guo et al. proposed a queuing model and developed a synthetic optimization method to optimize the performance of services. They analyzed and conducted the equation of each parameter of the services in the data centre. Then, by analyzing the queuing

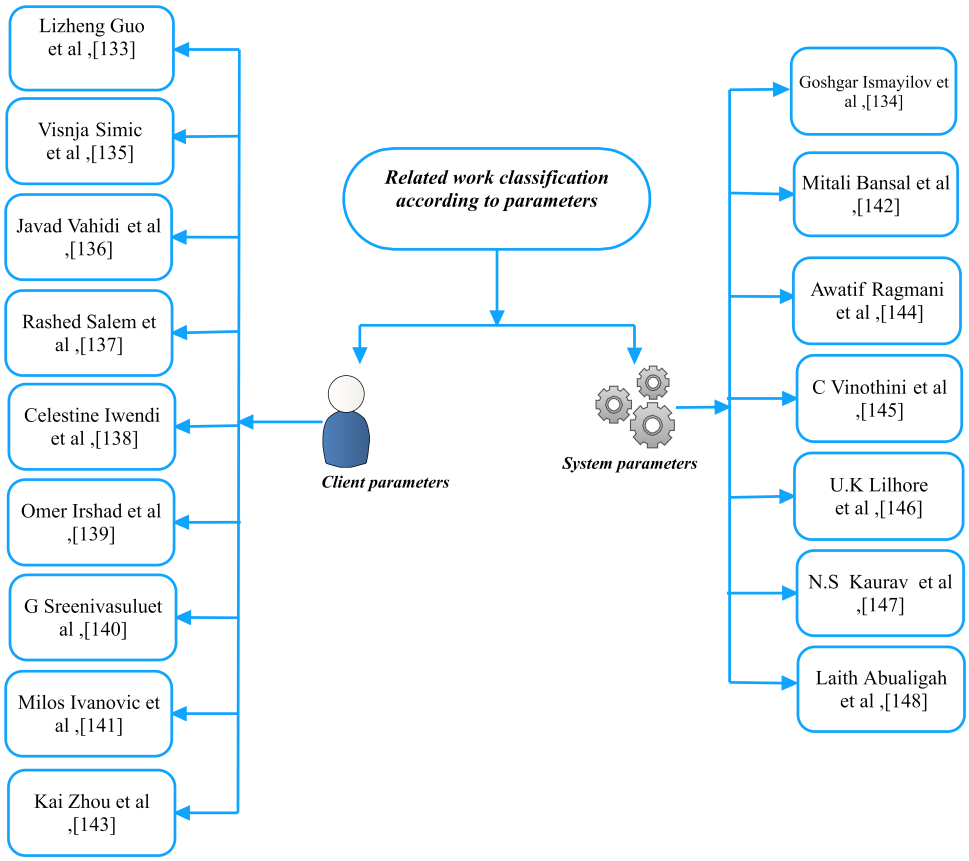


Figure IV.1: Related work classification based on parameters

system's performance parameters, they proposed the synthetic optimization mode, function, and strategy. Finally, they set up the simulation based on the synthetic optimization mode. By comparing and analyzing the simulation results to classical optimization methods, the authors showed that the proposed model can optimize the average wait time, queue length, and number of clients.

The authors in [134] proposed a prediction-based dynamic multi-objective evolutionary algorithm called the NN-DNSGA-II. They incorporated an ANN with the NSGA-II. The optimization goals were the minimization of makespan, cost, energy, and imbalance and maximizing reliability and utilization. The authors revealed that the NN-DNSGA-II significantly outperforms the other alternatives in most cases concerning metrics used for Dynamic Multi-objective Optimization Problems (DMOPs) with unknown actual Pareto-optimal fronts, including the number of non-dominated solutions, Schott's spacing, and the Hypervolume indicator.

The authors of [135] extended an existing parallel software framework, WoBinGO, for GA-based optimization to be used in a Cloud environment. They also proposed an intelligent decision support engine based on ANN and meta-heuristics, which allows the user to assess the framework behaviour on the underlying infrastructure regarding optimization duration and the cost of resource consumption. This assessment lets users decide whether they want faster-delivered results or lower infrastructure costs.

Authors of [136] recognized the role of the innovative Grasshopper Optimization

Algorithm (GOA). They have strongly highlighted the significance of such an algorithm for optimizing resource allocation in a Cloud computing environment. The proposed algorithm was simulated with MATLAB using eight datasets. Moreover, the authors compared GOA with GA and SEIRA algorithms to evaluate its performance precisely. Results revealed the efficiency of the proposed grasshopper algorithm in solving the resource allocation problem in the Cloud.

In [137], Salem et al. created a new algorithm (MOABC) derived from a combination of ABC and Multi-Objective Optimization. Therefore, access and optimized replica placement were presented at the appropriate best place for the minimum distance and least-cost path, for direct bees, and for shortest routes in the distance and lower cost. The proposed algorithm gave fast access to data and selected the best replica placement nearest to users. Additionally, the placement optimization provided more least-cost paths, better response times, and replication costs within the budget.

Authors of [138] used a hybrid metaheuristic algorithm, namely, the Whale Optimization Algorithm (WOA) with Simulated Annealing (SA), to optimize the energy consumption of sensors in IoT-based WSNs. To simulate the IoT network, the authors used the Xively IoT platform. Several performance metrics, such as load, residual energy, number of alive nodes, cost function, and temperature, were used to choose the optimal CHs in the IoT network. The proposed work in [138] was compared with several state-of-the-art approaches and yielded good results.

In [139], the authors presented a novel intelligent multi-agent-based performance optimization approach based on the Internet of Things (IoT) and deep learning paradigm. They took advantage of state-of-the-art probabilistic, recurrent neural networks and extended short-term memory models to intelligently predict the system's upcoming behaviour and optimization needs. They deployed the proposed performance optimization approach and showed significant performance gain compared to existing approaches.

In [140], a hybrid optimization model has been developed, allowing for an efficient task allocation to the virtual machines (VMs) in cloud computing. The task priorities are managed by using the hierarchy process. The authors have used BAT (Bandwidth-aware divisible task) and BAR models to consider the task properties and VM characteristics for task scheduling. They also used the Minimum Overload and Minimum Lease (MOML) preemption policy, successfully employed to reduce the VM load on the VMs. The performance of the proposed model was then compared with existing algorithms such as BAT and ACO (Ant Colony Optimization) algorithms. Consequently, the authors have proven that their model is efficient regarding resource, bandwidth, and memory utilization.

In [141], the authors proposed a decision support engine that recommends optimal framework parameters to achieve minimal total execution time and total cumulative uptime for a specified optimization problem. The engine solved a bi-criteria optimization problem and used surrogate models of the IaaS behaviour under various large-scale optimization loads as a fitness evaluator in MOGA (Multi-Objective Genetic Algorithms). According to the authors, the obtained results were promising, especially in the case of computationally heavy fitness evaluation functions.

In [142], the authors proposed an approach that integrates scheduling and resource cost chronology models, called the Multifaceted Optimization Scheduling Framework (MFOSF). According to the authors, this approach can manifest the relationship between the user's budget and the producer's cost when planning.

In [143], Zhou et al. proposed a cloud service optimization method based on an artificial ant colony algorithm and bee colony algorithm (DAABA). For the optimization model, the authors added the dynamic coefficient strategy and the reliability feedback update strategy to the mathematical model to strengthen the applicability of the farming season. Furthermore, the optimal fusion evaluation strategy was used to save optimization time by reducing useless iterations. In contrast, the iterative adjustment threshold strategy was adopted to improve the accuracy of cloud service findings by increasing the size of the bee colony.

Ragmani et al. [144] proposed a hybrid Fuzzy Ant Colony Optimization (FACO) algorithm for VM scheduling to guarantee high efficiency in a cloud environment. The proposed fuzzy module evaluates historical information to calculate the pheromone value and select a suitable server while keeping an optimal computing time [144]. Their study provides one of the first investigations into how to choose the optimal parameters of ant colony optimization algorithms using the Taguchi experimental design.

In [145], the authors presented a hybrid meta-heuristic algorithm based on the Firefly Optimization Algorithm and GA to optimize task scheduling for multiple tasks in the cloud computing platform. The developed system provides a distribution by reallocating the loads to the related VMs, taking into account the objective function of the VM. The system also increases resource utilization and communication costs during task scheduling. It efficiently decreases the processing time of the process compared to different techniques such as GA, Firefly Algorithm, and Modified Firefly Optimization Algorithm.

In [146], a hybrid load balancing model based on optimising a modified particle swarm was proposed. It includes improved metaheuristic firefly algorithms that boost cloud performance. The proposed approach has been focused on predictive workload allocation, which often primarily includes resource scalability, as well as a load balancing model allowing implementation of workflows and maximizing the use of uniformly load-distributed VMs [146].

The authors of [147] proposed an approach to improve the capability of data centres. It inefficiently allocates requests among VMs by using their current status in cloud computing with a GA [147]. The authors claim that their algorithm uses a modified GA-based approach, in which the best VM is selected by analyzing candidates with more fitness than others. The proposed approach significantly reduced the response time of servers and provided an effective load balancing among VMs [147].

In [148], an efficient optimization method for task scheduling was presented. It is based on a hybrid Multi-Verse Optimizer with a GA (MVO-GA). MVO-GA was proposed to enhance the performance of task transfer via the cloud network based on the cloud resources' workload. The proposed method works on multiple properties of cloud resources, namely speed, capacity, task size, number of tasks, VMs, and throughput. The proposed

Table IV.1: A comparative summary of some previous studies on performance optimization.

Research Paper	Environment	Used Parametres					Client/System Side	
		R	Av	C	Th	Rt	Client	System
[133]	Cloud Computing	-	-	-	-	+	+	-
[134]	Cloud Computing	+	-	-	+	+	-	+
[135]	Cloud Computing	-	-	-	-	-	+	-
[136]	Cloud Computing	-	-	-	+	-	+	-
[137]	Cloud Computing	-	-	+	+	-	+	-
[138]	Iot Network	-	-	-	+	-	+	-
[139]	Iot	-	-	-	-	-	+	-
[140]	Cloud Computing	-	-	-	-	+	+	-
[141]	Cloud Computing	-	-	-	-	+	+	-
[142]	Cloud Computing	-	-	-	+	-	-	+
[143]	Cloud Computing	+	-	-	-	-	+	-
[144]	Cloud Computing	-	-	-	-	-	-	+
[145]	Cloud Computing	-	-	-	-	-	-	+
[146]	Cloud Computing	-	-	-	-	-	-	+
[147]	Cloud Computing	-	-	-	-	+	-	+
[148]	Cloud Computing	-	-	-	-	-	-	+

method successfully optimized the task scheduling of many tasks [148]. Also, it optimized the transfer time of large cloud tasks, reflecting its effectiveness.

After analyzing the articles cited in Table (IV.1), we can classify them into two main classes as shown in figure IV.1 . The first class optimizes cloud performance on the client's side ([133], [135],[136],[137], [138], [139]). The second class contains articles that optimize cloud performance on the system side ([134], [142], [144], [145], [146], [147], [148]).
R, Av, C, Th, and Rt are Reliability, Availability, Cost, Throughput, and Response Time.

IV.3 Related work classification and analysis based on the nature of the used algorithms

As mentioned above, we present in this section an analysis by classifying the related work based on the nature of the used algorithms. We have chosen two main classes of algorithms: "Heuristic" and "non-heuristic".

IV.3.1 Heuristic Algorithms

Performance optimization using metaheuristics has become a prominent research area. Metaheuristic algorithms are characterized by their ability to explore the solution space and avoid local optimization efficiently. They have proven to be highly effective in tackling complex optimization challenges. In this section, we will focus on research using metaheuristics algorithms and suggest a comparative table (see Table IV.2) to introduce

an analysis of these studies.

In [134], authors proposed a prediction-based dynamic multi-objective evolutionary algorithm (NN-DNSGA-II) for dynamic workflow scheduling in cloud computing, incorporating artificial neural networks into the NSGA-II algorithm to improve the scheduling solution. The proposed algorithm is based on an accurate Pareto algorithm. The algorithm considers six objectives: minimization of "makespan, cost, energy, and degree of imbalance" and maximization of "reliability and utilization". The algorithm significantly outperforms other dynamic algorithms that are not based on prediction in the metrics used for emotional multi-objective optimization problems (DMOPs) where the actual Pareto optimal front is unknown (e.g., number of non-dominated solutions, shot spacing, hypervolume metrics). Based on a real-world application of the Pegasus workflow management system, this study examined the applicability of the NN-DNSGA-II algorithm to the dynamic workflow scheduling problem. The work also adapted five main non-prediction-based dynamic multi-objective algorithms for the same problem, providing alternative solutions for dynamic workflow scheduling in cloud computing. Performance evaluations based on different Pegasus workflows demonstrated that the NN-DNSGA-II algorithm outperforms the non-prediction-based algorithms regarding the number of non-dominated solutions, shot spacing, and hypervolume metrics.

The paper [135] presents a software framework for large-scale optimization using parallel Genetic Algorithms (GA) over cloud resources. The framework includes auto-scaling in the application layer to ensure user quality of service (QoS) in cost and performance. The software framework considered factors such as cost and performance to recommend optimal VM instance parameters for the best efficiency. The decision support engine in the framework optimized the cost/performance ratio, considering the user's specific requirements. The framework also utilized a Machine Learning (ML) surrogate model to predict the delivery time and execution cost, which helps make informed decisions about VM instance parameters. The recommendation of optimal VM instance parameters was based on the analysis of user QoS, price, performance, and the predictions made by the ML surrogate model.

The paper [136] highlights the importance of the Grasshopper Optimization Algorithm (GOA) for optimizing resource allocation in cloud computing environments. The proposed GOA algorithm is simulated in MATLAB and compared with GA and SEIRA algorithms. The results strongly confirm the applicability of GOA and its high ability to solve the resource allocation problem in cloud computing. The functions designed as the basic operators of GOA are found to explore the problem space properly, optimize the found answers and derive acceptable answers according to the allocation problem. This paper recommends further investigation and comparison of GOA and other optimization algorithms to provide practical solutions for resource allocation in cloud computing. Grasshopper Algorithm GOA is important in solving the optimization problem by providing features such as a gradient-free mechanism and avoiding localized over-optimization. GOA exhaustively explores the solution space and optimizes and finds the solutions found. Innovative algorithms like GOA can successfully optimize resource allocation in cloud computing, providing sufficient resources to users and maximizing resource utilization for

cloud providers.

The paper [137] proposes an artificial bee colony (ABC) algorithm combined with multi-objective optimization (MOO) for data replication optimization in cloud environments. The algorithm aims to achieve the highest efficiency and lowest cost in the system and provide low-cost and fast access to data replication. The MOABC algorithm provides access to replicas via the closest and lowest-cost route and optimizes the selection and placement of replicas. The algorithm outperforms other algorithms such as EFC, DCR2S, ACO, Genetic algorithm, and GASA in terms of effectiveness and efficiency.
The proposed algorithm is simulated using CloudSim, and its efficiency is demonstrated. In addition, the authors mentioned that they are considering validating their system in the Cloud and applying hybrid AI techniques to ensure its availability and improve data replication access and cost. According to the authors, the MOABC algorithm provides lower latency, optimal data replication placement, and improved data availability. It also provides less costly paths, better response times, and replication costs within budget.

The paper [138] addresses the energy optimization problem in IoT networks. This is crucial to increase the robustness and extend the network's lifetime. The proposed hybrid meta-heuristic algorithm, Whale Optimization Algorithm Simulated Annealing, is used to optimize the energy consumption of sensors in IoT networks. The approach considers various performance indicators, such as the number of surviving nodes, load, temperature, residual energy, and cost function, to select the optimal cluster head (CH) in the IoT network. The results show the superiority of the proposed hybrid approach over existing optimization algorithms such as the Artificial Bee Colony (ABC) algorithm, Genetic Algorithm (GA), Adaptive Gravity Search Algorithm (AGSA), and Whale Optimization Algorithm (WOA).
The proposed approach can be extended to consider additional performance metrics such as latency, node density, and link lifetime to optimize energy usage. The paper suggests directions for future research, including scalability testing in real-time applications and addressing security issues using existing solutions.

The WoBinGO framework in [141] provides a flexible approach to utilizing computing resources according to user requirements, avoiding unnecessary reservation of resources and reducing job waiting times. It hides the complexity of the underlying computing infrastructure from the user and reduces the burden of resource acquisition and dealing with different middleware. The framework has been evaluated through theoretical analysis and empirical studies, which show significant speedup in large-scale optimization problems and computationally intensive evaluations.
The framework is suitable for solving real-world applications such as calibrating leakage models for Visegrad power plants. Future improvements to the framework aim to leverage its inherent flexibility to manage cloud instances independently based on system load.

In [142], A multi-faceted optimized scheduling framework (MFOSF) based on particle swarm optimization (PSO) algorithm was proposed. It has effectively maximized the performance and cost of cloud computing. MFOSF combines a scheduling model and a resource cost timeline model to address the challenges of resource demand and multi-job

execution in cloud computing. The MFOSF-PSO method outperforms other models in cost, time interval, deadline, and resource utilization, showing a potential increase of 57.4% in the best-case scenario. The resource cost model of the framework reflects the relationship between task and resource costs and allows for a detailed definition of task requirements. The improved PBPSO algorithm helps prevent PSO from falling into a local optimum solution and improves the quality of the optimization results. The optimization approach proposed in this paper focuses on the task scheduler, which coordinates with the task manager and the global resource manager to allocate tasks to resources according to their requirements.

The proposed cloud service optimization method in [143], based on the Dynamic Artificial Bee Colony Algorithm (DAABA), was designed to be applied to agricultural machinery production in China, characterized by smaller and more decentralized facilities. The optimization model of the method includes a dynamic coefficient strategy and a reliability feedback update strategy, which is suitable for the peak farming season. Due to the optimal fusion evaluation strategy, the DAABA algorithm reduces unnecessary iterations and saves optimization time. It also uses an iteration adjustment threshold strategy to increase the size of the bee colony, thus improving the accuracy of the crowd service. It was verified that DAABA's performance is better than that of other algorithms and suitable for solving large-scale problems in the cloud service optimization of agricultural machinery.

The paper [144] proposed a FACO algorithm for load-balanced virtual machine scheduling in cloud computing architectures that is suitable for handling large and distributed networks, as verified by experimental results. By applying fuzzy logic to calculating pheromone probabilities, the computation time can be minimized, and the optimal parameters of the ACO algorithm can be selected using Taguchi's experimental design. The FACO algorithm improved load balancing in cloud architectures, and depending on the scenario, response times can be reduced up to 82%, processing time up to 90%, and total cost up to 9%. The proposed algorithm enabled the optimisation of resource utilization and prediction of the optimal configuration of the FACO algorithm. The FACO algorithm can be applied to define an efficient cloud architecture adapted to high-performance applications.

In[145], a new system was proposed. It improves resource utilization and communication costs during task scheduling on cloud computing platforms. This leads to increased efficiency and reduced processing time compared to other techniques such as GA, FA, and MFOA. The proposed cloud computing load balancing solution based on the Genetic Algorithm (GA) provided an optimized and high-quality solution to the resource allocation and load balancing problem.
The performance evaluation of the proposed method compared to the traditional genetic algorithm, firefly method, and modified firefly method shows improvements in communication cost, execution time, and resource utilization depending on the number of tasks. A hybrid meta-heuristic combining Modified Firefly Optimization Algorithm (MOF) and genetic algorithm techniques provides a timely and efficient optimization approach for resource allocation in cloud environments. The CloudSim simulation and the deployment solution tested on the Platform demonstrated the behaviour and performance results of the

hybrid meta-heuristic algorithm. They provided insight into its applicability in practice.

The paper [146] introduced a hybrid load balancing model based on modified particle swarm optimization and an improved metaheuristic firefly algorithm that can improve cloud performance and resource utilization. The model provided predictive workload allocation and resource scalability, enabling workflow execution and maximum utilization of evenly load-balanced virtual machines (VMs). According to experimental data, the proposed hybrid load-balancing approach outperforms existing approaches in terms of time span, resource utilization, degree of imbalance, and task migration. The model can be further evaluated for real-time loads and shows its potential for practical applications in dynamic cloud environments.

A load-balancing approach based on a modified genetic algorithm was proposed in [147]. It can significantly reduce server response times and provide efficient load balancing among virtual machines (VMs) in a cloud computing environment. The approach increased the capacity of data centres by efficiently allocating requests among VMs based on the current state of VMs. The evaluation results demonstrated the effectiveness of the proposed approach in reducing response times and achieving load balancing in a simulated environment. Since load balancing is an essential aspect of cloud-based systems, this paper focused on the design and implementation of load balancers suitable for cloud computing environments.
The practical implications of this paper included improving the performance of cloud-hosted services, enhancing resource optimization, and addressing technical issues such as load management and data security in cloud computing.

In [149], the authors introduced a hybrid load balancing technique that provided globally optimal fast convergence in load balancing between virtual machines (VMs) in cloud computing. This technique ensures that servers remain operational and responsive to user requests. The proposed GWO-PSO algorithm combines the advantages of fast convergence and global optimization to improve system efficiency and resource allocation. It reduces the overall response time by 12% compared to other algorithms. The algorithm has been implemented and tested using MATLAB software, showing good global optimization and fast convergence. The proposed algorithm can be used to optimize imaginative city concepts by incorporating cognitive Internet of Things (IoT) technologies.
The study highlights the potential of machine learning and deep learning in developing load-balancing mechanisms. While machine learning algorithms can effectively improve accuracy, the proposed technique focuses on achieving fast convergence and low response times. So, combining the proposed technique with machine learning provides accurate results with guaranteed success.

The authors of [148] proposed MVO-GA method for task scheduling in cloud computing that helps to reduce scheduling costs and improve the availability and accessibility of services in cloud environments. By optimizing the task scheduling process, the proposed method improved the performance of task transfer in cloud networks, resulting in more efficient data transfer and timely delivery of important tasks.
Integrating MVO and GA algorithms in the proposed method enabled better distribution

of routing tasks and workload balancing among the available cloud machines, resulting in reduced routing time. Simulation results show that the MVO-GA algorithm in the proposed method is better than using MVO and GA separately and is effective in optimizing the transfer time of many tasks in the Cloud. The proposed method's effectiveness in optimizing throughput in a real cloud environment can be further investigated by applying it to a larger cloud data transfer dataset.

The paper [150] proposed a method to balance multiple Quality of Service (QoS) factors and satisfy service composition constraints in a multi-cloud environment using Firefly Optimization Algorithm (FOA) and fuzzy logic. The proposed method aimed to improve the Quality of Service (QoS) in cloud computing by combining different types of services with the same functionality but different non-functionalities. Experimental results showed that the proposed method outperforms traditional methods in terms of response time, availability, and energy consumption. The proposed method applies to real-world scenarios where cloud service providers must combine atomic services from multiple clouds into a single service to meet complex customer requirements. Therefore, the method can help to dynamically provision virtualized scalable resources as a service over the internet and thus improve the overall performance and efficiency of the cloud computing system.

Table IV.2: A comparative summary of previous studies utilizing Heuristic algorithms

Paper	Algorithm	Targeted Application	Advantages	Limitations
[134]	NN-DNSGA-II	• Workflows in cloud computing for large-scale compute and data-intensive applications. • Periodical workflows in different domains such as physics, business, and meteorology.	• NN-DNSGA-II algorithm performs well regarding the number of non-dominated solutions. • Performance evaluation validates the applicability of NN-DNSGA-II algorithm.	• The NN-DNSGA-II algorithm cannot be further improved and optimized for dynamic workflow scheduling problems.
[135]	Parallel Genetic Algorithms (GAs)	• Real-world applications with computationally expensive evaluations. • Optimization methods exposed as an Internet service	• ML surrogate model for predicting delivery time and execution cost. • Recommends optimal VM instance's parameters for best efficiency.	• WoBinGo does not optimize cost and performance even more effectively.
[136]	Grasshopper Optimization Algorithm (GOA)	•Resource allocation in Cloud computing systems.	• Grasshopper Optimization Algorithm (GOA) is effective for resource allocation in Cloud computing. • GOA outperforms GA and SEIRA algorithms in terms of performance.	N/A
[137]	Artificial Bee Colony (ABC)	•optimizing data replication in cloud environments	•Multi-Objective Optimization (MOO) with ABC achieves highest efficiency and lowest costs. • MOABC algorithm finds optimal solution for replication placement in Cloud.	N/A

[138]	Whale Optimization Algorithm (WOA) with Simulated Annealing (SA)	• Energy Efficiency in IoT	• Minimizes energy consumption of sensors in IoT network •Increases network lifetime	•The proposed approach may face challenges when tested for scalability in real-time applications with many sensors, potentially hindering its ability to handle the increasing data volume effectively and processing demands inherent in such environments.
[141]	NSGA-II	• Solving optimization problems in Grid Computing	•Adaptive allocation of jobs with limited lifetime prevents endless waiting in batching queues. • Suitable for solving optimization problems with computationally expensive evaluations.	• The paper does not take into consideration the independently managing cloud instances based on the system load
[142]	•Enhanced PBPSO •FCFS Scheduling •Original PSO	•The paper proposed a multi-faceted optimization scheduling framework for cloud computing.	• MFOSF optimizes performance and cost in cloud computing. •MFOSF-PSO method is more effective, increasing performance by 57.4%.	N/A
[143]	Dynamic artificial ant-bee colony algorithm (DAABA).	•Cloud service optimization in agricultural equipment manufacturing	•Potential of parallelism, leading to strong robustness and high efficiency. •Strong global searching ability in the early stage	•Low accuracy in the early stage of the search algorithm • Slow convergent speed in the later stage of the search algorithm.

[144]	Hybrid fuzzy ant colony optimization algorithm (FACO)	• The targeted applications are high-performance cloud computing and virtual machine scheduling.	•Instant migration of virtual machines •Elasticity in adjusting storage capacity and computing resources to user requirements	•The paper does not evaluate the FACO algorithm within a real and multi-cloud computing architecture.
[145]	MOFA	•Optimization Approach in Cloud Environment	•Proposed method requires less time compared to GA, FA, and MFOA. • The Proposed method requires a lower cost of communications than GA, FA, and MFOA.	• The paper does not consider task scheduling in cloud computing platforms for multiple tasks, using hybrid meta-heuristic methodology.
[146]	MPSO	• The research focuses on cloud computing applications.	• Experimental data shows excellent performance compared to existing load balancing approaches. •Proposed approach offers predictive workload allocation and resource scalability.	The model does not evaluate for real-time loading in a real-time environment.
[147]	Modified genetic algorithm	• Load balancing in cloud computing environments.	•Proposed approach allocates efficient distribution of loads among virtual machines. • Experiment on Cloud Sim shows the promising response time for load balancing.	•The model does not consider user preferences to improve the load balancing efficiency .

[148]	MVO-GA	•Task scheduling in Cloud computing.	•Reduces scheduling costs in cloud computing. *bullet* Optimizes task scheduling for a large number of tasks.	• MVO-GA does not applied on a larger dataset of cloud data transfer.
[149]	Combined approach of GWO-PSO	•Load balancing in cloud computing	•The proposed technique reduces overall response time by 12% compared to other algorithms. •The GWO-PSO algorithm improves PSO to 97.253% in terms of convergence.	•The quality parameters are not evaluated.
[150]	Firefly Optimization Algorithm (FOA)	• Cloud services composition	•Lower energy consumption compared to CRIO-GWO method *bullet* outperforms previous methods in terms of response time and availability.	N/A

IV.3.2 Non-heuristic Algorithms

Non-heuristic algorithms are problem-solving procedures that do not employ heuristic techniques. They rely on exact, systematic, or exhaustive methods to solve problems. Several works have used this algorithm to deal with the optimization issue. The authors of [133] proposed a dynamic performance optimization model that can optimize the average waiting time, queue length, and number of customers for cloud computing with a queuing system. This optimization method can reduce the waiting time, shorten the queue length, and serve more customers, thus improving the quality of service (QoS) in cloud computing. The proposed model can be used to answer essential questions about cloud computing performance, such as determining the level of QoS that can be achieved for a given service and customer arrival rate, the number of required servers to meet specific QoS requirements, and the number of customers that can be served. The paper also highlights the societal implications of this research, which aims to reduce operational costs and carbon emissions by optimizing performance and energy consumption in cloud computing.

The paper [140] introduced a hybrid optimization model for task scheduling and virtual

machine allocation in cloud computing to address the problem of assigning appropriate virtual machines to tasks in the cloud environment. This hybrid model modifies the bandwidth-aware divisible task (BAT) scheduling model by using a hierarchical process to prioritize tasks before sending them to the scheduler and adding a bar system model to improve optimization. Minimum overload and minimum lease policies are used to reduce virtual machine overload in the data centre.
The proposed hybrid model's performance is evaluated using different parameters, and simulation results show its efficiency. The hybrid model is compared with existing algorithms such as BAT and ACO and has proved efficient regarding resource, bandwidth, and memory utilization. The proposed model can be used in real-time workflows to validate its efficiency further.

The paper [139] proposed an intelligent multi-agent-based performance optimization approach for distributed biological systems that addresses performance issues at the cache, persistence, and computation levels. The approach uses state-of-the-art probabilistic models from a deep learning paradigm to predict system behaviour and optimization needs and achieves significant performance improvements compared to existing approaches.
This approach intelligently persists and moves biological data objects between distributed system nodes, improving system responsiveness and user experience. The research work of [139] highlights the importance of performance optimization in providing an interactive and responsive experience to end system users, especially in the context of exponentially growing big biological data.
The distinctive features of the approach, such as optimization based on context and functional history data, autonomous and loopback mechanisms for continuous optimization, and IoT-based device communication, make it suitable for efficient storage and workflow-based biological systems. It reduces the need for object migration and provides a percentage performance improvement while ignoring application overhead and complexity. The work also presents a conceptual simulator and algorithm for object migration and provides a practical tool to verify and validate the proposed solution.

In [151], authors integrated machine learning (ML) into the data warehouse and increased efficiency by overcoming high maintenance costs and failure rates. ML algorithms optimize query performance and increase processing speed and overall system effectiveness. ML provides a personalized optimization approach to queries, tailoring responses to individual user needs and speech patterns. ML analyzes user requirements and behaviours and is very easy to use, allowing query optimization procedures to be refined and automated.

The paper [152] proposed a new approach, HunterPlus, which extends the existing GGCN scheduler and develops a new CNN scheduling model for task scheduling in cloud fog computing environments. Experimental results reveal that the CNN scheduler outperforms the GGCN-based model by at least 17% HunterPlus is the best scheduler among those considered in all evaluation metrics. Its consistent and uniform task allocation and migration behaviour make it stable. Adding a bidirectional layer to the GGCN scheduler tends to improve performance. Using CNN models in HunterPlus is effective but does not consider temporal information in the data. Future work could combine RNNs and CNN models to utilize temporal information to improve performance further. In addition

to energy consumption optimization, this paper also emphasizes the importance of job completion rate as a measure of scheduler reliability.

In [153], the authors proved that Edge computing (EC) is a promising computing model that can address the limitations of traditional cloud computing in handling the vast amount of data generated by the Internet of Things (IoT) and meet practical needs. Combining Artificial Intelligence (AI) with EC can enhance and optimize EC performance. This is because traditional non-AI methods are limited in their ability to cope with EC's complex and dynamic environment. EC can provide faster response times and more stable network conditions for the practical application of AI. Researchers should focus on providing higher computational overhead, privacy, and security performance in ECs to enable broader use of AI. The design and adoption of model-free methods and lightweight AI models are essential research directions to achieve efficient strategies and improve the efficiency of AI algorithms with limited computing and energy resources in ECs.

The paper [154] proposed an IWHOLF-TSC technique that improves task scheduling efficiency in cloud computing and addresses issues such as long scheduling times, high-cost consumption, and high virtual machine load.
The IWHOLF-TSC technique improves performance compared to other meta-heuristic algorithms by combining the Wild Horse Optimization (WHO) algorithm with the LF (Levy Flight Theory) algorithm. It is a powerful tool for load balancing and better resource optimization, including energy efficiency, leading to improved system performance and cost-effectiveness in cloud computing. According to the authors, the IWHOLF-TSC technique can be validated and evaluated using various metrics, thus demonstrating its effectiveness in different situations. In addition, it can be used for task scheduling in cloud computing and applied to resource allocation and clustering processes in future IoT-enabled cloud environments.

The resource optimization method proposed in [155] can be applied to long-range wide area networks (LoRaWAN) to reduce implementation and maintenance costs. Using variable neighborhood search (VNS) and the least-cost spanning tree algorithm, parameters such as propagation factor, bandwidth, and transmission power are optimized to minimize the total energy per useful bit and full data collection time. The results show that in networks with a small number of associated devices, it is preferable to use a small value for the spreading factor to reduce the energy consumption per helpful bit. On the other hand, in LoRaWANs with extensive coverage, a significant value of the spreading factor is preferred.
The optimization also considers device scaling over time and ensures equal allocation of time slots per propagation factor to avoid discrepancies in data collection times. Implementing the proposed resource allocation optimization method effectively enables LoRaWAN scaling using LoRa repeaters to increase coverage while minimizing implementation and maintenance costs.

In [156], the authors proved that AI could improve the performance of cloud applications by leveraging data-driven insights and automation. AI-based anomaly detection techniques can detect abnormal behaviour or underperformance of cloud applications at

an early stage. AI algorithms continuously monitor system metrics and compare them to established baselines to quickly detect potential problems and alert administrators. Timely intervention through AI-driven anomaly detection can minimize downtime and optimize performance and AI-driven resource allocation contributes to a consistent and reliable user experience, higher customer satisfaction, improved brand awareness, and increased revenue opportunities. Efficient use of resources through AI-driven resource allocation minimizes waste, lowers operating costs, and reduces energy consumption, thus supporting environmental sustainability by minimizing waste, lowering operating costs and reducing energy consumption. Machine learning algorithms can develop predictive models based on historical data to forecast resource needs and proactively adjust resource allocation. Proactive resource provisioning eliminates the pitfalls of under-resourcing and over-resourcing, ensuring optimal application performance and cost efficiency.

The paper [157] proposed a user signature-based identity management framework to optimize user authentication time in peer-to-peer distributed IoT blockchain networks. It evaluates various cryptographic techniques and hash functions built on the Modified Merkle Hash Tree (MMHT) data structure algorithm to enhance data security and integrity. The results show that using the AES-128 encryption algorithm with the MMHT algorithm and SHA3 hash function provides a minimum gain of 36% in time optimization compared to other algorithms. This improves the performance of user integrity checks in blockchain networks while ensuring network scalability. The practical design of identity signatures can be effectively used in decentralized IoT blockchain networks, providing high security against identity threats. This paper also highlights the importance of cryptographic algorithms and hash functions in user integrity verification and proposes an improved registration process for blockchain networks with identity providers.

Table IV.3: A comparative summary of previous studies utilizing Non-heuristic algorithms

Paper	Algorithm	Targeted Application	Advantages	Limitations
[133]	M/M/m Queueing System	• Performance Optimization for Cloud Computing	• Proposed model optimizes average wait time, average queue length, and number of customers.	• The model is not implemented in a real Cloud.
[140]	BAT	• Task scheduling and virtual machine allocation in cloud computing	• Hybrid model improves resource, bandwidth, and memory utilization. • MOML preemption policy reduces the load on virtual machines.	• The proposed does not employ real-time workflows.
[139]	BioDistSys system.	•Performance optimization of IoT-based biological systems.	• Optimizes cache, persistence, and processing nodes for better performance •Improves cache hits and system performance significantly • Existing performance optimization techniques do not consider the system's functional context history data.	
[151]	Machine learning algorithms for resource allocation and task scheduling in cloud computing.	•Optimization performance of data warehouse	• ML integration in data warehousing reduces latency and enhances query optimization.	• High failure rates of conventional data warehouses. •Difficulty in making changes and satisfying real-time data requirements.

[152]	HunterPlus	•Task Scheduling for Cloud-Fog Computing	•HunterPlus improves energy consumption per task and job completion rate. •Intelligent task-scheduling algorithms reduce energy consumption in cloud-fog environments	N/A
[153]	CNN, LSTM, Bi-RNN	•Edge Computing	•Deep Q network (DQN) algorithm can handle high complexity in edge computing. •Edge computing enables faster response speeds and network stability for AI applications.	*bullet* Traditional machine learning algorithms have limitations in solving optimization problems of edge computing.
[154]	IWHOLF	•Task scheduling in cloud computing	•Reduction of makespan and maximization of resource utilization in the cloud platform.	• Lengthy scheduling time
[155]	VNS	•Resource optimization in IoT	• Reduces implementation and maintenance costs. •Large spread factor for extended coverage LoRaWANs	•The impact of networks generated with different Graph variants is not investigated.
[156]	AI algorithms	•Cloud performance optimization.	•AI-driven resource allocation systems proactively adjust resource distribution. •Deep learning models excel at recognizing intricate resource allocation patterns.	N/A

[157]	MMHT	• Time Optimization in IoT.	• Ensures system performance and user integrity through identity management. •High security against identity threats in IoT blockchain networks.	N/A

IV.4 Discussion and critical analysis

After analyzing the two tables (IV.2 and IV.3), we can note that: Artificial intelligence algorithms, such as deep learning-based approaches and reinforcement learning, can process complex data and decision-making tasks. These algorithms can learn patterns and optimize performance based on large data sets and thus have the potential to provide highly accurate solutions tailored to specific cloud computing scenarios. However, since these algorithms require significant computational resources and large amounts of training data, they can be challenging to acquire and deploy in resource-constrained cloud environments.
On the other hand, metaheuristics are known for their versatility and efficiency in exploring the solution space without relying heavily on explicit training data or computational resources. They provide a more intuitive optimization approach, using stochastic processes and intelligent search strategies to find near-optimal solutions. Metaheuristics are often preferred when computational resources are limited or when real-time adaptation is important due to their ability to quickly adapt to changing conditions and requirements.

Ultimately, the choice between AI algorithms and meta-heuristic techniques depends on the specific optimization goals, constraints, and available resources. In some cases, a hybrid approach can be used to optimize performance in a cloud computing environment, combining the best of both AI and meta-heuristic techniques. This approach uses the learning capabilities of AI algorithms and the flexibility and efficiency of meta-heuristic techniques, sometimes providing the most effective solution.

In addition, we note that the papers in both sections (IV.1 and IV.2) optimize either the "Client" or "System" preferences. For this, our research introduces a model that optimizes cloud performance for clients and systems, positioning our research at the intersection of these two sides, as shown in Figure IV.2. Hence, by reviewing all the research work presented above, we can notice that most of the solutions out there are all focused either on helping customers or helping service providers by taking account their preferences, but not both, despite their use of different methods and algorithms. At our acknowledgement, no work considers the needs of both sides simultaneously, which leaves a big gap in making cloud services work better.

To remedy these shortcomings, we focused on a very specific objective: using Artificial

Intelligence approaches and MCDA techniques to propose a new multi-criteria decision model that focuses on optimizing operational costs and performance, helping to optimize the management of data centre resources, and representing the interaction between criteria. Our model aims to make finding cloud services easier while also dealing with all users and systems' different needs and limits. So, compared to what's been done in the literature, our idea looks at user and system preferences, as shown in Figure (IV.2). This means we can solve the problem of optimizing cloud service performances while ensuring everyone's needs are met.

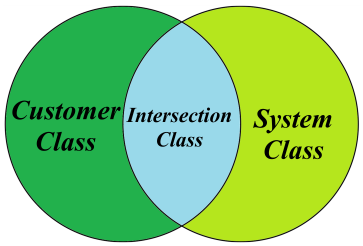


Figure IV.2: The contextual situation of our research work

IV.5 Conclusion

This chapter has provided valuable insights into the current work state in cloud services performance optimization. We have highlighted key findings, methodologies, and gaps in the literature. By synthesizing and critically evaluating prior studies, this review sets the stage for the present research, identifying areas where our contributions can be made. The diverse perspectives and methodologies in the reviewed literature contribute to a deeper understanding of the research problem and provide a context for the study's objectives and hypotheses. Indeed, we have categorized the related work into "Classification according to parameters" and "Analysis based on the nature of the used algorithms ". In addition, we have proposed comparative tables in each area. Overall, this chapter provided a comprehensive overview of the latest research in the performance optimization of cloud services and identifies key trends, challenges and future directions in this rapidly evolving field.

Chapter V

Proposed Models

V.1 Introduction

The successful development of cloud computing has led more and more people and businesses to use cloud computing. Cloud computing reduces costs on the one hand and increases efficiency on the other. Since users are very concerned about the quality of service, optimizing the performance of cloud computing has become crucial for the success of cloud computing [133]. Moreover, the number of cloud providers is increasing rapidly. Therefore, a significant challenge is choosing the cloud provider that best meets customer needs and optimizes cost and performance. In this context, we propose two contributions to help customers select the best provider that best meets their needs and optimizes cost and performance. Indeed, this chapter introduces two key contributions to enhance our understanding of optimizing the performance and cost of cloud services. These contributions are detailed into two parts: the first one presents a novel model for robust decision-making in cloud service selection through a multi-criteria decision support system. This model integrates a modified neural network, applying the ELECTRE technique and Tabu search algorithm to evaluate cloud services and optimize cost and performance, providing a comprehensive framework for decision-makers.

Whereas, in the second part (contribution), we consider a service with five criteria. Two criteria are specifically related to the customer side, two criteria focus on the cloud provider side, and one criterion is common to both the customer and the provider. However, the customer tries to minimize this criterion when the provider tries to maximize it. This is exactly why we chose a multi-criteria decision-making system. In many decision-making problems, different viewpoints are encountered, often leading to contradictions and different opinions.

V.2 Tabu-OptiNimbus

In our model, as shown in Figure (V.1), we've divided a sophisticated multilayer neural network tailored precisely to our unique needs, departing from conventional standards to

address the intricacies of cloud provider selection. At the forefront of this architecture lies the input layer, where each neuron symbolizes a distinct cloud provider, accompanied by its set of parameters representing crucial decision-making criteria.
Delving into the neural network's hidden layers, the first tier seamlessly integrates the renowned ELECTRE (Elimination and Choice Translating Reality) method. This algorithm meticulously evaluates and ranks the proposed providers, sieving out the most promising candidates. Transitioning to the second hidden layer, we deploy the Tabu search algorithm, famed for its prowess in navigating complex solution spaces. Here, it diligently sifts through the elite pool of providers identified by ELECTRE, striving to pinpoint the optimal solution aligning with the customer's bespoke needs. This iterative process harmonizes the strengths of both layers, ensuring that the chosen provider meets and exceeds expectations. By orchestrating this intricate dance between algorithms, our model optimizes the decision-making process, paving the way for tailored, efficient cloud solutions that elevate customer satisfaction.
.

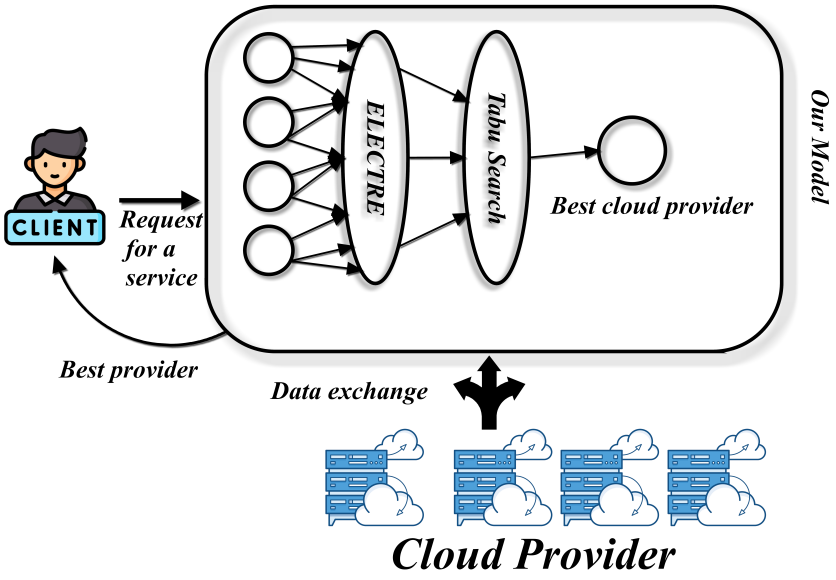


Figure V.1: An overview of the general architecture

V.2.1 Functionality of the Proposed Model

Our system's basis of the decision-making process is meticulously collected data from each cloud provider's data centre. This data includes critical metrics such as storage capacity allocated to end users, pricing structures, and uptimes, which comprehensively build the landscape of available options. These three key collected criteria are seamlessly incorporated into the first hidden layer of the neural network and feed the advanced analysis engine of the ELECTRE method.
Based on the performance of these critical parameters, ELECTRE can identify candidate providers. Once the ELECTRE method has identified the best performers, the flag is passed to the second hidden layer, where the challenging Tabu Search algorithm takes over. Tabu Search has the unique ability to navigate the complex solution space and

Table V.1: Decision matrix illustration

Alternatives		Criteria			
		C_1	C_2	...	C_n
	a_1	g_{11}	g_{12}	...	g_{1n}
	a_2	g_{21}	g_{22}	...	g_{2n}
	...	...	...	...	...
	a_n	g_{n1}	g_{n2}	...	g_{nn}

embarks on a relentless quest to identify the best cloud providers among the elite identified by ELECTRE. Our system identifies the selected cloud providers and orchestrates the final optimization, ensuring that the selected solution is tailored to meet user needs with unmatched efficiency and effectiveness.

V.2.1.1 ELECTRE Tri

ELECTRE II [158] is a multicriteria analysis method that solves decision problems with greater accuracy. This method was the first of the ELECTRE methods specifically designed to deal with ranking problems. The evaluation matrix is the starting point of the ELECTRE II method, in which alternatives (The alternatives, in our case, illustrate the services. Each service has five criteria) are evaluated on different criteria. It aims to rank actions from best to worst. Based on a total pre-ordering principle, ELECTRE II assumes that all actions are comparable.

ELECTRE consists of two main steps. The first step is the preparation of the decision matrix (see Table 1), where g_{ij} denotes the value of variant i concerning criterion j. The second step is the calculation of the concordance and discordance matrices [158] .
ELECTRE consists of two main steps. The first step is the preparation of the decision matrix (see Table V.1), where g_{ij} denotes the value of variant i concerning criterion j. The second step is the calculation of the concordance and discordance matrices.

In the first step, we sort the services of each class from the best to the worst by calculating the concordance, discordance, and dominant matrices.

V.2.1.2 Calculation of the Concordance Matrix

The concordance matrix is generated by summing the weights of the elements in the concordance set. The strength of the hypothesis that alternative A_i is at least as good as alternative A_j is evaluated using the concordance index between the pair of alternatives A_i and A_j which is calculated using formula (V.1)

$$c(a,b) = \frac{\sum k_j}{k} \quad \text{where} \quad g_{j(a)} \geq g_{j(b)} \forall_j \tag{V.1}$$

Where $g_{j(a)}$ and $g_{j(b)}$ are the sets of criteria for which a is equal or preferred to b, k_j is the weight of the j^{th} criterion.

V.2.1.3 Calculation of the Discordance Matrix

The discordance index D (a, b) is calculated by the formula (V.2) or (V.3):

$$D(a,b) = 0 \quad \text{if} \quad \forall j, g_{j(a)} \geq g_{j(b)} \tag{V.2}$$

Else

$$D(a,b) = \frac{1}{\sigma} MAXj[g_{j(b)} - g_{j(a)}] \tag{V.3}$$

$$\sigma = max \left| g_{j(b)} - g_{j(a)} \right| \tag{V.4}$$

After calculating the concordance and discordance indices for each pair of alternatives, two outranking relationships are constructed by comparing these indices with two pairs of threshold values: (C+, D+) and (C-, D-). The pair (C+, D+) is defined as the concordance and discordance thresholds for the strong outranking relationship, and the pair (C-, D-) is defined as the thresholds for the weak outranking relationship, where C+ > C- and D+ > D-. Then, outranking relationships are constructed according to the following two rules [158]:

- If $C(a,b) \geq C+, D(a,b) \leq D+$ and $C(a,b) \geq C(b,a)$, then alternative a is considered to strongly outperform alternative b. Likewise,
- If $C(a,b) \geq C-, D(a,b) \leq D-$ and $C(a,b) \geq C(b,a)$, then alternative a is considered to weakly outperform alternative b.

The values of C-, C+, D-, and D+ are given by the decision makers [158].

Service a is better than service b if the following strong outranking relation holds:

- If $C(a,b) \geq C+, D(a,b) \leq D+$ and $C(a,b) \geq C(b,a)$, then service a is considered to strongly outperform service b.

In the second step, we eliminate the services that are weakly outranking. We consider that a service a is weakly outranking with b when the following condition is true:

- If $C(a,b) \geq C-, D(a,b) \leq D-$ and $C(a,b) \geq C(b,a)$, then service a is considered to strongly outperform service b.

V.2.1.4 Modified Tabu Search

The first step in the process is using the ELECTRE method, which is explained in section(V.2.2); the second step is the "Modified Tabu Search."
The Tabu Search algorithm is a famous optimization method at the forefront of our innovative approach to improving and developing cloud services. The basic form of the TabuSearch algorithm is to explore the complexities of optimizing cloud services and navigate through many potential solutions with unparalleled efficiency. TabuSearch systematically explores the solution space using optimization and diversification strategies. Using a memory-based mechanism to guide the search skillfully avoids stagnation at the local optimum. Throughout this iterative process, the algorithm dynamically adjusts trajectories and iteratively refines existing solutions while avoiding previously explored paths to enable exploration of the solution landscape. Integrating the basic tabu search algorithm into the optimization framework of cloud services unlocks the potential for improved performance, streamlined resource allocation, and ultimately increased user satisfaction.
Basic Tabu Search Algorithm : (Algorithm 8)

Algorithm 8 Basic Tabu Search Algorithm

```
function TabuSearch(initial_solution):
    current_solution ← initial_solution
    best_solution ← initial_solution
    tabu_list ← empty list
while stopping_condition_not_met do
    neighbors ← generate_neighbors(current_solution)
    best_neighbor ← select_best_neighbor(neighbors, tabu_list)
    if aspiration_criteria_met(best_neighbor) then
        current_solution ← best_neighbor
    else
        current_solution ← select_next_solution(neighbors, tabu_list)
    end if
    update_tabu_list(tabu_list, current_solution)
    if evaluate_solution(current_solution) < evaluate_solution(best_solution) then
        best_solution ← current_solution
    end if
end while
Return best_solution
```

Modified Tabu Search Algorithm Because of the heterogeneous criteria, we must calculate their global performances. For that, we will use the weighted sum. The weighted sum determines each action's global performance a_i given by $V(a_i)$. This overall performance is calculated using the expression(V.6):

$$V(a_i) = \sum_{j=1}^{n} w_j \times e_{ij} \tag{V.5}$$

where w_j represents the coefficient of importance of the criterion j and e_{ij}, the evaluation of the action a_i with regard to the criterion j.
So, the modified tabu search algorithm is as follows (Algorithm 9):

Algorithm 9 Modified Tabu Search Algorithm

```
function TabuSearch(initial_service):
    current_service ← initial_service
    best_service ← initial_service
    tabu_list ← empty list
while stopping_condition_not_met do
        Weighted_Sum
    neighbors ← generate_neighbors(current_solution)
    best_neighbor ← select_best_neighbor(neighbors, tabu_list)
    if aspiration_criteria_met(best_neighbor) then
        current_service ← best_neighbor
    else
        current_service ← select_next_service(neighbors, tabu_list)
    end if
    update_tabu_list(tabu_list, current_service)
    if evaluate_service(current_service) < evaluate_service(best_service) then
        best_service ← current_service
    end if
end while
Return best_service
```

Table V.2: Performance table

	Storage	Price	Execution time
Dropbox	2	7	0.25
Box	10	8	0.80
Google Drive	15	9.99	0.73
Microsoft One Drive	5	7	0.25
Amazon	5	4	0.73
Apple iCloud	5	9.99	0.78
Hubic	25	5	0.83

V.2.2 Practical Application Analysis

This section discusses a simulation of multicriteria decision-making models to optimize the cost and performance of cloud users. We aim to demonstrate our proposed model's practical effectiveness and usefulness in cloud computing by examining specific case studies and scenarios. Through these case studies, we will explore different aspects of cloud service selection and highlight the model's ability to optimize costs while improving the performance of cloud users.

We analyze a customer scenario to demonstrate two key aspects. Firstly, we outline the sequential steps or procedures aligned with our model, and secondly, we delve into the interaction among the various components within the suggested architecture. We presuppose a situation where a customer asks a provider to purchase storage space. Despite having a list of potential cloud providers, determining which one aligns with the specific requirements is challenging. To commence, we present the performance table of cloud providers (refer to Table V.2), where the significance of each decision-making criterion is expressed through a weight represented by k_j.

Computation of Concordance Matrix To calculate the concordance matrix, we use formula (V.1):
$C(Dropbox, Box) = \frac{0+3+2}{8} = \frac{5}{8} = 0.625$
$C(MicrosoftOneDrive, Box) = \frac{3+3+2}{8} = \frac{8}{8} = 1$

Computation of Discordance Matrix The discordance index is calculated using the formula (V.2, V.3, and V.4):
$D(Dropbox, Box) = \frac{8}{23} = 0.34$

$\sigma = 25 - 2 = 23$ The limited concordance = 1
The limited discordance = 0.26
So, we have: Google Drive S Box, which means Google Drive overclass Box Microsoft One Drive S Dropbox; Amazon Cloud Drive S Apple iCloud. After analyzing the two matrixes, we obtained the four best providers: Google Drive, Microsoft One Drive, Amazon, Hubic.

Table V.3: Performances weight table

Criteria	Storage	Price	Execution time
Weight(k_j)	3	3	2

Tabu Search We've identified the top four providers and aim to pinpoint the optimal choice by utilizing the adapted Tabu search algorithm, which serves as the second concealed layer within our modified neural network. The algorithm's advancement is depicted in Figure 3.
$S : GoogleDrive, MicrosoftOneDrive, Amazon, Hubic$, S is the set of initial solutions.
We assume that Google Drive is s_0 and set $s^* = s_0$. s^* the best solution so far.
s:new solution from the neighborhood of s^*.
$f(s)$: objective function.
$f(s^*)$: the value of the best solution.
l: The tabu list is initially empty.
The stop condition is the number of elements in the tabu list, which is the number of aspects of the initial list -1.
The supposed weights for each criterion of the weighted sum are in Table (V.3).
We choose a solution from the neighbourhood s of s_0, $s = Hubic$.
We calculate the object function of each provider:
$f(googledrive) = 15 \times 3 + 9.99 \times 3 + 0.73 \times 2 = 351.5$
$f(Hubic) = 25 \times 3 + 5 \times 3 + 0.78 \times 2 = 481.56$
$f(googledrive) < f(hubic)$
So, we put hubic in tabu list, and we choose another solution from the initial solution list.
l:hubic.
s:Amazone.
$f(Amazone) = 43.07$
$f(googledrive) > f(amazon)$, we put google drive in the tabu list and $s^* = Amazon$.
We choose another solution from the initial list: $s = Microsoftonedrive$. $f(Microsoftonedrive) = 132,5. f(Amazon) < f(Microsoftonedrive)$. So, we put Microsoft on the tabu list.
Finally, the stop condition was checked, and Amazon was the best provider. The result of our modified neural network is Amazon.

V.3 MOOA-CSF: A Multi-objective Optimization Approach for Cloud Services Finding

Figure (V.1) shows the overall architecture of our system, which consists of five interconnected components.

[2] The first component of our system is a classification component using a multilayer neural network CC-ANN. This component consists of three layers. The first layer is the input layer, where the inputs represent the service criteria. As a result, the parameters of each cloud service are extracted considering the customer's requirements, and a vector

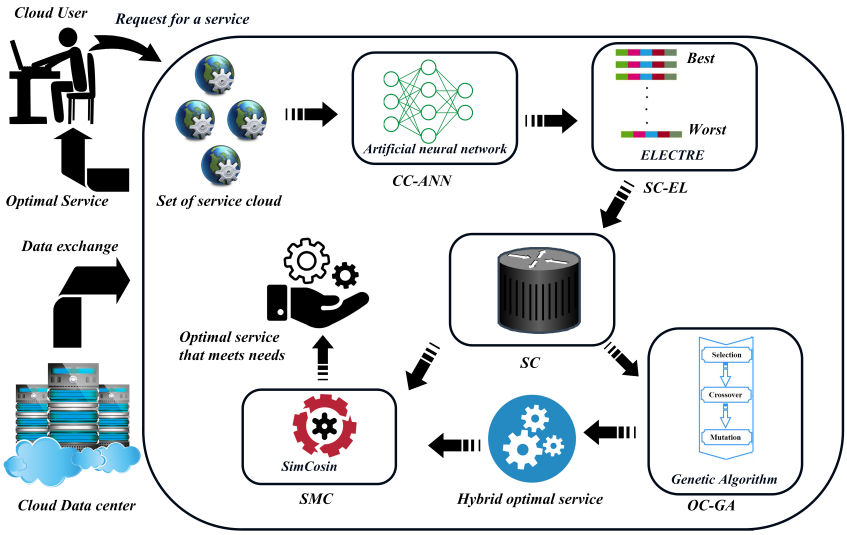


Figure V.2: An overview of the proposed general architecture

represents each service. The second layer is the hidden layer, which contains the functions that perform the classification. Finally, the third layer is the output layer, which produces output to three classes: CPC, which includes client-preferred services; SPC, which includes system-preferred services; and CSC. The second component of this architecture is called the ranking component SC-EL. It is based on the ELECTRE method, which ranks the services within each class and ranks them from best to worst. This ranking process assigns weights according to the importance of each criterion. The Storage Component SC acts as a central repository within the system and is responsible for storing all relevant data. The fourth component is the optimization component OC-GA, which works on the genetic algorithms (GA) principle. Its main function is to produce a new generation of cloud services derived from the first three classes described above; the services produced by GA are considered hybrid services that exist in theory but are not realized in practice. To address this, it is necessary to assess the similarities between these hybrid and existing services to address this. To achieve this, a fifth component, the similarity component SMC, is introduced, which helps to identify the best service (closest match) among the services obtained in the previous step. The whole process is shown in the sequence diagram in Figure V.2.

.

V.3.1 Classification Step

This step is performed by the CC-ANN component, as shown in Figure (V.3). The ANN comprises three layers: input, hidden, and output. The input layer is comprised of five nodes, denoting the criteria of a service, namely Reliability, Throughput, Availability, Cost, and Response Time. These criteria values are then propagated to the hidden layer. Within the hidden layer, the activation values are passed from neuron to neuron, where each neuron aggregates the received activations and updates its value using a transfer function. This process involves an anticipation mechanism. Subsequently, the difference between the predicted and actual values (errors) is propagated backward through the network. .

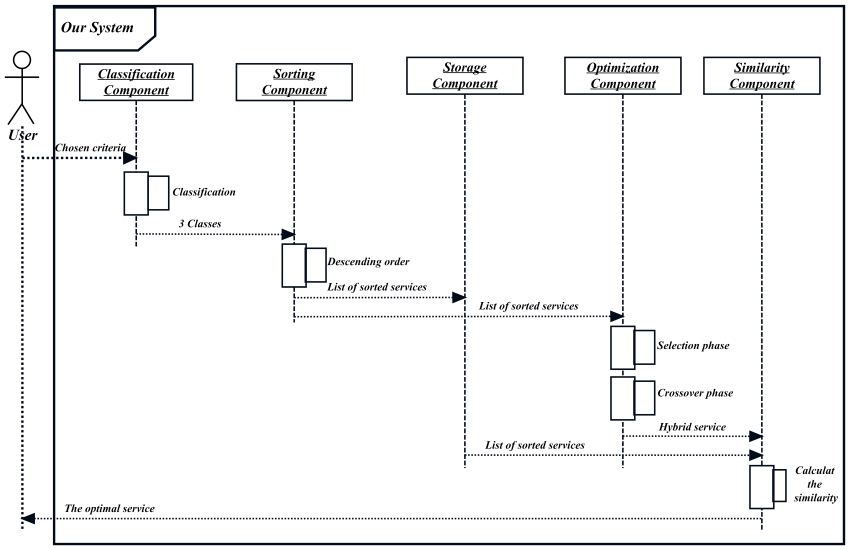


Figure V.3: Sequence Diagram of the general architecture

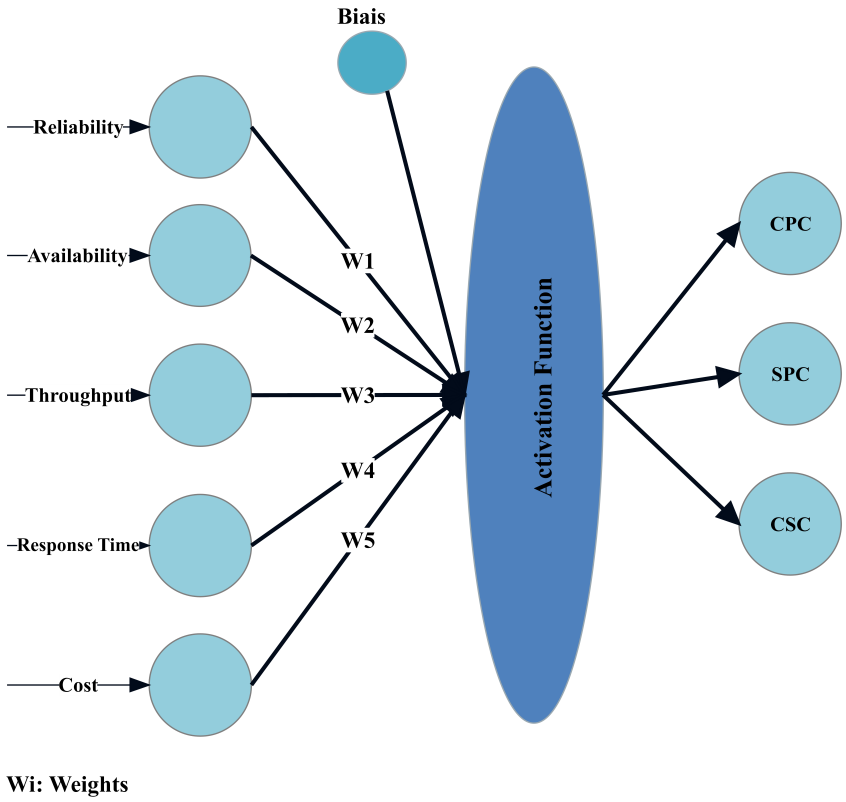


Figure V.4: The layers of the CC-ANN component

V.3.1.1 Activation Function

The activation function transfers the input values to an output signal. In this paper, we have chosen the hyperbolic tangent function (Tanh). The Tanh function is similar to the sigmoid function but symmetrical around the origin. This results in different signs of outputs from the previous layers being fed into the input of the next layer, as defined by the formula (V.6).

The Tanh function is continuous and differentiable, with values ranging between -1 and 1. This property often makes it easier for optimization algorithms to converge, especially in neural networks. Compared to the sigmoid function, the gradient of the Tanh function is steeper. Tanh is more commonly used than the sigmoid function because it has gradients that are not bound to vary in a certain direction, and it is also centred on zero [159].

$$Tanh(a) = \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}} \tag{V.6}$$

where a is the value of the neuron. Figure (V.4) shows the process of the classification component, where d denotes the desired goal and E is the error margin.

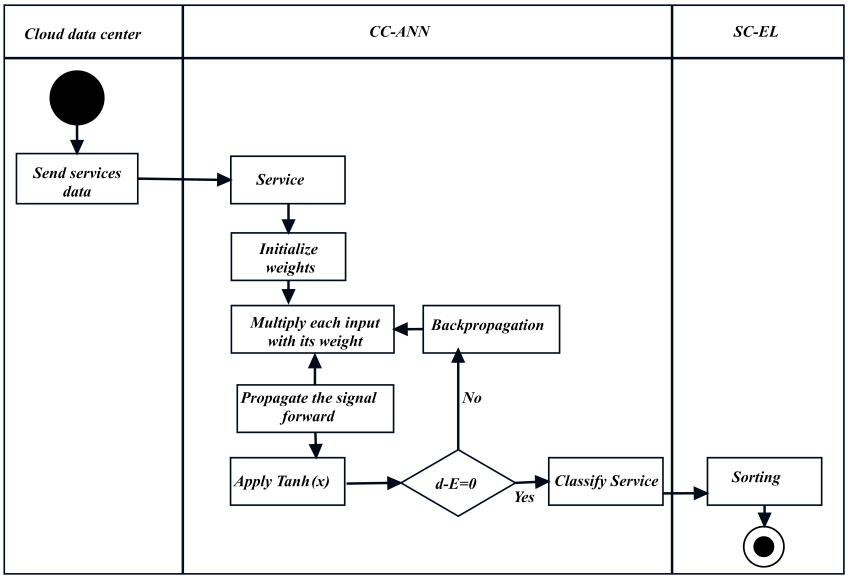


Figure V.5: Activity diagram of the CC-ANN component

.

V.3.1.2 ANN training

Training CC-ANN involves several steps:

1. **Data Collection:** Collect a set of data on cloud services, where each service is represented by features such as reliability, availability, cost, throughput, and response time, along with corresponding labels indicating their classes (e.g., client preference class (CPC), system preference class (SPC) and common service class (CSC)).

2. **Data Preprocessing:**As a preprocessing step for the dataset, features are normalized so that they scale. Similarly, missing values, if any, are handled, and the dataset is split into Training, Validation, and test sets.

3. **Model Architecture:**We designed the neural network architecture using CC-ANN. We fixed the number of layers, which is 3. Then, we select the type of fully connected

layers and the activation function. The output layer usually consists of softmax activation for classification tasks to produce class probabilities.

4. **Training:**Train the neural network on the training data using the compiled model. Use backpropagation to adjust the model weights to minimize the loss function iteratively.

5. **Validation:**Monitor the model's performance on the Validation set during Training to detect overfitting and adjust hyperparameters accordingly.

6. **Evaluation:**Evaluate the trained models on a test set to assess generalization performance.

7. **Deployment:**The trained model is deployed in the production environment to classify cloud services according to their characteristics.

V.3.2 Sorting and Elimination Step

This component performs according to the ELECTRE II principle. One pivotal component employed in cloud services' sorting and elimination process is ELECTRE II. This abbreviation "Elimination and Choice Expressing Reality" method offers a structured framework for multicriteria decision-making. By leveraging ELECTRE II, various criteria essential for selecting cloud services are systematically evaluated and weighted according to their significance. This facilitates a comprehensive assessment, considering diverse factors such as performance, cost, reliability, and scalability. Through a series of comparative analyses, ELECTRE II enables the identification of superior cloud service options while efficiently filtering out suboptimal choices. Its methodology ensures a rational and transparent decision-making process, ultimately enhancing the quality and efficiency of cloud service selection for businesses and organizations.

V.4 Optimization Step

This component is based on the principle of GA, a meta-heuristic approach to solving multi-objective optimization problems. GA is inspired by the principles of Darwin's theory of evolution and is often used as an evolutionary computational model in various fields of study [160]. Currently, GA's are recognized as a potent tool in optimization, having been applied in computer science, engineering, education, and stock market data mining optimization [160].
We have modified the GA to make it adaptive to our context. Its operation after modification is as follows: .

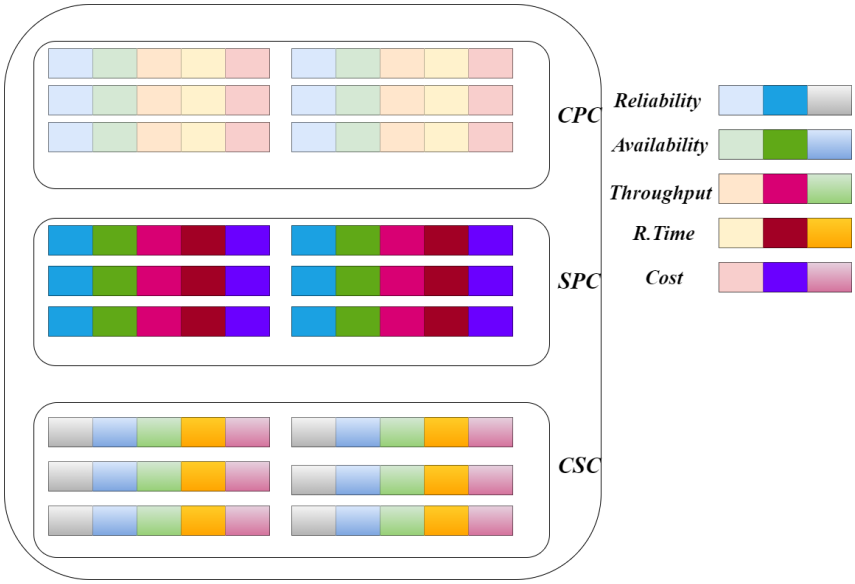


Figure V.6: Initial populations of CP, SP and CS classes

1. The selection step is a fundamental component of the evolutionary process that mimics the principle of "survival of the fittest" from natural selection. This step involves choosing individuals (services) from the current population to serve as parents for the next generation based on their fitness or performance in solving the optimization problem. The initial population in our approach is not generated randomly. Instead, we utilize the three classes obtained during the classification phase as our initial populations. Consequently, there are three separate initial populations, not just one (as shown in Figure V.6).

2. Before selection, each solution in the population must be evaluated based on predefined criteria or objectives. These criteria could include reliability, availability, cost, throughput, and response time. Each service's fitness is determined by how well it meets these criteria or objectives. For example, a solution comprising cloud services with low cost, high reliability, and fast response time would be considered more fit than a solution with higher costs and lower reliability. The population is evaluated by assigning a fitness value to each service so we can generate a new population.

3. The algorithm determines the termination of the search process based on specific predefined conditions. Typically, these conditions are met when the algorithm reaches a fixed number of generations or discovers a satisfactory solution.

4. If the termination condition is unsatisfied, the population proceeds with the selection step. During this step, one service is chosen from each class based on its fitness score, with higher fitness scores leading to a higher likelihood of selection.

5. Following the selection step, the algorithm proceeds to implement crossover on the chosen services, as illustrated in Figure (V.7). This stage involves creating new services for the subsequent generation through the process of crossing or recombination.
.

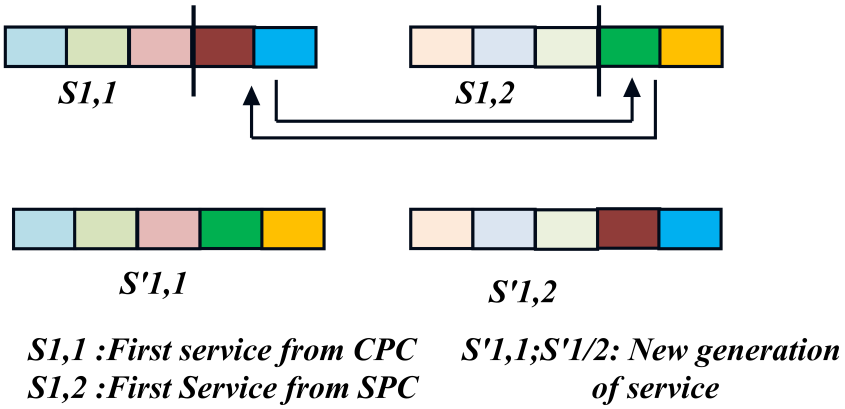


Figure V.7: Illustration of the crossover operation results

Each solution represents a combination of cloud services with specific features such as reliability, availability, cost, throughput, and response time. During the crossover step, pairs of parent solutions are selected based on criteria such as their fitness (i.e., how well they perform in achieving the desired objectives, like minimizing cost or maximizing throughput).
Once the parent solutions are selected, the crossover is applied to their genetic representations. This typically involves exchanging genetic information between the parents at random points or based on specific rules. For example, an arbitrary crossover point is chosen in a simple one-point crossover, and the genetic material beyond that point is swapped between the parents to create two offspring solutions. After crossover, the new offspring solutions inherit characteristics from both parents. In the genetic algorithm optimization process, these offspring solutions represent potential candidates for the next generation. By combining genetic information from multiple parent solutions, crossover allows for the exploration of different combinations of cloud services that may exhibit desirable characteristics or performance improvements compared to the parent solutions.

6. At this stage, the new population returns to the assessment step, and the process begins again. We call each cycle of this loop a *generation*.

7. When the termination condition is met, the algorithm breaks out of the loop and usually returns its final search results to the client/provider.

V.4.1 Similarity Step

The calculation of the degree of similarity between two services is ensured by the SMC, which is based on the similarity SimCosin(X, Y). We suppose that we have two services,

X and Y, represented by two vectors, each vector containing five criteria [161]:

$$X = [R_x, Av_x, C_x, Th_x, Rt_x]; Y = [R_y, Av_y, C_y, Th_y, Rt_y]$$

Additionally, the function SimCosin(X,Y) must satisfy the following properties [161]:
Property 1: $0 \leq \cos(X, Y) \leq 1$
Property 2: $cos(X, Y) = cos(Y, X)$
Property 3: $cos(X, Y) = 1; if X = Y.$

Here's how you can break down the calculation steps:

1. Compute the dot product of the feature vectors X and Y. This involves multiplying each feature value of X with the corresponding feature value of Y. And summing up these products.
   $X \cdot Y = x_1 \times y_1 + x_2 \times y_2 + ... + x_n \times y_n$ x_i and y_i are the feature values of X and Y respectively for the ith feature.

2. Compute the Euclidean norms of the feature vectors X and Y. This involves taking the square root of the sum of squares of each feature value. $\|X\| = \sqrt{x_1^2 + x_2^2 + ... + x_n^2}$ $\|Y\| = \sqrt{y_1^2 + y_2^2 + ... + y_n^2}$

3. Finally, divide the dot product by the product of the Euclidean norms to obtain the cosine similarity. $SimCosin(X, Y) = \frac{X \cdot Y}{\|X\|\|Y\|}$

The resulting value will be in the range [-1, 1], where 1 indicates perfect similarity, 0 indicates no similarity, and -1 indicates perfect dissimilarity.

This calculation measures how similar the feature vectors representing the two services are in their characteristics.

V.4.2 Case study

A simulation environment is established to validate our system. The simulation context proceeds as follows: we suppose we have several cloud service providers, each one providing a set of service. A client searches for optimal services that meet their needs among these services, and each service has (m) criteria.

. In the first phase, we develop a neural network to classify these services and assign each to the appropriate class.

As shown in Figure (V.8), we train the neural network using a dataset in the first step. We initialize our weights randomly, then feed-forward the values from one layer to the next. If the output does not equal the desired value, we back-propagate from the output neuron to the input neuron. We update the weights and feed forward the values. We repeat this

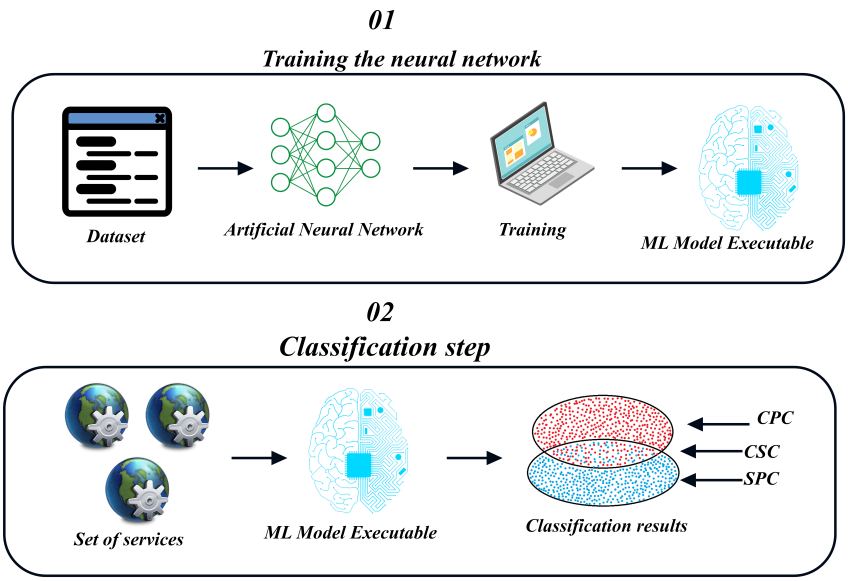


Figure V.8: Classification process of the CC-ANN component

Table V.4: Initialization of the five criteria weights

Criterion	Availability	Cost	Response Time	Throughput	Reliability
Weight	2	3	3	2	1

process until we find the desired values and obtain a model of a neural network.

In the Classification step (Figure V.8), we use the neural network model to classify services into three categories. Each class contains a set of services with the same value range. The first class (CPC) includes 916 services, the second (SPC) consists of 640 services, and the last one (CSC) has 444. After that, we will copy the list of these services and send it to the storage component. Then, we transfer the result to the sorting component.

As a second phase, the sorting component uses the ELECTRE method to sort and eliminate services to find the best one in each class. Five parameters or criteria characterize the decision problem. All criteria are advantage criteria, i.e., performance is better when the score is high.
The weights of the criteria are presented in Table (V.4).
Due to the large number of services in the dataset, we will take as a sample the services S6, S8, S9, S907, S908, and S910. Therefore, the service performance matrix of the first class, as illustrated in Table 4, is used to calculate the concordance matrix C (a,b), which is shown in Table 6. This is calculated using the formula (1) quoted in the previous section.
We calculate the discordance matrix D (a,b) of our illustrative example using formula (3); results are shown in Table (V.7). To obtain σ, we calculate, for each attribute, the differences between all its values in the dataset. Moreover, the attribute corresponding to the maximum value of these deviations is maintained. We then calculate σ, equal to the total value of the maintained attribute minus its minimum value (see formula

Table V.5: Performance matrix of the illustrative example

Service ID	Availability	Cost	Response Time	Throughput	Reliability
Service 1	59	150	1,020	1,804	78
Service 2	62	10	1,030	1,842	80
Service 3	88	74	1,115	1,735	74
Service 4	63	70	1,061	1,651	88
Service 5	84	34	1,120	2,190	98
Service 6	61	68	1,046	2,178	68
Service 7	64	62	1,500	2,120	95
Service 8	73	63	1,850	2,144	75
Service 9	81	124	1,244	2,060	86
Service 10	52	122	1,200	1,980	90
....					
Service 906	61	68	1,101	2,254	99
Service 907	64	62	1,175	1,655	43
Service 908	73	63	1,188	1,963	50
Service 909	81	117	1,990	1,889	65
Service 910	52	110	1,107	1,999	60
Service 911	83	53	1,330	2,025	77
Service 912	52	145	1,780	2,365	87
Service 913	85	130	1,640	2,458	92
Service 914	77	100	1,545	2,220	73
Service 915	62	95	1,256	1,933	70

Table V.6: The concordance matrix relating to the illustrative example

Service ID	Service 6	Service 8	Service 9	Service 907	Service 908	Service 910
Service 6	**1**	0,454	0,181	0,545	0,545	0,454
Service 8	0,545	**1**	0,454	**1**	**1**	0,727
Service 9	0,818	0,545	**1**	**1**	**1**	**1**
Service 907	0,454	**0**	**0**	**1**	**0**	0,454
Service 908	0,454	0,454	**0**	**1**	**1**	0,454
Service 910	0,545	0,272	**0**	0,545	0,545	**1**

Table V.7: The discordance matrix relating to the illustrative example

Service ID	Service 6	Service 8	Service 9	Service 907	Service 908	Service 910
Service 6	**0**	0,085	0,4	0,021	0,081	0,3
Service 8	0,035	**0**	0,435	**0**	**0**	0,335
Service 9	0,0008	0,004	**0**	**0**	**0**	**0**
Service 907	0,178	0,228	0,442	**0**	0,064	0,342
Service 908	0,128	0,178	0,435	**0**	**0**	0,335
Service 910	0,064	0,15	0,207	0,085	0,15	**0**

Table V.8: The dominant matrix relating to the illustrative example

Service ID	Service 6	Service 8	Service 9	Service 907	Service 908	Service 910
Service 6	Strong	-	-	-	-	-
Service 8	-	Strong	-	Strong	Strong	-
Service 9	Weak	-	Strong	Strong	Strong	Strong
Service 907	-	-	-	Strong	-	-
Service 908	-	-	-	Strong	Strong	-
Service 910	-	-	-	-	-	Strong

(4)). According to the QWS2 dataset, the maintained attribute is the Cost attribute; consequently, σ =140.

After calculating the concordance and discordance matrices, the dominant matrix is constructed from the two concordance and discordance indices. Thus, the following strong and weak outranking indices are obtained: C+=0,850, C-=0,750, D+=0,200, and D-=0,300. We obtain the dominant matrix as illustrated in Table (V.8).

Figure (V.9) shows the final rank between services. To get this graph, we follow these two properties:

- If service a outranks service b, an arrow starting at vertex a and ending at vertex b.
- If no outranking relation exists between the two services a and b, then no arrow can be drawn between the two vertices.

. After that, we sort services to eliminate those with weak outclass relations. A service with a weak outclass relation to another service means that the former is included. The remaining services, which have not been eliminated, are the best in each class. We use these services as an initial population. This latter is used to generate new populations based on the good services. We repeat the same process for the second and third classes, transferring the resulting services to the optimization component. The OC will consider each class as an initial population. At first, the OC crosses a service from the first class with the services from the second. Then, we evaluate the new service. We eliminate the service if the value is less than the fitness value. We will obtain a new population after crossing all services of the first class with all services of the second class. Then, we cross a service from the second class with the services of the third class, bringing a second

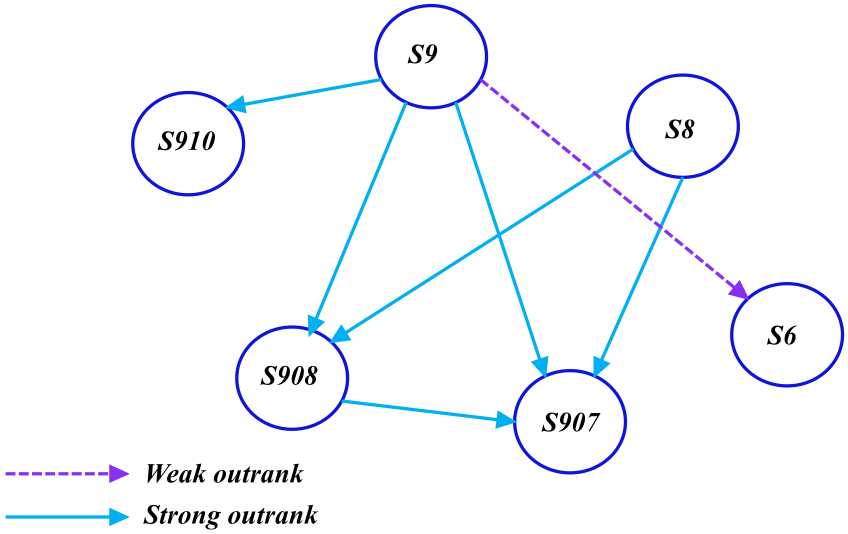


Figure V.9: The outrank relation between services relating to the illustrative example

population. After that, we travel the services of the new populations to get the final one. The final population contains hybrid services; the latter are abstract. To design the optimal service, we must calculate the similarity between services already stored in our storage component and those of the final population.

The similarity component (Figure V.10) calculates the similarity index between each class's best services and the final population's services. In our case, we use the Sim-Cos function. We keep the service if the similarity index is in the interval [0.8:1]. Furthermore, we keep all services that have a high index. Then, we send the list of these services to the client.

.

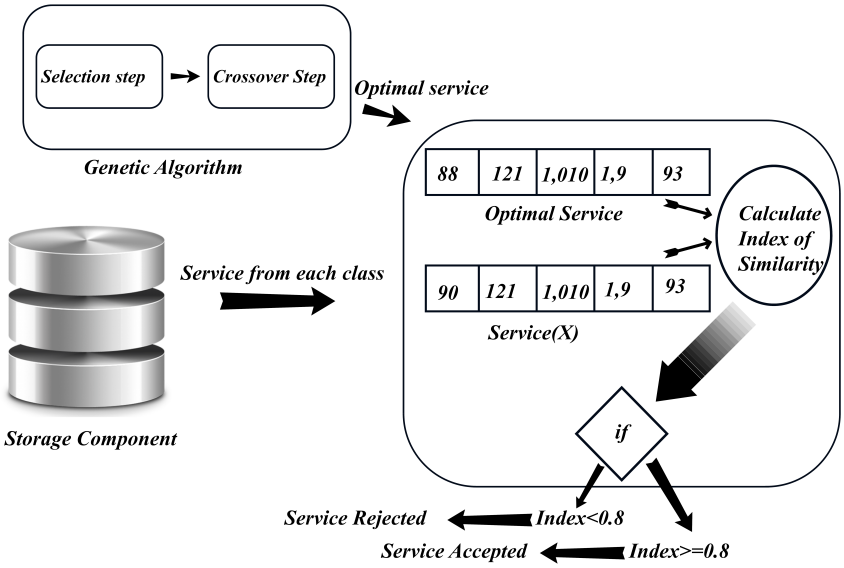


Figure V.10: The similarity component functioning

V.5 Conclusion

In conclusion, our Multicriteria Decision Model for Optimizing Costs and Performances for a Cloud User represents a significant step forward in achieving optimal price and performance outcomes in cloud computing environments. By seamlessly integrating supervised learning and multi-criteria techniques, our system offers a robust framework for cloud users to navigate the complex landscape of service selection.

At the core of our model lies an artificial neural network (ANN) that effectively categorizes cloud services based on their features, resulting in three distinct classes tailored to the client's needs, the system, and both. Leveraging the ELECTRE method, these classes are meticulously ordered, providing clarity and guidance in decision-making.

Furthermore, we have enhanced traditional optimization algorithms such as the genetic algorithm (GA) and the Tabu search to adapt to our system's unique requirements. This adaptation results in the theoretical creation of hybrid cloud services that optimally balance client and system preferences. To validate the practicality of these hybrid services, we employ similarity tests to assess their resemblance to other benefits within each class.

The performance of our model has been rigorously evaluated across various scenarios, demonstrating its efficiency and effectiveness in optimizing costs and performances for cloud users. Through comprehensive simulations, we have validated the efficacy of our approach, showcasing its potential to revolutionize decision-making processes in cloud computing environments.

Our Multicriteria Decision Model offers a comprehensive solution for cloud users seeking to optimize costs and performances. By leveraging advanced techniques such as supervised learning, multicriteria decision analysis, and enhanced optimization algorithms, our system empowers users to make informed decisions that align with their objectives and preferences, ultimately improving their cloud computing experience.

Chapter VI

Experimental results and evaluation

VI.1 Introduction

In the rapidly evolving landscape of cloud computing, the quest for optimal resource utilization while managing costs and ensuring performance remains paramount for cloud users. With the proliferation of cloud service providers and the diversity of service offerings, selecting the most suitable configuration can be challenging. Traditional decision-making approaches often prioritize cost considerations at the expense of performance or vice versa, leading to sub optimal outcomes.

To address this challenge, this experimental chapter presents a novel Multicriteria Decision Model tailored specifically for cloud users. By integrating multiple criteria such as cost, performance, reliability, and scalability, this model offers a comprehensive framework for decision-makers to navigate the complexities of cloud resource selection. Users can prioritize their preferences and constraints through a systematic evaluation process, leading to informed decisions that balance financial considerations and performance requirements.

This experimental chapter aims to demonstrate the effectiveness and versatility of the proposed model through real-world scenarios and case studies. By applying the model to diverse cloud computing environments, ranging from Infrastructure as a Service (IaaS) to Software as a Service (SaaS), this research seeks to provide empirical insights into its applicability across various contexts.

Moreover, the experimental chapter will explore the multicriteria decision model's practical implementation aspects, including data collection methodologies, analytical techniques, and Software tools utilized. By elucidating the model deployment and utilization process, this research endeavors to equip cloud users with the necessary knowledge and tools to leverage the model effectively in their decision-making processes.

Ultimately, the experimental chapter endeavors to contribute to advancing decision support systems in cloud computing by offering a robust framework that reconciles the often-competing objectives of cost optimization and performance enhancement. Through empirical validation and practical insights, this research endeavors to empower cloud users with the means to navigate the complexities of the cloud landscape with confidence and

efficiency.

VI.2 Experimental Environment

A simulation environment is established to validate our system. The simulation context proceeds as follows: we suppose we have several cloud service providers, each one providing a service. A client searches for optimal services that meet their needs among these services, and each service has (m) criteria.

The simulation environment is a PC with the following configuration: Nvidia GeForce GTX 1060 GDDR5, Intel Core i7-7700HQ CPU 2.80 GHz, and RAM 16 GB. The programming environment is Eclipse IDE 2020-09. The test data is based on the QWS2 dataset [162] where the number of services is 4000. The neural network was trained using 2600 services. Each service is composed of five criteria.

We leverage Java as our primary programming language alongside the CloudSim simulator to develop our innovative approach, known as MOOA-CSF. Java's versatility and robustness enable us to seamlessly integrate complex algorithms and implement intricate functionalities required for cloud optimization. With its object-oriented nature and the vast array of libraries, Java provides us with the flexibility to efficiently manage and manipulate data structures, simulate cloud environments, and orchestrate optimization strategies. CloudSim is a crucial tool in our development process, facilitating the emulation of diverse cloud scenarios and enabling realistic performance evaluations of our approach under varying conditions. By combining the power of Java and the capabilities of CloudSim, we can meticulously design, refine, and validate MOOA-CSF, ensuring its effectiveness in enhancing cloud service performance and meeting the demands of modern cloud computing environments.

VI.3 Experimental part of Tabu-OptiNimbus

VI.3.1 Experiment 1: Hypervolume Indicator (HV)

In this study, we assess Tabu-OptiNimbus alongside alternative methods using the hypervolume (HV) metric, a widely accepted measure for evaluating the performance of evolutionary multi-objective algorithms. HV gauges the volume enclosed by the solution set and a reference point, with higher values denoting superior outcomes. Figure (VI.1) illustrates that our approach yields the second-highest hypervolume indicator (HV), trailing the leading method by 6.72%. Notably, compared to other techniques, Tabu-OptiNimbus showcases improved performance, with gain rates of 0.97%, 2.05%, and 0.04% over GOA, FACO, and WoBingo, respectively. These HV values affirm the effectiveness of our Tabu-OptiNimbus approach in generating a compelling set of solutions.

VI.3.2 Experiment 2: MakeSpan

In this experiment, our aim is to thoroughly assess the MakeSpan of our approach and contrast it with several established optimization methods, including GOA, FACO, MOGA,

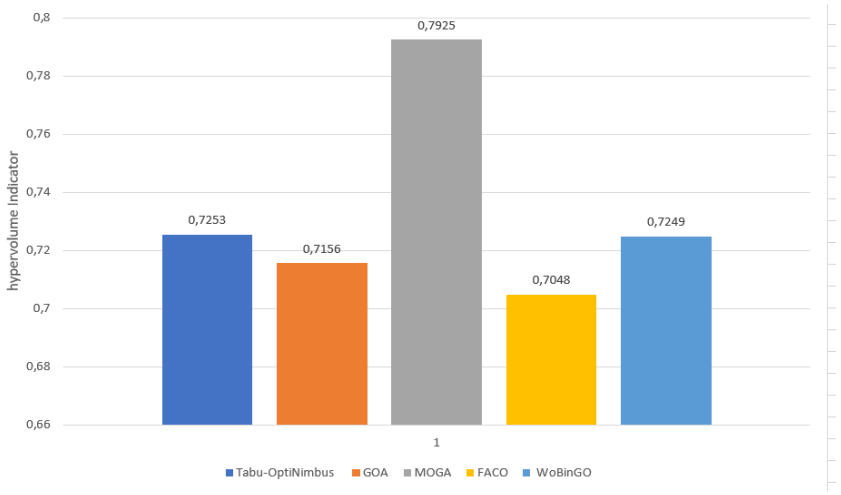
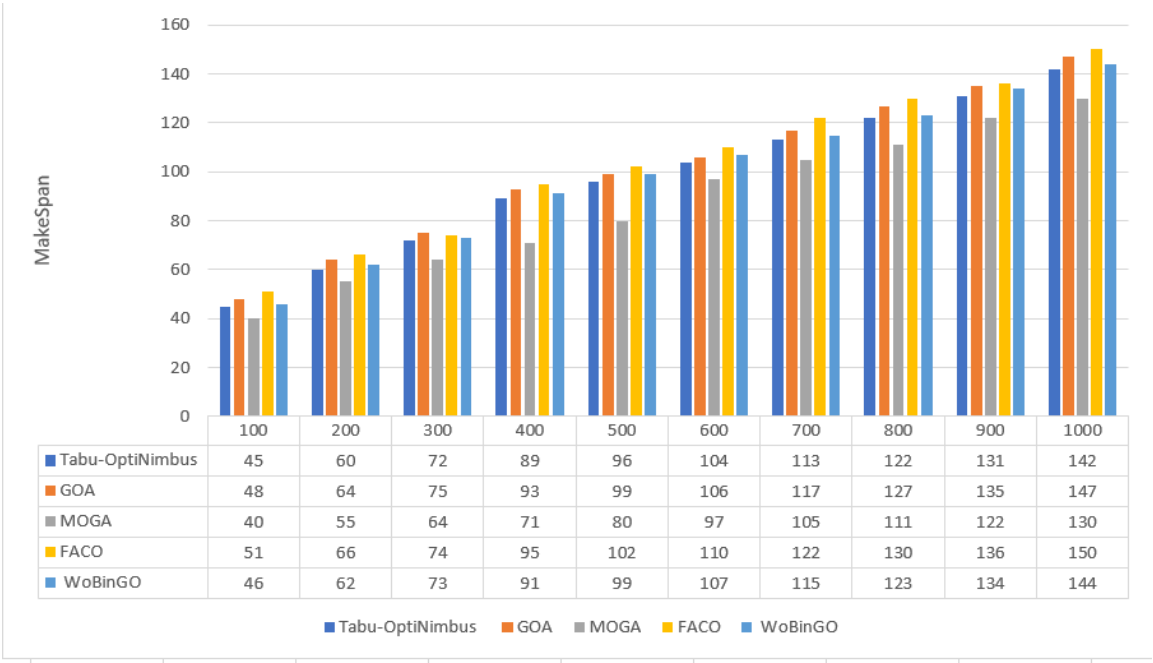


Figure VI.1: An analysis of the hypervolume metric.

and WoBingo. To ensure a comprehensive scalability analysis, we systematically varied the number of services from 100 to 1000, increasing in increments of 100 units. This deliberate approach allowed us to explore how each method performs across various service quantities. Upon scrutinizing the results, as depicted in Figure VI.2, we observed clear distinctions between our approach and the others. Despite witnessing some fluctuations in performance, our method consistently exhibited significantly better results than GOA, FACO, and WoBingo. However, it's worth noting that our approach fell short of the performance achieved by the MOGA approach. This nuanced comparison underscores the effectiveness and competitiveness of our proposed methodology in optimizing MakeSpan across various service load scenarios.



	100	200	300	400	500	600	700	800	900	1000
Tabu-OptiNimbus	45	60	72	89	96	104	113	122	131	142
GOA	48	64	75	93	99	106	117	127	135	147
MOGA	40	55	64	71	80	97	105	111	122	130
FACO	51	66	74	95	102	110	122	130	136	150
WoBinGO	46	62	73	91	99	107	115	123	134	144

Figure VI.2: The outcomes regarding the MakeSpan concerning the quantity of services

VI.3.3 Experiment 3:Response Time

In this investigation, we delve into the response time dynamics of Tabu-OptiNimbus under different service quantity scenarios, juxtaposing its performance against the GOA and MOGA strategies. We meticulously varied the number of services across a range from 100 to 1000 in a systematic manner. Figure VI.3 presents a visual representation of our findings, revealing that Tabu-OptiNimbus demonstrates a slight advantage over the MOGA approach regarding response time. However, it's noteworthy that the response time achieved by the GOA approach surpasses that of Tabu-OptiNimbus. Nevertheless, upon further examination, we observed that our approach exhibits enhanced response time performance when compared directly with MOGA. These observations shed light on the nuanced comparative performance of Tabu-OptiNimbus in different service quantity scenarios, highlighting its competitive edge in certain contexts over established optimization methods like MOGA.

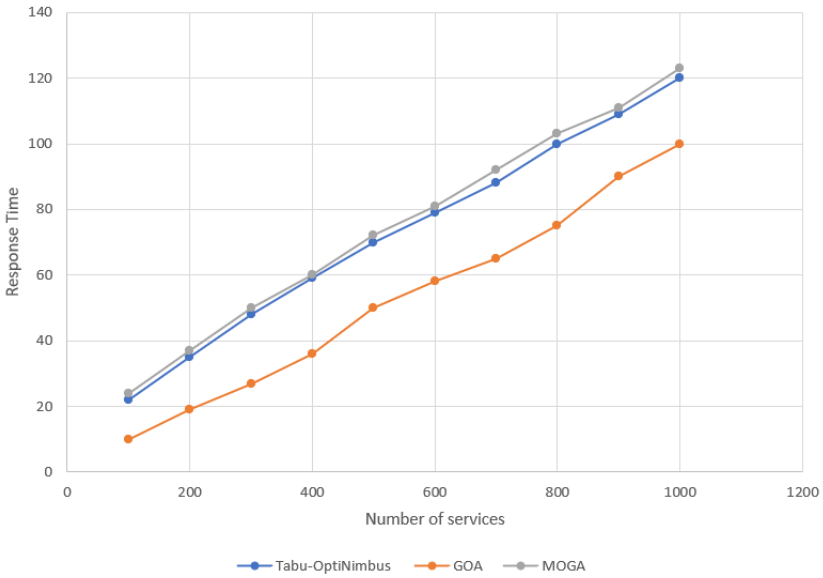


Figure VI.3: Impact of service quantity on response time.

VI.4 Experimental part of MOOA-CSF

VI.4.1 Experiment 1

This experimentation aims to show the average of services in the client-preferred class (CPC) before and after optimization. As shown in Figure VI.4, the response time, throughput, availability, reliability, and cost values before optimization are 1.12, 1.6, 83 %, 89 %, and 110 $. After optimization, availability increases from 83 % to 92 %, reliability from 89 % to 96 % and cost decreases from 110 $ to 93 $. Through this experimentation, we note that the values of three criteria have been optimized due to the number of crossed services.

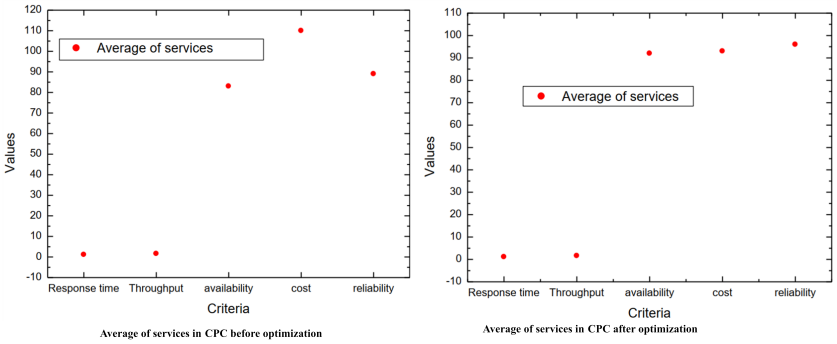


Figure VI.4: average of services in CPC before and after optimization

VI.4.2 Experiment 2

This experimentation aims to show the average of services in the SPC before and after optimization. As shown in Figure VI.5, the response time, throughput, availability, reliability, and cost values before optimization were respectively 2.1, 2.74, 74 %, 82 %, and 135 $. After optimization, the availability increases from 74 % to 89 %, reliability from 82 % to 93 % and the cost decreases from 135 $ to 100 $. In this experiment, we found that the number of crossing services led to optimizing the values of three criteria.

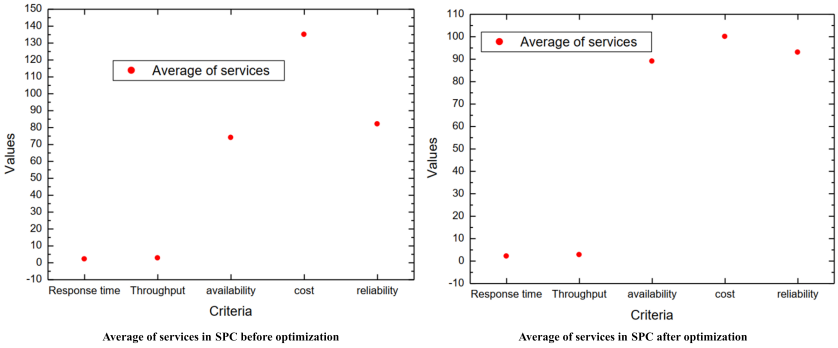


Figure VI.5: Average of services in SPC before and after optimization

VI.4.3 Experiment 3

This experimentation aims to show the average of services in the Common-Services-Class (CSC) before and after optimization. As shown in Figure VI.6, the response time values, throughput, availability, reliability, and cost before the optimization are 1.52, 2.35, 85 %, 81 %, and 120 $. After the optimization, the availability increases from 85 % to 96 %, the reliability increases from 81 % to 94 % and the cost decreases from 120 $ to 93 $. Due to the number of intersecting services, we observe through this experiment that the values of the three criteria have been optimized.

VI.4.4 Experiment 4

This experiment aims to evaluate our system according to the number of services. To reach this goal, we vary the number of services from less than 100 to up to 1000, then observe the values of the criteria. The experimental results are shown in Figure VI.7. When the

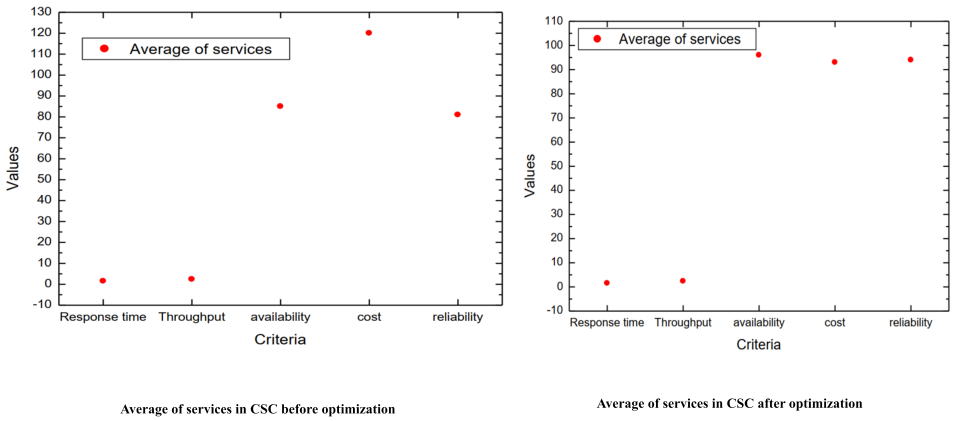


Figure VI.6: Average of services in CSC before and after optimization

number of services is more than 100, the availability has increased from 75.35 % to 82 %, the reliability has increased from 80.52 % to 85 %, the cost has decreased from 141.58 $ to 130 $, and the response time and throughput are unchanged. When the number of services is more than 500 and less than 1000, the availability has increased from 75.35 % to 90 %, the reliability has increased from 80.52 % to 92 %, the cost has decreased from 141.58 $ to 115.4 $, and the response time and throughput are unchanged. Finally, when the number of services is more than 1000, the availability has increased from 75.35 % to 99 %, the reliability has increased from 80.52 % to 98 %, the cost has decreased from 141.58 $ to 95 $, and response time and throughput are unchanged. From these results, we can conclude that the optimization rate increases accordingly with the number of services.

VI.4.5 Experiment 5

This experiment aims to assess the MakeSpan of our approach and compare it with well-known optimization methods like NN-DNSGA1, ABC, WOA, NSGA2, and DAABA. We systematically varied the number of services from 500 to 4500, increasing by 500 units, to thoroughly analyze the scalability of each method. Evaluation of the results, as illustrated in Figure VI.8, reveals apparent differences between our approach and the others. Notably, despite noticeable performance variations, our approach demonstrates significantly superior performance. This underscores the effectiveness and competitiveness of our proposed methodology in optimizing MakeSpan across different service loads.

VI.4.6 Experiment 6

In this experiment, we evaluate the response time of our approach according to the number of services. In this case, we compare our approach with the MOABC approach. We systematically varied the number of services from 100 to 5000. Figure VI.9 shows us that when the number of services is between 100 and 500, There is no significant difference between our approach and MOABC. After that, we observed that the response time of our approach was better than that of MOABC.

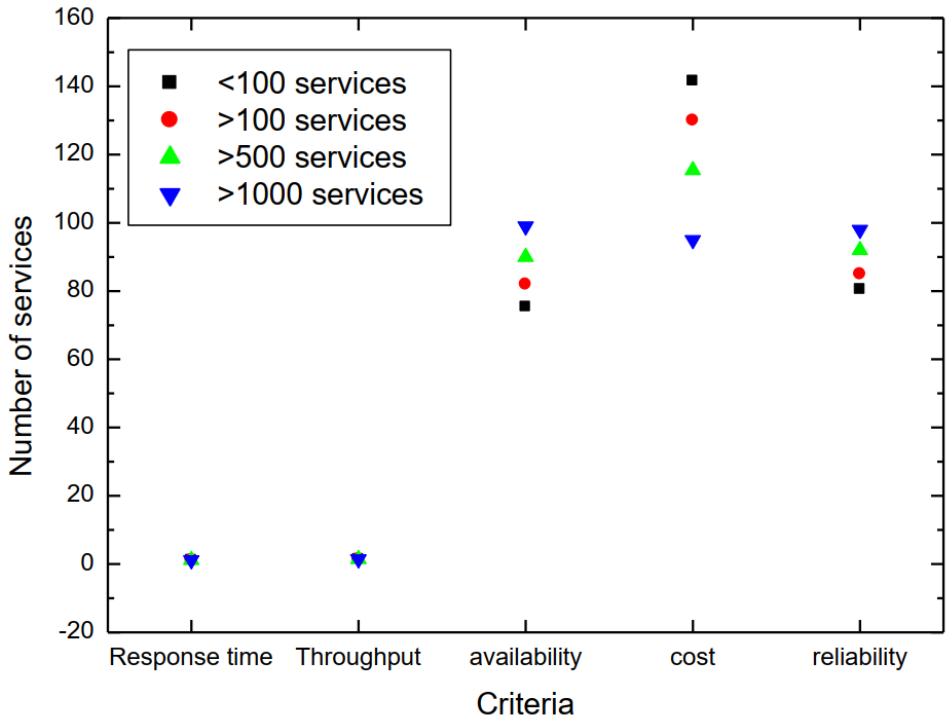
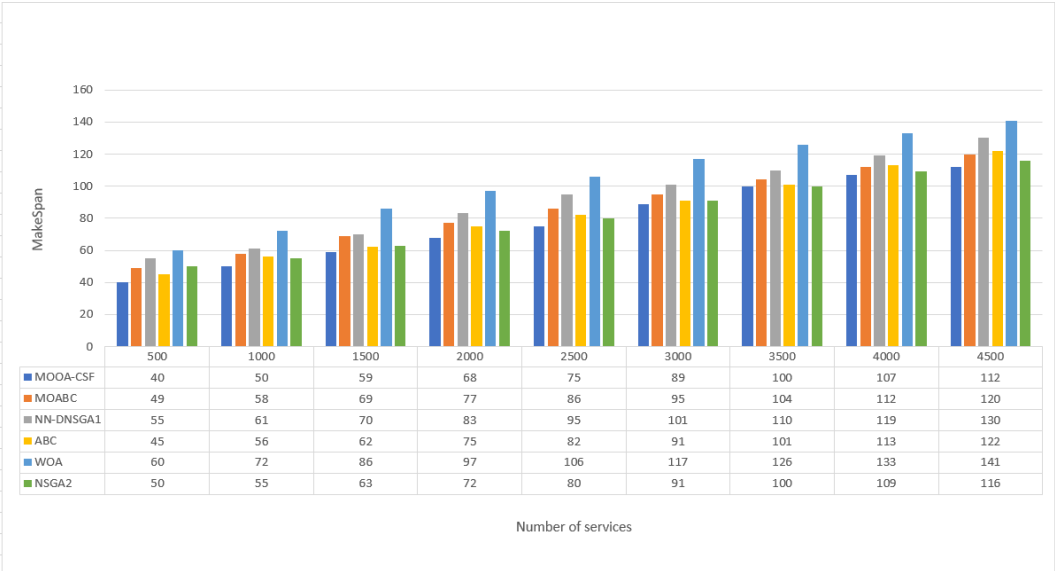


Figure VI.7: Impact of the services number on the optimization rate



	500	1000	1500	2000	2500	3000	3500	4000	4500
MOOA-CSF	40	50	59	68	75	89	100	107	112
MOABC	49	58	69	77	86	95	104	112	120
NN-DNSGA1	55	61	70	83	95	101	110	119	130
ABC	45	56	62	75	82	91	101	113	122
WOA	60	72	86	97	106	117	126	133	141
NSGA2	50	55	63	72	80	91	100	109	116

Figure VI.8: The results of the MakeSpan on the number of services

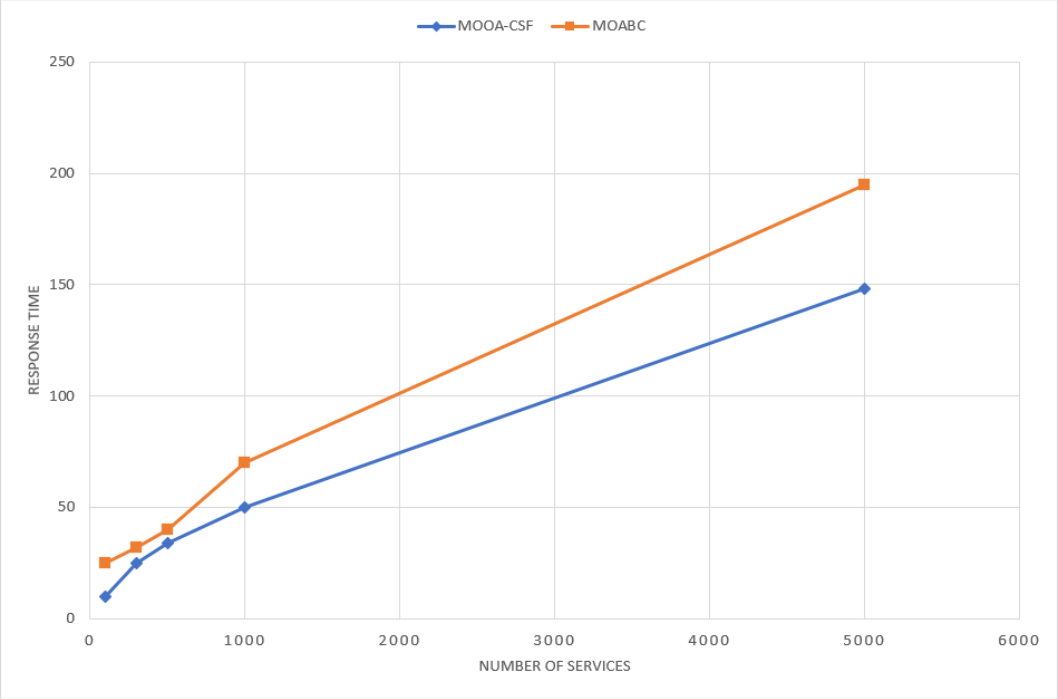


Figure VI.9: Effect of the number of service on Response time

VI.4.7 Experiment 7

In this experiment, MOOA-CSF is compared to other approaches based on the hypervolume indicator (HV), the most commonly used metric for comparing the performance of evolutionary multi-objective algorithms. HV is a unitary measure that calculates the volume of the area bounded by the set of solutions and a reference point, with a higher value indicating a better result. Figure (VI.10) shows that Our approach has the best hypervolume indicator (HV) with a slight superiority of about 6% compared to the next-best approach. Compared to other techniques, we observe that our approach outperforms the others, with a gain rate of 11%, 6%, 8, 76%, 23, 12% and 21, 1% compared to the ABC, WOA, DAABA, NN-DNSGA2 and NSGA2 respectively. The HV values confirm that our MOOA-CSF approach provides a much more effective set of solutions than the other approaches.

VI.5 Comparative Analysis: MOOA-CSF vs. Tabu-OptiNimbus Approaches

VI.5.1 The comparison between the HV indicators

The objective of this experiment is to contrast the HV indicators between two approaches: MOOA-CSF and Tabu-OptiNimbus. The data in Figure VI.11 indicate that the HV value associated with the MOOA-CSF approach surpasses that of the Tabu-OptiNimbus approach by 13.79%. This significant difference underscores the effectiveness and advancement achieved by the MOOA-CSF approach. It suggests that MOOA-CSF yields superior

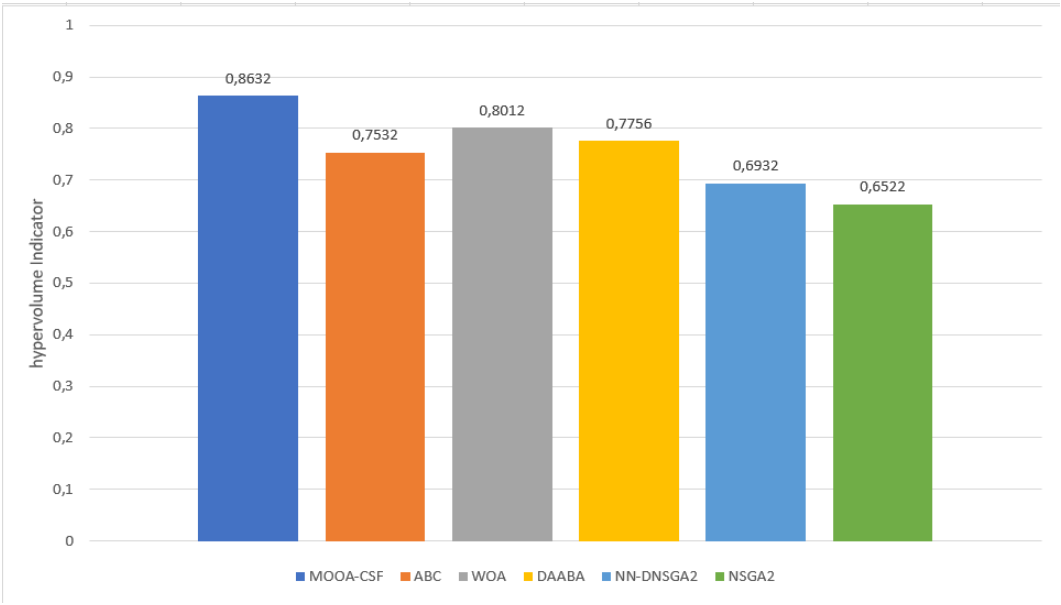


Figure VI.10: Comparison of the hypervolume indicator

outcomes in terms of HV, indicating its potency and progress in optimizing solutions compared to Tabu-OptiNimbus.

VI.5.2 Comparative Analysis of MakeSpan

In this experiment, we aim to compare the MakeSpan performance of two distinct approaches: MOOA-CSF and Tabu-OptiNimbus, as shown in figure (VI.12). The MakeSpan, representing the total time taken to complete a set of tasks, serves as a critical metric for evaluating the efficiency of optimization algorithms.

The comparison conducted in this experiment reveals that the MakeSpan associated with the MOOA-CSF approach outperforms that of the Tabu-OptiNimbus approach. This notable disparity underscores the effectiveness and advancement achieved by the MOOA-CSF approach. It suggests that MOOA-CSF yields superior outcomes in MakeSpan optimization, indicating its potency and progress in optimizing solutions compared to Tabu-OptiNimbus.

VI.5.3 Comparative Analysis of Response Time

In this experiment, our objective is to assess and contrast the response time performance between two distinct approaches: MOOA-CSF and Tabu-OptiNimbus. Response time, a pivotal metric for evaluating system efficiency, quantifies the duration from initiating a request to completing its processing, thus reflecting the system's responsiveness and effectiveness.

The comparison in the figure (VI.13) illustrates that the response time achieved by the MOOA-CSF approach outperforms that of the Tabu-OptiNimbus approach. This significant disparity underscores the effectiveness and progress achieved by the MOOA-CSF methodology. It indicates that MOOA-CSF delivers superior outcomes in optimizing response time, emphasizing its efficacy and advancement in optimizing solutions compared

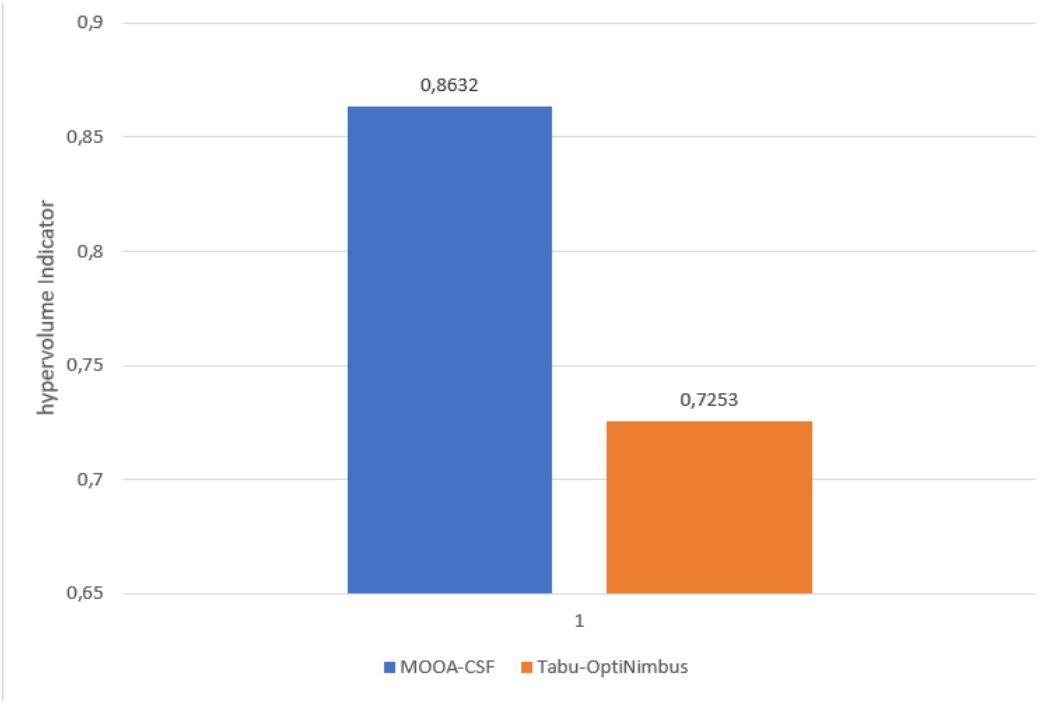


Figure VI.11: Comparison of the hypervolume of MOOA-CSF and Tabu-OptiNimbus

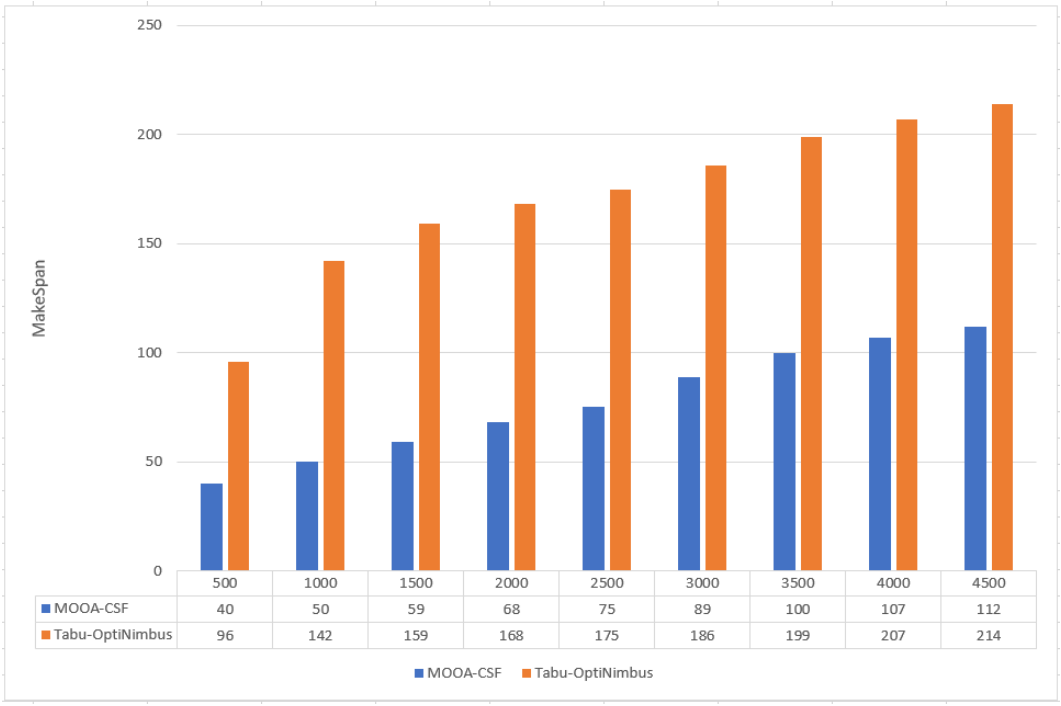


Figure VI.12: Comparative Analysis of MakeSpan

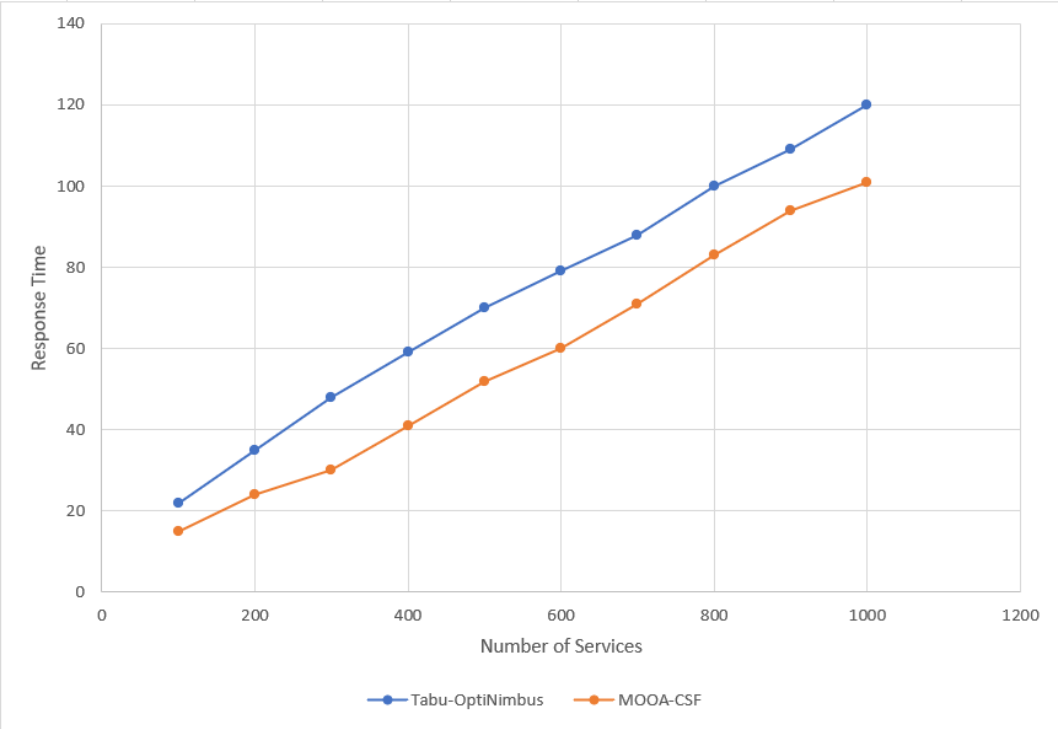


Figure VI.13: Comparative Analysis of Response time

to Tabu-OptiNimbus. This suggests that MOOA-CSF offers a more efficient and responsive approach for handling requests and processing tasks within cloud service environments.

VI.6 Analysis and discussion

This experimental section compares two Multicriteria Decision Models designed to tackle the complex challenges that cloud users encounter when optimizing costs and performance in their computing environments. We evaluated the MOOA-CSF using three key metrics: hypervolume, MakeSpan, and Response time. Our analysis demonstrates that MOOA-CSF exhibits superior effectiveness and responsiveness compared to Tabu-OptiNimbus.

In this comparison, we examined how well each model addressed the intricate demands of cloud users in terms of cost optimization and performance enhancement. By evaluating these models across multiple metrics, we gained insights into their respective strengths and weaknesses in meeting these challenges.

Now, let's delve into the comparison in more detail. The MOOA-CSF (Multi-Objective Optimization Algorithm with Cloud Service Framework) is specifically tailored to handle the multi-faceted nature of decision-making in cloud environments. It integrates various criteria, such as cost, performance, reliability, and scalability, to provide a comprehensive approach to cloud resource selection. This model employs advanced optimization techniques and decision-making methodologies to balance these criteria and generate high-quality solutions effectively.

On the other hand, Tabu-OptiNimbus utilizes the Tabu search algorithm to optimize cloud service performance. While both models aim to enhance cloud performance and

reduce costs, they may prioritize different aspects and employ different strategies to achieve their objectives.

The comparison conducted in this experimental section reveals that MOOA-CSF outperforms Tabu-OptiNimbus across the three metrics: hypervolume, makeSpan, and Response time. This suggests that MOOA-CSF is more effective and responsive in optimizing cloud resources than Tabu-OptiNimbus.

Overall, this comparison illuminates the strengths of each Multicriteria Decision Model and provides valuable insights for cloud users seeking to optimize their computing environments. By understanding how these models perform across various metrics, decision-makers can make more informed choices when selecting the most suitable approach for their needs.

VI.7 Conclusion

The experimental chapter presents a Multicriteria Decision Model tailored for cloud users to optimize costs and performance in their computing environments. The model offers a robust framework for informed decision-making in cloud resource selection by integrating cost, performance, reliability, and scalability criteria.

Throughout the chapter, empirical findings and case studies underscore the model's effectiveness and versatility across diverse cloud computing contexts. The model enables decision-makers to balance financial considerations and performance requirements by systematically prioritizing user preferences and constraints.

Moreover, practical insights into the model's implementation aspects—from data collection methodologies to analytical techniques and software tools—offer valuable guidance for users, enhancing its deployment effectiveness.

Given the increasing importance of robust decision support systems in cloud computing, the Multicriteria Decision Model represents a significant advancement. Future research can refine and extend the model to accommodate emerging trends and technologies, ensuring its relevance and effectiveness in addressing evolving user needs.

Ultimately, the Multicriteria Decision Model is a valuable tool for cloud users seeking to optimize resource utilization, minimize costs, and enhance performance in their computing environments. Through empirical validation and practical insights, this research aims to empower cloud users to navigate the complexities of the cloud landscape confidently and efficiently.

Conclusion and Perspectives

The main issue addressed in this thesis is how to optimize the cost and performance of cloud services? We thoroughly discussed our proposed solutions to address this problem in earlier chapters. In this chapter, we summarize our work in the first section and present potential future research directions in the second section.

Summary of the Work

Cloud computing transforms how data and programs are stored and accessed by shifting them to online platforms, removing the necessity of owning and controlling physical resources. It provides adaptable computing services that are accessible over the Internet and organized into categories such as SaaS, PaaS, and IaaS. Cloud service providers (CSPs) give users access to resources that serve various needs in sectors like business and education. Although cloud computing lowers expenses and allows for scalability, the plethora of providers presents challenges in decision-making. To tackle performance concerns like fluctuating workloads and security risks, continual innovation and adopting strategic approaches are vital for providers and organizations to meet evolving requirements.

This thesis delved into the challenge of "Optimizing performance and cost in cloud computing services," empowering consumers to select the most suitable cloud service based on defined criteria. It tackled a pertinent and intricate research problem, particularly relevant due to the rapid proliferation of cloud service providers. With each provider vying to offer superior services, consumers often lack the expertise to discern which services align best with their needs. Furthermore, accessing accurate information about multiple providers and their offerings presents another hurdle for consumers.

This situation may result in consumers overspending if they select an unsuitable service provider, posing a significant challenge for cloud users. To address this issue, the thesis developed two novel approaches to optimize cloud service performance. The first solution introduces a multi-criteria decision support system for cloud service selection. By integrating a neural network with ELECTRE and Tabu search algorithms, it enables stakeholders to balance cost and performance effectively, optimizing cloud service decisions. The second proposed solution is grounded in robust theoretical frameworks, drawing on Multi-Criteria Decision-Making (MCDM) principles and optimization algorithms.

The thesis is meticulously structured to ensure a logical flow of ideas. The first chapter begins with a detailed exploration of Cloud Computing and its various paradigms, establishing a solid foundation for subsequent discussions. The second chapter then delves into algorithms and optimization methods, providing insights into the technical complexities of enhancing efficiency within cloud environments. Building on this groundwork, the third chapter offers a comprehensive overview of Multi-Criteria Decision Making (MCDM), highlighting its relevance to cost and performance optimization in the cloud. In the fourth chapter, a thorough examination of related literature sets the context for understanding existing research and methodologies. Chapter five introduces innovative optimization approaches designed to address contemporary challenges in the field. Finally, the experimental chapter translates theory into practice, providing empirical validation and practical insights into the effectiveness of the proposed methodologies.

Perspectives

Additionally, we believe our work paves the way for further research opportunities. Numerous extensions can be suggested, all of which align with the overarching goal of enhancing the optimization performance of services within the cloud computing environment. The perspectives we present represent improvements in our approach.

- **Load Balancing:** load balancing plays a pivotal role in optimizing performance in cloud computing by facilitating efficient resource utilization, reducing response times, and improving system scalability and reliability. By effectively distributing workloads across various servers or resources, load balancing enhances optimising cloud-based services and applications.
- **Resource Utilization:** Optimizing resource utilization in the cloud service environment entails maximizing the utilization of CPU, memory, and storage resources while minimizing any wastage or inefficiencies.
- **Scalability:** Optimizing scalability ensures that a cloud service can efficiently adjust its resource capacity to meet fluctuating workload demands, thereby maintaining performance without disruption.

Bibliography

[1] Huda Ibrahim Mohamed. *Optimizing Energy Consumption of Cloud Computing IaaS*. PhD thesis, 2017.

[2] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.

[3] Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, and Samee Ullah Khan. The rise of "big data" on cloud computing: Review and open research issues. *Information systems*, 47:98–115, 2015.

[4] Rajkumar Buyya, Chee Shin Yeo, and Srikumar Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *2008 10th IEEE international conference on high performance computing and communications*, pages 5–13. Ieee, 2008.

[5] Peter Michael Garraghan. *Holistic cloud computing environmental quantification and behavioural analysis*. PhD thesis, University of Leeds, 2014.

[6] Amin Ziagham Ahwazi et al. Budget-aware scheduling algorithm for scientific workflow applications across multiple clouds. a mathematical optimization-based approach. Master's thesis, UiT Norges arktiske universitet, 2022.

[7] Haris Saric and Magnus Hornnes. Implementing cloud computing. a study of the impact on the it department. Master's thesis, University of Agder, 2020.

[8] Peter Mell and Timothy Grance. The nist definition of cloud computing: Recommendations of the national institute of standards and technology. special publication 800-145, 2011. *Prieiga per internetą:< http://csrc. nist. gov/publications/nistpubs/800-145/SP800-145. pdf*, 2021.

[9] Amandeep Verma and Sakshi Kaushal. Cloud computing security issues and challenges: a survey. In *Advances in Computing and Communications: First International Conference, ACC 2011, Kochi, India, July 22-24, 2011, Proceedings, Part IV 1*, pages 445–454. Springer, 2011.

[10] Dietmar Wiedemann and Jorg Strebel. Organizational determinants of corporate iaas usage. In *2011 IEEE 13th Conference on Commerce and Enterprise Computing*, pages 191–196. IEEE, 2011.

[11] Aleksandar Milić, Konstantin Simić, and Miloš Milutinović. Cloud computing environment for e-learning services for students with disabilities. *Continued Rise of the Cloud: Advances and Trends in Cloud Computing*, pages 363–381, 2014.

[12] N Pramod, Anil Kumar Muppalla, and KG Srinivasa. Limitations and challenges in cloud-based applications development. *Software Engineering Frameworks for the Cloud Computing Paradigm*, pages 55–75, 2013.

[13] RK Verma, S Dutta, SK Chaulya, AK Singh, and GM Prasad. Cloud computing: A new era in it industry. *International Journal of Computer Technology and Electronics Engineering*, 3(2):18–28, 2013.

[14] K Sukumaran. Prospectives of cloud computing applications in professional education. *Journal of Modern Education Review (ISSN 2155-7993)*, 1(2):89–98, 2011.

[15] Sushil Bhardwaj, Leena Jain, and Sandeep Jain. Cloud computing: A study of infrastructure as a service (iaas). *International Journal of engineering and information Technology*, 2(1):60–63, 2010.

[16] Sean Marston, Zhi Li, Subhajyoti Bandyopadhyay, Juheng Zhang, and Anand Ghalsasi. Cloud computing—the business perspective. *Decision support systems*, 51(1):176–189, 2011.

[17] Tharam Dillon, Chen Wu, and Elizabeth Chang. Cloud computing: issues and challenges. In *2010 24th IEEE international conference on advanced information networking and applications*, pages 27–33. Ieee, 2010.

[18] Xin Tan and Yongbeom Kim. Cloud computing for education: a case of using google docs in mba group projects. In *2011 International Conference on Business Computing and Global Informatization*, pages 641–644. IEEE, 2011.

[19] Prashant Gupta, Arumugam Seetharaman, and John Rudolph Raj. The usage and adoption of cloud computing by small and medium businesses. *International journal of information management*, 33(5):861–874, 2013.

[20] Darko Androcec. Data portability among providers o platform as a service. *Vedecké Práce Materiálovotechnologickej Fakulty Slovenskej Technickej Univerzity v Bratislave so Sídlom v Trnave*, 21(Special Issue):7, 2013.

[21] Christian Vecchiola, Xingchen Chu, Rajkumar Buyya, et al. Aneka: a software platform for .net-based cloud computing. *High speed and large scale scientific computing*, 18(3):267–295, 2009.

[22] Mohammad Manzurul Islam, Sarwar Morshed, and Parijat Goswami. Cloud computing: A survey on its limitations and potential solutions. *International Journal of Computer Science Issues (IJCSI)*, 10(4):159, 2013.

[23] AB Mutiara, Rina Refianti, and BA Witono. Developing a saas-cloud integrated development environment (ide) for c, c++, and java. *arXiv preprint arXiv:1411.5161*, 2014.

[24] Paul Ambrose and Ananth Chiravuri. An empirical investigation of cloud computing for personal use. 2010.

[25] Arto Ojala. Software-as-a-service revenue models. *IT Professional*, 15(3):54–59, 2012.

[26] Victor Chang, Gary Wills, and David De Roure. A review of cloud business models and sustainability. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 43–50. IEEE, 2010.

[27] Farzad Sabahi. Cloud computing ras issues and challenges. *The International Journal on Advances in ICT for Emerging Regions*, 4(2), 2011.

[28] Wolfgang Lehner and Kai-Uwe Sattler. Database as a service (dbaas). In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, pages 1216–1217. IEEE, 2010.

[29] Francesco Moscato, Rocco Aversa, Beniamino Di Martino, Teodor-Florin Fortiş, and Victor Munteanu. An analysis of mosaic ontology for cloud resources annotation. In *2011 federated conference on computer science and information systems (FedCSIS)*, pages 973–980. IEEE, 2011.

[30] Xiang Sheng, Jian Tang, Xuejie Xiao, and Guoliang Xue. Sensing as a service: Challenges, solutions and future directions. *IEEE Sensors journal*, 13(10):3733–3741, 2013.

[31] Abdulelah Almishal and Ahmed E Youssef. Cloud service providers: A comparative study. *International journal of computer applications & information technology*, 5(II), 2014.

[32] Fang Liu, Jin Tong, Jian Mao, Robert Bohn, John Messina, Lee Badger, and Dawn Leaf. Nist cloud computing reference architecture: Recommendations of the national institute of standards and technology (special publication 500-292), 2012.

[33] Americas Headquarters. Cisco data center infrastructure 2.5 design guide. *CISCO validated design I*, 2007.

[34] Hana Teyeb. *Integrated optimization in cloud environment*. PhD thesis, Université Paris-Saclay; Université Tunis El Manar. Faculté des Sciences ..., 2017.

[35] Md Hasanul Ferdaus, Manzur Murshed, Rodrigo N Calheiros, and Rajkumar Buyya. Network-aware virtual machine placement and migration in cloud data centers. In *Emerging research in cloud distributed computing systems*, pages 42–91. IGI Global, 2015.

[36] Albert Greenberg, James R Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A Maltz, Parveen Patel, and Sudipta Sengupta. Vl2: A scalable and flexible data center network. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 51–62, 2009.

[37] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang, and Songwu Lu. Bcube: a high performance, server-centric network architecture for modular data centers. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 63–74, 2009.

[38] Radhika Niranjan Mysore, Andreas Pamboris, Nathan Farrington, Nelson Huang, Pardis Miri, Sivasankar Radhakrishnan, Vikram Subramanya, and Amin Vahdat. Portland: a scalable fault-tolerant layer 2 data center network fabric. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 39–50, 2009.

[39] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *ACM SIGOPS operating systems review*, 37(5):164–177, 2003.

[40] Osama Harfoushi, Bader Alfawwaz, Nazeeh A Ghatasheh, Ruba Obiedat, Mua'ad M Abu-Faraj, and Hossam Faris. Data security issues and challenges in cloud computing: A conceptual analysis and review. *communications and Network*, 6(01):15–21, 2014.

[41] Radhika Garg and Burkhard Stiller. Factors affecting cloud adoption and their interrelations. In *Closer*, pages 87–94, 2015.

[42] Anisah Herdiyanti Prabowo, Marijn Janssen, and Joseph Barjis. A conceptual model for assessing the benefits of software as a service from different perspectives. In *Business Information Systems: 15th International Conference, BIS 2012, Vilnius, Lithuania, May 21-23, 2012. Proceedings 15*, pages 108–119. Springer, 2012.

[43] Selome Kostentinos Tesfatsion. *Energy-efficient cloud computing: Autonomic resource provisioning for datacenters.* PhD thesis, Umeå University, 2018.

[44] Stefan Frey, C Reich, and C Lüthje. Key performance indicators for cloud computing slas. In *The fifth international conference on emerging network intelligence, EMERGING*, pages 60–64, 2013.

[45] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)*, 35(3):268–308, 2003.

[46] Xin-She Yang. *Nature-inspired metaheuristic algorithms.* Luniver press, 2010.

[47] Entisar Alkayal. *Optimizing resource allocation using multi-objective particle swarm optimization in cloud computing systems.* PhD thesis, University of Southampton, 2018.

[48] Syed Hamid Hussain Madni, Shafi'i Muhammad Abd Latiff, Yahaya Coulibaly, and Shafi'i Muhammad Abdulhamid. An appraisal of meta-heuristic resource allocation techniques for iaas cloud. 2016.

[49] Michel Gendreau and Jean-Yves Potvin. Metaheuristics in combinatorial optimization. *Annals of Operations Research*, 140:189–213, 2005.

[50] Chun-Wei Tsai and Joel JPC Rodrigues. Metaheuristic scheduling for cloud: A survey. *IEEE Systems Journal*, 8(1):279–291, 2013.

[51] David S Johnson, Christos H Papadimitriou, and Mihalis Yannakakis. How easy is local search? *Journal of computer and system sciences*, 37(1):79–100, 1988.

[52] Nareyus I Lawrance Amaldass. *Metaheuristic approach for solving scheduling and financial derivative problems.* PhD thesis, Brunel University London, 2019.

[53] Yaneer Bar-Yam. *Dynamics of complex systems*. CRC Press, 2019.

[54] Thomas Weise. Global optimization algorithms-theory and application. *Self-Published Thomas Weise*, 361, 2009.

[55] CD Gelatt Jr. Optimization by simulated annealing. *Science*, 200:671–680, 1983.

[56] Vladimír Černỳ. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45:41–51, 1985.

[57] Daniel Rex Greening. *Simulated annealing with errors*. University of California, Los Angeles, 1995.

[58] Christian Blum and Xiaodong Li. Swarm intelligence in optimization. In *Swarm intelligence: introduction and applications*, pages 43–85. Springer, 2008.

[59] Ahmed Kattan and Shaheen Fatima. Pso as a meta-search for hyper-ga system to evolve optimal agendas for sequential multi-issue negotiation. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2012.

[60] Lizheng Guo, Guojin Shao, and Shuguang Zhao. Multi-objective task assignment in cloud computing by particle swarm optimization. In *2012 8th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–4. IEEE, 2012.

[61] Ioan Cristian Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information processing letters*, 85(6):317–325, 2003.

[62] Frans Van Den Bergh et al. *An analysis of particle swarm optimizers*. PhD thesis, University of Pretoria, 2007.

[63] Juan J Durillo, José García-Nieto, Antonio J Nebro, Carlos A Coello Coello, Francisco Luna, and Enrique Alba. Multi-objective particle swarm optimizers: An experimental comparison. In *Evolutionary Multi-Criterion Optimization: 5th International Conference, EMO 2009, Nantes, France, April 7-10, 2009. Proceedings 5*, pages 495–509. Springer, 2009.

[64] SR Shishira, A Kandasamy, and K Chandrasekaran. Survey on meta heuristic optimization techniques in cloud computing. In *2016 international conference on advances in computing, communications and informatics (ICACCI)*, pages 1434–1440. IEEE, 2016.

[65] Marco Dorigo. Ant colony optimization. *Scholarpedia*, 2(3):1461, 2007.

[66] Medhat A Tawfeek, Ashraf El-Sisi, Arabi E Keshk, and Fawzy A Torkey. Cloud task scheduling based on ant colony optimization. In *2013 8th international conference on computer engineering & systems (ICCES)*, pages 64–69. IEEE, 2013.

[67] Pisut Pongchairerks. Particle swarm optimization algorithm applied to scheduling problems, 2009.

[68] Dervis Karaboga and Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of global optimization*, 39:459–471, 2007.

[69] Fei Kang, Junjie Li, Haojin Li, Zhenyue Ma, and Qing Xu. An improved artificial bee colony algorithm. In *2010 2nd International Workshop on Intelligent Systems and Applications*, pages 1–4. IEEE, 2010.

[70] Dervis Karaboga et al. An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes university, engineering faculty, computer ..., 2005.

[71] Dervis Karaboga and Bahriye Basturk. Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems. In *International fuzzy systems association world congress*, pages 789–798. Springer, 2007.

[72] Charles Darwin. On the origins of species by means of natural selection. *London: Murray*, 247:1859, 1859.

[73] JH Holland. Genetic algorithms, scientific american, 1992. *URL http://www. fortunecity. com/emachines/e11/86/algo. html.*

[74] El-Ghazali Talbi. *Metaheuristics: from design to implementation.* John Wiley & Sons, 2009.

[75] Ahmed Jaber. *Hybrid Algorithm for Multi-objective Mixed-integer Non-convex Mechanical Design Optimization Problems.* PhD thesis, Université de Technologie de Troyes; Université Libanaise, 2021.

[76] Voratas Kachitvichyanukul. Comparison of three evolutionary algorithms: Ga, pso, and de. *Industrial Engineering and Management Systems*, 11(3):215–223, 2012.

[77] Gregory JE Rawlins. *Foundations of Genetic Algorithms 1991 (FOGA 1).* Elsevier, 2014.

[78] Rahat Sultana. Application of genetic algorithm in multi-objective optimization of an indeterminate structure with discontinuous space for support locations. 2016.

[79] Imran Ali Chaudhry and Abdul Munem Khan. Minimizing makespan for a no-wait flowshop using genetic algorithm. *Sadhana*, 37:695–707, 2012.

[80] G Subashini and MC Bhuvaneswari. Comparison of multi-objective evolutionary approaches for task scheduling in distributed computing systems. *Sadhana*, 37:675–694, 2012.

[81] Xiaodong Sheng and Qiang Li. Template-based genetic algorithm for qos-aware task scheduling in cloud computing. In *2016 International Conference on Advanced Cloud and Big Data (CBD)*, pages 25–30. IEEE, 2016.

[82] Richard A Caruana, Larry J Eshelman, and J David Schaffer. Representation and hidden bias ii: Eliminating defining length bias in genetic search via shuffle crossover. In *Proceedings of the 11th international joint conference on Artificial intelligence-Volume 1*, pages 750–755, 1989.

[83] Lashon Booker. Improving search in genetic algorithms. *Genetic algorithms and simulated annealing*, 1987.

[84] Fred Glover. Tabu search—part i. *ORSA Journal on computing*, 1(3):190–206, 1989.

[85] Fred Glover. Tabu search—part ii. *ORSA Journal on computing*, 2(1):4–32, 1990.

[86] Lemasri Piniganti. A survey of tabu search in combinatorial optimization. 2014.

[87] Evangelos Triantaphyllou and Evangelos Triantaphyllou. *Multi-criteria decision making methods*. Springer, 2000.

[88] Juan Li, Qingrui Li, Chao Liu, Samee Ullah Khan, and Nasir Ghani. Community-based collaborative information system for emergency management. *Computers & operations research*, 42:116–124, 2014.

[89] Abbas Mardani, Ahmad Jusoh, Khalil Nor, Zainab Khalifah, Norhayati Zakwan, and Alireza Valipour. Multiple criteria decision-making techniques and their applications–a review of the literature from 2000 to 2014. *Economic research-Ekonomska istraživanja*, 28(1):516–571, 2015.

[90] Ling Xu and Jian-Bo Yang. *Introduction to multi-criteria decision making and the evidential reasoning approach*, volume 106. Manchester School of Management Manchester, 2001.

[91] Gianluca Campanella and Rita A Ribeiro. A framework for dynamic multiple-criteria decision making. *Decision Support Systems*, 52(1):52–60, 2011.

[92] Gianluca Campanella and Rita A Ribeiro. Multiple attribute decision making in a dynamic environment. In *World Conference on Soft Computing*, pages 32–26, 2011.

[93] Aditya Chauhan and Rahul Vaish. Magnetic material selection using multiple attribute decision making approach. *Materials & Design (1980-2015)*, 36:1–5, 2012.

[94] Aarushi Singh and Sanjay Kumar Malik. Major mcdm techniques and their application-a review. *IOSR Journal of Engineering*, 4(5):15–25, 2014.

[95] Edmundas Kazimieras Zavadskas, Zenonas Turskis, and Simona Kildienė. State of art surveys of overviews on mcdm/madm methods. *Technological and economic development of economy*, 20(1):165–179, 2014.

[96] David Nijssen. *Improving spatiality in decision making for river basin management*. PhD thesis, Lehrstuhl für Hydrologie, Wasserwirtschaft und Umwelttechnik, Ruhr-Univ., 2013.

[97] Philippe Vincke. *Multicriteria decision-aid*. John Wiley & Sons, 1992.

[98] Michael Doumpos and Evangelos Grigoroudis. *Multicriteria decision aid and artificial intelligence: links, theory and applications*. John Wiley & Sons, 2013.

[99] Sanjay D Pohekar and Muthu Ramachandran. Application of multi-criteria decision making to sustainable energy planning—a review. *Renewable and sustainable energy reviews*, 8(4):365–381, 2004.

[100] Christina Diakaki, Evangelos Grigoroudis, Nikos Kabelis, Dionyssia Kolokotsa, Kostas Kalaitzakis, and George Stavrakakis. A multi-objective decision model for the improvement of energy efficiency in buildings. *Energy*, 35(12):5483–5496, 2010.

[101] Jiang-Jiang Wang, You-Yin Jing, Chun-Fa Zhang, and Jun-Hong Zhao. Review on multi-criteria decision analysis aid in sustainable energy decision-making. *Renewable and sustainable energy reviews*, 13(9):2263–2278, 2009.

[102] Arayeh Afsordegan. A contribution to multi-criteria decision making in sustainable energy management based on fuzzy and qualitative reasoning. 2015.

[103] Adel Guitouni and Jean-Marc Martel. Tentative guidelines to help choosing an appropriate mcda method. *European journal of operational research*, 109(2):501–521, 1998.

[104] Andrea De Montis, Pasquale De Toro, Bert Droste-Franke, Ines Omann, Sigrid Stagl, et al. Assessing the quality of different mcda methods. *Alternatives for environmental valuation*, 4:99, 2004.

[105] Yongwen Liu. *Cloud services selection based on rough set theory*. PhD thesis, Troyes, 2016.

[106] K Paul Yoon and Ching-Lai Hwang. *Multiple attribute decision making: an introduction*. Sage publications, 1995.

[107] Edmundas Kazimieras Zavadskas, Abbas Mardani, Zenonas Turskis, Ahmad Jusoh, and Khalil MD Nor. Development of topsis method to solve complicated decision-making problems—an overview on developments from 2000 to 2015. *International Journal of Information Technology & Decision Making*, 15(03):645–682, 2016.

[108] Shu-Jen Chen and Ching-Lai Hwang. Fuzzy multiple attribute decision making methods. In *Fuzzy multiple attribute decision making: Methods and applications*, pages 289–486. Springer, 1992.

[109] E Roghanian, J Rahimi, and A Ansari. Comparison of first aggregation and last aggregation in fuzzy group topsis. *Applied Mathematical Modelling*, 34(12):3754–3766, 2010.

[110] Zhongliang Yue. A method for group decision-making based on determining weights of decision makers using topsis. *Applied Mathematical Modelling*, 35(4):1926–1936, 2011.

[111] Hsu-Shih Shih, Huan-Jyh Shyur, and E Stanley Lee. An extension of topsis for group decision making. *Mathematical and computer modelling*, 45(7-8):801–813, 2007.

[112] Majid Behzadian, S Khanmohammadi Otaghsara, Morteza Yazdani, and Joshua Ignatius. A state-of the-art survey of topsis applications. *Expert Systems with applications*, 39(17):13051–13069, 2012.

[113] Cengiz Kahraman, Sezi Cevik Onar, and Basar Oztaysi. Fuzzy multicriteria decision-making: a literature review. *International journal of computational intelligence systems*, 8(4):637–666, 2015.

[114] Ching-Lai Hwang and Ming-Jeng Lin. *Group decision making under multiple criteria: methods and applications*, volume 281. Springer Science & Business Media, 2012.

[115] Daekook Kang, Wooseok Jang, and Yongtae Park. Evaluation of e-commerce websites using fuzzy hierarchical topsis based on es-qual. *Applied Soft Computing*, 42:53–65, 2016.

[116] Meysam Shaverdi, Iman Ramezani, Reza Tahmasebi, and Ali Asghar Anvary Rostamy. Combining fuzzy ahp and fuzzy topsis with financial ratios to design a novel performance evaluation model. *International Journal of Fuzzy Systems*, 18:248–262, 2016.

[117] Amir Viyanchi, Ali Rajabzadeh Ghatari, Hamid Reza Rasekh, and HamidReza SafiKhani. Administrative process and criteria ranking for drug entering health insurance list in iran-topsis-based consensus model. *Iranian journal of pharmaceutical research: IJPR*, 15(1):369, 2016.

[118] M Aghajani Mir, P Taherei Ghazvinei, NMN Sulaiman, NEA Basri, S Saheri, NZ Mahmood, A Jahan, RA Begum, and NJJOEM Aghamohammadi. Application of topsis and vikor improved versions in a multi criteria decision analysis to develop an optimized municipal solid waste management model. *Journal of environmental management*, 166:109–115, 2016.

[119] Sen Liu, Felix TS Chan, and Wenxue Ran. Decision making for the selection of cloud vendor: An improved approach under group decision-making with integrated weights and objective/subjective attributes. *Expert Systems with Applications*, 55:37–47, 2016.

[120] Bernard Roy. *Multicriteria methodology for decision aiding*, volume 12. Springer Science & Business Media, 1996.

[121] Luis Miguel Del Vasto Terrientes et al. *Hierarchical outranking methods for multicriteria decision aiding*. PhD thesis, Universitat Rovira i Virgili, 2015.

[122] Jose Rui Figueira, Salvatore Greco, Bernard Roy, and Roman Słowiński. Electre methods: Main features and recent developments. *Handbook of multicriteria analysis*, pages 51–89, 2010.

[123] Jean-Pierre Brans and Ph Vincke. Note—a preference ranking organisation method: (the promethee method for multiple criteria decision-making). *Management science*, 31(6):647–656, 1985.

[124] Majid Behzadian, Reza Baradaran Kazemzadeh, Amir Albadvi, and Mohammad Aghdasi. Promethee: A comprehensive literature review on methodologies and applications. *European journal of Operational research*, 200(1):198–215, 2010.

[125] Laila Oubahman and Szabolcs Duleba. Review of promethee method in transportation. *Production Engineering Archives*, 27(1):69–74, 2021.

[126] Hua Ma, Haibin Zhu, Zhigang Hu, Keqin Li, and Wensheng Tang. Time-aware trustworthiness ranking prediction for cloud services using interval neutrosophic set and electre. *Knowledge-Based Systems*, 138:27–45, 2017.

[127] M Mousavi, H Gitinavard, and SM Mousavi. A soft computing based-modified electre model for renewable energy policy selection with unknown information. *Renewable and Sustainable Energy Reviews*, 68:774–787, 2017.

[128] Burak Omer Saracoglu. An experimental research study on the solution of a private small hydropower plant investments selection problem by electre iii/iv, shannon's entropy, and saaty's subjective criteria weighting. *Advances in Decision Sciences*, 2015, 2015.

[129] Juscelino Almeida-Dias, José Rui Figueira, and Bernard Roy. Electre tri-c: A multiple criteria sorting method based on characteristic reference actions. *European Journal of Operational Research*, 204(3):565–580, 2010.

[130] Salvatore Greco, Jose Figueira, and Matthias Ehrgott. *Multiple criteria decision analysis*, volume 37. Springer, 2016.

[131] Martin Rogers and Michael Bruen. Choosing realistic values of indifference, preference and veto thresholds for use with environmental criteria within electre. *European Journal of Operational Research*, 107(3):542–551, 1998.

[132] Brunno e Souza Rodrigues, Carla Martins Floriano, Valdecy Pereira, and Marcos Costa Roboredo. An algorithm to elicitate electre ii, iii and iv parameters. *Data Technologies and Applications*, 55(1):82–96, 2021.

[133] Lizheng Guo, Tao Yan, Shuguang Zhao, Changyuan Jiang, et al. Dynamic performance optimization for cloud computing using m/m/m queueing system. *Journal of applied mathematics*, 2014, 2014.

[134] Goshgar Ismayilov and Haluk Rahmi Topcuoglu. Neural network based multi-objective evolutionary algorithm for dynamic workflow scheduling in cloud computing. *Future Generation computer systems*, 102:307–322, 2020.

[135] Visnja Simic, Boban Stojanovic, and Milos Ivanovic. Optimizing the performance of optimization in the cloud environment–an intelligent auto-scaling approach. *Future Generation Computer Systems*, 101:909–920, 2019.

[136] Javad Vahidi and Maral Rahmati. Optimization of resource allocation in cloud computing by grasshopper optimization algorithm. In *2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI)*, pages 839–844. IEEE, 2019.

[137] Rashed Salem, Mustafa Abdul Salam, Hatem Abdelkader, and Ahmed Awad Mohamed. An artificial bee colony algorithm for data replication optimization in cloud environments. *IEEE Access*, 8:51841–51852, 2019.

[138] Celestine Iwendi, Praveen Kumar Reddy Maddikunta, Thippa Reddy Gadekallu, Kuruva Lakshmanna, Ali Kashif Bashir, and Md Jalil Piran. A metaheuristic optimization approach for energy efficiency in the iot networks. *Software: Practice and Experience*, 51(12):2558–2571, 2021.

[139] Omer Irshad, Muhammad Usman Ghani Khan, Razi Iqbal, Shakila Basheer, and Ali Kashif Bashir. Performance optimization of iot based biological systems using deep learning. *Computer Communications*, 155:24–31, 2020.

[140] G Sreenivasulu and Ilango Paramasivam. Hybrid optimization algorithm for task scheduling and virtual machine allocation in cloud computing. *Evolutionary Intelligence*, 14:1015–1022, 2021.

[141] Milos Ivanovic, Visnja Simic, Boban Stojanovic, Ana Kaplarevic-Malisic, and Branko Marovic. Elastic grid resource provisioning with wobingo: A parallel framework for genetic algorithm based optimization. *Future Generation Computer Systems*, 42:44–54, 2015.

[142] Mitali Bansal and Sanjay Kumar Malik. A multi-faceted optimization scheduling framework based on the particle swarm optimization algorithm in cloud computing. *Sustainable Computing: Informatics and Systems*, 28:100429, 2020.

[143] Kai Zhou, Yongzhao Wen, Wanying Wu, Zhiyong Ni, Tianguo Jin, and Xiaojun Long. Cloud service optimization method based on dynamic artificial ant-bee colony algorithm in agricultural equipment manufacturing. *Mathematical Problems in Engineering*, 2020:1–11, 2020.

[144] Awatif Ragmani, Amina Elomri, Noreddine Abghour, Khalid Moussaid, and Mohammed Rida. Faco: A hybrid fuzzy ant colony optimization algorithm for virtual machine scheduling in high-performance cloud computing. *Journal of Ambient Intelligence and Humanized Computing*, 11:3975–3987, 2020.

[145] C Vinothini, P Balasubramanie, and J Priya. Hybrid of meta heuristic firefly and genetic algorithm for optimization approach in the cloud environment. *Webology*, 17(1):297–305, 2020.

[146] Umesh Kumar Lilhore, Sarita Simaiya, Shikha Maheshwari, Advin Manhar, and Santosh Kumar. Cloud performance evaluation: hybrid load balancing model based on modified particle swarm optimization and improved metaheuristic firefly algorithms. *Int J Adv Sci Technol*, 29(5):12315–12331, 2020.

[147] Neeraj Singh Kaurav and Pradeep Yadav. A genetic algorithm-based load balancing approach for resource optimization for cloud computing environment. *Int J Inf Comput Sci*, 6(3):175–184, 2019.

[148] Laith Abualigah and Muhammad Alkhrabsheh. Amended hybrid multi-verse optimizer with genetic algorithm for solving task scheduling problem in cloud computing. *The Journal of Supercomputing*, 78(1):740–765, 2022.

[149] Mana Saleh Al Reshan, Darakhshan Syed, Noman Islam, Asadullah Shaikh, Mohammed Hamdi, Mohamed A Elmagzoub, Ghulam Muhammad, and Kashif Hussain Talpur. A fast converging and globally optimized approach for load balancing in cloud computing. *IEEE Access*, 11:11390–11404, 2023.

[150] Wenzhi Wang and Zhanqiao Liu. Cloud service composition using firefly optimization algorithm and fuzzy logic. *International Journal of Advanced Computer Science and Applications*, 14(3), 2023.

[151] Sina Ahmadi. Optimizing data warehousing performance through machine learning algorithms in the cloud. *International Journal of Science and Research (IJSR)*, 12(12):1859–1867, 2023.

[152] Sundas Iftikhar, Mirza Mohammad Mufleh Ahmad, Shreshth Tuli, Deepraj Chowdhury, Minxian Xu, Sukhpal Singh Gill, and Steve Uhlig. Hunterplus: Ai based energy-efficient task scheduling for cloud–fog computing environments. *Internet of Things*, 21:100667, 2023.

[153] Haochen Hua, Yutong Li, Tonghe Wang, Nanqing Dong, Wei Li, and Junwei Cao. Edge computing with artificial intelligence: A machine learning perspective. *ACM Computing Surveys*, 55(9):1–35, 2023.

[154] G Saravanan, S Neelakandan, P Ezhumalai, and Sudhanshu Maurya. Improved wild horse optimization with levy flight algorithm for effective task scheduling in cloud computing. *Journal of Cloud Computing*, 12(1):24, 2023.

[155] Matheus Araujo Gava, Helder Roberto Oliveira Rocha, Menno Jan Faber, Marcelo Eduardo Vieira Segatto, Heinrich Wörtche, and Jair Adriano Lima Silva. Optimizing resources and increasing the coverage of internet-of-things (iot) networks: An approach based on lorawan. *Sensors*, 23(3):1239, 2023.

[156] Arjun Reddy Kunduru. Artificial intelligence usage in cloud application performance improvement. *Central Asian Journal of Mathematical Theory and Computer Sciences*, 4(8):42–47, 2023.

[157] Ammar Riadh Kairaldeen, Nor Fadzilah Abdullah, Asma Abu-Samah, and Rosdiadee Nordin. Peer-to-peer user identity verification time optimization in iot blockchain network. *Sensors*, 23(4):2106, 2023.

[158] GIUSEPPE Aiello, M Enea, and GM Galante. A multi-objective approach to facility layout problem by genetic search algorithm and electre method. *Robotics and Computer-Integrated Manufacturing*, 22(5-6):447–455, 2006.

[159] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *Towards Data Sci*, 6(12):310–316, 2017.

[160] Li Lin, Longbing Cao, Jiaqi Wang, and Chengqi Zhang. The applications of genetic algorithms in stock market data mining optimisation. *Management Information Systems*, 2004.

[161] P Biswas, S Pramanik, and BC Giri. Cosine similarity measure based multi-attribute decisionmaking with trapezoidal fuzzy neutrosophic numbers. neutrosophic sets and system, 8 (2015). *Kalyan Mondal and Surapati Pramanik, Decision Making Based on Some similarity Measures under Interval Rough Neutrosophic Environment*, pages 102–110, 2015.

[162] Eyhab Al-Masri and Qusay H Mahmoud. Investigating web services on the world wide web. In *Proceedings of the 17th international conference on World Wide Web*, pages 795–804, 2008.