République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université A.MIRA-BEJAIA

Faculté des Sciences Exactes
Département d'Informatique
Laboratoire d'Informatique Médicale et des Environnements Dynamiques et intelligents (LIMED)

THÈSE

EN VUE DE L'OBTENTION DU DIPLOME DE DOCTORAT

Domaine : Mathématiques et Informatique Filière : Informatique
Spécialité : Réseaux et systèmes distribués

Présentée par
HAMECHE Salma

Thème

Addressing the Challenges of Service Composition in the Internet of Things and Cyber-Physical-Social Systems: Scalable Algorithms for QoS, energy, and user Mobility

Soutenue le : 09/04/2026 **Devant le Jury composé de :**

Nom et Prénom	Grade		
Mme EL BOUHISSI Houda	MCA	Univ. de Béjaia	Président
Mr TARI Abdelkamel	Professeur	Univ. de Bejaia	Rapporteur
Mr KHANOUCHE Mohamed Essaid	MC	Univ. Claude Bernard Lyon1	Co-rapporteur
Mr FARAH Zoubeyr	Professeur	Univ. de Béjaia	Examinateur
Mr ELMIR Youssef	Professeur	ESTIN, Béjaia	Examinateur

Année Universitaire : 2025/2026

Dedication

Every achievement stems from the love and patience of those around us; behind each page of this thesis are the hearts that have supported me tirelessly.

To my beloved parents,

No words can truly express the depth of my love and gratitude for you. Throughout these long and demanding years, your prayers, your patience, and your quiet strength have been my greatest source of courage. Even in the moments when my work kept me away and I could not be as present as I wished, your hearts never wavered. I ask your forgiveness for the times I was absent and for the moments I could not be there as I should have been.

To my brothers and sisters,

Thank you for the affection, the laughter, and the strength you have brought into my life. Each of you has, in your own way, helped me persevere through this journey. A special thought goes to my sister Sarah, whose generous support and reassuring presence have meant more than words can express.

To my mother-in-law, sisters-in-law, and brothers-in-law,

Thank you for your kindness, warmth, and the family spirit you have always extended to me.

To my nieces and nephews,

Your innocence, smiles, and joyful energy have been a gentle reminder of what truly matters.

To my husband,

You have been my anchor, my guiding light, and my greatest source of strength throughout this journey. From the first step to the final page, you have remained steadfast by my side, encouraging me when doubt crept in, and celebrating each small victory as if it were your own. Your love, patience, and devotion have been extraordinary. This thesis is a testament not only to my efforts but also to your unwavering support, sacrifices, and belief in me.

To my son, Abderahmane,

Your arrival into this world has brought immense joy and light into our lives, inspiring me to complete this thesis. May Allah guide you always on the path of knowledge, wisdom, and virtue, and bless you in every step of your life.

Acknowledgments

Praise be to **Allah Almighty**, the Lord of the worlds, whose mercy and blessing have enlightened my path throughout the completion of this thesis. He granted me the patience to endure, the strength to persevere, and the wisdom to overcome the challenges encountered along the way. Without Allah's guidance and benediction, this achievement would not have been possible.

I would like to express my sincere gratitude to my thesis supervisor, KHANOUCHE Mohamed Essaid, for his supervision, continuous guidance, and the insightful comments he provided throughout this work. His scientific rigor, encouragement, and high expectations have motivated me to persevere and continually improve the quality of my research.

I extend my sincere gratitude to my thesis co-supervisor, TARI Abdelkamel, for his encouragement, valuable advice, and the knowledge he imparted to me. I am especially grateful for the inspiration and support that he provided throughout the course of my studies, which greatly contributed to the development of my research.

My sincere thanks go to the members of the examination jury for accepting to evaluate my thesis. Their expertise, valuable comments, and suggestions have greatly contributed to improving the quality of this research.

I would like to thank all my teachers who have guided and supported me throughout my academic journey. Their dedication, patience and passion for teaching have deeply inspired me and contributed greatly to my learning and personal development.

I am also profoundly grateful to my friends and colleagues, whose kindness, camaraderie, and insightful discussions have enriched this path, making it both inspiring and enjoyable.

To all those who, in one way or another, have contributed to the completion of this thesis, I wish to extend my heartfelt gratitude.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
ARO	Artificial Rabbit Optimization
BA	Bat Algorithm
BPEL	Business Process Execution Language
CGA	Cultural Genetic Algorithm
CI	Critical Infrastructure
COAP	Constrained Application Protocol
CORBA	Common Object Request Broker Architecture
CPSS	Cyber-Physical-Social Systems
CPS	Cyber-Physical Systems
CPST	Cyber-Physical-Social-Thinking hyperspace
CSC	Cloud Services Composition
CSS	Cyber-Social Systems
CSSA	Chaotic Salp Swarm Algorithm
DABC	Discrete Artificial Bee Colony
DAG	Directed Acyclic Graph
DASC	Dependency-Aware Service Composition
DO	Dandelion Optimizer
DPWS	Devices Profile for Web Services
EBA	Experience-Based Algorithm
EFACO	Enhanced Flying Ant Colony Optimization
EQSA	Energy-centered and QoS-aware Services Selection
EHRs	Electronic Health Records

ES	Eagle Strategy
FACO	Flying Ant Colony Optimization
FCNS	Fuzzy Continuous Neighbourhood Search
GA	Genetic Algorithm
GA-SC	Genetic Algorithm-based Services Composition
GT-EQCA	Group Teaching-based Energy-Efficient and QoS-Aware Services Composition
GTO	Group Teaching Optimization
GWO	Grey Wolf Optimizer
HMM	Hidden Markov Models
IBA	Individual-Based Algorithm
ICT	Information and Communication Technology
ILP	Integer Linear Programming
IoT	Internet of Things
IP	Integer Programming
ITL	Improved Teaching–Learning
ITL-QCA	Improved Teaching–Learning-Based QoS-Aware Composition Algorithm
IW-SC	Invasive Weed Optimization-based Service Composition
J2EE	Java 2 Platform, Enterprise Edition
KH	Krill-Herd
LIP	Linear Integer Programming
LS-SCA	Learning-based Swarm Optimization-aware Service Composition Algorithm
LPG	Layered Path Generation Graph
M2M	Machine-to-Machine
MCDM	Multiple Criteria Decision Making
MILP	Mixed Integer Linear Programming
MIP	Mixed-Integer Programming
MO-SCOS	Multi-objective Service Composition and Optimal Selection
MOGWO	Multi-objective Grey Wolf Optimizer

MQoS	Mobility-aware QoS
MSA	Modified Simulated Annealing
MWOA	Modified Whale Optimization Algorithm
NBA	Novel Bat Algorithm
NSGA	Non-Dominated Sorting Genetic Algorithm
OASIS	Organization for the Advancement of Structured Information Standards
P2P	Peer-to-Peer
PBA	Partition-Based Algorithm
PCPSO	Parallel Cloud Particle Swarm Optimization
POC	Pareto Optimal Compositions
PS-SC	Particle Swarm Optimization-based Services Composition
PSGW-SC	Particle Swarm and Grey Wolf Optimization-based Service Composition
PSO	Particle Swarm Optimization
QoE	Quality of Experience
QoS	Quality of Service
QC-NBA	QoS-Aware Services Composition using a Novel Bat Algorithm
REST	Representational State Transfer
RFID	Radio Frequency Identification
SA	Simulated Annealing
SAW	Simple Additive Weighting
SDPLC	Single-Dimensional Perturbation Logical Chaos Strategy
SFLGA-SC	Shuffled Frog Leaping and Genetic Algorithm-based Service Composition
SFGA	Shuffled Frog Leaping Algorithm and Genetic Algorithm
SHTs	Smart Home Technologies
SGs	Smart Grids
SOAP	Simple Object Access Protocol
SOC	Service-Oriented Computing
SOA	Service-Oriented Architecture
SSA	Salp Swarm Algorithm

SS	Improved Salp Swarm
SRA	Strategic Research Agenda
TL-SC	Teaching Learning Optimization-based Services Composition
TLBO	Teaching–Learning-Based Optimization
UPnP	Universal Plug and Play
VOO	Vector Ordinal Optimization
VOPC	Vector Ordered Performance Curve
WOA	Whale Optimization Algorithm
WS-BPEL	Web Services Business Process Execution Language
WSFL	Web Services Flow Language
WSNs	Wireless Sensor Networks
WSDL	Web Services Description Language
WSC-MDP	Web Service Composition as a Markov Decision Process
XLANG	XML-based Language for Web Services

General Introduction

In recent years, the proliferation of computing devices such as smartphones, digital tablets, and connected objects has significantly transformed our daily lives. These devices provide a wide range of functions, including assistance, monitoring, communication, and information retrieval, highlighting their important role in modern society. This technological evolution has led to the emergence of complex systems integrating the cyber, physical, and social worlds, known as *cyber-physical-social systems (CPSS)* [Sheth et al., 2013, Murakami, 2012]. The latter are based on modular, interoperable services as a paradigm inherited from Service-Oriented Computing (SOC), and incorporate the Internet of Things (IoT) as a key factor for interacting with the physical world, through connected devices that collect and transmit contextual data in real time. A CPSS combines computing elements, physical devices, and human interactions to create intelligent environments that provide transparent smart services accessible anytime and anywhere, while continually adapting to diverse contexts and usage patterns. For example, smart buildings can guide their visitors, configure their electronic devices, and offer contextualized services, thereby improving user comfort, optimizing resource utilization, and enhancing the overall efficiency of the environment. CPSS thus seamlessly integrates computers and connected devices into our daily lives, automatically adapting to our behaviors and preferences without requiring our continuous attention. A CPSS relies on infrastructure for computing, communication, and control that is typically composed of multiple layers for each of the three worlds and contains a variety of resources and services. Implementing a CPSS requires effective strategies to handle the fluctuations in QoS requirements, environmental conditions, and user preferences at design time, as well as the dynamism of system behavior during run-time. These challenges arise from the diversity of component types and the highly dynamic nature of application environments [Smirnov et al., 2015]. Among the most important strategies, the service composition process plays a key role in coordinating and organizing essential elements in the cyber, physical, and social worlds.

The service composition is a mechanism that allows assembling software components to enable the development and deployment of fast, cost-effective, and scalable applications [Benatallah et al., 2005]. This mechanism favors the creation of more sophisticated applications by combining existing services to provide added-value functionalities that cannot be achieved by individual basic services [Zeng et al., 2004]. This process results

in a *composite service* that integrates multiple atomic or basic services, each responsible for a specific functionality. These services are orchestrated according to predefined control and data flows to collectively fulfill complex user requirements. However, the service composition is not limited to functional aspects and must account for non-functional requirements, which are essential for assessing the overall quality and suitability of the resulting composite service. The non-functional aspects include Quality of Service (QoS) attributes, such as response time, reliability, availability, and cost [Huang et al., 2009], along with other critical factors, including energy consumption, user mobility, context-awareness, and resource constraints. Such considerations are crucial in dynamic and heterogeneous environments such as IoT and CPSS, where services are often distributed, mobile, and constrained by limited energy. To ensure that the resulting composite service meets both user requirements and system constraints, users may impose global non-functional constraints on the composition process. These constraints often include QoS requirements, such as limiting the overall cost below a specific threshold or ensuring that total energy consumption remains within an acceptable range. Considering such global non-functional constraints, the service composition problem becomes a Non-deterministic Polynomial-time hard (NP-hard) problem, where the main challenge is to find a sub-optimal composition within a reasonable computation time [Ardagna and Pernici, 2007]. Consistently, a key issue in service composition is *how to intelligently and effectively select the most relevant services that not only meet the user's functional requirements but also satisfy the imposed non-functional constraints.*

In this thesis, we focus on the problem of service composition based on non-functional properties in the context of CPSS and IoT environments. The goal is to develop methods that enable the construction of new applications by combining existing services to provide functionalities that none of them can provide individually. The service composition process involves aggregating multiple candidate services to create a composite service while optimizing non-functional properties. However, in large-scale CPSS environments, several research issues arise related to efficient management of QoS and energy, user mobility, and the stochastic nature of services. Among these, the optimal management of QoS and energy is crucial in CPSS environments, as it directly impacts system performance and efficiency. The QoS refers to the level of service performance with respect to the user's needs, including parameters such as execution time, cost, reliability, availability, and others. The QoS attributes have been exploited as non-functional properties to differentiate between functionally equivalent services during the service composition process [Zeng et al., 2004]. In terms of energy, services belonging to the composition are typically hosted by devices with low-autonomy batteries that cannot be easily replaced or recharged in the case of large-scale service deployment [Guinard et al., 2010]. During the execution of a composition, these services may become unavailable due to intensive battery use, resulting in composite services with low availability and no guarantee of successful execution [Khanouche et al., 2016].

Furthermore, mobility poses a major challenge for service composition, as it requires continuous coordination among mobile services to ensure the seamless and uninterrupted satisfaction of user requirements. Unlike static environments, where service availability and location are relatively stable, mobile contexts introduce continuous fluctuations in service accessibility and quality. As users and devices move across networks or environments, services may appear or disappear dynamically, leading to perturbations in the composition process. Moreover, mobility often causes intermittent connectivity, fluctuating bandwidth, and increased latency, which affect service performance. These factors make service composition more challenging as predicting service availability and reliability over time becomes increasingly complex. In addition, mobility induces a high degree of stochastic behavior in services, meaning that their state and performance can vary unpredictably depending on location, mobility patterns, or environmental conditions. This uncertainty increases the risk of service failure, degradation in execution quality, or unavailability of required components during execution. These issues are particularly critical in scenarios where services are hosted on highly mobile and resource-constrained devices, such as smartphones, wearables, or drones. As a result, user and service mobility not only introduces significant complexity into the composition process but also affects the stability, continuity, and overall efficiency of the resulting composite service. [Deng et al., 2017a, Deng et al., 2017b]

The first objective of this thesis is to design a service composition approach in IoT environments that considers both energy consumption and QoS criteria. IoT systems often consist of devices with limited energy resources, which poses challenges for service availability. Furthermore, the dynamic nature of these systems leads to frequent changes in service availability and performance. In large-scale IoT environments with thousands of entities, managing QoS and energy efficiently becomes challenging, as services may fail or provide insufficient quality, and limited device energy can further disturb execution, especially since batteries are often not easily replaced or recharged in large-scale deployments. Existing service composition approaches often focus on either energy efficiency or QoS metrics, but rarely both simultaneously. As a result, service compositions may not meet user requirements or cause rapid discharge of device batteries. Therefore, the first objective of this thesis is to develop an optimization algorithm that: (i) extends the battery life of devices by reducing energy consumption and (ii) satisfies user QoS requirements by optimizing metrics such as availability, cost, reliability, response time, reputation, and throughput.

The second objective of this thesis is to extend the service composition approach to CPSS environments by integrating mobility, QoS, and energy consumption. Mobility is a critical factor in CPSS, as both users and services frequently move across different locations, leading to fluctuations in service availability, reliability, and overall composition quality. These dynamic changes increase the risk of interruptions, service unavailability,

failures, or a decline in QoS over time. Accordingly, the second objective of this thesis focuses on developing an optimization algorithm that: (i) integrates a mobility model to dynamically adapt service composition according to user trajectories and locations, ensuring seamless connectivity and stable service availability, (ii) incorporates an energy consumption model to select services efficiently, manage energy consumption, and extend the battery life of the device while maintaining high-quality compositions, and (iii) satisfies the user's global constraints and optimize QoS metrics, including execution time, cost, reliability, availability, and throughput.

To achieve our research objectives, the remaining parts of this thesis are structured into four chapters as follows:

Chapter 1 first provides an overview of the Service-Oriented Computing (SOC), Internet of Things (IoT), and Cyber-Physical-Social Systems (CPSS) paradigms. Next, concepts related to the service composition are introduced, including functional and non-functional properties of services, the selection mechanism, and the composition process. Finally, an analysis of research challenges related to service composition is also carried out in the context of IoT and CPSS environments.

Chapter 2 is devoted to a comprehensive study of service composition approaches in the literature, focusing on key criteria such as QoS, energy efficiency, and mobility. The chapter begins by presenting conventional QoS-based service composition approaches that aim to optimize various QoS parameters, such as availability, reliability, execution time, cost, etc. Next, the chapter addresses energy-efficient service composition approaches that aim to reduce the energy consumption of services, a crucial factor in resource-constrained environments. Furthermore, the chapter integrates mobility-aware service composition approaches in scenarios where services or users are mobile, requiring dynamic adaptation of the service composition process to maintain QoS. Finally, a comparative analysis is presented to highlight the strengths and limitations of the discussed approaches.

Chapter 3 introduces the first contribution of the thesis, the group teaching-based energy efficient and QoS-aware service composition algorithm (GT-EQCA), designed to address energy consumption and QoS challenges within IoT environments. The chapter begins by defining the foundational models used in the GT-EQCA algorithm and by outlining the service composition problem with a focus on energy and QoS issues. Subsequently, the Group Teaching Optimization (GTO) method is described to provide the foundations for the GT-EQCA approach, which is carried out in three phases: (i) Pruning of IoT concrete services that cannot provide the required user's QoS level; (ii) Selection of top-k IoT concrete services using the relative Pareto dominance principle; (iii) Finding the composite service with sub-optimal QoS (i.e., the composition that satisfies global QoS constraints and has the highest utility value in terms of QoS

and energy). Finally, the performance of the GT-EQCA approach is assessed through a comparison with four relevant baseline approaches by considering metrics such as composition time, energy consumption, and QoS utility.

Chapter 4 extends the service composition problem introduced in the previous chapter to a more complex and dynamic context of large-scale CPSS environments. This chapter presents the second contribution of this thesis, the learning-based swarm optimization-aware service composition algorithm (LS-SCA) that simultaneously integrates user mobility, energy efficiency, and QoS constraints, which become increasingly critical in such environments. The chapter begins by formalizing the service composition problem in the context of CPSS and introducing two key models: a mobility model and an energy consumption model that guide the service selection decision. A utility function is then proposed to evaluate composite services by integrating mobility awareness, energy consumption, and QoS attributes. Subsequently, the Two-Phase Learning-based Swarm Optimization (TPLSO) method used in the LS-SCA algorithm is described. A motivation scenario is also presented to illustrate the impact of user mobility on energy consumption and QoS. Finally, the LS-SCA algorithm is evaluated through extensive simulations in comparison with several baseline approaches by considering key metrics such as utility value, energy consumption, availability, and composition time. A mobility assessment is also performed to analyze the impact of user movement and velocity on aspects such as energy consumption and service availability.

At the end of this work, a synthesis of the key contributions presented throughout this thesis is given, and several perspectives are outlined to improve and extend this study.

Chapter I

From Service-Oriented Computing to Cyber-Physical-Social Systems: Foundational Concepts and Definitions

1 Introduction

In this chapter, we introduce the fundamental paradigms underlying the evolution from service-oriented computing (SOC) to cyber-physical-social systems (CPSS). We begin by introducing the SOC paradigm that uses services as modular elements in the development of software applications. This paradigm is related to service-oriented architecture (SOA), which provides a structured framework for organizing and integrating services. Then, we define the Internet of Things (IoT) as a network of physical devices equipped with sensors, software, and other technologies, connected to exchange data via the Internet. After that, we move on to CPSS, beginning with its definition, outlining the architecture of cyber-physical systems (CPS), and extending this to include social dimensions. We then identify the key components of CPSS and present various applications and case studies that illustrate its implementation. Finally, we present the concept of service composition that involves combining multiple services to create more complex applications. We give the characteristics of services, introduce their types, and their levels of granularity. In addition, we examine the service composition lifecycle, categorize the composition methods, and discuss the inherent challenges of service composition in CPSS and IoT environments.

2 Service-Oriented Computing

Service-oriented computing (SOC) is a computational paradigm that mainly uses services as the core components for application development. The primary strength of SOC lies in its ability to promote loose coupling among services, allowing individual services to be independently developed, deployed, and maintained. This modular approach allows

organizations to build complex applications by combining smaller, reusable services, thus enhancing development efficiency and reducing costs. SOC is associated with service-oriented architecture (SOA) that provides a structured approach to organize software applications and infrastructure into an interacting set of services. This organization enables efficient communication and interoperability between heterogeneous systems and platforms [Papazoglou, 2003].

In the SOC environment, services are accessible via standardized protocols such as Simple Object Access Protocol (SOAP) and Representational State Transfer (REST), which allow service components to communicate across different programming languages and systems, maintaining platform independence. This architecture enables services to function as modular units that can be dynamically integrated into larger workflows, allowing companies to quickly adapt to changes and reconfigure services to meet evolving needs [Huhns and Singh, 2005]. A SOA can be implemented using a wide range of technologies, such as Java 2 Platform, Enterprise Edition (J2EE), and .NET. These technologies provide the infrastructure necessary to create, deploy, and manage services.

2.1 Service-Oriented Architecture

Service-oriented architecture (SOA) is the architectural model that implements the principles of SOC by defining the structural framework to organize and coordinate services within software systems. The idea behind this model is to provide a way to organize, standardize, and manage services so that they can communicate and work together, regardless of their underlying platforms, technologies, or providers. This architecture emphasizes a set of fundamental requirements to facilitate the rapid and cost-effective development of applications through the following key requirements: [Papazoglou and Van Den Heuvel, 2007]

- **Technology independent:** services within SOA are designed to be independent of specific implementation technologies or standards on the client and service sides. For instance, services can rely on universal communication protocols such as SOAP or REST, which support interactions between different technology platforms.
- **Loose coupling:** this characteristic refers to the fact that the internal workings of the service, and the service itself, do not need pre-existing knowledge of the client's usage context. This enables services to be reusable and easily substitutable on demand, enhancing scalability and flexibility.
- **Support location transparency:** the consumer can access a service without knowing its location. Service descriptions, including functional and non-functional characteristics, are typically published in a service registry using protocols such as Web Services Description Language (WSDL). Service consumers can locate and invoke services using these descriptions without requiring details on where or how they are deployed.

2.2 Foundations of Service-Oriented Architecture

The foundations of SOA are built on three core entities: the service provider, the service consumer, and the service registry that interact to enable efficient, flexible, and scalable service-based systems (see Figure I.1).

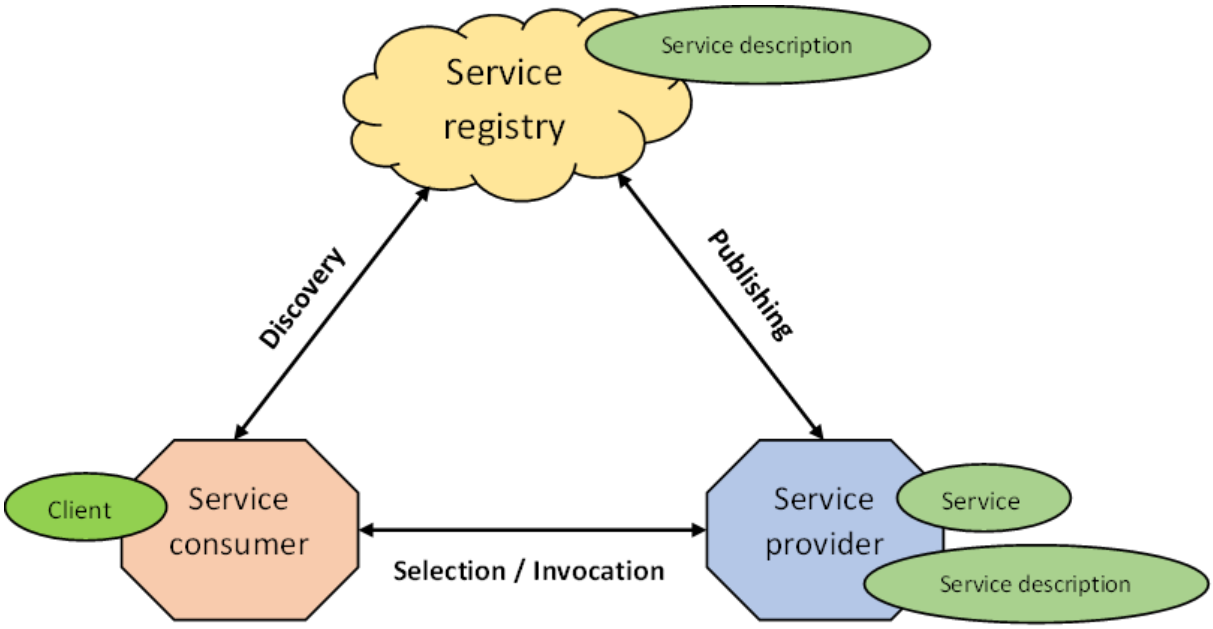


Figure I.1: Interactions between actors in a SOA model.

- **Service provider:** This entity is responsible for creating, hosting, and maintaining services. Providers publish service descriptions in a service registry that includes functional and non-functional properties to help consumers understand each service's capabilities and requirements.
- **Service consumer:** Also referred to as the client or user, this entity initiates requests to the service provider. Consumers search the service registry to discover services that meet their requirements. After finding the most suitable service, the consumer can invoke it according to the specifications provided in the service description.
- **Service registry:** This entity acts as a central directory where service descriptions from various providers are stored. This registry serves as a broker, connecting service providers with consumers and enabling efficient service discovery across the SOA ecosystem.

Each entity plays a distinct role. The interaction between these entities ensures the execution of the fundamental processes involved in publishing, discovering, selecting, and invoking services.

- **Publishing:** The service provider publishes a description of a service in the registry, which includes the service's functionalities, its communication protocols, and its non-functional properties. This description allows the service to be easily discovered and used by potential consumers.

- **Discovery :** refers to the process by which a service consumer searches the service registry for services that meet specific requirements. Consumers can initiate a search by submitting descriptions of the required services to the registry manager, which then returns a list of matching services.
- **Selection :** From the services found in the discovery process, the consumer selects the one that best meets its functional and non-functional requirements, such as Quality of Service (QoS) attributes (e.g., response time, availability, reliability), energy efficiency, or contextual relevance (e.g., location, time of use, or device compatibility).
- **Invocation :** After the selection, the consumer invokes the chosen service using a standard communication protocol such as SOAP, allowing secure and reliable communication across distributed systems.

By promoting reusability and interoperability, SOC has become a core architectural approach in fields such as business process management, enterprise integration, and cloud computing. However, traditional SOC architectures encounter limitations when applied to dynamic and heterogeneous environments that require real-time interaction with physical devices, as well as greater scalability and responsiveness. These limitations have led to the extension of SOC principles into new paradigms, particularly the Internet of Things (IoT) and Cyber-Physical Systems (CPS).

3 Internet of Things

The Internet of Things (IoT) is the practical realization of ubiquitous computing, where technology is naturally integrated into everyday objects. This powerful concept opens the way to many applications that link the physical and digital worlds, including smart homes, healthcare, smart cities, logistics, security, etc [Aoudia, 2022]. Despite its promise, IoT faces a range of technical and non-technical challenges that must be addressed to fully achieve its potential. These challenges include scalability issues due to the large number of connected devices, energy constraints due to limited device resources, interoperability across heterogeneous platforms, as well as privacy, security, and trust problems.

3.1 Definition of Things

A *thing* refers to a physical or virtual entity that interacts within a digital network environment. This entity moves through time and space and can be uniquely identified through an assigned identification number, name, and/or location address. A thing is therefore easily readable, recognizable, locatable, addressable, and controllable via the Internet [Atzori et al., 2010, Borgia, 2014].

In this definition, an object represents a concrete form of a thing linking the abstract notion to identifiable entities. Objects can be active or passive. Active objects

are equipped with electronic components that enable them to process data, sense environmental conditions (e.g., temperature, humidity, light), communicate, and sometimes act autonomously. Passive objects, such as RFID tags, lack processing power but can be identified and tracked within the IoT [Want, 2006]. The computing capacity of each device varies, influencing the complexity of tasks it can perform, from basic calculations to facial recognition, depending on available resources [Guinard et al., 2010]. These resources, coupled with communication interfaces, make objects capable of interacting with various wired and wireless networks.

Beyond their hardware resources, physical objects possess various characteristics related to their usage contexts. Some objects are mobile, meaning they can change position over time. These mobile items can be transportable, such as a smartphone, book, or clothing, or autonomous with motor capabilities, such as cars, drones, or pets, enabling them to be tracked within IoT systems [Borgia, 2014]. Conversely, many devices are stationary, such as refrigerators, furniture, and electric meters, remaining fixed for most of their lifecycle [Atzori et al., 2010].

Power sources further differentiate these objects. Some have continuous access to electricity, like typical household appliances, while others rely on batteries, such as wireless sensors or wearable devices, making their operational span dependent on battery life. This factor varies depending on whether the object is easily rechargeable, such as a mobile phone, or is designed to operate for long periods without recharging, such as tracking devices used to monitor the migration of farm animals [Borgia, 2014].

3.2 Definition of Internet of Things

Several definitions of IoT are provided in the literature. Among the most significant, some emphasize its technological and infrastructural dimension [Benghozi et al., 2008, Atzori et al., 2010], others reflect its role as a communication architecture linking sensors and actuators [Dorsemaine et al., 2015], and others stress its vision of continuous and ubiquitous connectivity [Rahmani et al., 2018].

The Internet of Things is defined as a network of physical and virtual entities that are uniquely identifiable, equipped with sensors, actuators, and communication capabilities, and can interact and exchange data through standardized and unified electronic identification systems and mobile wireless devices. This enables seamless data recovery, storage, transfer, and processing across the physical and virtual domains [Benghozi et al., 2008]. According to [Atzori et al., 2010], the IoT is based on the fundamental idea of an ubiquitous presence of various ”things” or objects such as RFID tags, sensors, actuators, and mobile devices that, through unique addressing schemes, can interact and collaborate with each other to achieve shared objectives. The Internet of Things is the assembly of infrastructures that link sensors and/or actuators with limited computing capabilities

and enable them to be managed, accessed, and have their generated data transmitted over the Internet without the need for human-to-human or computer-to-computer interaction [Dorsemaine et al., 2015]. According to [Rahmani et al., 2018], the IoT is a vision of connecting a large number of objects with unique identifications continuously and anywhere using IoT communication protocols. IoT devices can include cell phones, wearables, machines, and doors.

In this thesis, we define the IoT as a paradigm in which a large number of physical and virtual objects are interconnected via standardized protocols over the Internet. These objects, equipped with computing resources, can collect and process data (e.g., temperature, location, status) from their environment, interact, and collaborate autonomously or under human supervision. This capability enables the seamless acquisition, storage, transfer, and processing of data between physical and virtual domains, promoting innovative services while addressing technological, economic, and societal challenges.

3.3 Application domains of IoT

The applications of IoT are various and impact nearly every aspect of daily life, including society, industry, and environment (see Table I.1). The Internet of Things Strategic Research Agenda (SRA) [1], published in 2009, identified key application domains, including energy, healthcare, buildings, transport, smart living, and smart cities. Additionally, the IoT-I [2] project conducted a survey in 2010 that identified 65 IoT application scenarios grouped into 14 domains, including transportation, smart home, smart city, lifestyle, retail, agriculture, smart factory, supply chain, emergency, healthcare, user interaction, culture and tourism, environment, and energy [Porkodi and Bhuvaneswari, 2014]. Some of these IoT applications are detailed below.

3.3.1 Smart cities

The main goal of smart city strategies is to improve urban performance using information and communication technologies to provide efficient services and enhance existing infrastructure [Albino et al., 2015]. A wide range of applications can be deployed within smart cities [Nandury and Begum, 2015, Gharaibeh et al., 2017, Morello et al., 2017, Du et al., 2018], such as smart streetlights, smart buildings, smart grids, smart water distribution, smart farming, pollution detection, and smart surveillance. For example, a smart streetlight system can automatically adjust its brightness based on real-time data from motion sensors, conserving energy while improving safety.

[1] Cluster of European Research Projects on the Internet of Things (CERP-IoT), *Internet of Things – Strategic Research Agenda*, January 2009, available at: https://www.internet-of-things-research.eu

[2] https://cordis.europa.eu/article/id/123927-ioti-survey-on-critical-privacy-factors/en

Table I.1: IoT application domains .

Domain	Objective	Applications
Society	Improving and developing society, cities, and people	Smart Cities, Smart Animal Farming, Smart Agriculture, Healthcare, Home Automation, Energy, Defense, Medical Technology, Ticketing, Smart Buildings
Environment	Protecting, monitoring, and managing natural resources	Smart Environment, Smart Metering, Smart Water Recycling, Disaster Alerting
Industry	Activities related to financial and commercial transactions between companies and organizations	Retail, Logistics, Supply Chain Management, Automotive, Industrial Control, Aerospace and Aviation

3.3.2 Smart agriculture and smart water

In agriculture, IoT applications optimize crop production by monitoring environmental factors such as soil moisture, temperature, humidity, and sunlight exposure. For example, smart greenhouse systems use IoT sensors to adjust conditions and maximize production. Meanwhile, smart water management systems monitor water quality and levels in rivers, reservoirs, and dams, while also detecting leaks and pressure variations in pipelines to prevent wastage and reduce maintenance costs.

3.3.3 Health care

In remote patient monitoring, IoT devices track vital signs in real time, such as heart rate, blood pressure, and glucose levels. For example, wearable devices monitor glucose levels of diabetic patients, alerting the patient and the healthcare provider when levels are dangerously high or low [Aljohani and Alenazi, 2020]. Figure I.2 illustrates an IoT use case in healthcare where wearable sensors collect medical data and store them as Electronic Health Records (EHRs) on a secure cloud database. Healthcare providers can access these EHR data to simplify patient management and develop personalized treatment plans. Another scenario is hospital asset tracking, where IoT tags are placed on medical equipment and patient records to improve hospital workflow and reduce the time spent locating equipment.

3.3.4 Retail and logistics

IoT solutions in retail and logistics provide significant operational advantages. In a retail store, a smart inventory management system uses RFID tags to track product movement

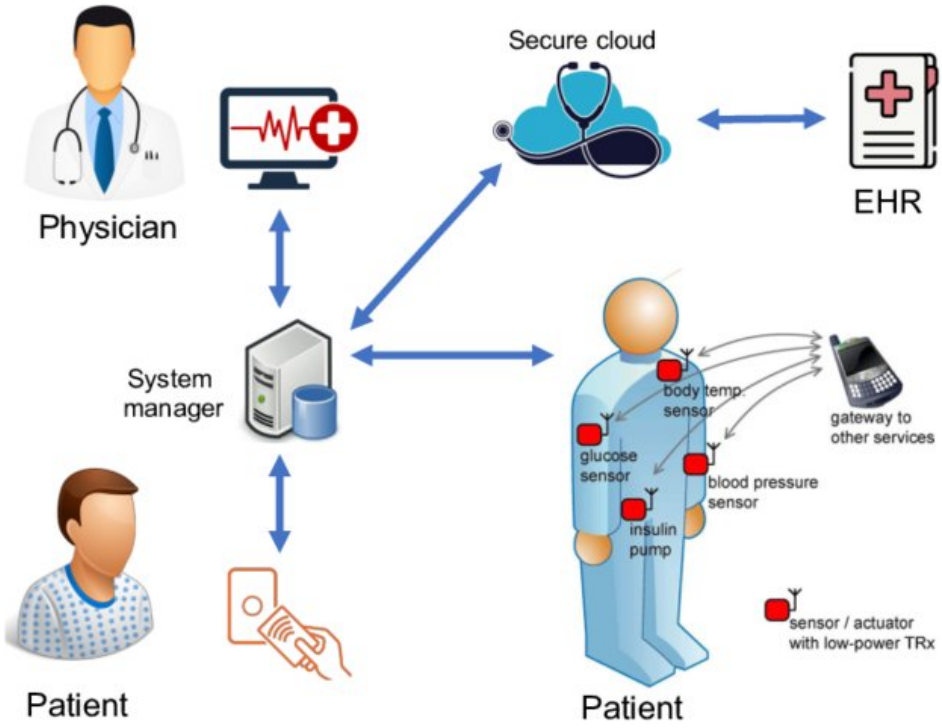


Figure I.2: Use case scenario of IoT in healthcare.

in real time and automatically place new orders when stock levels are low. Automated checkout systems streamline the shopping experience and reduce waiting times. Additionally, smart logistics solutions enable real-time tracking of goods throughout the supply chain, while IoT sensors monitor the condition of perishable goods during transit to ensure optimal storage.

3.3.5 Security and emergencies

IoT technologies are increasingly used in security and emergency services to enhance safety and streamline response efforts. For example, perimeter security systems use IoT sensors to detect unauthorized entry into restricted areas and trigger immediate alarms. Other applications include radiation monitoring at nuclear power plants, flood detection with sensors along rivers or basements, and liquid presence detection in sensitive environments, such as warehouses.

4 Cyber-Physical Social Systems

In recent years, technological progress has transformed isolated devices into unified control systems that operate together within specific networks. These advances encompass a wide range of smart devices, social networks, and IoT devices interconnected via the Internet in a large-scale ecosystem. This transformation has fundamentally reshaped business models and ecosystems across diverse fields, opening new perspectives for interactive and data-driven applications. As a natural extension of this evolution, cyber-physical systems

(CPS) have expanded to incorporate human and social dimensions, giving rise to cyber-physical-social systems (CPSS) [Wang, 2010].

4.1 Definition of cyber-physical-social systems

Most literature studies share a common understanding that a CPSS is composed of three interlinked subsystems: (i) the human-based social system that includes human actors, connected devices, and social platforms offering human-centered services; (ii) the software-based system, representing the cyber domain, which provides software-based services via on-premises or cloud infrastructures; (iii) the thing-based system including sensors, actuators, gateways, and underlying physical infrastructures [Yilma et al., 2019].

The concept of CPSS has been defined in various ways, reflecting its multidimensional nature. For instance, CPSS can be described as a combination of CPS and cyber-social systems (CSS) that enable smart interactions across cyber, physical, and social domains. A CPS refers to devices like communicators and business-processing tools, while a CSS represents social networks such as Facebook, Twitter, and YouTube. A CPSS goes beyond devices connectivity and machine-to-machine communication by connecting people to allow them to interact with the physical world [Zeng et al., 2020]. In other words, a CPSS is defined as a system that captures synergetic interaction between computing and human experience, offering holistic computational solutions across the (physical, cyber, and social) dimensions [Sheth et al., 2013, Murakami, 2012]. According to [Yilma et al., 2019], a CPSS can be conceptualized as an integrated environment in which humans and machines interact across physical and virtual spaces. The components of a CPSS are three interconnected subsystems: the cyber, the physical, and the social (see Figure I.3) [Yilma et al., 2019].

The definitions of CPSS agree on the fundamental role of humans within this system. When coming to the social dimension, the majority of research falls into one of two categories:

- *Human as a sensor:* Some approaches conceptualize humans as information sources or *sensors*, integrating social data (observations and experiences) with cyber-physical systems to meet a variety of application requirements [Su et al., 2017, Wang, 2010, Gharib et al., 2017, Huang et al., 2016].
- *Human as a system component:* Several studies consider humans as active system components who shape and influence the CPSS [Liu et al., 2011, De et al., 2017, Sheth et al., 2013, Wang et al., 2017].

4.2 Architecture of cyber-physical-social systems

CPSS have traditionally focused on technical and physical components, with humans viewed as external to the system, and emphasize core cyber aspects such as confidentiality, integrity, and availability [Yilma et al., 2021]. The CPSS architecture is closely similar to that of the IoT, comprising three main layers: the perception layer, the

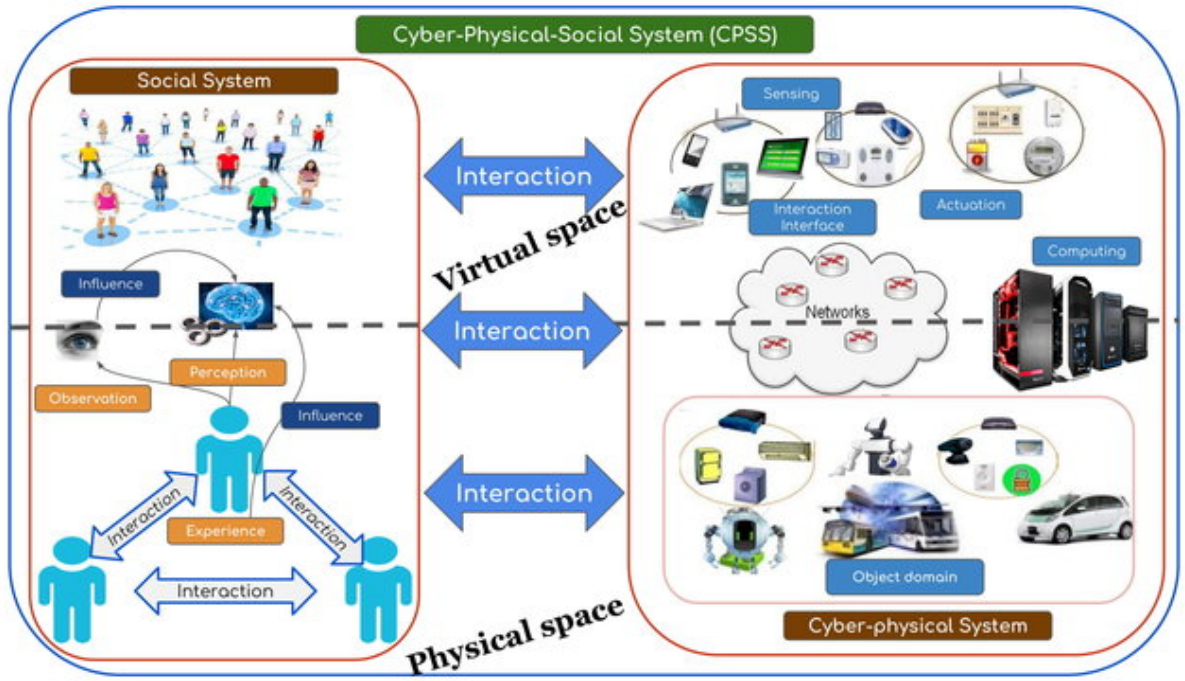


Figure I.3: Overview of a cyber-physical and social system.

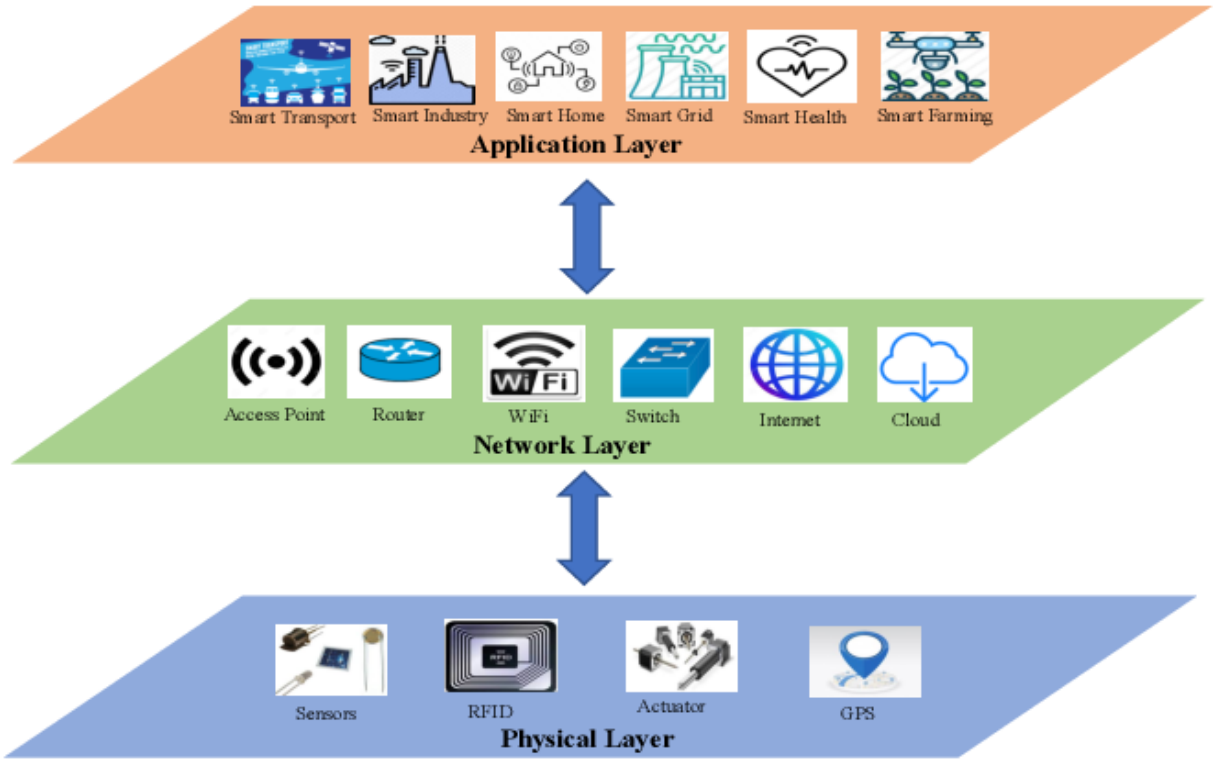


Figure I.4: A typical layer-based CPSS architecture.

network layer, and the application layer (see Figure I.4) [Chaganti et al., 2021]. The perception layer includes sensors and actuators that gather environmental data, while the network layer manages communication and enables data exchange between system components. The application layer processes data, enabling data-driven decision-making [Yaacoub et al., 2020, Yilma et al., 2021, Mahmoud et al., 2015].

With the rise of social media, CPSSs have emerged to explore how humans integrate within systems spanning the cyber, physical, and social domains [Yilma et al., 2021, Smirnov et al., 2014]. Unlike traditional human-in-the-loop systems, where human interactions follow predefined actions or decision-support roles [Sowe et al., 2016], a CPSS considers the social impacts and continuous interactions between people and the system, where each influences the other [Yilma et al., 2021].

4.3 Components of CPSS

CPSS have three primary components as shown in Figure I.5 [Gharib et al., 2017]:

- **Cyber System:** This component includes technical elements, such as computers, networks, and data-processing technologies. While humans may interact with these elements (e.g., read/ write information), they are not considered as an integral part of the system. An example of a cyber system is a healthcare information system that allows physicians, nurses, and patients to access medical records, although these users have no direct influence on the core system functions.
- **Cyber-Physical System (CPS):** This element integrates a digital (cyber) system with a physical one that potentially involves human interaction [Bondavalli et al., 2016]. In a CPS, the focus is placed on the interaction between cyber and physical elements, with human involvement not inherently integrated into the system. More specifically, sensors collect information from the physical world, which is then stored and processed by the cyber system. For example, in managing a diabetes patient's care, the cyber system could uses sensors to monitor blood glucose levels, analyze this data, and adjust insulin flow using actuators without a direct social interaction.
- **Cyber-Physical-Social System (CPSS):** CPSS integrates digital systems, controlled physical objects, and social components that include interactions among people. In a CPSS, both cyber-physical and social interactions are taken into account, with an emphasis on the human relationships within the system. Designing a CPSS with these social interactions can increase system trustworthiness by anticipating and addressing potential vulnerabilities within human interactions. For instance, in the case where a patient mistrusts his healthcare provider, he may be less likely to adhere to treatment, potentially impacting his health outcomes.

4.4 Applications and case studies of CPSS

The CPSS encompasses a wide range of applications such as smart cities [Cassandras, 2016], smart homes [Rahman, 2017], smart tourism [Ashari et al., 2020, Nisiotis et al., 2020], intelligent transportation [Xiong et al., 2015], social manufacturing [Ding and Jiang, 2018, Jiang et al., 2016], Critical Infrastructure, smart healthcare [Tian et al., 2019], smart energy [Camtepe and Yener, 2007, Xue and Yu, 2017],

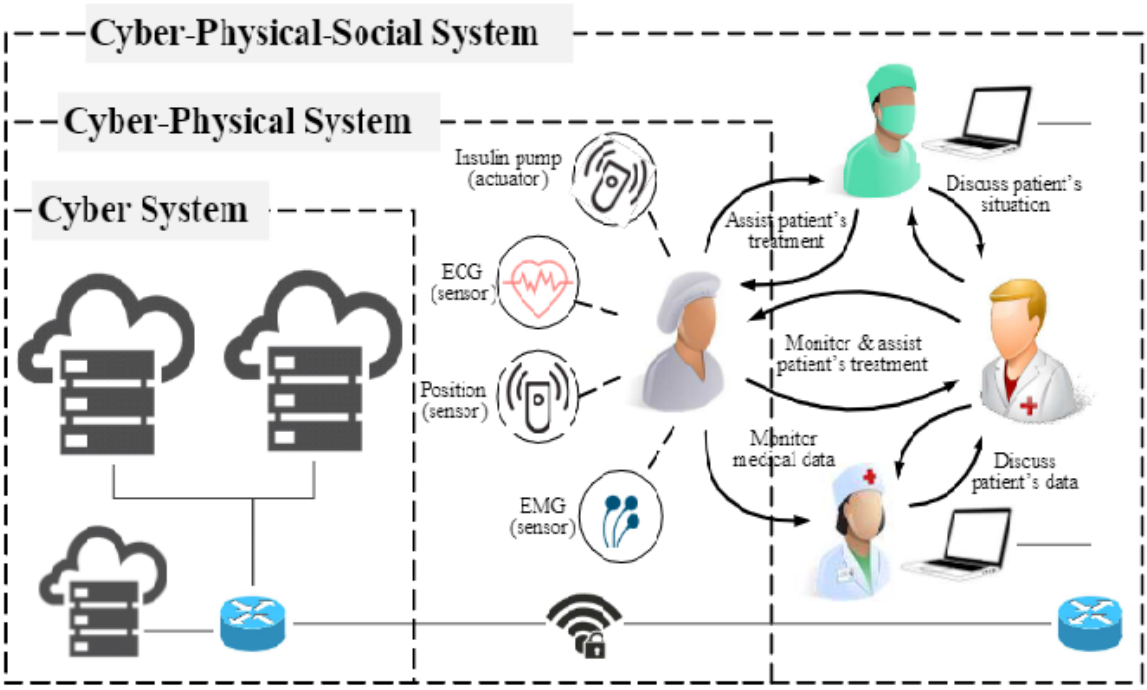


Figure I.5: The evolution of CPSS.

humanoid robots [Rahman, 2017], and safety and security systems [Gati et al., 2021, Sharma et al., 2020].

- **Smart cities:** Smart cities rely on CPSS to improve urban management and sustainability by integrating information and communication technology with urban infrastructure [Angelidou, 2015, Anthopoulos, 2015]. IoT and sensor networks further support adaptability to residents' needs through predictive services such as smart parking and environmental monitoring [Cassandras, 2016, De et al., 2017, Guo et al., 2015].
- **Smart homes:** Smart home technologies (SHTs) link devices, sensors, and monitors to offer automated and personalized domestic control [Kim et al., 2020]. An effective SHT design considers user preferences and integrates psychological and social factors [Smirnov et al., 2015].
- **Intelligent transportation:** a CPSS enables *Transportation 5.0* to enhance traffic management by integrating social and human dynamics [Xiong et al., 2015]. This approach also supports autonomous driving through systems that predict driver behavior [Dressler, 2018].
- **Smart healthcare:** Smart healthcare provides various services, from health management to virtual assistants [Tian et al., 2019]. Although privacy challenges persist, advancements in CPS suggest the potential of CPSS frameworks in healthcare to support secure and socially integrated systems [Haque et al., 2014, Zhang et al., 2015].
- **Smart energy:** Smart energy systems focus on improving efficiency and reducing costs, often using Smart Grids for networked management. A CPSS in energy systems introduces models that incorporate environmental, economic, and behavioral factors [Camtepe and Yener, 2007, Xue and Yu, 2017].

- **Safety and security:** Privacy and security remain central aspects in CPSS. They require robust measures to protect data integrity while meeting the specific needs of these systems, including human decision-making [Gati et al., 2021, Sharma et al., 2020].

5 Service composition

In modern computing paradigms, the service is a key concept that can be seen as an abstraction of the functionality of a software entity. Functional and non-functional characteristics, such as performance, cost, and availability, enrich this abstraction. Services are described, published, and made accessible for discovery by other entities, whether software or human users. They can then be executed individually or combined with other services to meet specific needs. This approach enhances the reusability of high software components, improves interoperability, and simplifies the development of complex distributed applications.

5.1 Service concept

There are several definitions of the service concepts. In the following, we give some of the most pertinent definitions of the service concept, which highlight its core characteristics and significance.

A service is a self-contained, open, and self-describing component that supports the rapid, low-cost composition of distributed applications [Papazoglou and Van Den Heuvel, 2007]. This component is defined by a formal interface and is characterized by loose coupling, ensuring the independence between the client and the service. On the one hand, the client does not require knowledge of the service's underlying implementation technology or execution platform. On the other hand, the service operates independently of the client's context.

The Organization for the Advancement of Structured Information Standards (OASIS)[3] defines the service as a mechanism to access one or more capabilities via a prescribed service interface. This interface specifies the service's functionalities and the policies governing access to them, while the internal implementation remains opaque to the client. This ensures technological neutrality and fosters interoperability [MacKenzie et al., 2006].

A service can be described as a software resource with a service description that enables discovery and interaction [Arsanjani, 2004]. This view highlights the importance of service discovery, where users can locate services that match their needs based on a service registry or directory.

The business-oriented view of services is also prevalent in the literature. According to [Han et al., 2009], the service represents an abstraction of business logic and processes, encapsulated into reusable units aligned with organizational goals.

[3] http://docs.oasis-open.org/soa-rm/v1.0/

Based on these definitions, a unified vision emerges. A service is a self-contained, reusable software entity that provides access to specific capabilities through a standardized interface. This entity operates independently of its users and can be integrated into various contexts within distributed environments. This abstraction serves as a key enabler for flexible, scalable, and interoperable systems, driving innovation across different application domains.

5.1.1 Characteristics of a service

A service is characterized by functional and non-functional properties. The functional properties describe what the service is capable of performing, such as the accomplished tasks or the provided functionalities, while the non-functional properties define the quality of these functionalities, considering factors such as performance, reliability, security, and availability [Huang et al., 2009]. To better understand this concept, consider a CPSS-based smart traffic management service that assists urban authorities in monitoring and controlling traffic flow in cities. The functional properties include collecting traffic data in real time from road sensors and connected vehicles, detecting congestion or accidents, and dynamically adjusting traffic lights to optimize flow. The non-functional properties ensure that the service processes and analyses data with minimal latency, maintains high reliability to avoid disruptions in traffic control, and remains available at all times to operate under varying traffic and environmental conditions.

5.1.1.1 Functional properties

The functional properties describe the core behavior of a service, specifying how it performs tasks and interacts with other services or entities in a system. These properties include the service's inputs, outputs, post-conditions, and effects. Inputs represent the data required for the service's execution, while outputs are the produced results. Post-conditions specify the expected state of the system after the service's execution, while effects refer to broader system changes, such as database updates or the executed processes.

5.1.1.2 Non-functional properties

The non-functional properties of a service define its qualitative attributes, such as performance, efficiency, adaptability, and overall execution quality. Unlike functional properties, non-functional properties specify how well a service performs under varying conditions. The non-functional properties of a service encompass:

- A set of Quality of Service (QoS) attributes that allow for an objective evaluation of a service. The QoS attribute values can be obtained from different sources. Some

attributes, such as cost, are explicitly provided by service providers, ensuring transparency in service offerings. Others, such as response time, can be measured based on previous service executions, providing an empirical basis for performance assessment. Additionally, qualitative attributes such as reputation or user satisfaction are often derived from user feedback and comments, offering an experiential perspective on service quality [Liu et al., 2004].

- Energy efficiency refers to the ability of a service to minimize energy consumption while providing the expected level of performance. In dynamic environments, where resource availability and workload can fluctuate, energy-efficient services adapt their operation to balance user requirements with sustainable energy consumption. This property not only reduces operational costs but also promotes system scalability and ensures long-term sustainability.
- User specifications refer to the specific user's requirements and preferences that a service must meet. These specifications may include personalized performance metrics, user-defined thresholds, and particular conditions of use. Incorporating user preferences into the service design allows the application to provide a more personalized experience. For instance, in scenarios where a quick response is required, such as addressing a medical emergency involving an elderly person, assigning a higher preference to the response time parameter ensures that the selected services prioritize rapidity.
- Dynamic adaptability refers to the service's ability to maintain consistent performance and provide uninterrupted access while accommodating the changing conditions of users who may be in different physical locations or moving between varying network environments. This aspect is essential for mobile users or IoT applications, where seamless service delivery must be ensured across various contexts. Dynamic adaptability involves adjusting service parameters in real-time to optimize performance and energy consumption based on current conditions. For example, a service might adjust the data transmission rate to accommodate fluctuations in network bandwidth, ensuring efficient operation without over-consuming energy.
- A set of attributes reflecting the Quality of Experience (QoE) that measures the subjective perception of the user regarding the service's performance. This measure encompasses the user's perceptions, feelings, and attitudes toward the service, considering factors such as usability, accessibility, responsiveness, and reliability. Unlike QoS, which usually focuses on technical parameters from the provider's perspective, QoE emphasizes the end-user's viewpoint, evaluating how well the service meets their expectations and needs.

5.1.2 Type of service

To better understand service composition, it is necessary to distinguish between the different types of services. There are two main types: concrete services and abstract services. A concrete service is an implementation that executes a specific functionality and is described by functional and non-functional properties. An abstract service is a conceptual

entity that groups functionally equivalent concrete services that differ only in their non-functional attributes [Khanouche et al., 2020b]. More details about these two types of services are provided in Chapter III.

5.1.3 Granularity of service

A service can be *atomic* or *composite* in an IoT or CPSS environment. An atomic service is a stand-alone access point to an application that operates independently, without relying on other services to fulfill user requests [Sheng et al., 2014]. Each atomic service provides a programmatic interface, typically using protocols such as SOAP (Simple Object Access Protocol) and WSDL (Web Services Description Language) to define its operations and communication methods.

A composite service integrates several services (atomic or composite) to work together to perform complex user requests. For example, a travel planning service can be seen as a composite service that combines functionalities such as flight booking, hotel reservation, and tourist attraction search, where each of these functionalities represents a component service [Sheng et al., 2014].

Composite services can be distinguished on two levels: an abstract composite service and a concrete composite one. According to [Casati and Shan, 2002] and [Papazoglou and Van Den Heuvel, 2007], an abstract composite service defines a sequence of tasks required to achieve a given objective, but without specifying the concrete services that will execute them. In contrast, a concrete composite service assigns each task to a specific service instance, thus producing an executable composition.

5.2 Service selection

The service selection process aims to select, from a set of functionally equivalent services, those that best meet the user's functional and non-functional requirements. The discovery module then requests the service directory to identify services that can fulfill each abstract service in the user-provided composition plan by performing a functional mapping between the abstract services and the descriptions of available services. A set of concrete services is obtained for each abstract service, and the selection process chooses a concrete service for each abstract service to form a composite service that satisfies the user's requirements.

The service selection process is used to differentiate between services that offer the same functionality based on their non-functional properties, generally related to QoS [Maximilien and Singh, 2004]. This process is often used for composition, substitution, or configuration [O'Sullivan et al., 2002, Orriëns et al., 2003]. In the context of service composition, service selection is a crucial step to ensure that the final composite service meets the user's requirements.

5.3 Service composition

The functionalities provided by an intelligent environment are abstracted as services. In the case when a user requires a functionality that no single atomic service can provide within the environment, it becomes necessary to combine multiple services to fulfill this request. This process is referred to as service composition [Benatallah et al., 2005]. The service composition can be classified as static or dynamic based on the timing of the composition process. The static composition is performed at design time, where the composition plan and the services to be used are predefined and fixed before the application deployment, preventing adjustments during execution. In contrast, dynamic composition occurs at runtime, enabling the system to adapt to real-time changes in user requirements, service availability, or quality metrics. In the following, we provide definitions of the service composition concept.

5.3.1 Definition of service composition

The service composition is the ability to create added-value services by combining existing services offered by different organizations [Casati and Shan, 2002]. Similarly, [Khalaf and Leymann, 2003] defines the service composition as the aggregation and combination of a set of services to achieve a common goal. In summary, the service composition specifies which services to invoke, their execution order, the data exchanged, and the strategies for handling exceptions. This process can also be seen as a mechanism for integrating various services into a unified application [Benatallah et al., 2005]. Among these definitions, the one proposed by [Benatallah et al., 2005] is the most widely referenced in the literature.

These definitions converge on the idea that the primary goal of the service composition process is to create more sophisticated applications or services by combining several basic services to provide new functionalities that none of these services can provide individually. The resulting service is commonly referred to as *a composite service*, while the individual services that make up a composition are called *atomic services*. In other words, the composition process involves assembling several existing service classes into a coherent and structured plan. This process also includes identifying the optimal match between the non-functional properties of these classes and their corresponding concrete services, known as *candidate services*.

5.3.2 Service composition life-cycle

The composition process facilitates the transition from an abstract specification to an executable composition. The service composition life-cycle is a sequence of steps aimed at efficiently designing, implementing, and maintaining composite services (see Figure I.6). This life cycle is divided into four main phases: (i) the definition phase; (ii) the discovery phase; (iii) the selection phase; (iv) the deployment and execution phase [Sheng et al., 2014].

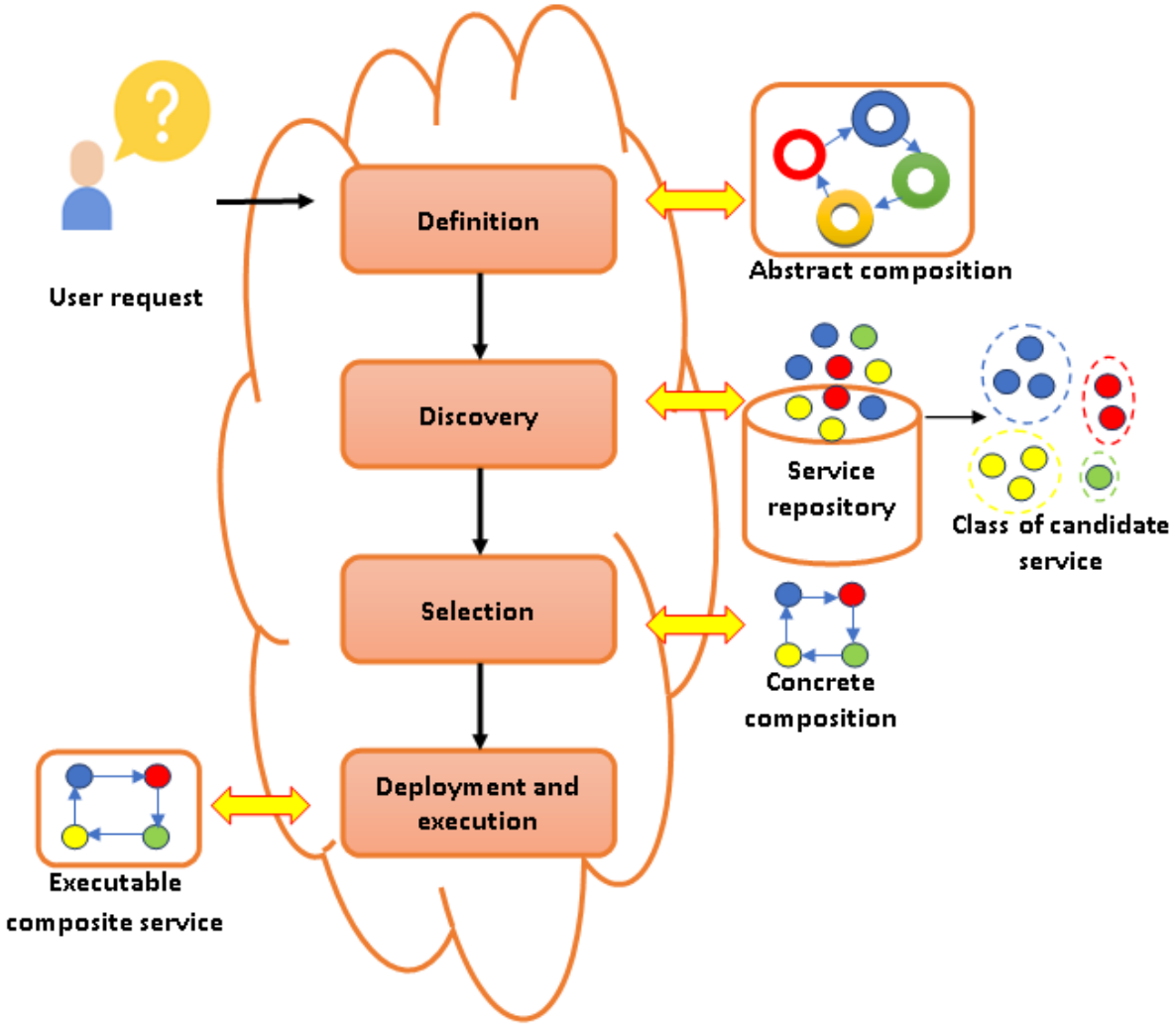


Figure I.6: Service composition life-cycle.

- **Definition phase.** Focuses on identifying the desired functionalities of the service resulting from the composition. Users or service requesters specify their functional and non-functional requirements, including details about the expected service behavior, QoS parameters, and exception handling. These requirements are then translated into an abstract description that serves as a blueprint for the composite service. This abstract description defines a sequence of description, control, and data flows among them, along with any exceptional scenarios that need to be managed. The abstract description also specifies how individual services should interact and ensures that their integration aligns with the desired QoS requirements. To support this process, advanced techniques such as ontologies and automated tools are often employed to decompose user's requests into structured components, ensuring that the specifications are both comprehensive and aligned with the capabilities of available services.
- **Discovery phase.** The process begins with a search of the service repository to discover candidate services that match the functional requirements defined during the definition phase. This discovery is typically based on the syntactic or semantic de-

scriptions provided in service metadata. The result of this discovery process is a list of candidate services, each capable of fulfilling similar functionalities but differing in their non-functional properties (such as QoS attributes).

- **Selection phase.** After identifying a set of candidate services for each abstract service in the composition plan, the selection phase evaluates these candidates based on their non-functional properties. The goal is to choose the best concrete service for each required functionality, ensuring that the resulting composite service satisfies both functional and non-functional requirements.
- **Deployment and execution phase.** The selected services are linked and instantiated to transform the composition plan into an executable composite service, which is then invoked to meet the user's needs. Once deployed, the service passes to execution, where an instance of the composite service is created, and the execution is orchestrated by invoking the individual component services in the sequence specified by the abstract description. In parallel, the runtime engine monitors execution, tracks progress, and collects performance metrics. The engine also manages exceptions by detecting and addressing errors, such as service failures or unexpected behavior, to maintain uninterrupted operation. Additionally, the engine may dynamically adapt the composite service by reconfiguring the abstract description or substituting components in response to environmental changes or evolving requirements.

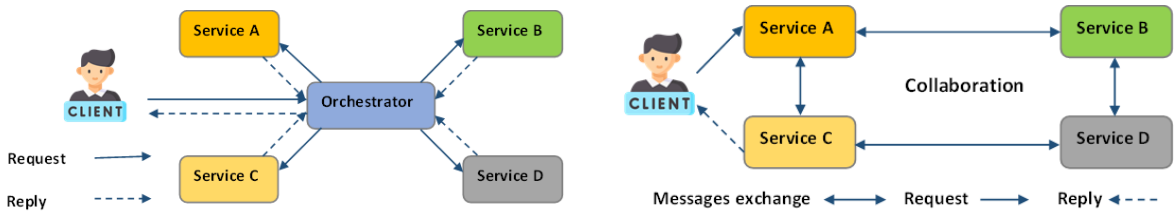
5.3.3 Typology of composition methods

The diversity of application areas and requirements has led to the development of various composition methods, each designed to meet specific challenges and objectives. These methods can be classified according to key dimensions such as automation level, architectural approach, execution environment, and optimization objectives.

5.3.3.1 Classification based on the architectural model

The architectural model determines how the atomic services are coordinated, monitored, and executed within a composite service. The two main approaches are orchestration and choreography, which are based on workflow and share the common goal of coordinating service workflows. The latter involves several autonomous services, each with a well-defined role and a specific relationship with others. However, these two approaches differ in how they coordinate the services involved in the composition process. Orchestration and choreography, therefore, refer to the description of the interactions among a set of component services to achieve a specific goal related to the user's needs (see Figure I.7).

The orchestration is a centralized method of service composition in which a central entity, called an *orchestrator* or *execution engine*, coordinates the instantiation and execution of abstract services according to the concrete available services [Zribi, 2014]. This process is based on a predefined interaction model that manages exchanges between services following a specific control logic [Benatallah et al., 2005]. The orchestrator manages



(a) Orchestration. (b) Choreography.

Figure I.7: Classification of composition approaches based on the architectural model

the sequence of services, whether atomic or composite, by centralizing their invocation and ensuring their integration into a coherent definition.

The choreography is a decentralized and dynamic service composition approach without a central entity controlling the interactions. The services collaborate as autonomous entities, each playing a specific role and managing its own interactions. This model is similar to a peer-to-peer (P2P) architecture in which services coordinate to achieve a common goal. Unlike orchestration, choreography does not rely on predefined execution sequences but allows services to dynamically choose their subsequent interactions, progressively building the composition [Barros et al., 2006, Peltz, 2003].

5.3.3.2 Classification based on the automation level

Depending on the degree of user involvement in defining the composition plan, three types of service composition are distinguished: manual, semi-automatic, and automatic.

- **Manual composition:** This technique depends on the expertise of a user in defining and creating the schema of the service composition. In this approach, the user takes a central role by manually specifying the sequence of invocation and execution for the component services. The process starts with a user request specifying the required functionalities. A service composition plan is then designed to address these requirements. Once the plan is completed, it is submitted to an execution engine to execute the composition process.
- **Semi-automatic composition :** This approach uses graphical tools to assist users through the service composition process. These tools offer a range of graphical operators that help users design composite services step by step and provide recommendations for selecting the appropriate services to include in the composition. After the composite service definition, the resulting composition plan is submitted to an execution engine for implementation. Additionally, some tools support schema storage, allowing for easier reuse in future compositions.
- **Automatic composition :** In this type of composition, the user specifies their requirements as a service request. The system then handles the entire composition process automatically and transparently, without requiring further user intervention.

5.3.3.3 Classification based on execution environment

The execution environment defines how and when the composition process is performed. In this context, there are two main types of composition: static composition and dynamic composition [Yachir, 2014].

- **Static composition:** This kind of composition refers to a process defined during the design and deployment phases of the system architecture. The services to be composed are pre-identified, selected, interconnected, and deployed by the designer. This approach results in fixed compositions, with services and their connections predefined. The static composition is cost-effective when the system and services remain relatively stable. In practice, static service composition approaches are widely adopted in the industry.
- **Dynamic composition:** Unlike static composition, dynamic composition occurs at runtime in response to a specific request. The candidate services are discovered upon receiving the user's request and then interconnected based on the user's requirements to create a composite service. The composition process is executed dynamically, relying on the services' availability at that moment. This approach is well-suited to dynamic environments where available services frequently change, and user expectations are variable and personalized. However, implementing dynamic composition is challenging due to changes in the user's context and service conditions in such environments.

6 Challenges of service composition in CPSS and IoT

The IoT and CPSS environments pose significant challenges, particularly regarding service composition. In this section, we define the main challenges associated with service composition in IoT and CPSS environments.

6.1 Energy constraints

In the context of IoT and CPSS environments, a large number of smart objects is interconnected to provide a variety of functionalities, such as monitoring environmental conditions, processing collected information, and transmitting data to other systems or users. These services, designed to meet the specific needs of the user or to achieve particular business objectives, are mainly deployed in devices (e,g., smartphones, tablets) that are often limited in terms of resources, including energy, storage capacity, computing power, and communication interfaces [Yang and Li, 2014]. Usually powered by batteries that are difficult to replace or recharge, these devices must efficiently manage their energy consumption to extend their autonomy. Consequently, when designing applications for these environments, it is essential to minimize service energy consumption to ensure their availability as long as possible. A typical example is the fitness tracker: a compact, lightweight device that monitors physical activity, heart rate, and sleep patterns. Despite its small size, this device constantly collects data, synchronizes with smartphone

applications, and maintains extended autonomy. This efficiency is achieved by optimizing energy consumption and focusing on specific health-related functions, rather than running complex applications.

6.2 Mobility

The dynamic nature of IoT and CPSS environments, characterized by frequent appearance and disappearance of devices and services, requires continuous coordination between devices and services to ensure seamless service execution. The user's mobility can lead to frequent changes in network topology, affecting the availability and reliability of services. For example, when devices move, they may leave network coverage areas, causing services to become temporarily unavailable or permanently inaccessible. This dynamic behavior requires continuous monitoring and adaptation of the service composition to maintain the desired QoS. In addition, mobile devices often operate under strict energy constraints. The intensive use of battery power can cause devices to shut down or degrade service, as some functionalities may be restricted to conserve energy, further impacting service availability. Therefore, energy-efficient service composition approaches are essential to extend device lifetime and ensure consistent service delivery. Incorporating energy consumption models into the service composition process can help in selecting services that balance functionality with energy efficiency, thereby enhancing overall energy efficiency and prolonging device lifetime .

6.3 Scalability

The scalability in the IoT and CPSS environments refers to a system's ability to handle a progressive increase in the number of users or services without a decrease in performance or efficiency [Hamzei and Navimipour, 2018]. With the rapid rise of hardware technologies, the number of connected devices has reached around 18.8 billion by the end of 2024[4] and is projected to increase further to around 27 billion by 2025. This rapid expansion makes scalability crucial for the realization of SOC systems [Sarkar et al., 2014]. In these environments, workloads are typically measured in terms of the number of processed requests [Hamzei and Navimipour, 2018] or the volume of generated data streams [Sun and Ansari, 2016]. Scalable approaches aim to improve QoS and energy efficiency by maintaining constant performance levels despite high workloads. To achieve this, key parameters must be optimized, including energy consumption, cost, latency, and response time, while improving throughput and reliability. In this context, energy efficiency is a critical aspect in IoT and CPSS environments due to the limited resources of connected devices, while QoS remains essential to ensure the overall effectiveness and efficiency of services

[4] According to the *State of IoT 2024* report by IoT Analytics: https://iot-analytics.com/number-connected-iot-devices/.

6.4 Interoperability

Interoperability is a critical factor for seamless data exchange and resource sharing among users, services, and smart objects within the IoT and CPPS. This heterogeneity is apparent at several levels, such as the diversity of hardware architectures, the incompatibility of communication infrastructures, and the variety of software platforms [Asghari et al., 2018]. To address this challenge, software technologies such as middleware have been developed to provide a high level of abstraction that conceals disparities between communication technologies, hardware resources, operating systems, and programming languages. However, the proliferation of middleware introduces a new form of heterogeneity at the interaction protocol level. Standards such as Universal Plug-and-Play (UPnP), Devices Profile for Web Services (DPWS), and Constrained Application Protocol (COAP) have been proposed to harmonize these interactions and enhance interoperability [Yachir, 2014]. In the context of CPSS, interoperability is particularly significant given the integration of human and social dimensions into cyber-physical systems. The middleware must manage environmental dynamics, including the appearance and disappearance of devices and services, the integration of new applications or devices, and user mobility. Furthermore, the middleware should provide mechanisms that facilitate the creation, deployment, and utilization of new applications through service composition methods.

6.5 Security and privacy

The integration of multiple information sources, such as sensors and cameras, raises major privacy and security concerns in IoT and CPSS environments. Data protection and access control policies must be rigorously defined when designing intelligent pervasive systems, enabling users to interact securely with services without compromising their privacy. The confidentiality aspect is another major concern in these environments, with an emphasis on protecting the preferences of users and service providers. The pervasive use of diversified services can, intentionally or unintentionally, lead to the disclosure of personal or sensitive information to unauthorized third parties. Consequently, it is crucial to develop security models that respect the privacy preferences of users and service providers, to protect critical information during the service composition process [Asghari et al., 2018].

7 Conclusion

In this chapter, we have explored the evolution of computing paradigms, from SOC to CPSS via the IoT, which use modular services to develop complex, adaptive software applications. The SOC is based on SOA, which enables services to be organized and integrated in a structured way. The IoT extends this concept by connecting several objects equipped with sensors and software, facilitating data exchange and interaction between devices and systems via the Internet. CPSS integrates cyber and physical dimensions by adding a social component, creating intelligent ecosystems that can detect the environ-

ment, analyze user requirements, and provide adapted services. The service composition is a central process that enables the creation of added-value services from basic ones. This process involves the careful selection of services considering their functional and non-functional properties, to meet specific user's requirements. Service composition in IoT and CPSS environments presents various challenges, such as energy constraints, mobility of components, scalability, interoperability between heterogeneous devices, and the concerns associated with security and confidentiality. These challenges require advanced mechanisms to ensure efficient and secure service composition.

Chapter II

State of the art of service composition approaches

1 Introduction

The service composition issue is a particularly interesting topic that attracts great interest within the research community, as shown by the large number of studies devoted to this topic in the literature. Unlike conventional approaches, which often focus on specific aspects such as QoS or functional performance, the emergence of IoT and CPSS has highlighted the need to simultaneously consider multiple criteria to satisfy the growing demands of applications. In particular, QoS, energy, and mobility are emerging as crucial criteria, requiring a re-examination and enhancement of existing service composition approaches.

This chapter aims to provide a classification of existing service composition approaches by analyzing the main research contributions in the literature. This classification distinguishes between conventional QoS and energy-aware approaches and mobility-aware approaches based on the resolution methods they employ. In the second part, we compare these approaches with respect to their ability to address key requirements in service-oriented environments, such as scalability, energy efficiency, and factors specific to dynamic and mobile contexts. The goal is to assess their relevance and effectiveness in different scenarios. Finally, a critical discussion of the reviewed approaches is provided to highlight their limitations and to emphasize the motivations of the contributions proposed in this thesis.

2 Taxonomy of service composition approaches

The service composition constitutes a key mechanism for the development and deployment of flexible and scalable applications. This process enables the creation of more sophisticated services by combining pre-existing ones, generating added-value functionalities that each basic service could not provide individually. This process relies on the need to respond to user demands by orchestrating a set of services to form applications

tailored to specific user's requirements. During the optimization process, users can impose global QoS constraints on the resulting composition, which can take various forms, such as keeping the cost below a given threshold or ensuring that the total energy consumption remains within a predefined range. Since the service composition is classified as an NP-hard (non-deterministic polynomial) problem, the challenge is to achieve a sub-optimal composition that satisfies these constraints within a reasonable computation time [Ardagna and Pernici, 2007]. As outlined in Chapter I, the key challenges of service composition in IoT and CPSS environments primarily concern mobility, energy efficiency, and QoS guarantees. These dimensions not only determine the feasibility of compositions in large-scale and dynamic contexts but also provide a foundation for classifying existing approaches. Accordingly, the literature has widely explored the service composition problem and several approaches have been proposed to deal with this issue. These approaches can be divided into two distinct categories: (i) conventional service composition approaches that optimize QoS or/and energy without explicitly addressing mobility, and (ii) mobility-aware service composition approaches, which explicitly integrate user and service mobility into the composition process (see Figure II.1).

2.1 Conventional QoS and energy-aware service composition approaches

The service composition has given rise to several conventional approaches that aim to optimize QoS and energy aspects. These approaches can be divided into two categories. QoS-aware service composition approaches focus on satisfying the user-defined QoS constraints, employing a variety of resolution methods including integer linear programming, decomposition of QoS constraints, Pareto dominance, reinforcement learning, and bio-inspired meta-heuristics. Energy-aware service composition approaches aim to minimize energy consumption either in combination with QoS optimization or, in some cases, by focusing exclusively on reducing the energy usage of the service.

2.1.1 QoS-aware service composition approaches

Most studies of service composition have focused on QoS parameters (e.g., response time, availability, reliability, cost) to ensure that the composite service meets user requirements and respects the predefined QoS constraints. Existing approaches aim to optimize these QoS attributes through various techniques, including exact, heuristic, and meta-heuristic algorithms, as well as machine learning methods.

2.1.1.1 Linear programming-based approaches

Linear Programming-based approaches focus on formulating the service composition problem as a mathematical optimization problem, where the objective is to maximize or minimize a linear combination of QoS attributes subject to constraints

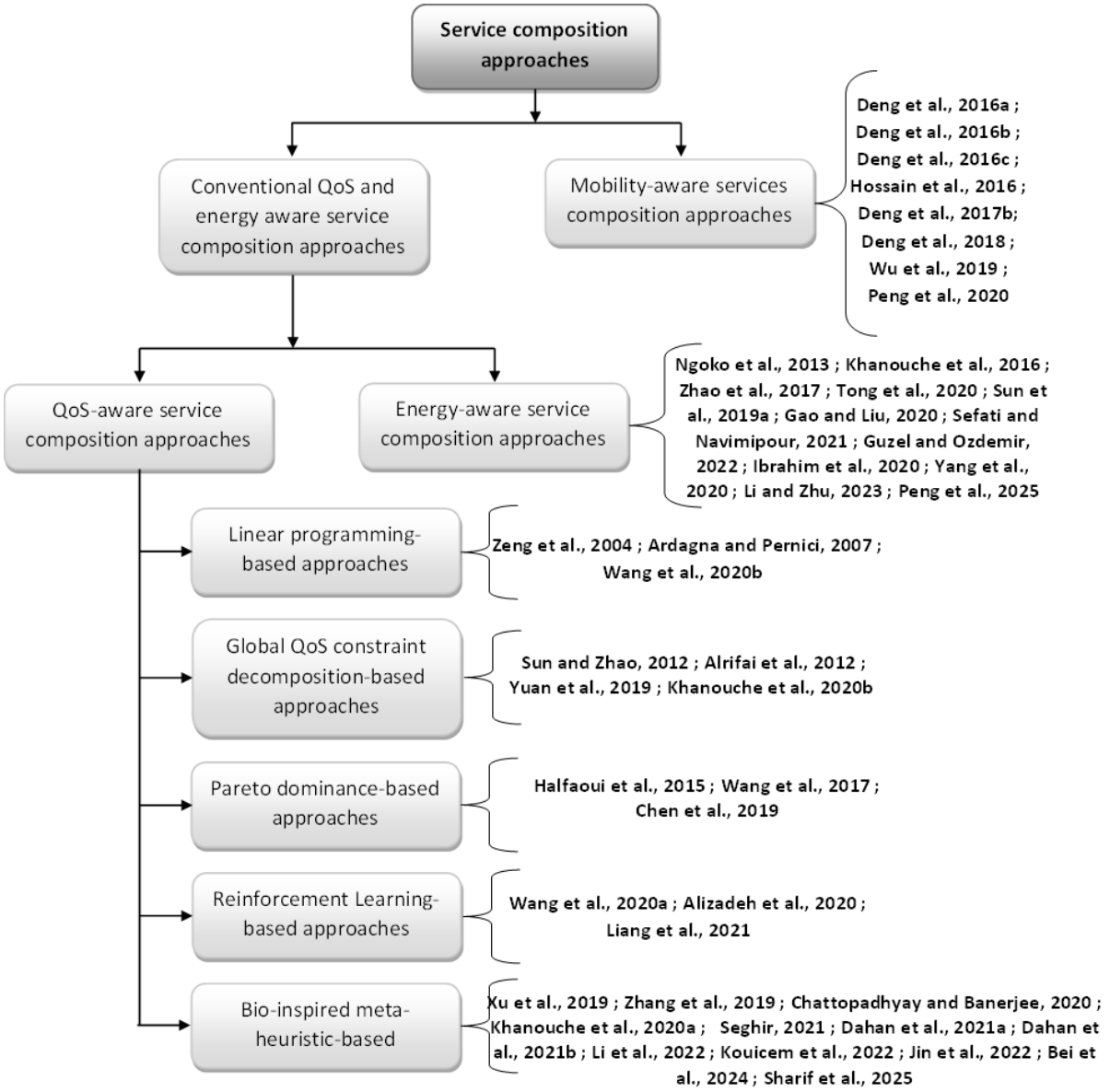


Figure II.1: Taxonomy of existing service composition approaches.

[Zeng et al., 2004, Ardagna and Pernici, 2007, Wang et al., 2020b].

In [Zeng et al., 2004], local and global optimization approaches are exploited to solve the optimal QoS-aware service composition. The first approach focuses on local service selection by optimizing QoS for each service class. For each abstract service, a Multiple Criteria Decision Making (MCDM) [Hwang and Yoon, 1981] technique is used to select the optimal service based on its QoS attributes and the user's preferences. This selection process involves calculating a QoS score that takes into account the user-assigned weights for each criterion and the defined constraints. However, constraints can only be expressed on an individual service class. The concrete service that satisfies local QoS constraints and has the highest score is selected for the composition process. Although this approach has a polynomial computation time, it does not consider global QoS in the optimization

process. In the second approach, the service selection problem is formulated as an Integer Linear Programming (ILP) problem with a set of variables, an objective function to be maximized, and QoS constraints to be satisfied. This approach aims to ensure optimal global QoS but, in some cases, can lead to high computation times due to the exponential complexity of the service composition problem. In addition, to maintain optimal QoS for the composite service, this approach proposes re-selection in the case of service failure or degradation of their QoS during composition execution.

The local selection approach is inadequate for service composition with global QoS constraints, whereas the global selection approach using ILP achieves acceptable computation times only when the number of services is small. Since the time complexity of ILP-based service composition approaches increases exponentially with the number of services, this method is impractical for large-scale service composition or environments where QoS values frequently fluctuate.

In [Ardagna and Pernici, 2007], the services selection issue is addressed considering severe QoS constraints for the execution of the composite service, i.e., limited resources that make the problem close to unfeasibility conditions (e.g., bounded execution time). More specifically, the ILP method is enhanced by using a QoS constraints relaxation strategy to handle cases where a feasible composition cannot be found, and by applying a loop peeling technique to deal with complexity arising from loop structures in the composition. The loop peeling technique is a loop unrolling in which loop iterations are transformed into a sequence of conditional branches, each evaluating whether the loop l should continue with the next iteration according to the probability distribution p_h^l. The probability distribution $\{p_0^l, ..., p_{NI}^l\}$ of the number of loop iterations is specified such as $\sum_{h=0}^{NI^l} p_h^l = 1$, where p_h^l indicates the probability that the loop l will be executed in h iterations. This technique aims to reduce the computational complexity of the service composition process and manage the execution time more efficiently. Additionally, multiple execution paths can be derived from the abstract composite service. Each execution path ep_k has an associated execution probability $freq_k$, which is the product of the probabilities of its constituent branches. The service selection is then formulated as a Mixed Integer Linear Programming (MILP) problem to maximize the composite service's aggregated QoS, considering all possible execution paths ep_k and their respective execution probabilities $freq_k$. To calculate the aggregated QoS of a composite service Y associated with an execution path ep_k, the Simple Additive Weighting (SAW) technique [Loeckx, 1981] is used to assign a score $score_k(Y)$ based on the weighted sum of QoS attributes. In scenarios where a feasible composition is not found, a negotiation process is initiated between the user and the service provider to relax some QoS constraints following service-oriented negotiation algorithms [Faratin et al., 1998, Comuzzi and Pernici, 2005]. This involves an alternating sequence of offers and counteroffers until an agreement is reached or the negotiation is terminated due to a deadline expiration.

The enhanced ILP method with loop peeling and QoS constraint relaxation provides a more effective approach to service composition under severe QoS constraints, especially when dealing with complex loop structures. However, the computational complexity of this approach can become prohibitive as the number of services and their interactions increase. This can lead to significant delays, especially in real-time applications. Additionally, this approach relies on the accuracy of the predefined probability distributions for loop iterations, which may not always accurately reflect runtime behavior. Finally, the QoS constraints negotiation process can introduce additional complexity and potential delays in reaching an acceptable service composition.

In [Wang et al., 2020b], a service composition algorithm based on the concept of cultural distance is proposed to enhance the reliability of the composite service and user satisfaction in a CPSS environment. To identify services that best meet user needs, cultural distance is introduced as a new QoS metric to quantify differences between the cultural profiles of the host and target countries across multiple dimensions, enabling service composition in CPSS to better align with the user's diversity [Kogut and Singh, 1988]. This metric is expressed as $CD_j = \frac{1}{m}\sum_{i=1}^{m}\frac{(I_{ij}-I_{ik})^2}{V_i}$, where m is the number of cultural dimensions, I_{ij} $(i = 1, \ldots, m)$ is the score of the i^{th} dimension for country j, k is the host country, and V_i is the variance of the i^{th} dimension. The index j ranges from 1 to n $(j \neq k)$, where n is the total number of countries. First, the degree of preference is calculated by evaluating each service's suitability relative to the user's cultural distance. The degree of preference D_{us} is defined as the difference between the cultural distance of the user CD_u and the cultural distance of the service CD_s, given by D_{us} =-$|CD_u$-$CD_s|$. A smaller cultural difference results in a higher degree of preference. Services with low preference degrees are dropped to improve user satisfaction and reduce composition time. Second, the calculation of the preference vector involves creating a preference vector $w = (w_1, ..., w_k, ..., w_r)$ that represents the user's preferences for QoS attributes such as $0 < w_k < 1$, and $\sum_{k=1}^{r} x_k = 1$. The value r represents the number of QoS attributes, and $w_k = \frac{\sum_{i=1}^{n} q_k(ds_i)}{\sum_{k=1}^{r}\sum_{i=1}^{n} q_k(ds_i)}$ is the preference value of the k^{th} QoS attribute, n is the number of abstract services and $ds_i = \arg\max \{D_{us}\}/s \in S_i$ represents the service with the highest preference degree among the candidate services in S_i, where S_i is the set of services corresponding to the i^{th} abstract service. Finally, a 0–1 Mixed-Integer Programming (MIP) algorithm is used in the service composition phase to identify the most appropriate service for each abstract service in the composition under global QoS constraints.

This approach enhances user satisfaction by incorporating cultural distance as a QoS metric, thus providing services that better align with users' cultural backgrounds. However, the focus on cultural distance may include other critical QoS attributes such as response time, cost, and availability. Furthermore, the computational complexity of the

$0-1$ MIP algorithm further reduces its practical application in real-time scenarios.

2.1.1.2 QoS constraint decomposition-based approaches

Several service composition approaches adopt a strategy to decompose global QoS constraints into local constraints [Sun and Zhao, 2012, Alrifai et al., 2012, Yuan et al., 2019, Khanouche et al., 2020b]. In these approaches, the range of each QoS attribute of candidate services is divided into a set of discrete values known as quality levels. The global QoS constraints are then decomposed into local constraints for each abstract service by solving an optimal-quality-level problem. The resulting local constraints are used to select the candidate service for each abstract service in the composition.

In [Alrifai et al., 2012], a two phases QoS-aware service composition algorithm that combines global optimization and local selection is proposed. The first phase uses the MIP algorithm to optimally decompose global QoS constraints into local constraints for each abstract service by introducing the concept of local quality levels. For a QoS attribute q_k, local quality levels are discrete values representing the candidate services within the same abstract service AS_j. These levels are determined through a three-step procedure. The candidate services of class AS_j are first sorted by the QoS attribute value q_k. The minimum and maximum values of the attribute q_k are then directly added to the set of local quality levels. The remaining candidate services are finally divided into $d-2$ equal subsets, and one candidate service is randomly selected from each subset to add its q_k value to the set of local quality levels. Thus, d local quality levels are obtained and used as local constraints for the corresponding QoS attributes. For each global QoS constraint, the decomposition involves selecting a local quality level q_{jk} from each abstract service AS_j. This problem is formulated as a Mixed-ILP problem, aiming to find a set of local constraints for each abstract service that covers as many candidate services as possible without violating any global constraints. A binary decision variable x_{jk}^z is associated with each local quality level q_{jk}^z, where $x_{jk}^z = 1$ if q_{jk}^z is selected as the local constraint for the QoS attribute q_k in the abstract service AS_j, and $x_{jk}^z = 0$ otherwise. To ensure that global constraints are satisfied, additional constraints are added to the MIP model, and the non-linear aggregation functions are normalized to match this model. In the second phase, a local selection is performed for each abstract service to choose the best candidate service in terms of QoS. The obtained local constraints are used as upper bounds for the QoS values of candidate services and services that do not meet these bounds are filtered out from the selection process.

This approach satisfies global QoS requirements by decomposing them into local constraints and selecting candidate services accordingly. However, the effectiveness of this approach depends on the accuracy of the derived local constraints. If they are too restrictive or poorly aligned with global QoS requirements, the selection may exclude suitable candidates or even fail to produce a feasible composition.

In [Yuan et al., 2019], a QoS constraints decomposition-based approach is proposed in the context of dynamic services selection. Initially, each global QoS constraint C_i is decomposed into n local constraints $\{c_1, ..., c_n\}$, where n is the number of abstract services. These local constraints are designed to satisfy the global constraints, while remaining broad enough to include all candidate services. For each QoS attribute of the candidate services within an abstract service, quality levels are initialized by dividing the attribute's value range into discrete quality values. The k^{th} QoS attribute value of the m^{th} candidate service of abstract service S_i is denoted as q_{ik}^m. The d^{th} quality level of the k^{th} attribute for a candidate service of abstract service S_i is defined as

$$L_{ik}^d = L_{ik}^{d-1} + d * \Delta \tag{II.1}$$

where $d \in [1, p_k]$ and p_k is the number of quality levels for the k^{th} QoS attribute. The step size Δ is computed as :

$$\Delta = \frac{q_{ik}^{max} - q_{ik}^{min}}{p_k} \tag{II.2}$$

where q_{ik}^{max} and q_{ik}^{min} are, respectively, the maximum and minimum values of the k^{th} QoS attribute for abstract service S_i, and $q_{ik}^{min} = L_{ik}^0$.

A three-steps Adaptive Adjustment method for the number of Quality Levels (AAQL) based on fuzzy logic is proposed to adapt the number of quality levels automatically. The first step, fuzzification, transforms QoS values of candidate services into fuzzy sets using membership functions. For instance, a response time of $200ms$ might belong to the fuzzy sets $Fast$ (0.7) and $Medium$ (0.3). Formally, for any QoS value u of a service, $f(u)$ represents the degree to which u belongs to a given fuzzy set. The second step, fuzzy inference, evaluates the fuzzified QoS values using the fuzzy rule base, expressed as IF–THEN statements. For example: $IF\ Cost = Expensive\ AND\ ResponseTime = Slow\ THEN\ Rank = Low$. These rules determine how combinations of QoS attributes affect service rankings. Finally, defuzzification converts the fuzzy output into an accurate value. The AAQL method automatically adjusts the number of quality levels based on user preferences, utility function values, and time cost. To address the combinatorial nature of QoS constraint decomposition, a Cultural Genetic Algorithm (CGA) is employed to find the near-to-optimal decomposition of global QoS constraints. These latter are then used to select candidate services from each abstract service in a parallel manner. The number of candidate services is reduced based on local constraints, and the service with the highest utility belonging to the reduced set is selected as the final local component service.

This approach uses fuzzy logic to adaptively adjust quality levels based on user preferences, utility values, and time cost. This adds a layer of personalization and ensures stable performance even with varying QoS attribute values. However, reliance on fuzzy logic and the CGA method results in high computational complexity for large-scale

service compositions. The process of initializing and adjusting quality levels requires careful configuration and fine-tuning, which may impact the robustness and consistency of the service selection process. Additionally, this approach may be unsuitable when the number of fuzzy sets and rules must be expanded to improve inference capabilities.

In [Khanouche et al., 2020b], a QoS-aware service composition approach is introduced to enhance the composition feasibility and reduce the search space. This composition approach consists of three main phases: decomposition of global QoS constraints, filtering of candidate services, and progressive composition. Initially, the user-specified global QoS constraints are decomposed into local constraints assigned to each abstract service. This decomposition depends on the type of attribute (e.g., additive, multiplicative). In the filtering phase, candidate services are filtered in two steps. First, candidate services are preselected based on QoS flexibility, with threshold values derived from local constraints and adjusted according to a flexibility factor. Candidate services that do not meet the obtained threshold are removed. Second, the QoS-based Pareto dominance relationship is used to prune dominated candidate services, resulting in the set of Pareto optimal candidate services ($POCS$) for each abstract service. In the third phase, a progressive composition strategy is employed where candidate services in ($POCS$) are sorted according to their utility values, and the partial composition algorithm (PCA) [Yu and Bouguettaya, 2011] is recursively applied to find the set of Pareto optimal compositions (POC). Specifically, ($POCS(AS^1)$) is combined with ($POCS(AS^2)$) using the PCA algorithm to find the first partial set of POCS, which is then combined with ($POCS(AS^3)$), and so on, until obtaining the final set of POC.

2.1.1.3 Pareto dominance-based approaches

Pareto dominance-based approaches focus on treating the service composition problem as a multi-objective optimization problem, where the goal is to simultaneously optimize the concurrent QoS attributes without aggregating them into a single objective function [Halfaoui et al., 2015, Wang et al., 2017, Chen et al., 2019]. These approaches identify a set of non-dominated (Pareto-optimal) solutions that represent the best compromises between QoS criteria.

In [Halfaoui et al., 2015], a fuzzy Pareto dominance-based service selection approach is proposed to sort the candidate services of every abstract service and select the k best services for the composition process. The Fuzzy Dominated Score $\mu_{\epsilon,\lambda}(u, v)$ is defined using a monotone comparison function that expresses the degree to which a value u is dominated by another value v. It is given by:

$$\mu_{\epsilon,\lambda}(u,v) = \begin{cases} 0, & \text{if } u - v > \epsilon, \\ \dfrac{|u - v - \epsilon|}{\lambda + \epsilon}, & \text{if } \lambda + \epsilon \leq (u - v) < \epsilon, \\ 1, & \text{if } u - v < \lambda + \epsilon, \end{cases} \tag{II.3}$$

where $u = q_k(s_i)$ and $v = q_k(s_j)$ represent the values of the k^{th} QoS attribute for services s_i and s_j, respectively. The parameters ϵ and λ are defined within the range $[-1, 0]$, under the condition $\epsilon + \lambda \geq -1$. The Fuzzy-Dominated score $FDet(Si, Sj)$ is then defined to express the degree to which the service S_i is dominated by the service S_j. Formally :

$$FDet(Si, Sj) = \frac{1}{d} \sum_{k=1}^{d} \mu_{\epsilon\lambda}(q_k(s_i), q_k(s_j)) \tag{II.4}$$

To rank the candidate services within each abstract service, pairwise comparisons are performed among all candidate services of the same abstract service according to an averaged fuzzy dominated score denoted as :

$$AFDetS(S_i) = \frac{1}{\mid S \mid -1} \sum_{j=1, i \neq j}^{n} FDet(S_i, S_j) \tag{II.5}$$

This score allows for establishing an ascending order of candidate services based on their QoS attributes. The candidate service with the lowest average fuzzy dominance score is considered as the best service. The top k candidate services are consequently selected for the composition process.

This approach effectively reduces the composition search space by identifying the top k services, thus improving the efficiency of the selection process. However, a key limitation of this approach lies in its scalability, as it requires extensive pairwise comparisons, resulting in high computational complexity. This complexity can become significant, particularly when dealing with large sets of candidate services, limiting the applicability of this approach in dynamic and large-scale environments.

In [Wang et al., 2017], a three-phase reliable service composition approach is proposed in the context of cyber-physical and social systems. The first phase aims to reduce the search space required for service composition by identifying and retaining only the candidate services that are not dominated by any other candidate services based on their QoS attributes. The Skyline algorithm based on the dominance relationship is used to filter out redundant candidate services and determine which services are as good or better in all QoS attributes and better in at least one QoS attribute. The remaining dominant candidate services form the Pareto optimal set, from which the service composition is determined. The second phase aims to calculate QoS fluctuations, using the coefficient of variation to transform QoS values into a qualitative measure representing the degree of QoS uncertainty for a candidate service. Based on this qualitative measure, candidate

services are ranked using a utility function that retains services with low QoS values variation and filters out those having high QoS values variation. The use of the coefficient of variation ensures the reliability of service composition and further reduces the search space of composition. The third phase solves the service composition problem using an ILP method to select the most reliable candidate services while ensuring user efficiency in CPSSs.

The QoS fluctuation computation used in this approach enhances the reliability of the service selection process by keeping only candidate services with low variation in their QoS values, thus favoring stable services and minimizing uncertainty. However, reliance on the Skyline algorithm may result in dropping out some promising services that do not strictly meet the non-dominance criteria but could still offer an acceptable QoS.

In [Chen et al., 2019], a Dependency-Aware Service Composition (DASC) approach is proposed to find Pareto-optimal compositions efficiently. This approach uses the dependency information between QoS attribute values and involves two main steps: (1) pre-processing of candidate services and (2) service composition considering QoS attributes dependencies. During the pre-processing step, a pruning is used to reduce the search space for each abstract service belonging to the composition. A service S_i is correlated with a service S_j, considering their QoS attribute values, if their combined use in a composition results in a QoS gain compared to their individual use. Candidate services without correlation and those that are dominated by other services are eliminated to reduce the number of candidate services in each abstract service. Despite this pruning, the composition search space containing non-dominated candidate services may still large. To address this problem, the DASC approach relaxes the optimization objectives by searching approximate Pareto-optimal compositions rather than exact ones. This involves ordering the service composition solution using a multi-objective approach by organizing them into hierarchical layers. Each layer contains a set of Pareto-optimal compositions that remain after removing the compositions from the previous layers. For instance, Layer 1 contains the compositions that are not dominated by any others, i.e., the Pareto-optimal compositions. Once these are identified and excluded, the next group of non-dominated compositions forms Layer 2, and the process continues until all compositions are classified. This layered structure provides a hierarchical organization of the compositions, facilitating the identification and selection of the most promising ones. In the second step, the DASC approach uses Vector Ordinal Optimization (VOO) techniques to estimate the performance of compositions, calculate the required alignment probability, and determine the number of layers (s) needed to guarantee that at least k best compositions are obtained with high probability. The VOO process first uses a simplified model to estimate the performance of candidate compositions and organizes them into multiple layers. A subset of the most promising layers is then selected for further evaluation. Considering the dependencies among candidate services, the exact

QoS values of the compositions in these layers are computed using a brute-force method, and the k best compositions are finally identified as near-to-optimal compositions.

The DASC approach improves service composition by modeling the dependencies between QoS attributes. However, the search space remains large even after eliminating uncorrelated candidate services. This problem results from the exponential size of the composition space, particularly in scenarios with high candidate services per abstract service.

2.1.1.4 Reinforcement Learning-based approaches

Reinforcement Learning-based approaches focus on modeling the service composition problem as a sequential decision-making process, where an agent learns optimal composition strategies by interacting with the environment to maximize a cumulative reward based on QoS objectives. Several studies have employed this approach to achieve adaptive and efficient service compositions [Wang et al., 2020a, Alizadeh et al., 2020, Liang et al., 2021].

In [Wang et al., 2020a], a service composition approach that combines reinforcement learning with skyline computing is proposed to improve the efficiency and effectiveness of Web service composition. The Q-learning method learns optimal policies for selecting services through trial-and-error interactions in a dynamic environment. At the same time, the Branch-and-Bound skyline algorithm filters out non-optimal services by considering multiple QoS attributes simultaneously. This approach models Web service composition as a Markov Decision Process (WSC-MDP) represented by a 6-tuple $(S, s_0, s_\Gamma, A, T, R)$, where S is the set of states, s_0 is the initial state, s_Γ is the set of terminal states, $A(S)$ is the set of services executable in state s, T is the state transition probability function, and R is the reward function. In the Q-learning algorithm, the agent aims to learn a policy that maximizes the cumulative reward by updating a Q-value function $Q(s, a)$, which represents the expected utility of taking action a in state s. The Q-value is updated using the following formula:

$$Q(s,a) \longleftarrow Q(s,a) + \alpha[r + \gamma max_{\alpha'} Q(s',a') - Q(s,a)] \tag{II.6}$$

where α is the learning rate, r is the immediate reward received after taking action a, γ is the discount factor, and s' is the next state. The Q-learning algorithm updates the Q-value function $Q(s, a)$ until the cumulative reward converges to an optimal value, ensuring optimal or near-to-optimal service composition. Skyline computing is applied to reduce the action space by filtering out non-optimal services based on QoS attributes, allowing Q-learning to focus on the most promising candidate services.

The combination of the Q-learning approach with skyline computing enables an adaptive and efficient service composition process to ensure optimal or near-to-optimal

service composition. However, tuning hyperparameters, such as the learning rate α and discount factor γ is crucial for optimal performance but can be challenging and time-consuming. Moreover, although skyline computing effectively reduces the search space, a large number of services may remain to be evaluated, especially in scenarios involving high dimensionality of QoS attributes and large number of services.

In [Alizadeh et al., 2020], a reinforcement learning-based service composition approach is proposed for Web-of-Things environments without prior knowledge of user preferences regarding QoS attributes. The problem is formulated as a Vector-Valued Markov Decision Process (VMDP) defined by the tuple $(T, S, A, P_t(.|s,a), \overline{r}_t)$, where T is the set of decision time steps, S is the finite set of states that represents abstract services, $A(s)$ is the set of actions available in state s, representing the concrete services, $P_t(s'|s,a)$ is the state transition probability distribution that refers to the probability of moving from state s to state s' when action a is taken at time t, and $\overline{r}_t$ is the vector-valued reward function including multiple QoS attributes such as response time, availability, and cost. The objective is to determine the optimal service combination that maximizes these QoS attributes. To achieve this, an interactive value iteration algorithm (IVI-SC) starts by assigning a zero vector to the terminal states. For each abstract service at each time step, the algorithm evaluates all possible concrete services using a comparison method that includes three steps. The Pareto dominance first checks if a service is better than another across all QoS attributes. When Pareto dominance doesn't yield a clear preference, the K-dominance checks whether this service is better for any possible weight combination of QoS attributes within the defined polytope W. In cases where neither dominance criterion can determine the better service, the user is queried to compare the QoS vectors of the services. The IVI-SC algorithm iteratively updates the policy that maps states to actions based on these comparisons and the user feedback. This process is repeated until the policy converges to an optimal solution that provides the best trade-off among the multiple QoS attributes. To compare two workflows, value functions are vector-valued, meaning that they return QoS vectors. These vectors are then scaled into a single utility value, allowing workflows to be compared directly. The optimal workflow is identified by combining these comparisons through a backward induction method, solving the following Bellman equation :

$$v_t^*(s) = max_{a \in A(s)}(r_t(s,a) + \gamma \sum_{s' \in s} p_t(s'|s,a) v_{t+1}^*(s')) \tag{II.7}$$

and the Q-value function $Q_t(s,a)$ as follow:

$$Q_t(s,a) = r_t(s,a) + \gamma \sum_{s' \in s} p_t(s'|s,a) v_{t+1}^*(s'). \tag{II.8}$$

The optimal action a_t^* at each state and time step is determined by $arg\ max_{\alpha \in A(s)}\{Q_t(s,a)\}$. By iteratively applying these principles and incorporating user feedback, the algorithm efficiently learns the user's preferences and converges to an

optimal service composition policy, ensuring the best possible QoS.

The VMDP-based service composition approach demonstrates a high adaptability to user preferences without prior knowledge, an effective handling of multiple QoS attributes, and a solid formal foundation. Moreover, the interactive learning process ensures personalized service composition by incorporating user feedback, and the approach's scalability enables efficient management of larger problems. Additionally, the complexity of this algorithm is polynomial with respect to the number of abstract services, concrete services, and QoS attributes, making it suitable for practical applications.

2.1.1.5 Bio-inspired meta-heuristic-based approaches

Several service composition approaches proposed in the literature are based on bio-inspired meta-heuristics. In this context, the QoS-aware service composition is formalized as a combinatorial optimization problem and solved using a discrete artificial bee colony-based (ABC) approach [Xu et al., 2019], an non-dominated sorting genetic algorithm (NSGA)-II [Zhang et al., 2019, Chattopadhyay and Banerjee, 2020], an improved teaching learning-based (ITL) approach [Khanouche et al., 2020a], an extended fuzzy version of the genetic algorithm [Seghir, 2021], an ant colony optimization (ACO) approach [Dahan et al., 2021a], an enhanced Flying Ant Colony Optimization (FACO) approach [Dahan et al., 2021b], an improved salp swarm (SS) algorithm combined with a chaos strategy [Li et al., 2022], a novel bat algorithm (NBA) [Kouicem et al., 2022], and an eagle strategy (ES) using uniform mutation and a modified whale optimization approach (WOA) [Jin et al., 2022], an improved Ant Colony Optimization (ACO) approach [Bei et al., 2024], an artificial rabbit optimization (ARO) combined with a dandelion optimizer (DO) approach [Sharif et al., 2025]. Most of these approaches will be presented in detail below.

In [Xu et al., 2019], an approximate approach based on the discrete artificial bee colony algorithm is proposed to address the QoS-aware service selection problem. This algorithm uses specialized neighborhood search strategies to handle discrete optimization problems, such as service selection. The algorithm begins by initializing a population of artificial bees, each representing a candidate composite service. These composite services are formed by selecting one concrete service for each abstract service in the composition plan to achieve the overall QoS while satisfying specific constraints. This approach is divided into three phases: employed bee phase, onlooker bee phase, and scout bee phase. In the used bee phase, each composition explores its neighborhood to find a new composite service with a higher utility value. Using predefined QoS thresholds, neighboring services are identified according to the rules of the Individual-Based Algorithm (IBA). In each iteration, only one candidate service belonging to the composition is replaced to generate a neighboring composition. If the new composition has a better utility value, it replaces

the current one. This neighborhood exploration reduces computational time and ensures convergence toward a high-quality composition. In the onlooker bee phase, compositions are selected using a probability-based mechanism, with higher utility compositions more likely to be selected for further improvement. A neighborhood search is performed using the Partition-Based Algorithm (PBA) or the Experience-Based Algorithm (EBA). The PBA algorithm organizes candidate services into partially ordered subsets based on their QoS attribute values, therefore enabling the neighborhood search to focus on subsets more likely to contain optimal solutions. The EBA algorithm uses historical optimization data, along with clustering and statistical analysis, to guide the search process toward promising regions. The compositions that are not improved after several iterations enter the scout bee phase, where the search space is randomly explored to generate new composite services, which are then initialized and evaluated according to their utility, and compositions that do not improve after several iterations are discarded. Throughout these phases, the algorithm dynamically adjusts the neighborhood size, assuming that near-to-optimal compositions are located close to existing good ones. This preserves the quality of candidate compositions while improving search efficiency. The algorithm iterates through these three phases until the maximum number of iterations is reached or the convergence criteria are met, selecting the best solution as the optimal composite service.

This approach introduces three tailored neighborhood search strategies (IBA, PBA, and EBA), which dynamically adjust the neighborhood sizes to maintain changes in utility values within a predefined gap, thereby approximating the continuity of near-to-optimal compositions in discrete spaces. However, this approximation depends on the choice of predefined QoS thresholds and the effectiveness of historical optimization data, which do not always find optimal results.

In [Zhang et al., 2019], an improved Non-dominated Sorting Genetic Algorithm-II (NSGA-II) is proposed to optimize the composition of shared manufacturing services by addressing both the short-term utility of consumers and the long-term utility of providers. However, these two objectives may conflict, as maximizing consumer satisfaction in the short term does not always align with ensuring sustainable benefits for providers in the long term, making the optimization problem more complex. To solve this, the service composition problem is formulated as a bi-objective optimization problem, with the consumer's short-term utility model aiming to minimize delivery time and cost, while maximizing product quality. Exponential utility functions are used to quantify consumer satisfaction with respect to these QoS attributes (e.g., delivery time, cost). In contrast, the provider's long-term utility model considers potential future tasks and their associated utilities, focusing on the provider's trust and the probability of future task allocations during the execution period. The provider's utility calculation incorporates factors such as cost, time, and trust, balancing the trade-off between accepting current tasks and

anticipating future ones. To solve this bi-objective optimization problem, an improved version of the NSGA-II algorithm is used, incorporating a crowding distance mechanism to ensure a uniform distribution of Pareto optimal compositions and non-dominated sorting to identify the Pareto front. Furthermore, a Tabu search strategy is integrated into the optimization process to efficiently explore the search space by preventing the reuse of suboptimal compositions, thus helping to avoid local optima. Additionally, an improved k-means technique is applied to group similar service compositions more accurately, therefore enhancing diversity and search capability.

In [Khanouche et al., 2020a], the Improved Teaching-Learning-Based QoS-Aware Composition Algorithm (ITL-QCA) is proposed, where the service composition problem is modeled as a teaching and learning process and solved using the TLBO (Improved Teaching-Learning-Based Optimization) method. This approach optimizes compositions by iteratively improving a population of potential solutions through interactions between a teacher and learners in a classroom setting. The algorithm operates in several phases, starting with the initialization of several parameters, including population size ($Popsize$), maximum number of iterations ($Maxiter$), learning probability (P_c), and iteration-specific parameters (P_m). The population size is the number of composite services belonging to the population. Each composite service $CC_l =< cs^1_{j,l}, ..., cs^i_{j,l}, ..., cs^m_{j,l} >$ is randomly chosen, where $1 \leq j \leq n$ and $1 \leq l \leq Popsize$. The algorithm performs the teaching phase, where the teacher is selected as the best composite service in terms of QoS utility value. Each composition in the population is improved at iteration t according to the best composition $CC^{(t)}_{teacher}$ and the mean composition $CC^{(t)}_{Mean}$ in the population, as follows :

$$CC_l^{(t+1)} = CC_l^{(t)} \oplus (CC^{(t)}_{teacher} \ominus CC^{(t)}_{Mean}) \tag{II.9}$$

In the learning phase, composite services are improved by learning from others according to a learning probability P_c. For each composition in the population, a random number r is generated and if $r < P_c$, each composition $CC_l^{(t)}$ is improved in iteration t according to the best neighboring composition $CC_n^{(t)}$ in terms of QoS utility value and a neighboring composition $CC_v^{(t)}$ chosen randomly, as follows :

$$CC_l^{(t+1)} = CC^{(t)}l \oplus (CC_n^{(t)} \ominus CC_l^{(t)}) \oplus (CC_v^{(t)} \ominus CC_l^{(t)}) \tag{II.10}$$

In the case where $r \geq P_c$, the composition $CC_l^{(t)}$ is improved according to a composition $CC_e^{(t)}$ randomly selected from the population, as follows:

$$CC_l^{(t+1)} = \begin{cases} CC_l^{(t)} \oplus \left(CC_l^{(t)} \ominus CC_e^{(t)}\right), & \text{if } CC_l^{(t)} > CC_e^{(t)}, \\ CC_l^{(t)} \oplus \left(CC_e^{(t)} \ominus CC_l^{(t)}\right), & \text{if } CC_e^{(t)} > CC_l^{(t)}. \end{cases} \tag{II.11}$$

This phase allows the algorithm to refine compositions by learning from the best performing ones or from other members of the population. The self-learning phase enables each composite service to autonomously enhance its QoS attributes, without direct interaction with other compositions, based on historical changes over the last two iterations.

If the composition has changed over the previous two iterations, it is updated using the following formula :

$$CC_l^{(t+1)} = CC_l^{(t)} \oplus (CC_l^{(t)} \ominus CC_l^{(t-1)}) \tag{II.12}$$

If there was no change, the composition is refined using a random number generated from a normal distribution:

$$CC_l^{(t+1)} = normrnd(a, b) \tag{II.13}$$

$$a = (CC_{teacher}^{(t)} \oplus CC_{Mean}^{(t)})/2 \tag{II.14}$$

$$b = abs(CC_{teacher}^{(t)} \ominus CC_{Mean}^{(t)}) \tag{II.15}$$

The algorithm iterates through these three phases until the maximum number of iterations ($Maxiter$) is reached or the algorithm converges to an optimal or near-to-optimal composition.

The multi-phase structure of this approach, comprising teaching, learning, and self-learning phases, ensures continuous improvement by allowing compositions to evolve based on the most efficient ones, interactions with other compositions, and self-refinement over time. However, reliance on stochastic elements, such as random neighbor selection, introduces variability that can affect the convergence stability. Additionally, the lack of dynamic parameter adjustment during iterations can limit the adaptability of the approach in complex environments.

In [Dahan et al., 2021b], an Enhanced Flying Ant Colony Optimization (EFACO)-based approach is proposed to solve the QoS-aware service composition problem. This approach improves the exploration and exploitation capabilities of the conventional Flying Ant Colony Optimization (FACO) method, thus reducing computation time. The EFACO algorithm introduces three modifications: restricting the flying ant process, employing a novel neighboring selection method, and implementing a multi-pheromone mechanism. In the conventional FACO algorithm, the flying ant process searches for the nearest neighbors in each iteration, which is computationally expensive. To overcome this limitation, the EFACO algorithm introduces an adaptive flag that manages the execution of the flying ant process, activating the indicator only when the newly obtained composition outperforms all previous ones. This restriction reduces execution time but may slightly decrease the quality of the composition. The neighboring selection method is modified to improve performance. Instead of examining all neighboring services, the EFACO algorithm randomly selects a subset, which is then sorted by the QoS distance to a service chosen at random from the best compositions found so far. This distance measures the difference between two services by combining the variations in their QoS attributes, including cost, response time, availability, and reliability. As

distance decreases, services are more similar in terms of QoS. The EFACO algorithm balances the exploration of distant services in the initial stages of the search with the exploitation of closer, and more promising services in the subsequent iterations, thereby improving the ability of the algorithm to refine compositions efficiently. The EFACO algorithm introduces a multi-pheromone mechanism, where each QoS attribute has its own influence metric rather than a single combined metric. This allows a better handling of multi-objective optimization by independently evaluating each QoS attribute during the composition process. The influence values for each attribute are computed by combining availability, reliability, cost, and response time, ensuring that each QoS attribute influences the decision-making process, guiding the selection process to the best possible compositions.

The EFACO approach reduces computational time and enhances search efficiency for optimal service compositions compared to the conventional FACO algorithm. However, this refined design brings complexities, including the need for multiple parameters and pheromone updates, which can be computationally intensive.

In [Li et al., 2022], an enhanced Salp Swarm Algorithm (SSA) is proposed to improve search efficiency and accuracy in the QoS-aware service composition problem. This algorithm integrates a chaotic mapping method, named the Chaotic Salp Swarm Algorithm (CSSA), which includes a preprocessing phase using the Fuzzy Continuous Neighborhood Search (FCNS) method. Services with similar QoS attribute values are first grouped into clusters, guiding the composition algorithm towards high quality regions of the search space. Candidate services within each cluster are then ranked according to their aggregated QoS values. From each cluster, representative services are selected based on their rankings to refine the search space and guide the optimization process toward the most promising service compositions. The CSSA algorithm is used to optimize the service selection considering candidate services resulting from the preprocessing process. The initialization phase involves generating an initial population of composite services, where each composite service is represented by the indexes of its concrete services. The CSSA algorithm consists of exploration and exploitation phases. In the exploration phase, the leader composite service guides the search by updating the indexes of its concrete services as follows:

$$X_j^1 = \begin{cases} F_j + (ub_j - lb_j) \cdot c_1 + lb_j, & \text{if } c_2 \geq 0.5, \\ F_j - (ub_j - lb_j) \cdot c_1 + lb_j, & \text{if } c_2 < 0.5 \end{cases} \tag{II.16}$$

where c_1 and c_2 are random values within the interval [0,1], X_j^1 denotes the index of the leader composite service in the j^{th} abstract service, ub_j and lb_j are the upper and lower bounds for that abstract service, and F_j represents the objective function value in the j^{th} abstract service. In the exploitation phase, follower composite services are improved based on the leader's trajectory, incorporating small chaotic perturbations to

maintain diversity and prevent premature convergence. To enhance local exploitation, a single-dimensional perturbation logical chaos strategy is used to update the follower composite services, which preserves the dimensional information of the optimal solution by exploring variations around the optimal solution.

The CSSA approach effectively improves service composition by clustering similar services and focusing on promising search regions. The use of chaotic sequences helps to avoid getting stuck in local optima and enhances exploration. However, the approach can be sensitive to parameter settings, rely on accurate clustering, and may take a long time when dealing with large-scale optimization problems.

In [Kouicem et al., 2022], a QoS-aware service composition approach is proposed using a novel bat algorithm (QC-NBA) for IoT-based applications. The QC-NBA approach consists of five steps: initialization, composition evaluation, iteration process, adaptive learning, and best composition identification. In the initialization phase, a population of bats is randomly generated, where each bat represents a candidate composite service. Each bat is defined by its position, velocity, frequency, and pulse rate, which guide the search process. The parameters such as loudness A_i, pulse rate R_i, frequency F_i, population size ($nPop$), and maximum number of iterations ($MaxIt$) are initialized to control the bats' behavior during the optimization process. In the composition evaluation phase, each composition is evaluated using a fitness function. The algorithm proceeds by successive iterations in which global and local searches are repeated until reaching the maximum number of iterations ($MaxIt$). During the global search (exploration), a new composition new_{xi} is generated from X_i according to a quantum or mechanical behavior. The choice between the two search behaviors is determined by a probability value that depends on the population size and the predefined probability bounds. In quantum behavior, new compositions are generated based on the best composition, along with small random movements proportional to the distance between the average position of the population and the current composition. This allows the algorithm to explore a wide range of possible service combinations. In mechanical behavior, compositions are updated by adjusting their positions based on velocity and adaptive frequency, with a focus on improving the quality of selected services. To further refine compositions, a local search phase updates the best composition using a Gaussian adjustment, as follows:

$$x_{ij}^{t+1} = g_j^t \cdot (1 + rand \cdot N(0, \sigma^2)) \tag{II.17}$$

where g_j^t represents the current best composition at the iteration t. and σ^2 is calculated as follow :

$$\sigma^2 = |A_i^t - A_{mean}^t| + \epsilon \tag{II.18}$$

In the adaptive learning phase, each composite service is evaluated according to the utility value in terms of QoS, whereas the loudness A_i and pulse emission R_i are updated

as follows :

$$A_i^{t+1} = \alpha \cdot A_i^t, \quad R_i^{t+1} = r_i^0 \cdot (1 - e^{-\gamma \cdot t}) \tag{II.19}$$

where α and γ are constants. Finally, in the best composition identification phase, all candidate compositions are sorted, and the one with the highest fitness value is selected as the best one. To avoid local optima, the algorithm reinitializes the loudness A_i and adjusts the pulse rate R_i if no improvement is observed over a predefined number of iterations.

The QC-NBA approach improves service composition in IoT applications by integrating habitat selection and Doppler effect compensation, which enhances exploration and exploitation capabilities of the BA method. Although adaptive parameter tuning, such as loudness and pulse rate, further optimizes the process by making the algorithm more adaptable to the evolving search context, the approach's effectiveness depends heavily on appropriate tuning, which can be challenging and time-consuming.

2.1.2 Energy-aware service composition approaches

The aforementioned QoS-aware service composition approaches have limitations regarding energy consumption, since most services are provided by smart devices with limited battery capacity. To address this issue, several service composition approaches have been proposed to optimize the QoS and energy consumption of services [Ngoko et al., 2013, Khanouche et al., 2016, Tong et al., 2020, Sefati and Navimipour, 2021, Guzel and Ozdemir, 2022, Ibrahim et al., 2020, Yang et al., 2020, Li and Zhu, 2023, Peng et al., 2025] or only energy consumption [Zhao et al., 2017, Sun et al., 2019a, Gao and Liu, 2020].

In [Ngoko et al., 2013], a service selection approach using Mixed ILP is proposed, where the energy consumed by a service is considered as a QoS attribute. The hierarchical service graph (HSG) model [Goldman et al., 2012] is used to represent a service composition as a form of business processes communicating via message transmission. Each business process is represented as a decomposition sub-graph composed of abstract services linked by logical connectors (AND, XOR, etc.), and each abstract service is associated with several concrete services characterized by two attributes: response time and energy consumption. The service selection problem is solved by considering constraints on the composite service, involving two positive upper bounds on the composite service for response time and energy consumption. This requires choosing a single concrete service for each abstract service to optimize the global utility value in terms of response time and energy consumption, while accounting for the user preferences. Accordingly, the SRT_LP_gen algorithm [Goldman et al., 2012] is extended to support the selection of concrete services, compute the Service Response Time (SRT) and

Energy Consumption (EC), and incorporate global QoS constraints such as Service Level Agreements (SLAs) and penalties. The modified version transforms the problem into a mixed-MILP, where binary variables indicate which concrete service corresponds to each abstract service. Additional constraints ensure that both SRT and EC are aggregated correctly across the workflow, while respecting SLA thresholds. The objective function combines SRT and EC into a single utility value to be minimized, enabling the selection of a composite service that balances response time and energy efficiency. When QoS limits make the problem infeasible, the approach estimates feasible intervals for SRT and EC.

The MILP-based service selection approach optimizes response time and energy consumption, ensuring the selection of optimal concrete services that satisfy user-defined QoS constraints. However, the approach faces computational challenges as the number of services and constraints grows, leading to scalability issues that can limit the application of this approach in large-scale service environments.

In [Khanouche et al., 2016], an Energy-centered and QoS-aware services Selection (EQSA) Algorithm is introduced to optimize service composition in IoT environments with a focus on balancing energy consumption and QoS. The EQSA algorithm has two main phases: the preselection of services based on QoS requirements and the selection of services using the concept of relative dominance. In the preselection phase, the QoS-aware service selection is formulated as a multi-objective problem and solved using the lexicographic optimization method. This method consists of solving a sequence of single-objective subproblems, where each subproblem deals with one QoS attribute. The QoS attributes are ranked according to user preferences, and the algorithm identifies candidate services that meet these prioritized attributes while allowing for a slight reduction in quality to save energy. A QoS threshold value is then calculated from the best QoS value and a tolerance factor that characterizes the quality reduction allowed by the user. This threshold value determines the QoS level required to satisfy the user's need, and the candidate services that do not satisfy the obtained threshold are filtered out. This process is repeated for all of the QoS attributes in the order of their importance. The filtering phase reduces the search space of candidate services and, consequently, the selection time. In the second phase, the algorithm selects the most suitable concrete services from the preselected ones based on their relative dominance calculated by combining two components: QoS-based dominance and energy profile-based dominance. The best services in terms of relative dominance are selected for the composition process.

The EQSA algorithm considers both energy consumption and QoS requirements, making the approach well-suited to IoT environments where balancing QoS and energy efficiency is crucial. Despite the fact that the EQSA approach is efficient in terms of time complexity and provides high composition availability by minimizing services' energy consumption, the algorithm does not account for the QoS constraints imposed on the

whole composition.

In [Tong et al., 2020], a QoS and energy-aware service composition approach is proposed to extend the lifetime of the sensor network. This approach has four phases. The first phase involves filtering candidate services based on their functional requirements to ensure that only relevant services are considered for the composition process. In the second phase, global QoS constraints are decomposed into local constraints using the k-means clustering approach. Specifically, the candidate services of each abstract service are grouped into K clusters, and the MIP method is used to derive the optimal decomposition of global QoS constraints into local QoS constraints. After clustering, each abstract service is represented by several clusters, each described by a centroid. These centroids define different quality levels (QL). Each QL corresponds to an intermediate value between two neighboring centroids and serves as a local constraint in the service selection process. In the third phase, candidate services are selected within each cluster based on their QoS, residual energy, and running state. The quality level $QL_{i,j}$ for each abstract service is determined by:

$$QL_{i,j} = \frac{1}{2} \times QoS_i(c_j \cdot centroid) + QoS_i(c_{j+1} \cdot centroid) \tag{II.20}$$

To evaluate the utility of each quality level, the value z_i is associated with each quality level as follow :

$$z_i = \omega_i \cdot \frac{N(QL_{i,j})}{N(total)} \times \frac{U(QL_{i,j})}{U(max)} \tag{II.21}$$

where $N(QL_{i,j})$ is the number of candidate services qualified under $QL_{i,j}$, $U(QL_{i,j})$ is the average utility, $N(total)$ is the total number of individual services. $U(max)$ is the highest utility available. In the fourth phase, the composition is dynamically adapted by monitoring the residual energy of the services. More precisely, a monitoring process continuously tracks the energy levels of selected services and triggers a service re-selection under certain conditions, such as a degradation in QoS below a specified threshold.

This approach offers a good trade-off between satisfying the user's overall QoS constraints and extending the lifetime of the sensor network. When QoS declines, a re-selection of candidate services will be carried out, which can increase the execution time and energy consumption of the composition. Furthermore, when selecting services, the approach favors services that are already running, which does not allow for load balancing between candidate services.

In [Ibrahim et al., 2020], an energy-aware service composition approach using a hybrid Shuffled Frog Leaping Algorithm and Genetic Algorithm (SFGA) is proposed to minimize the energy consumption of mobile cloud providers. The utility value of the composition is calculated as the weighted sum of three key QoS criteria: cost, energy

consumption, and response time. In this approach, the population is represented as a group of solutions, referred to as frogs, and each solution (i.e., composite service) is represented as a frog where its position corresponds to the indexes of the selected candidate services. Frogs are divided into groups called memeplexes, where each group evolves independently to improve solutions. The hybrid algorithm proceeds as follows. First, the initial parameters of the SFLA and GA methods are initialized, and a set of memes forms the initial population, where each meme consists of a set of memo types. The fitness of these memes is then calculated, and the memes are sorted according to their fitness values. These memes are then randomly divided into multiple memeplexes. Each memeplex helps to explore the composition search space in parallel, with the solutions evolving through local and global optimization processes. Within each memeplex, frogs influence and evolve based on their fitness values, exchanging information through GA operators such as two-point crossover and one-point mutation. After the evolution process, the memeplexes are shuffled to form a new population for the next iteration. These steps are repeated iteratively until reaching a predefined stopping condition.

The hybrid SFGA approach uses parallelism, since the search is enhanced by allowing multiple memeplexes to explore the composition space simultaneously. This parallel processing accelerates convergence and reduces computation time, making the approach well-suited for large-scale service composition problems. However, the performance of the SFGA approach depends on the parameter tuning.

In [Yang et al., 2020], an enhanced multi-objective gray wolf optimizer (EMOGWO) is proposed to solve the multi-objective service composition and optimal selection (MO-SCOS) problem in cloud manufacturing, considering QoS and energy consumption. Initially, the concept of Pareto dominance is employed to handle multi-objective optimization by comparing compositions based on their QoS and energy values to identify the Pareto optimal set. To address these issues, the EMOGWO introduces several enhancements over the standard MOGWO to address challenges such as poor exploitation and premature convergence to local optima. First, a novel method including a backward learning strategy is introduced for initialization. This method enhances the exploration of the initial population by generating new candidate compositions outside the current search boundaries. This reflection mechanism increases diversity and promotes more exploration in the optimization process. Second, the EMOGWO method adjusts a control parameter using a nonlinear function to better balance global exploration and local exploitation during the search phases. Third, the EMOGWO method enhances the search process by incorporating the Cauchy mutation, which is applied to boost exploration around the leader solutions. The mutation process generates a new composition by slightly modifying the current one based on a random value obtained from a Cauchy distribution. Furthermore, the algorithm's leader selection strategy, based on roulette-wheel selection, ensures that the α, β, and δ wolves are chosen

from an external archive of non-dominated solutions, promoting solution diversity and preventing convergence to local optima. The archive maintains a set of Pareto optimal compositions by inserting new non-dominated compositions and using a grid mechanism to replace crowded archive members when necessary. The search space is represented as a matrix P, where each row corresponds to an abstract service and its elements denote the candidate services. A position vector X_i represents a possible composite service by selecting a candidate from each row. The EMOGWO algorithm begins with a population of compositions and iteratively improves them. At each iteration, the algorithm identifies the three best compositions according to QoS and energy criteria. These best compositions guide the adjustment of the other compositions, ensuring that the search balances exploration of new possibilities with exploitation of promising areas. In this way, the algorithm progressively refines the population while maintaining an archive of non-dominated solutions. The process continues until the algorithm reaches the stopping condition by returning the archive, which contains the Pareto optimal compositions for the MO-SCOS problem.

In the EMOGWO approach, the adaptive control parameters facilitates a transition between global exploration and local exploitation, enabling the algorithm to progressively refine the obtained solutions. However, the need to maintain and update an archive of non-dominated solutions using a grid mechanism can lead to a significant increase in the computation time, especially in large-scale service composition scenarios.

In [Sefati and Navimipour, 2021], an improved approach combining a hidden Markov model (HMM) and the ACO method is proposed to address the QoS-aware IoT service composition problem in dynamic environments. The HMM model is first used to predict values of QoS attributes. After estimating these values, two matrices are created: the Emission and Transition matrices, which describe how services change between states and their associated QoS values. The Viterbi algorithm is applied to refine the process by sorting and improving service selection based on QoS attributes, facilitating the identification of optimal services. The ACO method is then employed to optimize the service composition process. This method starts by defining a set of IoT nodes $N = \{N_1, ..., N_n\}$, where each node provides various services. A set of ants, m, is initialized, and each ant begins by randomly selecting a starting IoT node. The ants iteratively build a service composition by selecting services from subsequent nodes using a probabilistic decision rule. The probability of selecting a service at node N_j from node N_k at time t is given by:

$$p_{kj}^{i}(t) = 1 + \frac{[\tau_{k,j(t)}][(n_{kl})]}{\sum_{C_i \in S_r(i)} \tau^k(t)[(n_{kl})]x^2} \tag{II.22}$$

where $\tau_{k,j(t)}$ represents the pheromone level between nodes N_k and N_j, and n_{kl} is the heuristic value representing the quality of the service obtained from the QoS criteria. Once a composite service has been created, the pheromone levels associated with the selected services are updated. Services that have contributed to improving compositions

receive higher pheromone levels, while less efficient services receive lower pheromone levels. This process continues until reaching a stopping condition (e.g., the convergence of the algorithm or after a predetermined number of iterations). The final result is the best service composition with the best QoS.

The Hidden Markov Model employed in this approach is used to predict QoS attributes and refine service selection through the Viterbi algorithm. However, achieving accurate QoS predictions with HMM requires a substantial amount of data, which can be difficult to obtain in real-world scenarios.

In [Li and Zhu, 2023], the service composition problem is solved by focusing on the energy consumption of file transfers between clouds, as well as the optimal QoS for user requests. The proposed approach uses a genetic algorithm to select atomic services from multiple cloud providers while balancing QoS attributes, including response time, cost, reliability, availability, reputation, and security. Furthermore, an energy consumption model is introduced to account for the user's energy consumption and data transmission between different cloud centers. Each individual in the GA population represents a candidate composition, encoded as a mapping of abstract services to atomic services across clouds. The utility function simultaneously maximizes overall QoS satisfaction, minimizes user energy consumption, and reduces transmission energy between clouds. The search process is guided by genetic operators, such as selection, crossover, mutation, and updating to improve the population. The iterative process continues until a predefined stopping condition is satisfied, such as reaching a maximum number of iterations or observing no significant improvement in the utility composition. The algorithm then retains the best solutions found.

2.2 Mobility-aware service composition approaches

The QoS and energy-aware service composition approaches are inappropriate for mobile environments requiring more flexible and dynamic service composition techniques that can be adapted to user's mobility. Unfortunately, few existing approaches deal with the service composition problem that accounts for the mobility aspect [Deng et al., 2016a, Deng et al., 2016b, Deng et al., 2016c, Hossain et al., 2016, Deng et al., 2017b, Deng et al., 2018, Wu et al., 2019, Peng et al., 2020] .

In [Deng et al., 2016a], a mobility-enabled service composition approach is introduced to address the challenge of user mobility in dynamic service environments. A mobility model is first proposed to track and manage users' movements during the service composition and execution phases. This model is based on user properties, such as location, velocity, and direction, as well as environmental factors, including network conditions and

connectivity fluctuations. A mobility-aware QoS (MQoS) computation rule is then proposed to dynamically calculate QoS values of services under mobility constraints, focusing exclusively on response time. This metric is calculated taking into account the mobile network latency for both input and output data transmission, as follows:

$$MQoS_s = t_{di} + Q_s + t_{do} \tag{II.23}$$

where t_{di} and t_{do} represent the input and output data latency, respectively, and Q_s is the response time of the service. The latency values are obtained by dividing the volume of transmitted data by the quality of the mobile network available at the user's location. Finally, service selection in a mobile environment is modelled as a classroom learning process and solved using the Teaching-Learning-Based Optimization (TLBO) algorithm in two phases. In the teacher phase, the current best composition, called the teacher, guides the other potential compositions (learners) toward improved QoS values by using the teacher's performance. A composition is updated according to the teacher's composition and the average QoS of all compositions, guided by random and teaching factors to reinforce improvement. In the learner phase, the compositions interact with each other to exchange knowledge and further enhance their QoS, based on their own performance and the changing mobility conditions. Each composition of the population randomly selects another composition as a learning target. If the chosen composition has a lower QoS utility value, the current composition approximates its own solution. Conversely, in the case where the chosen composition has a higher QoS utility value, the current composition is updated toward a better solution. Through these phases, the TLBO algorithm iteratively improves the service composition by minimizing the global QoS value, which in this case represents the overall response time.

The mobility-enabled service composition approach proposed in [Deng et al., 2016a] effectively adapts to user movement and changing network conditions, making the approach well-suited for dynamic mobile environments. However, only response time is considered as the QoS attribute, while other important factors such as cost, reliability, and energy consumption are not taken into account during the composition process.

In [Deng et al., 2016b], a Mobile Service Sharing Community (MSSC) architecture is proposed to address the challenges associated with moving service users and providers. The MSSC allows users to share mobile services while moving within specific areas. To model the movement of the user and the provider, the Random Waypoint (RWP) model is extended to form a critical point-based Random Waypoint (CRWP) model. In this model, users move only between predefined critical points within a convex region. A user randomly selects a starting critical point and stays there for a random duration within an interval $[p_{min}, p_{max}]$. After the pause, another critical point is randomly chosen, and the user moves toward this point at a speed randomly selected from $[v_{min}, v_{max}]$. This process is repeated, producing a trajectory described as a sequence of trips, where each

trip includes the destination point, the pause time, and the velocity. For the MSSC architecture, several key concepts are formally defined. A mobile user is represented as a four-tuple $u = (uid, umt, S, l)$, where uid is the unique identification of the user, umt represents the user's moving trajectory, S is the set of services the user can offer, and l is the sensing distance of the user's mobile device. A mobile service is defined as a triple $s = (sid, e)$, where sid is the service identification, and e is the response time of the service. A composition request is represented as a two-tuple $cr = (T, R)$, where $T = \{t_1, ..., t_n\}$ is a set of abstract services (tasks), and $R = r(t_i, t_j)|t_i, t_j \in T$ defines the relations between tasks, with $r(t_i, t_j) = 1$ indicating that the inputs of t_j depend on the outputs of t_i. The mobile service composition, denoted as msc, is a triple $msc = (h, S, L)$, where h represents the composition request, S is the set of services in the composition, and L is the total response time of the composition. The goal is to select services from different providers to form an optimal service composition with minimal response time while considering the user mobility. Based on the MSSC architecture and the introduced mobility model, the mobility-aware service composition problem is formalized as an optimization problem and solved using the Krill-Herd (KH) algorithm. The latter starts by initializing a population of krill individuals, each representing a set of composite services. Over successive iterations, the compositions are updated based on three motions: (i) motion induced by other krill that corresponds to learning from neighbor compositions, (ii) foraging motion that uses global knowledge, and (iii) random diffusion to maintain diversity and avoid premature convergence. The position of each krill, representing a service composition, is iteratively updated based on movements that balance learning and exploration. The algorithm proceeds until a stopping condition is satisfied, such as reaching a maximum number of iterations or observing no further improvement. At this point, the best composition obtained is considered the optimal or near-optimal solution.

This approach adapts to user mobility and dynamic network conditions, while the krill-heard algorithm balances exploration and exploitation to avoid premature convergence. However, only response time is considered as a QoS attribute, ignoring other important aspects such as cost, reliability, or energy consumption.

In [Hossain et al., 2016], a big data-driven service composition approach for mobile environments is proposed to optimize the selection of interconnected services offered by various cloud providers (computation, big data storage, messaging, media services, and payment services). The approach has two key phases. The first phase uses the k-means clustering technique to group services based on their QoS attributes. The k-means algorithm starts by initializing a predefined number of clusters, k, and assigning each service to the closest cluster based on the Euclidean distance between the service's QoS values and the centroids of the clusters. The centroid of each cluster is iteratively updated by recalculating the average QoS values of the services assigned to the cluster,

and the process continues until the clusters converge. This clustering helps to reduce the search space by identifying representative services from each cluster, thus simplifying the selection process. Furthermore, the Parallel Cloud Particle Swarm Optimization (PCPSO) method is employed to further enhance the efficiency of service selection. The PCPSO algorithm partitions the service set and runs the optimization process in parallel, significantly reducing the computation time required to handle the large volume of data in a mobile cloud environment. In the second phase, the approach employs the PSO method to select the best combination of services among the candidate service set resulting from the first phase.

This approach addresses the challenges of large-scale mobile cloud environments by combining clustering and parallel optimization. The k-means clustering reduces the search space by grouping services with similar QoS attributes, while PCPSO accelerates computation through parallel execution. This makes this approach scalable and suitable for big data contexts. However, the efficiency of the approach depends on the quality of clustering, which may ignore some promising services, and the use of the PSO method may lead to premature convergence, thus avoiding to find optimal service compositions.

In [Deng et al., 2017b], a service selection approach based on the genetic algorithm is proposed to minimize the energy consumption. The mobile path is represented as a set of time intervals, location points, and a function mapping time to location, allowing for the prediction of signal strength at any given point along the path. This approach allows the evaluation of energy consumption, comprising three parts: uploading input data, standby power during service execution, and downloading results. In this approach, a composite service corresponds to a chromosome, a candidate service to a gene, an abstract service to a locus, and the energy consumption of a composite service to a fitness value. A chromosome with a high fitness value indicates a composite service with lower energy consumption. The genetic algorithm begins with parameter initialization, including the population size, the number of iterations it, the number of crossovers per iteration ct, and the number of mutations per iteration mt. Then, the initial composite services are generated and placed in the composite service set $ChrSet$. For each abstract service, a candidate service is randomly selected to create an initial composite service. The crossover operation aims to generate new composite services with lower energy consumption for the next generation. This is achieved by combining two composite services and swapping candidate services of the same abstract services to produce two new composite services. The single-point crossover operator is employed for this process, and the resulting composite services from crossover, which may have lower energy consumption than their parents, are added to $ChrSet$. The mutation operator replaces a candidate service in a composite service with another candidate service randomly chosen from the same abstract service. This operator may produce new composite services that consume less energy compared to their predecessors, and

they are thus added to $ChrSet$. The selection phase starts by calculating the fitness value of each composite service in $ChrSet$. In each iteration it, the composite service with the highest fitness value in terms of energy consumption is saved in $CurOptChr$. This sub-optimal composite service is compared to the previous sub-optimal service $OptChr$, and the one with the lowest energy consumption is recorded as the new $OptChr$. During the selection process, cq composite services are chosen from $ChrSet$, and the selection probabilities favor the services that consume the least energy. This process is repeated until the predefined number of iterations it is reached. Finally, the composite service $OptChr$ with the lowest energy consumption is returned as the sub-optimal service.

This approach focuses on minimizing energy consumption. However, giving priority to the reduction of energy consumption may have a negative impact on important factors, such as QoS parameters. As a result, the composite services generated may not fully meet the QoS user requirements during the composition process.

In [Deng et al., 2018], a dependable service composition approach considering the mobility of users and providers is proposed. The overall QoS and the risk of invoking the service composition in a mobile environment are considered as two primary properties. The overall QoS is computed using integration functions that aggregate the individual QoS values across the candidate services. The risk of mobile service composition represents the probability that the overall composition fails during execution. It is defined by the highest individual risk among the selected services, meaning that the overall reliability depends on the weakest component. If one service becomes unavailable, the entire composition may fail, making the most vulnerable service the determining factor of the overall risk. Formally, the risk of a service composition is modeled as follows :

$$Risk_{sc} = max_{i=1,\dots,n} r_i \tag{II.24}$$

where r_i is the risk that the service provider of the selected service of the i^{th} abstract service becomes unavailable during the execution due to factors such as moving out of communication range. The risk of each abstract service is computed as follows :

$$r_i = 1 - Prob(X_i^p \geq dur_i^p) \tag{II.25}$$

where $X_i^p \geq dur_i^p$ represents the probability that the service provider p will remain within the required communication range for the duration dur_i^p of the i^{th} abstract service. This probability can be estimated using mobility prediction methods based on signal strength detection. The approach formulates the service composition problem as an optimization problem that accounts for QoS and risk. To solve this problem, a Modified Simulated Annealing (MSA) algorithm is applied, where a mutation mechanism similar to genetic algorithms is introduced to improve the solution space exploration more effectively. The MSA begins by generating an initial composition randomly from the search space. The algorithm then iteratively generates new compositions by making slight modifications

to the current composition using a mutation operator similar to the mutation process in genetic algorithms that randomly replaces one or more service components with alternative candidates. The next step is to evaluate whether the new composition should be accepted or rejected based on the objective function. A worse composition may still be accepted with a certain probability to avoid being trapped in local optima. As the temperature decreases, the algorithm becomes more selective, accepting only better compositions as the algorithm converges toward an optimal solution. This iterative process continues through successive cooling steps until the temperature reaches a predefined minimum threshold, after which the process terminates and the best solution found is returned as the final composition.

3 Comparative study of service composition approaches

Table II.1 summarize the approaches presented in this chapter by classifying them according to the following criteria: (F1) resolution method; (F2) QoS awareness; (F3) energy awareness; (F4) mobility awareness; (F5) meta-heuristics evolution process that describes how the optimization method evolves the population of the compositions over iterations; (F6) specific parameters that indicates whether the used method requires tuning particular algorithm parameters; (F7) filtering process that specifies whether the approach discards the unpromising candidate services before carrying out the composition.

Table II.1: Comparison of existing service composition approaches.

	Comparison metrics						
Approach	F1	F2	F3	F4	F5	F6	F7
[Deng et al., 2016a]	TLBO	✓	✗	✓	Overall population	No	No
[Deng et al., 2016b]	Krill-heard	✓	✗	✓	Overall population	Yes	No
[Deng et al., 2016c]	Differential evolutionary	✓	✗	✓	Overall population	Yes	Yes
[Hossain et al., 2016]	PSO and Clustering	✓	✗	✓	Overall population	Yes	Yes
[Deng et al., 2017b]	GA	✗	✓	✓	Overall population	Yes	No
[Deng et al., 2018]	Modified simulated annealing	✓	✗	✓	Overall population	Yes	No
[Wu et al., 2019]	GA and simulated annealing	✗	✗	✓	Overall population	Yes	No
[Peng et al., 2020]	Krill-heard	✓	✗	✓	Overall population	Yes	No
[Ngoko et al., 2013]	Integer programming	✓	✓	✗	✗	/	No
[Khanouche et al., 2016]	Lexicographic method	✓	✓	✗	✗	/	Yes
[Zhao et al., 2017]	PSO and GA	✗	✓	✗	Overall population	Yes	No
[Tong et al., 2020]	QoS decomposition	✓	✓	✗	✗	/	No
[Sun et al., 2019a]	PSO and GWO	✗	✓	✗	Overall population	Yes	No
[Gao and Liu, 2020]	GA and PSO	✗	✓	✗	Overall population	Yes	No

[Sefati and Navimipour, 2021]	ACO and Hidden Markov Model	✓	✓	✗	Overall population	Yes	No
[Guzel and Ozdemir, 2022]	NSGA II	✓	✓	✗	Overall population	Yes	No
[Ibrahim et al., 2020]	SFLA and GA	✓	✓	✗	Partial population	Yes	No
[Yang et al., 2020]	GWO	✓	✓	✗	Overall population	Yes	Yes
[Li and Zhu, 2023]	GA	✓	✓	✗	Overall population	Yes	No
[Peng et al., 2025]	NSGA II	✓	✓	✗	Overall population	Yes	No
[Zeng et al., 2004]	Integer programming	✓	✗	✗	✗	/	No
[Ardagna and Pernici, 2007]	Integer programming	✓	✗	✗	✗	/	No
[Wang et al., 2020b]	Integer programming	✓	✗	✗	✗	/	Yes
[Sun and Zhao, 2012]	QoS decomposition	✓	✗	✗	✗	/	No
[Alrifai et al., 2012]	QoS decomposition	✓	✗	✗	✗	/	No
[Yuan et al., 2019]	QoS decomposition	✓	✗	✗	✗	/	No
[Khanouche et al., 2020b]	QoS decomposition	✓	✗	✗	✗	/	Yes
[Halfaoui et al., 2015]	Pareto dominance	✓	✗	✗	✗	/	Yes
[Wang et al., 2017]	Pareto dominance	✓	✗	✗	✗	/	Yes
[Chen et al., 2019]	Pareto dominance	✓	✗	✗	✗	/	Yes
[Wang et al., 2020a]	RL and Pareto dominance	✓	✗	✗	✗	/	Yes
[Alizadeh et al., 2020]	RL	✓	✗	✗	✗	/	No
[Liang et al., 2021]	Deep RL	✓	✗	✗	✗	/	No
[Xu et al., 2019]	Artificial Bee Colony	✓	✗	✗	Overall population	Yes	No
[Zhang et al., 2019]	NSGA II	✓	✗	✗	Overall population	Yes	No
[Chattopadhyay and Banerjee, 2020]	NSGA	✓	✗	✗	Overall population	Yes	Yes
[Khanouche et al., 2020a]	Improved teaching learning	✓	✗	✗	Overall population	No	No
[Seghir, 2021]	Fuzzy GA	✓	✗	✗	Overall population	Yes	No
[Dahan et al., 2021a]	ACO and GA	✓	✗	✗	Overall population	Yes	No
[Dahan et al., 2021b]	Enhanced FACO	✓	✗	✗	Overall population	Yes	No
[Li et al., 2022]	Improved salp swarm	✓	✗	✗	Overall population	Yes	Yes
[Kouicem et al., 2022]	Novel BA	✓	✗	✗	Overall population	Yes	No
[Jin et al., 2022]	Eagle strategy and improved Whale optimization	✓	✗	✗	Overall population	Yes	No
[Bei et al., 2024]	Improved ACO	✓	✗	✗	Overall population	Yes	No
[Sharif et al., 2025]	ARO and DO	✓	✗	✗	Overall population	Yes	No

Extensive research has been conducted on service composition, focusing mainly on three optimization dimensions: QoS, energy, and mobility. Early studies have addressed QoS-aware service composition [Wang et al., 2020b, Seghir, 2021, Sefati and Navimipour, 2021, Khanouche et al., 2020a, Dahan et al., 2021b, Kouicem et al., 2022, Sun and Zhao, 2012, Chen et al., 2019, Wang et al., 2017, Alizadeh et al., 2020, Wang et al., 2020a], while others have focused on energy-aware service composition [Zhao et al., 2017, Sun et al., 2019a, Gao and Liu, 2020]. More recent works have explored hybrid approaches that simultaneously consider QoS and energy [Guzel and Ozdemir, 2022, Ibrahim et al., 2020, Yang et al., 2020], or that integrate mo-

bility aspect to deal with dynamic environments [Deng et al., 2016a, Deng et al., 2016b, Deng et al., 2016c, Hossain et al., 2016, Deng et al., 2017b, Deng et al., 2018].

The proposed QoS-aware service composition approaches [Wang et al., 2020b, Seghir, 2021, Sefati and Navimipour, 2021, Khanouche et al., 2020a, Dahan et al., 2021b, Kouicem et al., 2022, Sun and Zhao, 2012, Chen et al., 2019, Wang et al., 2017, Alizadeh et al., 2020, Wang et al., 2020a] have proven to be efficient in addressing users' QoS requirements, ensuring that services maintain high quality, such as response time, reliability, and throughput. However, most of these approaches often ignore the constraints imposed by the physical environment, particularly when resource-limited or battery-powered devices provide services. In dynamic and mobile contexts, ignoring the energy dimension can result in a significant decrease in service availability, as continuous execution of energy services quickly depletes device batteries and leads to unexpected failures. Furthermore, the random nature of mobile environments amplifies this problem: fluctuating network conditions, frequent handovers, and service migration often cause temporary unavailability or QoS degradation. In such contexts, conventional QoS-aware service composition approaches fail to capture the link between device energy, mobility, and service performance. They tend to offer solutions that are theoretically optimal in terms of QoS but impractical in the real-word, as they do not guarantee continuity of composition over time and are inadequate to the dynamic nature of large-scale mobile environments.

To address this limitation and enhance service availability, energy-aware service composition approaches [Zhao et al., 2017, Sun et al., 2019a, Gao and Liu, 2020] have been proposed to reduce the energy consumption and thus increase the lifetime of devices. However, these approaches often degrade QoS, as they do not consider users' QoS requirements. In such cases, when a service in a composition becomes unavailable or its quality deteriorates, the service needs to be replaced to ensure continuous operation and maintain the required QoS levels.

To address this issue, more balanced approaches, namely QoS and energy-aware service composition approaches, have been proposed [Guzel and Ozdemir, 2022, Ibrahim et al., 2020, Yang et al., 2020]. These approaches aim to simultaneously optimize energy and QoS. However, achieving a good trade-off between these two objectives remains a major challenge. The complexity of this problem increases in mobile environments, where services are hosted by heterogeneous and resource-constrained of users and devices such as smartphones, robots, and drones. In this context, both QoS attributes and energy profiles vary dynamically, further complicating the composition process.

Despite these efforts, only few approaches consider mobility in the context

of service composition [Deng et al., 2016a, Deng et al., 2016b, Deng et al., 2016c, Hossain et al., 2016, Deng et al., 2017b, Deng et al., 2018, Wu et al., 2019, Peng et al., 2020], but do not simultaneously address QoS and energy issues. This can lead to (*i*) a decrease in the QoS of the composition and/or (*ii*) an unbalanced energy consumption among services, which reduces the availability of the composition. Moreover, most of these approaches are based on simplified mobility models, such as Random Waypoint, which randomly define the user's movement and fail to capture users' social behaviors, thus limiting the practical relevance of their results.

Another significant shortcoming concerns the size of the candidate service space. Many service composition approaches consider the entire set of candidate services during the composition process [Khanouche et al., 2020b, Kouicem et al., 2022, Ardagna and Pernici, 2007, Alrifai et al., 2012, Yuan et al., 2019, Chattopadhyay and Banerjee, 2020, Zeng et al., 2004, Xu et al., 2019, Dahan et al., 2021a, Jin et al., 2022, Sun and Zhao, 2012, Seghir, 2021]. While this exhaustive exploration can theoretically increase the chances of finding an optimal solution, it also dramatically expands the search space, especially in large-scale environments with thousands of candidate services for each abstract service. As a result, the optimization process becomes very expensive in terms of computation resources, increasing the composition time. Furthermore, including all candidate services in the composition often introduces low quality or redundant services, which can reduce the utility of the resulting composition.

Moreover, most of the population-based service composition approaches cited above [Xu et al., 2019, Dahan et al., 2021a, Jin et al., 2022, Seghir, 2021, Sun et al., 2019a, Ibrahim et al., 2020, Sefati and Navimipour, 2021, Li and Zhu, 2023, Li et al., 2022] require careful tuning of several specific algorithm parameters to achieve a near-to-optimal composite service within a reasonable time. This requirement not only increases the computation time but also limits the practical applicability of these approaches in real-world scenarios, where service environments are dynamic, heterogeneous, and unpredictable.

Finally, most of meta-heuristics-based service composition approaches [Zhao et al., 2017, Gao and Liu, 2020, Guzel and Ozdemir, 2022, Deng et al., 2016a, Deng et al., 2016b, Deng et al., 2016c, Hossain et al., 2016, Deng et al., 2017b, Deng et al., 2018, Wu et al., 2019, Peng et al., 2020, Seghir, 2021, Sefati and Navimipour, 2021, Yang et al., 2020, Khanouche et al., 2020a, Dahan et al., 2021b, Kouicem et al., 2022, Jin et al., 2022, Zhang et al., 2019, Li et al., 2022] improve the entire population of candidate compositions through a given number of iterations. Although this strategy can provide good results, the process can lead to premature convergence and high computation time, especially in large-scale

environments where the composition space grows exponentially with the number of candidate services. As a result, maintaining population diversity becomes difficult, leading to suboptimal solutions that cannot adapt to dynamic and unpredictable contexts.

Given the aforementioned limitations, including the lack of balanced consideration between QoS, energy, and mobility constraints, the limited capacity of existing algorithms to improve the population of candidate compositions throughout the iterative process, the need for manual parameter tuning, and the absence of service filtering process, it becomes evident that existing service composition approaches remain unsuitable for large-scale and dynamic environments. Therefore, developing service composition algorithms that simultaneously ensure QoS guarantees, energy efficiency, and adaptability to user mobility remains an open research problem. Existing approaches often consider only one or two of these dimensions, which compromises the service quality, the energy sustainability of devices, or the adaptability of the composition to mobile contexts. Moreover, scalability issues related to the large candidate service space, the dependence on algorithmic parameters tuning, and the limited convergence capacity of these methods highlight the need to design efficient optimization mechanisms. In this context, the objective of this thesis is to propose new service composition approaches that overcome these limitations by ensuring a balanced trade-off between QoS and energy consumption, while integrating mobility aspect to better reflect the dynamics nature of Cyber-Physical-Social Systems.

4 Conclusion

In this chapter, we have reviewed different service composition approaches, focusing on key criteria such as QoS awareness, energy efficiency, mobility considerations, and optimization techniques. Although many of the existing approaches successfully address QoS-aware service composition, they often overlook critical aspects such as energy constraints, which are especially relevant in environments with resource-limited devices. Energy-aware approaches improve service availability but can compromise service quality, leading to the degradation of performance. The trade-off between QoS and energy consumption has driven the development of hybrid approaches that aim to balance both factors. However, achieving an optimal balance between these two objectives remains a challenge, particularly in mobile environments where services are hosted on devices with fluctuating energy and QoS. Furthermore, while several mobility-aware approaches have been introduced, few of these methods simultaneously address both QoS and energy issues. To overcome these limitations, the aim of the next chapter is to present the energy-efficient and QoS-aware service composition approach proposed in the context of Internet of Things. This approach decreases the composition time and reduces the energy consumption of the composition, while satisfying the global QoS constraints.

Chapter III

A group teaching optimization-based approach for energy and QoS-aware Internet of Things service composition

1 Introduction

In the context of large-scale IoT, service composition poses several critical problems, particularly the optimal management of energy and QoS. This complexity arises from the dynamic and stochastic behavior of services, in which QoS parameters are frequently affected by unpredictable changes. Therefore, services may appear or disappear, become unavailable, deteriorate in quality, or fail to satisfy user requirements. In addition, energy constraints further complicate the composition process, as the services belonging to the composition are typically hosted by devices with limited battery life, leading to service interruptions during execution. Several existing service composition approaches have limited computation time, QoS utility, and composition lifetime because they do not simultaneously address energy and QoS constraints, consider all services during the composition process, or usually require tuning specific algorithm parameters.

The present chapter proposes the Group Teaching-based Energy-Efficient and QoS-Aware Services Composition algorithm (GT-EQCA) to address the aforementioned challenges. By integrating energy efficiency and QoS awareness into the composition process, the GT-EQCA algorithm aims to address the limitations of existing approaches and improve overall composition performance. The remainder of this chapter is structured as follows. Section 2 defines the models used in the proposed approach and formally introduces the problem of service composition, accounting for energy consumption and QoS constraints. Section 3 outlines the Group Teaching Optimization (GTO) method, whereas Section 4 details the proposed GT-EQCA approach. Section 5 evaluates its performance through comparisons with existing baseline algorithms, and Section 6 summarizes the key findings of the GT-EQCA algorithm.

2 Models and problem description

2.1 Service model

An IoT service refers to an autonomous software component that provides a well-defined functionality, such as data sensing, data analysis, or information search. Each IoT service is characterized by two dimensions: (*i*) Functional properties that include the semantics of the performed operations, the behavioral specification (including preconditions, post-conditions, and exceptions), and the service interface; (*ii*) Non-functional properties that describe quality-related attributes such as cost, reliability, robustness, contextual relevance, and energy efficiency. In the context of IoT environments, an IoT service could be a concrete service or an abstract service.

2.1.1 Concrete service

An concrete IoT service, denoted cs_j, refers to a specific operation that, upon invocation, processes input data to produce output data. It is characterized not only by its functional characteristics but also by non-functional attributes, including QoS parameters (e.g., execution time), energy-related metrics (e.g., residual energy), and contextual data (e.g., localization). Formally, a concrete service is described by a vector: $cs_j = \{InputData(cs_j), OutputData(cs_j), QoS(cs_j), EE(cs_j)\}$, where $InputData(cs_j)$ and $OutputData(cs_j)$ represent, respectively, the inputs and outputs of the service cs_j, $QoS(cs_j)$ denotes the vector of QoS parameters, and $EE(cs_j)$ refers to the energy efficiency of the service cs_j (see Figure III.1).

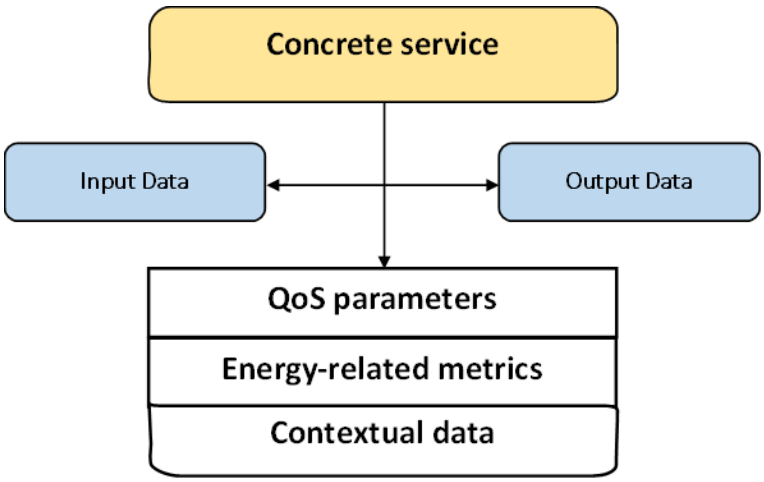


Figure III.1: Representation of a concrete service.

2.1.2 Abstract service

An abstract service, denoted as $AS^i = \{cs_1^i, \ldots, cs_j^i, \ldots, cs_n^i\}$, represents a set of n concrete IoT services that share common inputs and outputs, but differ in their non-functional characteristics (see Figure III.2). These concrete services are functionally equivalent as

they perform the same operations but differ in aspects such as energy efficiency and QoS parameters. Abstract services usually serve as abstract classes because they lack a fixed concrete implementation and are dynamic, since their concrete realization depends on the concrete service chosen at runtime. This selection process must consider non-functional properties, especially those related to QoS parameters and energy efficiency. Consequently, an abstract service encapsulates a set of concrete services with similar functionalities, thus introducing flexibility and adaptability in service provision and utilization.

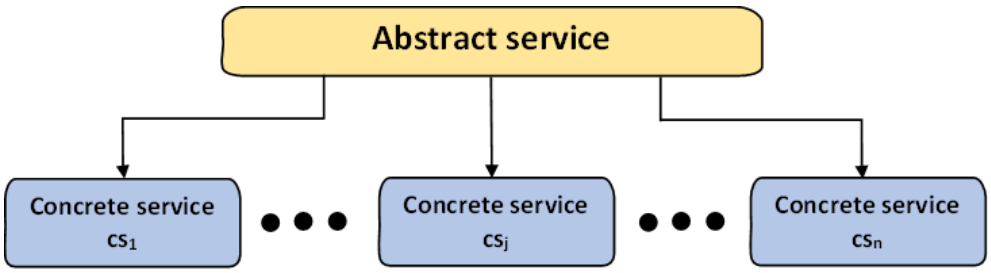


Figure III.2: Representation of an abstract service.

2.2 Energy-related metrics

In this thesis, each concrete IoT service cs_j is characterized by two energy-related metrics: (i) the residual energy $RE(cs_j)$, which represents the remaining energy of the IoT device that hosts the service, and (ii) the energy consumption $EC(cs_j)$, which quantifies the energy consumed during the execution of the service.

The residual energy of a concrete service cs_j is computed as follows:

$$RE(cs_j) = ED(cs_j) - TH(cs_j) \tag{III.1}$$

where $ED(cs_j)$ denotes the energy capacity of the device hosting the service and $TH(cs_j)$ refers to a critical energy threshold below which the device can no longer deliver the service.

The energy consumed by the concrete IoT service cs_j during its execution is calculated using the following equation:

$$EC(cs_j) = EC_{unit}(cs_j) \cdot ExecTime(cs_j) \tag{III.2}$$

where $EC_{unit}(cs_j)$ represents the rate of power consumed per time unit, and $ExecTime(cs_j)$ is the duration required to complete the execution of the service cs_j.

The energy efficiency of a concrete IoT service, denoted $EE(cs_j)$, reflects the proportion of residual energy remaining on the device after the service execution. It is formally defined by the following formula:

$$EE(cs_j) = 1 - \frac{EC(cs_j)}{RE(cs_j)} \tag{III.3}$$

Consider two concrete IoT services, cs_j^i and cs_k^i, which belong to the same abstract service AS^i and have energy efficiency $EE(cs_j^i)$ and $EE(cs_k^i)$, respectively. The service cs_j^i is considered more efficient than the service cs_k^i if and only if $EE(cs_j^i) > EE(cs_k^i)$.

2.3 Service composition model

In the context of IoT service composition, the user's functional requirements are typically specified as an abstract composition model $AC = \{AS^1, \cdots, AS^i, \cdots, AS^m\}$, where m abstract services are interconnected via basic composition patterns that specify how these services are linked. These patterns include sequence $\rightarrow$, parallel $\otimes$, conditional $\oplus$, and loop $\odot$ structures [Yu et al., 2007]. In a sequential structure, an abstract service AS_{i+1} is invoked only after the execution of its predecessor AS_i. Conversely, in a parallel structure, several services $AS_1, \cdots, AS_p$ within the parallel branch are executed concurrently. The conditional structure enables selecting a single execution path AS_i among multiple alternatives with probability P_i. The loop structure allows iterative execution of a service AS_i for a predetermined number of iterations before progressing to the subsequent service. Figure III.3 presents an example of a service composition involving six abstract services that can be represented as $AC =< \odot AS_1 \rightarrow (AS_2 \oplus (AS_3 \rightarrow (AS_4 \otimes AS_5))) \rightarrow AS_6 >$. Each abstract service AS^i within the composition can be instantiated by n_i $(1 \leq n_i \leq n)$ functionally equivalent concrete IoT services, which differ in terms of their non-functional properties.

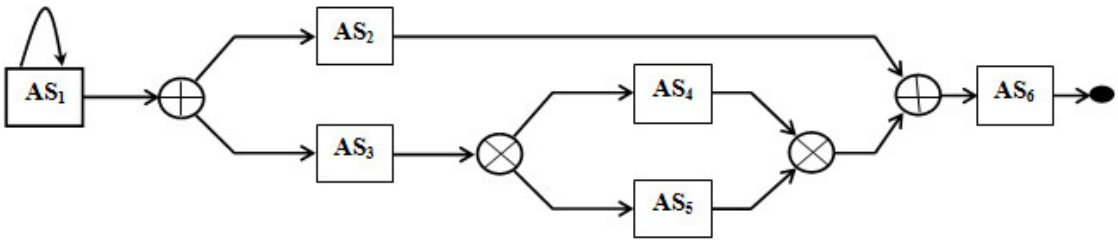


Figure III.3: Example of abstract services composition including basic patterns.

2.3.1 QoS of concrete service

The quality of a given concrete IoT service cs_j^i is typically represented by a vector comprising k parameters $QoS(cs_j^i) = (q_{1,j}^i, \cdots, q_{q,j}^i, \cdots, q_{k,j}^i)$, where $q_{q,j}^i$ denotes the value of the q^{th} QoS parameters associated with the service. Each criterion reflects a specific aspect of the service's performance and usability. The QoS criteria are generally categorized into positive or negative attributes. A positive QoS attribute, denoted by Q^+, is one for which higher values indicate better performance (e.g., availability, throughput), and should therefore be maximized. In contrast, a negative QoS attribute, denoted by Q^-, is one for which lower values reflect better performance (e.g., cost, response time), and should therefore be minimized. Table III.1 presents the QoS criteria that are used in the service computing domain [Sun and Zhao, 2012].

Table III.1: The QoS criteria used in the services computing domain.

QoS criterion	**Description**
Response Time	The duration required by an IoT service to process and fulfill a user's request. This duration may include delay introduced by network transmission during the invocation process.
Throughput	The number of service requests handled by an IoT service within a specific time interval.
Availability	The degree to which an IoT service is operational and accessible during a specific period of time.
Reliability	The ability of an IoT service to correctly fulfill the operations defined in its description and without failure.
Cost	The price that a user pays to access and use an IoT service provided by a third party.
Reputation	The level of trust associated with an IoT service, typically derived from the historical user feedback and service performance.
Security	The quality aspect related to the confidentiality, authentication, and non-repudiation of the parties involved in an IoT service use.

2.3.2 QoS of composite service

The objective of QoS-aware IoT service composition is to select, for each abstract service, the most suitable concrete IoT service that optimizes QoS attributes, such that the resulting composite concrete service $CC =< cs^1_{j_j}, \cdots, cs^i_{j_j}, \cdots, cs^m_{j_j} > (j_j \in \{1..n\})$ maximizes overall utility while satisfying the global QoS constraints specified by the user. The quality of the concrete composition CC is expressed by a vector $QoS(CC) = (Q_1, \cdots, Q_q, \cdots, Q_k)$, where Q_q $(1 \leq q \leq k)$ corresponds to the aggregated value of the q^{th} QoS criterion across all the selected concrete services belonging to the composition CC. The aggregation method depends on the nature of the QoS criterion and the composition patterns. Table III.2 summarizes four main aggregation functions that are commonly used (summation, product, minimum, and maximum) in the context of service composition [Khanouche et al., 2019a, Alrifai et al., 2012].

2.3.3 Global QoS constraints

Global QoS constraints play an essential role in the service composition process, serving as guidelines to ensure that the resulting composite service meets the user's non-functional requirements. These constraints represent the upper and lower bounds imposed on the QoS attributes of the composite service, where the upper limits bound the attributes

Table III.2: The aggregation of QoS attributes for different composition patterns.

QoS attributes	Aggregation pattern			
	Sequential	Parallel	Conditional	Loop
Summation	$\sum_{j=1}^{m} q_{q,j}$	$\sum_{j=1}^{m} q_{q,j}$	$\sum_{j=1}^{m} q_{q,j} * p_j$	$n * q_{q,j}$
Product	$\prod_{j=1}^{m} q_{q,j}$	$\prod_{j=1}^{m} q_{q,j}$	$\sum_{j=1}^{m} q_{q,j} * p_j$	$q_{q,j}^{n}$
Minimum	$\sum_{j=1}^{m} q_{q,j}$	$min_j(q_{q,j})$	$\sum_{j=1}^{m} q_{q,j} * p_j$	$n * q_{q,j}$
Maximum	$\sum_{j=1}^{m} q_{q,j}$	$max_j(q_{q,j})$	$\sum_{j=1}^{m} q_{q,j} * p_j$	$n * q_{q,j}$

to maximize, and those to minimize are bounded by lower limits [Alrifai et al., 2012]. For example, in an e-commerce application, users may require a payment service that minimizes transaction costs while ensuring high reliability, providing seamless and efficient processing of purchases. Global QoS constraints guide the optimization process, leading the composition algorithm to search for solutions that not only maximize the QoS utility but also respect these predefined limits.

2.4 Utility function

In an IoT environment, each service s is assessed through a utility value $f(s)$, defined as:

$$f(s) = \sum_{q=1}^{k} \omega_q \cdot qos_q{}'(s) \tag{III.4}$$

where ω_q is the user's preference assigned to the q^{th} QoS attribute within a specific application domain, and $qos_q{}'(s)$ is the normalized value of the q^{th} QoS criterion, computed as follows:

$$qos_q{}'(s) = \begin{cases} \frac{qos_q(Max) - qos_q(s)}{qos_q(Max) - qos_q(Min)} & \text{if } q \in Q^- \\ \frac{qos_q(s) - qos_q(Min)}{qos_q(Max) - qos_q(Min)} & \text{if } q \in Q^+ \end{cases} \tag{III.5}$$

such as $qos_q(Max)$ and $qos_q(Min)$ represent the maximum and minimum values of the q^{th} QoS attribute, while $qos_q(s)$ corresponds to the current value of this attribute. This normalization ensures that each QoS metric is mapped to a comparable scale before aggregation.

2.5 Energy-efficient and QoS-aware service composition

In IoT environments, users often specify their non-functional requirements as constraints on the QoS parameters of the composite service [Khanouche et al., 2019a]. These constraints typically take the form of *lower bounds* for negative QoS attributes (Q^-), and

upper bounds for positive QoS attributes (Q^+). The objective of the energy-efficient and QoS-aware service composition issue is to identify the appropriate concrete services to realize each requested functionality in such a way that the resulting composition: (*i*) complies with the needed global QoS constraints, (*ii*) achieves the highest utility value in terms of QoS, and (*iii*) reduces as effectively as possible the energy consumption of the selected IoT services.

3 Group teaching optimization method

The Group Teaching Optimization (GTO) algorithm is a novel meta-heuristic inspired by the group teaching mechanism [Zhang and Jin, 2020]. This algorithm operates through four main stages: (*i*) teacher assignment phase, (*ii*) ability grouping phase, (*iii*) teacher learning phase, and (*iv*) student learning phase. Note that the GTO algorithm requires only the definition of the population size and a stopping condition, without specific control parameters, making it suitable for optimization across various problem domains.

3.1 Teacher assignment

The role of teacher assignment is to enhance students' learning performance within the population. The three best students in terms of fitness value are chosen to guide the optimization process. The teacher at iteration t, denoted by T^t, is defined by the equation:

$$T^t = \begin{cases} x^t_{First} & \text{if } x^t_{First} > C \\ C & \text{otherwise,} \end{cases} \tag{III.6}$$

$$C = \frac{x^t_{First} + x^t_{Second} + x^t_{Third}}{3} \tag{III.7}$$

where x^t_{First}, x^t_{Second}, and x^t_{Third} represent the best three students in terms of fitness value in the current population.

3.2 Ability grouping

The students in the population are categorized into two distinct subgroups according to their fitness values, which reflect their learning performance. The first subgroup, called the *outstanding group*, consists of students with the highest fitness values, reflecting a high ability to learn new knowledge. The second subgroup, known as the *average group*, includes students with relatively lower fitness values, representing a lower capacity to acquire knowledge.

3.3 Teacher learning

The teacher uses different teaching strategies adapted to the abilities of the two aforementioned groups of students. The aim is to optimize knowledge dissemination based on

each group's learning potential. More specifically, *the teacher learning I* is designed to enhance the outstanding group, while *the teacher learning II* is devoted to improve the average group.

During *Teacher learning I*, a teacher aims to enhance the knowledge of the entire class as much as possible. Consequently, students belonging to the outstanding group can improve their knowledge using the following equation :

$$X_{Teacher,i}^{t+1} = X_i^t + a \times T^t - F \times (b \times M^t + c \times X_i^t) \tag{III.8}$$

$$M^t = \frac{1}{N} \sum_{i=1}^{n} X_i^t \tag{III.9}$$

$$b + c = 1 \tag{III.10}$$

where X_i^t denotes the knowledge of the i^{th} student at iteration t, N is the number of students, T^t is the knowledge of the current teacher, and M^t represents the average knowledge of students at iteration t. The value F is a teaching factor set to either 1 or 2, while a, b, and c are random values within the interval $[0, 1]$. The outcome, $X_{Teacher,i}^{t+1}$, reflects the student's updated knowledge derived from the teaching process.

The teacher learning II addresses students with lower learning performance who receive focused support from the teacher to enhance their knowledge through the following formula:

$$X_{Teacher,i}^{t+1} = X_i^t + 2 \times (d \times (T^t - X_i^t)) \tag{III.11}$$

where d is a random value within the interval $[0, 1]$ that adjusts the degree of knowledge update. When the new knowledge level obtained from this phase fails to enhance the student's performance, the following formula is applied to preserve the better composition:

$$X_{Teacher,i}^{t+1} = \begin{cases} X_{Teacher,i}^{t+1} & \text{if } f(X_{Teacher,i}^{t+1}) > f(X_i^t) \\ X_i^t & \text{otherwise} \end{cases} \tag{III.12}$$

This formula ensures that the search process is not negatively affected by non-productive learning iterations. If the new composition generated during the teacher learning phase does not yield a better fitness value, the algorithm retains the current composition X_i^t. This maintains the best solutions throughout the optimization process and prevents the algorithm from converging prematurely to suboptimal solutions.

3.4 Student learning

Students have the opportunity to improve their knowledge independently during out-of-class time. This can occur either through self-learning or via peer interactions. Formally, the update of a student's knowledge is done as follows:

$$X_{Student,i}^{t+1} = \begin{cases} X_{Teacher,i}^{t+1} + C_1 + C_2 & \text{if } H \\ X_{Teacher,i}^{t+1} - C_1 + C_2 & \text{otherwise} \end{cases} \tag{III.13}$$

$$H = f(X^{t+1}_{Teacher,i}) \geq f(X^{t+1}_{Teacher,j}) \tag{III.14}$$

$$C_1 = e \times (X^{t+1}_{Teacher,i} - X^{t+1}_{Teacher,j}) \tag{III.15}$$

$$C_2 = g \times (X^{t+1}_{Teacher,i} - X^{t}_{i}) \tag{III.16}$$

where e and g are random values within the interval $[0, 1]$. The term $X^{t+1}_{Student,i}$ denotes the knowledge of the i^{th} student at iteration $(t + 1)$, obtained through learning during the student learning phase. Similarly, $X^{t+1}_{Teacher,j}$ corresponds to the knowledge gained by the j^{th} student from the teacher in the previous teacher learning phase. In cases where a student fails to improve their knowledge through this interaction phase, the following adjustment is made using equation (III.17):

$$X^{t+1}_{i} = \begin{cases} X^{t+1}_{Teacher,i} & \text{if } f(X^{t+1}_{Teacher,i}) \geq f(X^{t+1}_{Student,i}) \\ X^{t+1}_{Student,i} & \text{otherwise} \end{cases} \tag{III.17}$$

4 The proposed GT-EQCA approach

Several existing service composition approaches have notable shortcomings, particularly their limited ability to address QoS and energy issues simultaneously. To overcome these limitations, this thesis proposes a Group Teaching-based Energy-efficient and QoS-aware Composition Algorithm (GT-EQCA) [Hameche et al., 2024a] for IoT service environments. The proposed approach has three phases: (*i*) Pruning concrete IoT services that do not meet the user's QoS requirements, (*ii*) Selecting the *top-k* concrete IoT services based on the concept of relative Pareto dominance, and (*iii*) Finding the composite service that provides sub-optimal QoS (i.e., a composition that satisfies the global QoS constraints while maximizing the utility value with respect to energy and QoS) using the GTO algorithm (see Figure III.4).

4.1 Pruning of concrete IoT services

This phase filters out IoT candidate services that do not satisfy the user's QoS requirements. The process of selecting appropriate IoT services based on multiple QoS criteria constitutes a multi-objective combinatorial optimization problem [Khanouche et al., 2016]. To address such issues, several resolution techniques have been proposed in the literature, such as the *lexicographic optimization method*. The latter decomposes a multi-objective optimization problem into a series of single-objective sub-problems, each dealing with a single QoS criterion. The sub-problems are then solved sequentially, following a predefined priority order among the criteria [Fishburn, 1974] (see Figure III.5).

The lexicographic optimization method is particularly applicable when there is an order of importance between the objectives of a given problem. In this method, the first objective is given priority over the second, the second over the third, and so

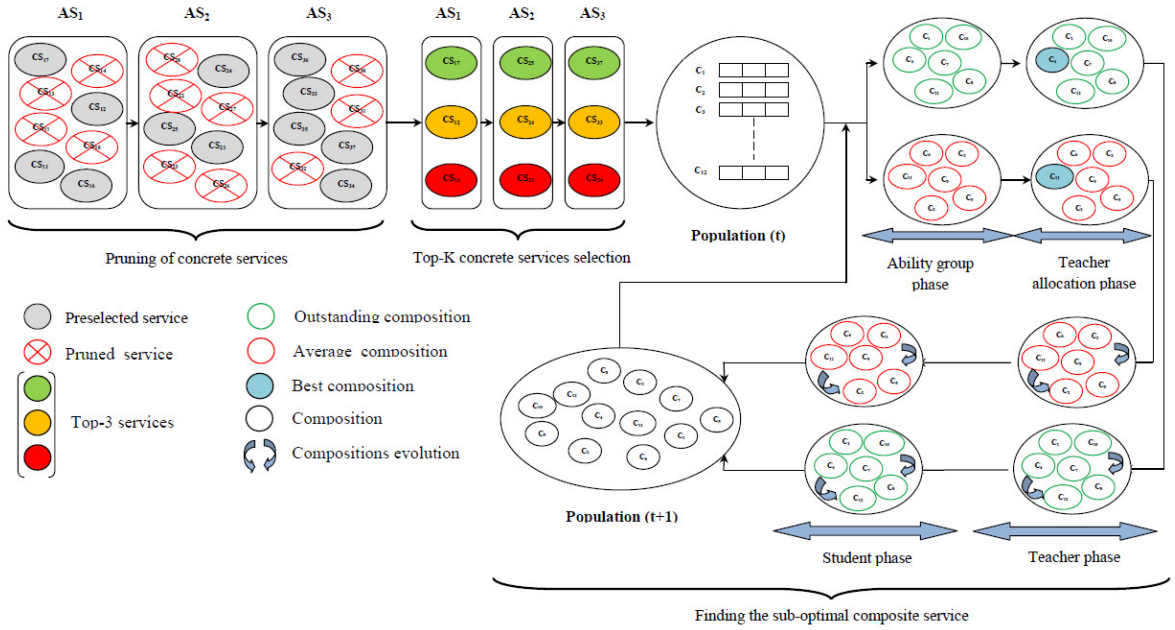


Figure III.4: Phases of the GT-EQCA algorithm.

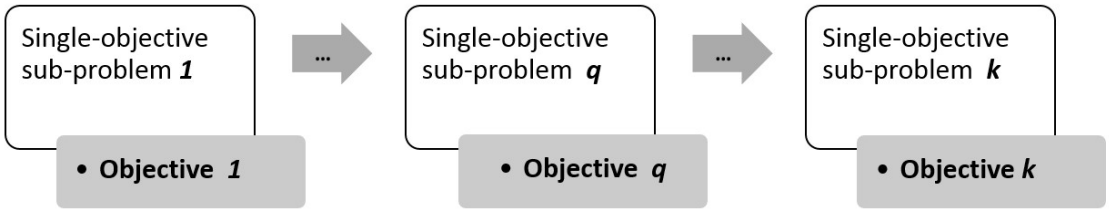


Figure III.5: Multi-criteria lexicographic optimization technique.

forth. In the context of IoT service selection, the priority order of the problems' objectives corresponds to the user's preferences attributed to the QoS criteria. This makes the lexicographic optimization method well-suited for solving QoS-aware IoT service selection problems. Moreover, existing studies have demonstrated the effectiveness of this method in significantly reducing the search space when selecting IoT services [Khanouche et al., 2019a, Khanouche et al., 2016].

In this phase, the QoS criteria are ranked in ascending order of the user's preferences (line 1 of Algorithm 1). The most important criterion is assigned the highest priority (rank 1), whereas the less important criterion receives the lowest priority (rank k). For each abstract service AS^i within the composition, the concrete service cs^i_j that has the optimal value $qos^i_q(Optimal)$ for the highest-priority QoS criterion is first identified (line 5 of Algorithm 1). Based on this optimal value, a threshold is then computed to filter out concrete services that fail to meet the expected user's QoS level. This threshold represents a lower bound of quality $qos^i_q(LowB)$ for positive criteria ($q \in Q^+$, line 8) or an upper bound $qos^i_q(UppB)$ for negative ones ($q \in Q^-$, line 15). Formally:

$$qos^i_q(LowB) = qos^i_q(Optimal) - qos^i_q(Decrease) \tag{III.18}$$

$$qos_q^i(UppB) = qos_q^i(Optimal) + qos_q^i(Decrease) \tag{III.19}$$

where $qos_q^i(Decrease)$ denotes the acceptable reduction of quality with respect to the q^{th} QoS criterion, defined as:

$$qos_q^i(Decrease) = qos_q^i(Optimal) \cdot TF_q \tag{III.20}$$

Such as, TF_q is the tolerance factor that measures the acceptable reduction of the quality regarding the q^{th} criterion, ranging between 0 and 1, and varies according to the user's QoS preferences. The goal of this reduction in quality is to conserve energy without compromising the user's satisfaction. This improves service availability and extends the composition lifetime. Finally, candidate services that do not satisfy the obtained threshold are filtered out to reduce the search space and composition time (lines 9–13 and 16–20 of Algorithm 1).

This filtering process is repeated for each QoS criterion according to the order of priority. At the end of this phase, for each abstract service AS_i, a set of preselected concrete services $PCS(AS_i)$ is obtained that meet the user's QoS requirements. The steps of the first phase of the GT-EQCA algorithm are detailed in Algorithm 1.

4.2 Top-k selection of concrete IoT services

Assuming two concrete IoT services cs_u^i and cs_v^i characterized by their respective QoS vectors $QoS(cs_u^i) = (q_{1,u}^i, ..., q_{q,u}^i, ..., q_{k,u}^i)$ and $QoS(cs_v^i) = (q_{1,v}^i, ..., q_{q,v}^i, ..., q_{k,v}^i)$ and belonging to the preselected service set $PCS(AS^i)$ associated with the abstract service AS^i, where all criteria are considered to have equal priority.

4.2.1 Pareto dominance based on QoS

A concrete service cs_u^i is said to dominate another service cs_v^i with respect to QoS, referred to as $cs_u^i \succ cs_v^i$, if none of the QoS criteria of cs_u^i is worse than a component of $QoS(cs_v^i)$, and at least one criterion is strictly better than a component of $QoS(cs_v^i)$. Formally, $cs_u^i \succ cs_v^i$ holds if (i) for all $q \in Q^+$ (resp. Q^-), $q_{q,u}^i \geq q_{q,v}^i$ (resp. $q_{q,u}^i \leq q_{q,v}^i$), and (ii) there exists at least one $q \in Q^+$ (resp. Q^-) such that $q_{q,u}^i > q_{q,v}^i$ (resp. $q_{q,u}^i < q_{q,v}^i$).

Let us consider the example shown in Table III.3, where four candidate services cs_1^i, cs_2^i, cs_3^i, and cs_4^i are assessed using their energy efficiency and three QoS criteria: availability (AV), response time (RT), and cost (CT). This example shows that for each $j \in [2, 4]$, the concrete service cs_j^i dominates cs_1^i in the Pareto sense, as cs_j^i provides better values for all considered QoS criteria.

4.2.2 QoS attribute-based Pareto dominance

The Pareto dominance score of a concrete service cs_j^i with respect to the q^{th} QoS criterion, denoted as $PD(cs_j^i, q)$, corresponds to the number of services in the set $PCS(AS^i)$ whose value for the q^{th} criterion is worse than that of cs_j^i [Khanouche et al., 2016]. This score

Algorithm 1 Concrete IoT services pruning of the GT-EQCA algorithm.

Require: The composition $AC = \{AS^1, \cdots, AS^m\}$.
The user's preferences $W =< w_1, \cdots, w_k >$.
The tolerance factors $TF =< TF_1, \cdots, TF_k >$.

```
Rank the QoS criteria based on the user's preferences;
for each AS^i ∈ AC do
    PCS(AS^i) = AS^i;
    for q = 1 to k do
        Determine cs^i_j / qos^i_{q,j} = qos^i_q(Optimal) ;
        qos^i_q(Decrease) = qos^i_q(Optimal) · TF_q
        if q ∈ Q^+ then
            Calculate qos^i_q(LowB) using formula III.18;
            for each cs^i_j ∈ AS^i do
                if qos^i_{q,j} < qos^i_q(LowB) then
                    PCS(AS^i) = PCS(AS^i) − {cs^i_j};
                end if
            end for
        else
            Calculate qos^i_q(UppB) using formula III.19;
            for each cs^i_j ∈ AS^i do
                if qos^i_{q,j} > qos^i_q(UppB) then
                    PCS(AS^i) = PCS(AS^i) − {cs^i_j};
                end if
            end for
        end if
    end for
end for
```

Ensure: Preselected concrete IoT services $PCS(AS^i)/i \in [0, m]$

Table III.3: QoS metrics and energy efficiency values for four candidate services.

Services	QoS Criteria			Energy Efficiency
	AV (%)	RT (ms)	CT (€)	
cs^i_1	90	22	12	0.13
cs^i_2	95	21	9	0.80
cs^i_3	93	20	11	0.50
cs^i_4	91	18	10	0.06

takes an integer value ranging from 1 to $|PCS(AS^i)|$. A dominance value equal to 1 indicates that the service has the lowest value for the QoS criterion, while a value equal to $|PCS(AS^i)|$ reflects the best value among all candidate services for the considered QoS attribute.

Table III.4 presents the Pareto dominance scores based on the QoS criterion for the four candidate services described in the previous example. For instance, the highest availability-based Pareto dominance (i.e., AV-based PD) is equal to 4 and is assigned to the candidate service cs_2^i providing the highest availability, whereas the lowest availability-based Pareto dominance, which is equal to 1, is given to the candidate service cs_1^i that has the lowest availability.

Table III.4: QoS attribute and energy efficiency-based dominance scores for four concrete services.

Services	QoS Dominance			EE-based PD	RPD
	AV-based PD	RT-based PD	CT-based PD		
cs_1^i	1	1	1	2	2
cs_2^i	4	2	4	4	13.6
cs_3^i	3	3	2	3	7.5
cs_4^i	2	4	3	1	3.1

4.2.3 Energy efficiency-based Pareto dominance

The energy efficiency-based Pareto dominance of a concrete service cs_j^i, denoted as $PD(cs_j^i, EE)$, measures the number of services within the set $PCS(AS^i)$ that have a lower energy efficiency than the service cs_j^i [Khanouche et al., 2016]. Formally:

$$PD(cs_j^i, EE) = |Z| \tag{III.21}$$

$$Z = \{cs_k^i \in PCS(AS^i) \ / \ EE(cs_j^i) > EE(cs_k^i)\} \tag{III.22}$$

The value $|PCS(AS^i)|$ represents the highest value of Pareto dominance based on energy efficiency and is attributed to the concrete service having the highest energy efficiency. In contrast, the value 1 represents the lowest value of Pareto dominance based on energy efficiency and is assigned to the concrete service with the lowest energy efficiency in the set $PCS(AS^i)$.

In the previous example, the concrete service cs_4^i achieves the highest energy efficiency. It is then assigned the highest energy efficiency-based Pareto dominance (i.e., EE-based PD), which is equal to 4. Conversely, the concrete service cs_2^i, having the lowest energy

efficiency among all of the services, is associated with the lowest energy efficiency-based Pareto dominance, which is equal to 1 (see Table III.4).

4.2.4 Relative Pareto dominance

The relative Pareto dominance score $RPD(cs_j^i)$ for a concrete service cs_j^i is a scalar value calculated from three key components: the energy efficiency-based Pareto dominance $PD(cs_j^i, EE)$, the Pareto dominance values associated with various QoS criteria, and the user's preferences ω_q with respect to each QoS criterion q [Khanouche et al., 2016]. Formally:

$$RPD(cs_j^i) = PD(cs_j^i, EE) \cdot \sum_{q=1}^{k} PD(cs_j^i, q) \cdot \omega_q \tag{III.23}$$

In the example above, assuming that 0.2, 0.3, and 0.5 are weights associated with availability, response time, and cost, respectively. The RPD values for the four concrete services are shown in Table III.4.

4.2.5 Top-k concrete services

The *top-k*(AS^i) concrete services, associated with the abstract service AS^i, refer to the k best concrete services in terms of relative Pareto dominance extracted from the preselected candidate set $PCS(AS^i)$. These services are considered the most suitable to balance QoS and energy efficiency. For instance, in the previous example, the *top-2*(AS^i) concrete services with respect to relative Pareto dominance is the set including services cs_2^i and cs_3^i.

The second phase of the GT-EQCA algorithm focuses on refining the selection process by retaining only the *top-k* concrete services from each abstract service AS^i for the subsequent composition phase. This strategy significantly reduces the search space and computation time required by the algorithm while improving the overall QoS utility of the resulting composition. The value of k is carefully selected to reduce composition time, while maximizing composition quality and minimizing energy consumption. The algorithm 2 describes the steps of finding the *top-k* concrete services for each abstract service involved in the composition.

4.3 Finding of the sub-optimal composite service

The third phase of the GT-EQCA algorithm aims to identify the sub-optimal composition that offers the best balance between energy efficiency and QoS, using the GTO method. In this context, each student $X_s = \{x_{j,s}^1, \cdots, x_{j,s}^i, \cdots, x_{j,s}^D\}$ within the population corresponds to a composite service $C_s = \{x_{j,s}^1, \cdots, x_{j,s}^i, \cdots, x_{j,s}^m\}$, where $x_{j,s}^i$ represents the index of the j^{th} concrete service for the i^{th} abstract service in the s^{th} composition. The teacher refers to the composite service having the best utility in terms of energy and

Algorithm 2 Top-k concrete IoT services selection of the GT-EQCA algorithm

Require: The composition set $AC = \{PCS^1, \cdots, PCS^m\}$,
An integer k,
User preference weights $W = \langle w_1, \cdots, w_k \rangle$.

for each abstract service $AS^i \in AC$ **do**
  for each candidate service $cs^i_j \in PCS(AS^i)$ **do**
    for each QoS criterion q **do**
      Compute $PD(cs^i_j, q)$;
    end for
    Compute $PD(cs^i_j, EE)$ using Eq. (III.21);
    Calculate relative Pareto dominance $RPD(cs^i_j)$ using Eq. (III.23);
  end for
  Select the *top-k*(AS^i) services based on relative Pareto dominance;
end for

Ensure: The *top-k* concrete services for each $PCS(AS^i)$ where $i \in [1, m]$.

QoS. Table III.5 illustrates the correspondence between the GT-EQCA algorithm and the energy and QoS-aware services composition problem.

Table III.5: The GT-EQCA algorithm versus the energy and QoS-aware service composition problem.

GT-EQCA algorithm	Energy and QoS-aware services composition
Class	The population of compositions
Student	A composite service in the population
Knowledge	The indexes of candidate services in composition
Teacher	The best composition in terms of a QoS utility value and energy
Fitness function	Utility value in terms of QoS and energy

The GT-EQCA algorithm does not require the adjustment of any specific parameters other than only two basic ones: the population size N and the maximum number of iterations T_{Max}. A population $P(t)$ of N feasible compositions (i.e., compositions that meet the QoS constraints) is first randomly generated by considering the *top-k* candidate services for each abstract service within the composition. Each feasible composition is denoted as $C_s = \{x^1_{j,s}, \cdots, x^i_{j,s}, \cdots, x^m_{j,s}\}$, where m represents the total number of abstract services, and $x^i_{j,s}$ refers to the chosen concrete service for the i^{th} abstract service in the s^{th} composition. This population is then iteratively improved to find a sub-optimal composition in terms of energy consumption and QoS. The GT-EQCA algorithm proceeds through four phases.

In the *teacher assignment phase*, the algorithm computes the utility value of each composition in terms of energy efficiency and QoS. The three compositions with the highest utility value in the population are selected to guide the remaining compositions.

The teacher composition $C^t_{Teacher}$ is then determined using the following formula:

$$C^t_{Teacher} = \begin{cases} C^t_{First} & if \quad C^t_{First} > C \\ C & otherwise, \end{cases} \tag{III.24}$$

$$C = \frac{C^t_{First} + C^t_{Second} + C^t_{Third}}{3} \tag{III.25}$$

where C^t_{First}, C^t_{Second}, and C^t_{Third} denote the compositions with the first, second, and third best utility value in terms of energy and QoS

In the *ability grouping phase*, the goal is to guide the search process using the teacher composition more effectively. Accordingly, the compositions are sorted according to their utility values and divided into two distinct groups. The best half compositions form the outstanding composition group C^t_{Good} and the remaining compositions constitute the average composition group C^t_{Bad}. To enhance the convergence speed of the algorithm, both groups use the same teacher throughout their evolution.

In the *teacher learning phase*, each composition within the outstanding group C^t_{Good} undergoes an evolution process guided by the teacher composition $C^t_{Teacher}$ and the average composition C^t_{Mean} of the group. Specifically, the improvement of the compositions belonging to the outstanding group is determined by the following equations :

$$C^{t+1}_{Teacher,i} = C^t_i + a \times L \tag{III.26}$$

$$L = (C^t_{Teacher} - F \times (b \times C^t_{Mean} + c \times C^t_i)) \tag{III.27}$$

$$C^t_{Mean} = \frac{1}{N} \sum_{i=1}^{n} C^t_i \tag{III.28}$$

$$b + c = 1 \tag{III.29}$$

where N denotes the total number of compositions in the population, and C^t_i along with $C^t_{Teacher}$ are respectively the i^{th} and the teacher composition at iteration t. The term C^t_{Mean} represents the average composition calculated from the outstanding group, while F is a teaching factor that takes either the value 1 or 2. The composition $C^{t+1}_{Teacher,i}$ corresponds to the composition i improved by the teacher at iteration t. The parameters a, b, and c are randomly selected from the interval $[0, 1]$. For the group C^t_{Bad}, which contains the less effective compositions, the teacher focuses more on the average group than on the outstanding group. In this context, each composition of the average group is improved using the following equation:

$$C^{t+1}_{Teacher,i} = C^t_i + 2 \times (d \times (C^t_{Teacher} - C^t_i)) \tag{III.30}$$

where d is a randomly generated value within the interval $[0, 1]$. Moreover, suppose a composition shows no improvement after the teacher phase. In this case, a verification

step is performed to determine whether the updated composition should be retained or replaced, as defined by the following formula :

$$C_{Teacher,i}^{t+1} = \begin{cases} C_{Teacher,i}^{t+1} & if\ f(C_{Teacher,i}^{t+1}) > f(C_i^t) \\ C_i^t & otherwise \end{cases} \tag{III.31}$$

During *the student learning phase*, each composition in the C_{Good}^t and C_{Bad}^t groups is enhanced using two distinct strategies. The first strategy improves the composition through its components, while the second strategy enhances the composition through its neighboring ones. Formally, these two improvement mechanisms are defined as follows:

$$C_{Student,i}^{t+1} = \begin{cases} C_{Teacher,i}^{t+1} + C_1 + C_2 & if\ H \\ C_{Teacher,i}^{t+1} - C_1 + C_2 & otherwise \end{cases} \tag{III.32}$$

$$H = f(C_{Teacher,i}^{t+1}) \geq f(C_{Teacher,j}^{t+1}) \tag{III.33}$$

$$C_1 = e \times (C_{Teacher,i}^{t+1} - C_{Teacher,j}^{t+1}) \tag{III.34}$$

$$C_2 = g \times (C_{Teacher,i}^{t+1} - C_i^t) \tag{III.35}$$

where e and g are two randomly generated numbers between 0 and 1. The term $C_{Student,i}^{t+1}$ represents the composition C_i improved at iteration t from the student phase. Meanwhile, $C_{Teacher,j}^{t+1}$ refers to the version of composition C_j that was updated during the teacher phase. In the case where a composition does not gain improvements through the student phase, an alternative update formula is applied as follows:

$$C_i^{t+1} = \begin{cases} C_{Teacher,i}^{t+1} & \text{if } f(C_{Teacher,i}^{t+1}) \geq C_{Student,i}^{t+1} \\ C_{Student,i}^{t+1} & \text{if } f(C_{Teacher,i}^{t+1}) < C_{Student,i}^{t+1} \end{cases} \tag{III.36}$$

Once the process of finding the sub-optimal composite service is achieved, the algorithm selects and returns the feasible composite service with the highest utility value in terms of QoS and energy as the final sub-optimal composition. The steps of the third phase of the GT-EQCA algorithm are presented in Algorithm 3.

4.4 Complexity analysis

The time complexity analysis of the GT-EQCA algorithm is based on five factors: m, the number of service classes that constitute an abstract composite service; n, the number of candidate services per service class (assumed to be the same for all classes); k, the number of QoS attributes; N, the population size; Tx, the maximum number of iterations.

The time complexity of ranking QoS criteria based on user's preferences is logarithmic, that is, $O(k \log k)$. For each service class, the time required to identify the service with the best quality value for a specific QoS attribute is $O(n)$. Similarly, filtering candidate services that do not meet the required QoS threshold values also requires $O(n)$. Thus, the time complexity of the first phase is $O(k \log k) + O(k \cdot m \cdot n)$. However, since the number of QoS attributes, k, is much smaller than the number of service classes m and

Algorithm 3 Sub-optimal composition finding of the GT-EQCA algorithm

Require: The initial population P^0,
The stopping condition TC,
The size of population N,
The *top-k*(AS^i) for each $PCS(AS^i)$ $/i \in [1, m]$.

$t = 0$;
while (TC is not satisfied) **do**
  Calculate the utility values of compositions in P^t;
  Sort the compositions in ascending order of their utility values;
  Divide the population into two sub-population P^t_{Good} and P^t_{Bad};
  Find the outstanding C^t_{Good} and the average C^t_{Bad} compositions;
  Find $C^t_{teacher}$ using Eq. (III.24);
  for each composite service $C^t_s \in P^t_{Good}$ **do**
    Perform the teacher phase using Eq. (III.26)–(III.29) and (III.31);
    Perform the student phase using Eq. (III.32);
    Update C^{t+1}_s using Eq. (III.36);
    Add C^{t+1}_s to the population P^{t+1}_{Good};
  end for
  for each composite service $C^t_s \in P^t_{Bad}$ **do**
    Perform the teacher phase using Eq. (III.30) and (III.31);
    Perform the student phase using Eq. (III.32);
    Update C^{t+1}_s using Eq. (III.36);
    Add C^{t+1}_s to the population P^{t+1}_{Bad};
  end for
  $P^{t+1} = P^{t+1}_{Good} + P^{t+1}_{Bad}$;
  $t = t + 1$;
end while

Ensure: A new population of composite services.

the number of concrete services n, the overall time complexity of the first phase becomes $O(n \cdot m)$.

The second phase involves ranking concrete services based on dominance, with time complexity $O(k \cdot p \log p)$, where p is the number of candidate services obtained from the first phase. Similarly, calculating the dominance based on the energy efficiency has a time complexity of $O(p \log p)$. Furthermore, identifying the candidate service that maximizes relative dominance has time complexity $O(p \log p)$. For a composition with m service classes, the time complexity of the second phase is $O(k \cdot m \cdot p \log p)$. Given that the number of QoS criteria k is much smaller than the number of service classes m, the complexity of the second phase becomes $O(m \cdot p \log p)$.

The main computational time in the third phase depends on performing and updating candidate services during the teacher and student learning phases. Each phase has a time

complexity of $O(m)$, and for each composition in the population, the time complexity for both phases is $O(m)$. For the whole population, the time complexity is $O(N \cdot m)$. Taking into account the maximum number of iterations, Tx, the total time complexity of the third phase becomes $O(Tx \cdot N \cdot m)$.

In summary, the overall time complexity of the GT-EQCA algorithm is on the order of $O(m.n) + O(m.p \log p) + O(Tx.N.m)$.

5 Performance study

This section analyses the performance of the proposed GT-EQCA algorithm. First, we examine how different values of the tolerance factor and *top-k* influence its performance. Then, we compare its performance with that of four relevant service composition approaches.

5.1 Simulation environment and dataset

A series of large-scale service composition scenarios was conducted. The simulations were carried out using MATLAB R2015b on a 64-bit Windows operating system, an Intel Core i5-5005U processor at 2.20 GHz, and 8 GB of RAM. To simplify the analysis while preserving generality, all simulated compositions adopt a sequential execution model. This choice is justified by the fact that other composition patterns, such as loops, conditional, and parallel branches, can be represented using a sequential structure [Alrifai et al., 2012, Geebelen et al., 2014]. The sequential model thus provides a unified pattern for the simple analysis and interpretation of the composition process.

The simulation scenarios use the QWS dataset [Al-Masri and Mahmoud, 2008], which provides values for nine QoS criteria for 2507 services. The QoS parameters considered in the simulation scenarios include response time, availability, reliability, throughput, and success rate. Since existing datasets do not simultaneously contain QoS criterion values and energy features of IoT services, the realistic energy model proposed in [Furthmüller and Waldhorst, 2012] is used to define the energy characteristics of the services in the QWS dataset. Each device is initialized with a battery level $C_{initial}$ randomly chosen from the interval $[0.7 \cdot C_{max}, C_{max}]$, where $C_{max} = 1500\ mA.h$ denotes the maximum battery charge. The energy consumed by an IoT service for every invocation is uniformly distributed from the interval $[10^2\ mA \cdot s, 10^4\ mA \cdot s]$. After invocation, the residual energy is updated accordingly. To ensure energy efficiency during composition, only services hosted on devices with residual energy higher than the threshold $C_{Threshold} = 0.3 \cdot C_{max}$ are considered. The integration of this energy model with the QWS dataset results in a realistic dataset suitable for evaluating large-scale IoT service composition algorithms.

5.2 Baselines and performance metrics

We evaluate the performance of the GT-EQCA algorithm by comparing them with four well-known service composition baseline approaches: the Teaching Learning optimization-based Services Composition (TL-SC) approach [Deng et al., 2016b], the Genetic Algorithm-based Services Composition (GA-SC) approach [Deng et al., 2017b], the Particle Swarm optimization-based Services Composition (PS-SC) approach [Khanam et al., 2018], and the Particle Swarm optimization and Gray Wolf optimizer-based Services Composition (PSGW-SC) approach [Sun et al., 2019b]. For each simulation scenario, the results are averaged over 50 independent runs, and the comparative analysis is based on the following key performance metrics:

- *Composition time:* indicates the time required by each algorithm (GT-EQCA, PSGW-SC, GA-SC, PS-SC, and TL-SC) to determine the sub-optimal composite service.
- *Energy consumption of the composite service:* refers to the total energy consumed by the set of concrete services selected in the sub-optimal composite service.
- *Utility value of the composite service:* represents the overall QoS utility value achieved by the sub-optimal composite service obtained by each algorithm.

5.3 Performance analysis of the GT-EQCA algorithm

The influence of the tolerance factor and the *top-k* parameter on the performance of the GT-EQCA algorithm is evaluated through a service composition scenario involving 5 abstract services, each associated with a number of concrete services varying from 2000 to 10000 and randomly selected from the QWS dataset in each simulation scenario. Furthermore, five QoS constraints are imposed on the overall composition and are determined using a statistical approach that considers the average QoS values of candidate services for each abstract service in the composition [Cho et al., 2015]. The population size is set to 20 compositions, and the algorithm is executed for a maximum of 100 iterations.

5.3.1 Impact of the tolerance factor values

This simulation scenarios investigate the impact of the tolerance factor on the preselection time, energy consumption, and the QoS utility value achieved by the GT-EQCA algorithm.

As illustrated in Figure III.6, the tolerance factor has a positive correlation with the preselection time. Specifically, when the tolerance factor is relatively low (i.e., $TF \leq 0.4$), the GT-EQCA algorithm requires less time for the preselection phase. In contrast, when the tolerance factor exceeds 0.4 (i.e., $TF > 0.4$), a large number of candidate services satisfy the QoS threshold values, resulting in a significant increase in the preselection time required by the GT-EQCA algorithm.

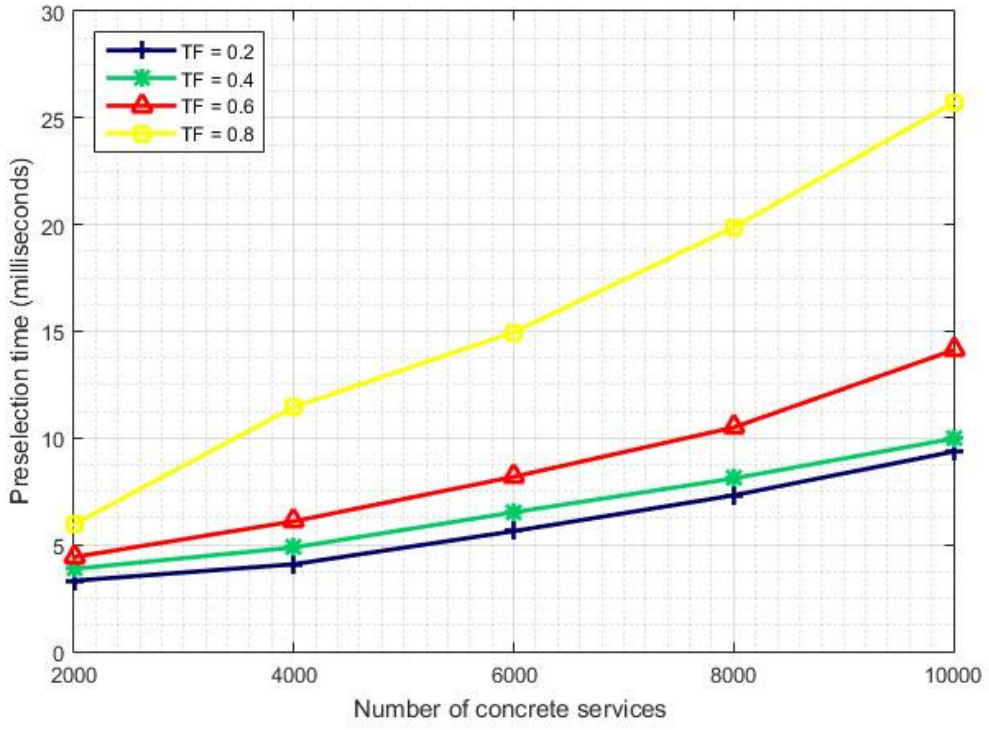


Figure III.6: Impact of the tolerance factor on the preselection time for different numbers of candidate services.

As shown in Figure III.7, the energy consumption of the composition is inversely proportional to the tolerance factor. Specifically, energy consumption tends to decrease when the tolerance factor is relatively high (i.e., $TF > 0.4$) and to increase when it is low (i.e., $TF \leq 0.4$). A higher tolerance factor increases the probability of identifying candidate services with higher energy efficiency, thus reducing the overall energy consumption of the composite service. However, this benefit results in a significant increase in the preselection time (see Figure III.6).

Figure III.8 shows that the QoS utility of the composite service is relatively high when the tolerance factor is low and decreases slightly as this factor increases. However, this utility remains stable as the number of concrete services increases. This stability can be explained by the fact that the preselection process effectively selects the most relevant candidate services using the lexicographic ordering and the *top-k* selection mechanism. This approach promotes services with higher QoS attributes, ensuring that only the most appropriate candidate services are retained for the composition process. However, although the utility value is maximized when the tolerance factor is set to 0.2, this results in a significantly high energy consumption.

In summary, a tolerance factor that exceeds 0.4 leads to a significant increase in the preselection time, and the QoS utility of the composition hardly decreases. However, when this factor is lower than 0.4, energy consumption increases substantially. Therefore, setting the tolerance factor at 0.4 offers a good trade-off between energy consumption,

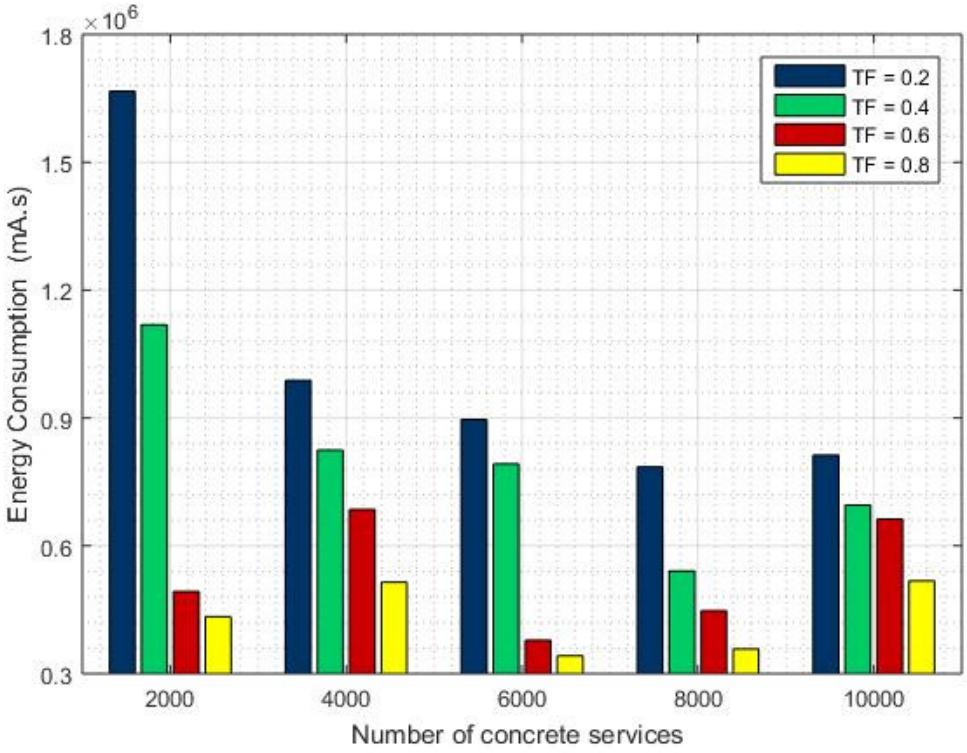


Figure III.7: Impact of the tolerance factor on the energy consumption for different numbers of candidate services.

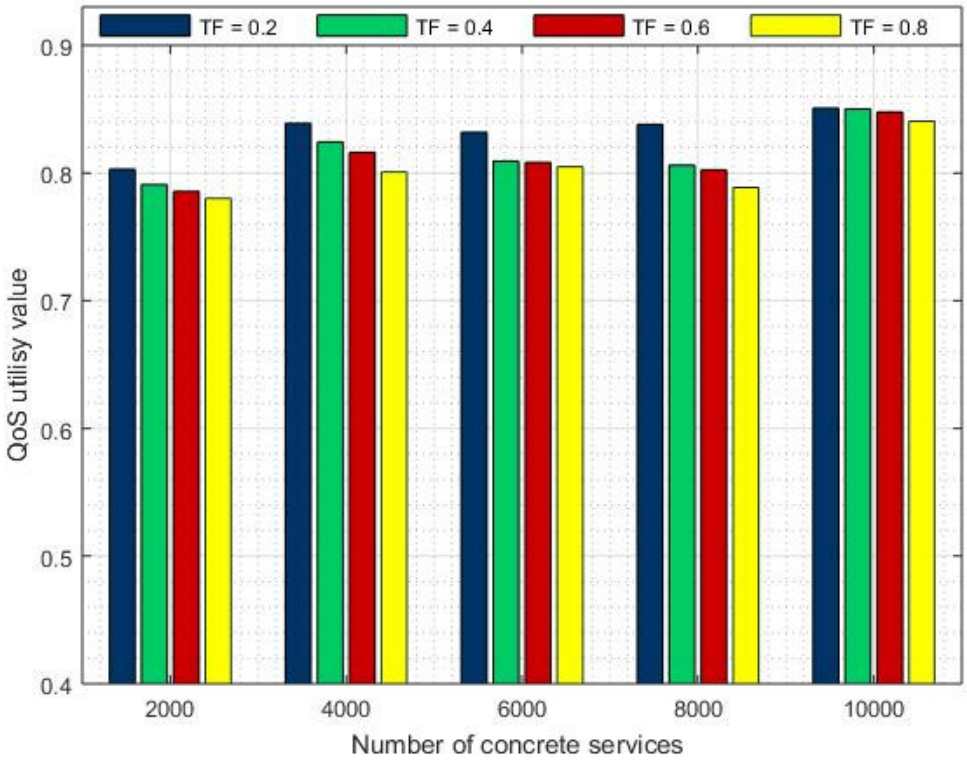


Figure III.8: Impact of the tolerance factor on the composition utility for different numbers of candidate services.

QoS utility, and preselection time in the context of the GT-EQCA algorithm.

5.3.2 Impact of the top-k values

This simulation scenario explores how the variation of the *top-k* parameter influences the composition time, energy consumption, and the QoS utility value of the GT-EQCA algorithm.

As illustrated in Figure III.9, the composition time increases proportionally to the value of the *top-k* parameter, regardless of the number of candidate services. This increase is mainly due to the expansion of the search space as the *top-k* value increases, leading to an increase in composition time. However, the number of candidate services does not significantly affect composition time, since the algorithm primarily selects the best services from the preselected candidates set based on their relevance and quality, rather than exploring the entire set of services. Figure III.10 illustrates that the energy consumption decreases with the *top-k* value. This reduction results from pruning concrete services that are inefficient in terms of energy consumption, therefore improving the overall efficiency of the composition process. Figure III.11 demonstrates that when the *top-k* value decreases, the QoS of the composite service does not decline significantly. This is because the preselection phase effectively filters out services that do not meet the user's QoS constraints, ensuring that only the most suitable services are selected for the composition process.

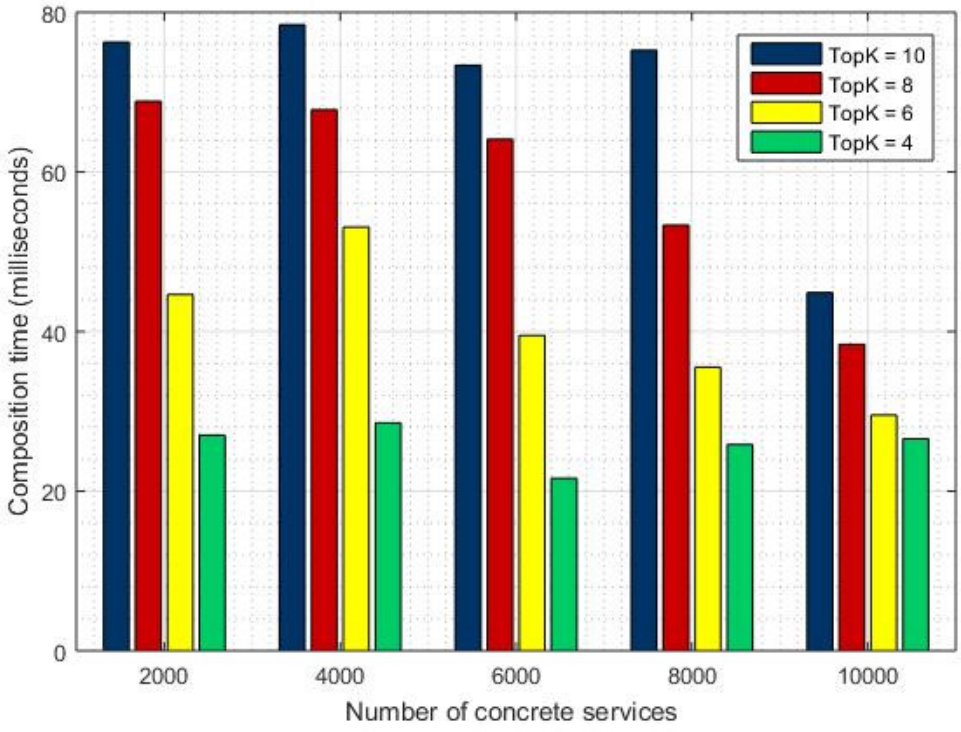


Figure III.9: Impact of the top-k services on the composition time for different numbers of candidate services.

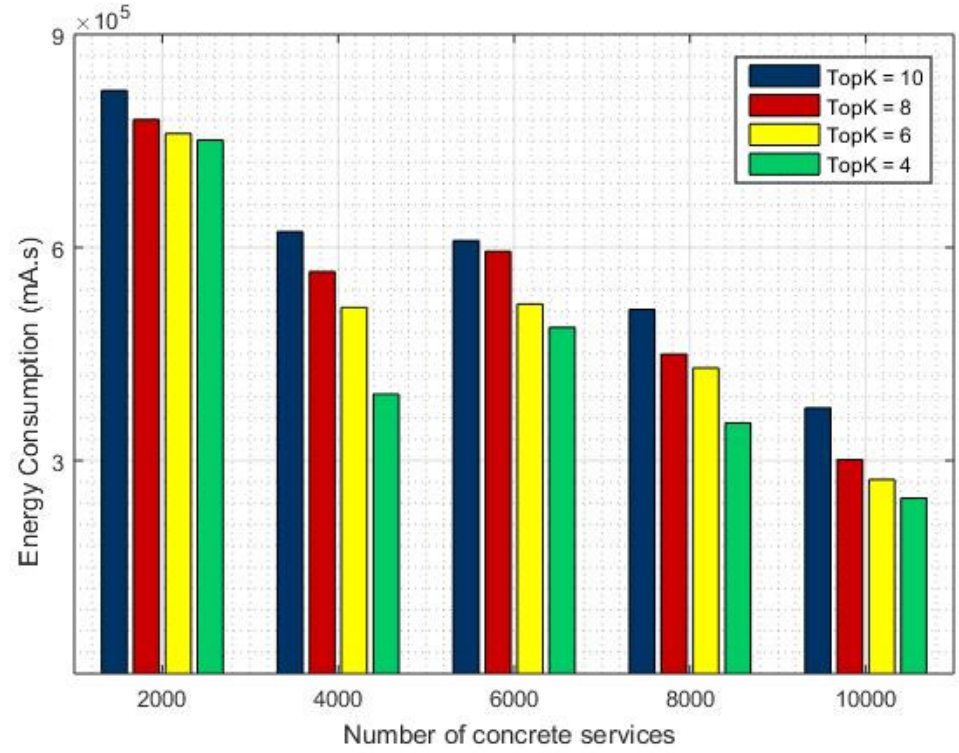


Figure III.10: Impact of the top-k services on the energy consumption for different numbers of candidate services.

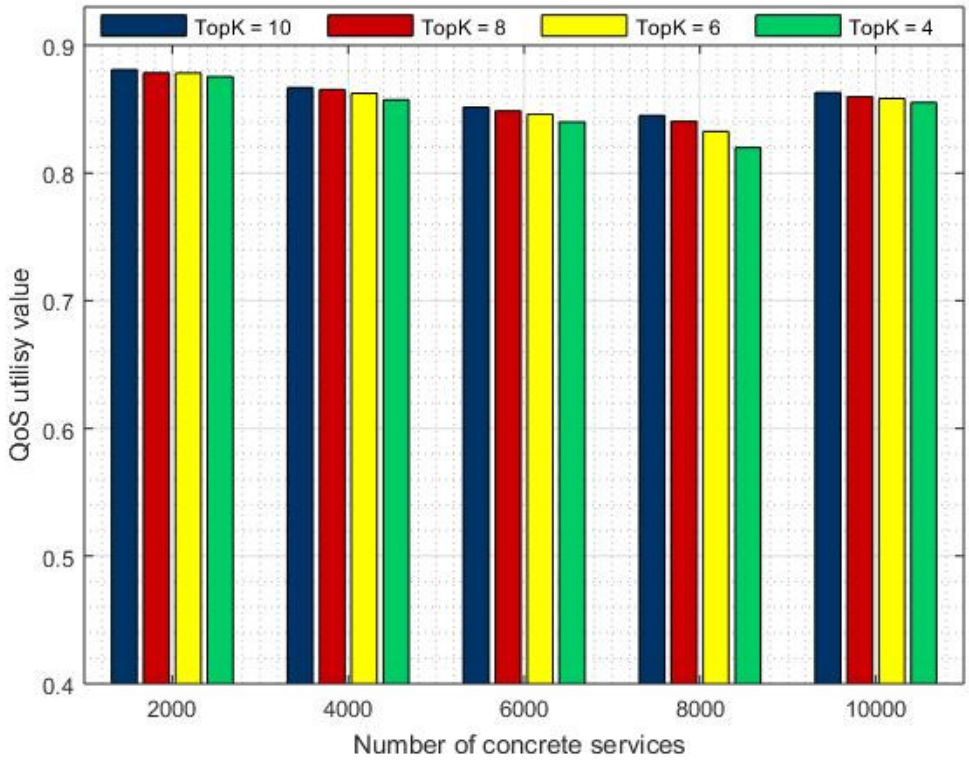


Figure III.11: Impact of the top-k services on the composition utility for different numbers of candidate services.

On the light of this scenario, we can conclude that when *top-k* is equal to 4, the composition time and energy consumption are lower than those obtained in the case of the other values of *top-k*. At the same time, the QoS utility decreases slightly and in an insignificant manner. This means that the QoS of the resulting composite service is not affected by the decrease of the *top-k* value. Accordingly, the best value of *top-k* that gives good performance with the GT-EQCA algorithm is equal to 4.

5.4 Performance comparison and discussion

The simulation scenarios carried out in this section consider a service composition of 5 abstract services, each associated with a randomly selected set of 2000 to 10000 concrete services extracted from the QWS dataset. The number of user's QoS constraints is set to 5, where each constraint is derived using a statistical approach based on the average QoS values of concrete services belonging to each abstract service of the composition [Cho et al., 2015]. All algorithms are evaluated using a population size of 20 compositions and a maximum of 100 iterations. For the GT-EQCA algorithm, the tolerance factor is set to 0.4 and the *top-k* value to 4. For the GA-SC algorithm, the crossover and mutation probabilities are set to 0.7 and 0.3, respectively. For the PS-SC and PSGW-SC algorithms, the inertia weight is set to 0.5, whereas the coefficients C_1 and C_2 are equal to 2.

5.4.1 Composition Time

This simulation scenario aims to evaluate and compare the composition times of the GT-EQCA algorithm with those of the PS-SC, PSGW-SC, TL-SC, and GA-SC algorithms. As illustrated in Figure III.12, the GT-EQCA algorithm outperforms the baseline approaches in terms of composition time, even as the number of concrete services increases. Specifically, the GT-EQCA algorithm can find a sub-optimal composite service in less than 25 milliseconds, while the PS-SC, GA-SC, TL-SC, and PSGW-SC algorithms require over 48, 75, 331, and 581 milliseconds, respectively, when the number of concrete services per abstract service is equal to 10000. The GT-EQCA algorithm demonstrates a significantly faster composition process compared to the PS-SC, GA-SC, TL-SC, and PSGW-SC algorithms. This outcome can be explained by the fact that the strategy adopted in each phase of the proposed algorithm plays a key role in reducing the overall composition time. More precisely, the GT-EQCA algorithm begins with a preselection phase that filters out concrete services that do not satisfy the user's QoS constraints and applies a relative Pareto dominance approach to retain only the most promising concrete services for the composition process. This two-step filtering process significantly reduces the search space, thus accelerating the composition process. Moreover, the exploration mechanism used in the GT-EQCA algorithm is designed to guide the search toward high-quality compositions, leading to rapid convergence and further reducing the time required to find a sub-optimal service composition.

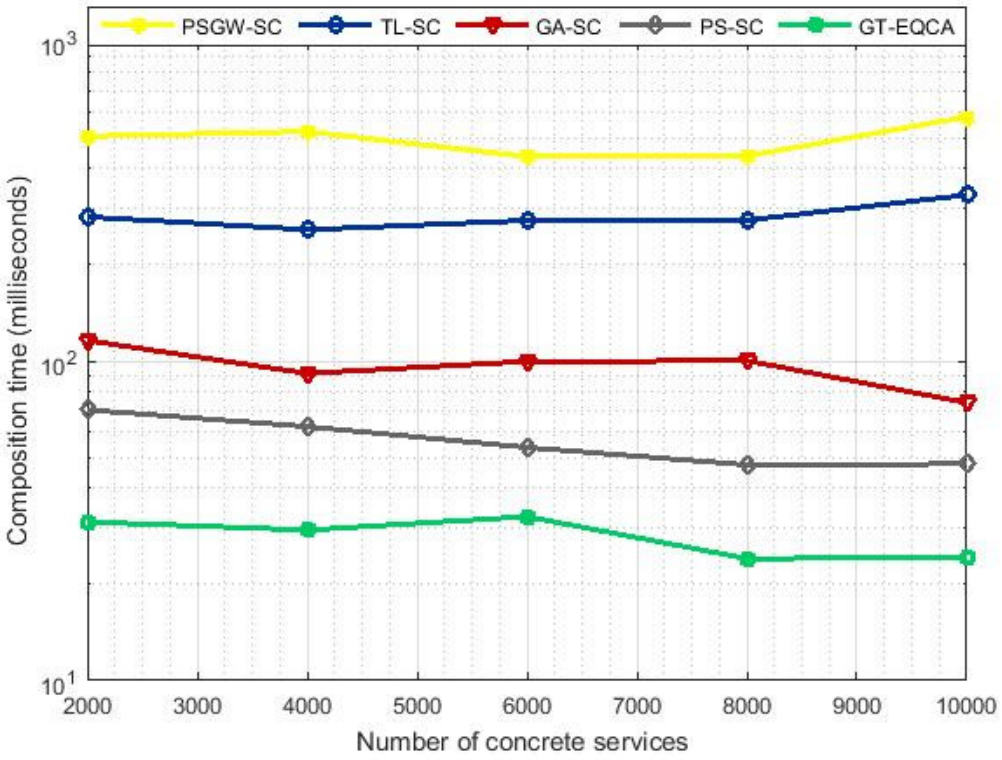


Figure III.12: Impact of the concrete services' number on the composition time.

5.4.2 Energy consumption

This simulation scenario compares the energy consumption of the GT-EQCA algorithm with those of PS-SC, PSGW-SC, GA-SC, and TL-SC algorithms under varying the number of concrete services. As shown in Figure III.13, when each abstract service is associated with 10000 concrete services, the PSGW-SC, GA-SC, PS-SC, and TL-SC algorithms consume approximately 78.66%, 88.84%, 93.1%, and 94.6% more energy than the GT-EQCA algorithm. The main factor contributing to the GT-EQCA algorithm's lower energy consumption is the consideration of the energy profile of concrete services during the selection process. Moreover, applying the relative Pareto dominance principle facilitates the selection of the most energy-efficient concrete services. This principle ensures that the algorithm favors services that not only fulfill the user's QoS constraints but also minimize energy consumption. As a result, the energy savings achieved by the GT-EQCA algorithm enhance the availability of the services involved in the composition and extend its overall lifetime.

5.4.3 Utility value of the composition

This simulation scenario compares the QoS utility obtained with the GT-EQCA algorithm against the TL-SC, PS-SC, PSGW-SC, and GA-SC algorithms under varying numbers of concrete services. As illustrated in Figure III.14, the QoS utility achieved by the GT-EQCA algorithm remains high, ranging from 0.85 to 0.87, as the number of concrete services varies from 2000 to 10000. This utility value is significantly higher compared to

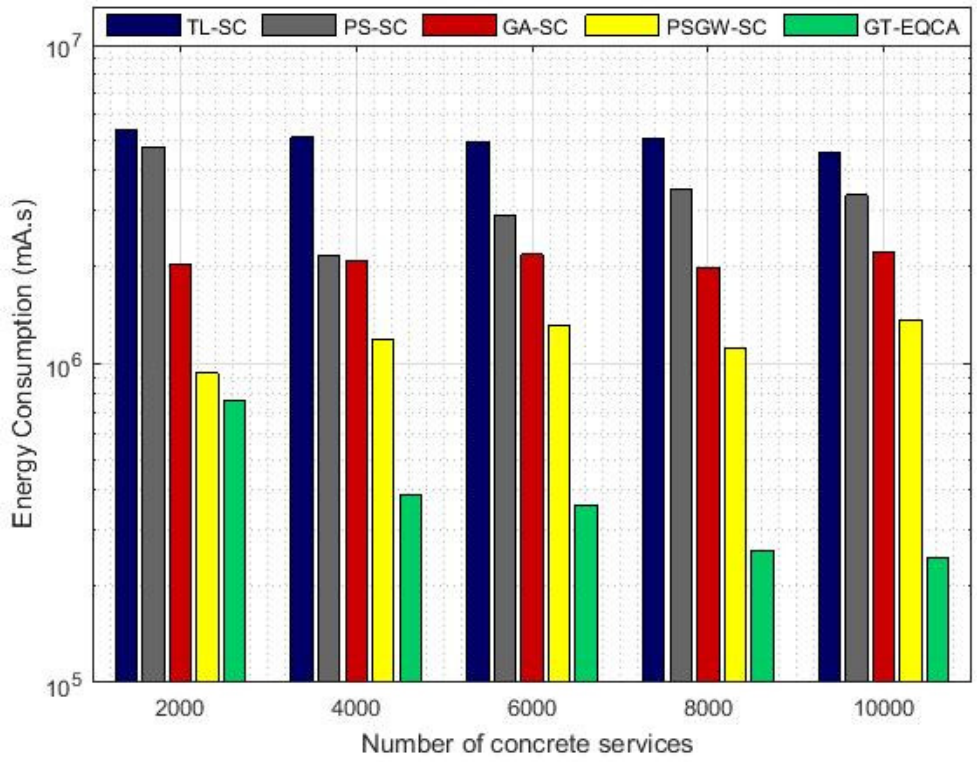


Figure III.13: Impact of the concrete services' number on the energy consumption.

those obtained by the TL-SC, PS-SC, GA-SC, and PSGW-SC approaches, which range, respectively, from 0.69 to 0.70, 0.68 to 0.70, 0.65 to 0.67, and 0.63 to 0.64. This outcome can be explained by the GT-EQCA algorithm's use of the relative Pareto dominance principle to select the *top-k* concrete services with the highest QoS values, enhancing the overall utility of the composition. Moreover, the GT-EQCA algorithm outperforms other approaches by employing a self-learning process and the interaction with other compositions, which avoids an early convergence to a sub-optimal composition in terms of QoS. Additionally, each composition of the GT-EQCA algorithm is improved using its appropriate learning strategy, enabling the generation of new compositions that offer a higher QoS utility value. It should be noted that the PSGW-SC algorithm has the lowest QoS utility among the other algorithms. This approach combines the PSO and GWO methods to guide the service composition process. However, their combination fails to improve the algorithm's exploration or exploitation capabilities, limiting their contribution to improving the composition process.

6 Conclusion

This chapter proposed a novel algorithm to address the energy-efficient and QoS-aware service composition problem within the IoT environment. The proposed GT-EQCA approach enhances composition availability by minimizing the energy consumption of services while satisfying the user's QoS constraints. The process begins with a service prun-

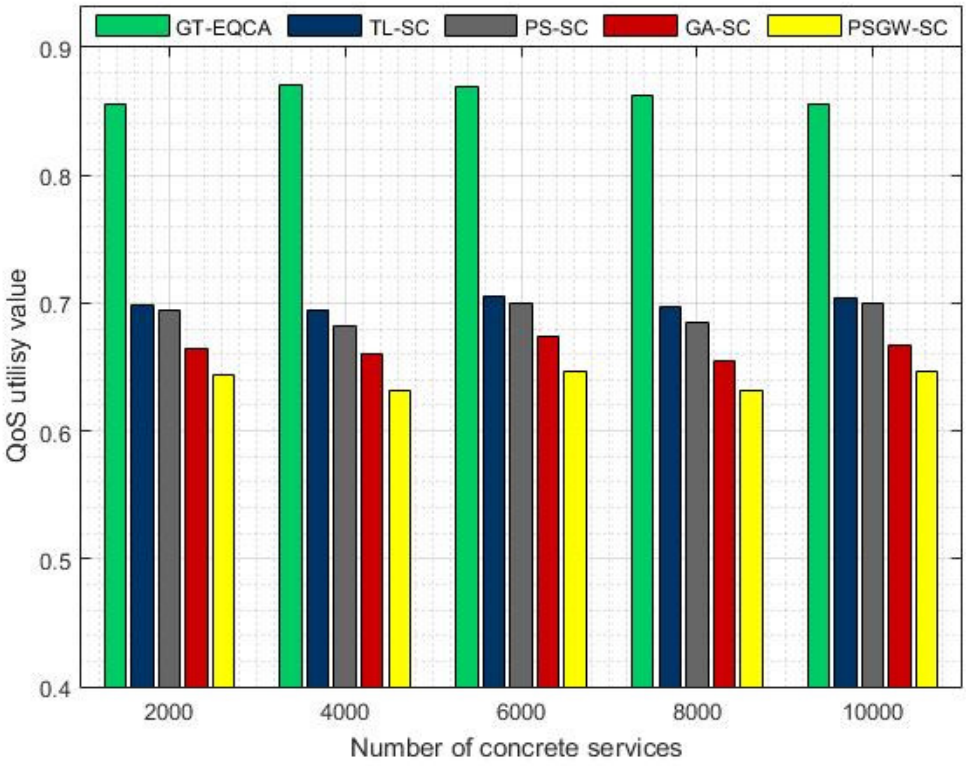


Figure III.14: Impact of the concrete services' number on the QoS utility of the composition.

ing phase that filters out concrete services that do not meet predefined QoS constraints, ensuring that only the relevant services are selected for the next phase of composition. Subsequently, the algorithm exploits the concept of relative Pareto dominance to select the most suitable concrete services based on both QoS and energy efficiency. Finally, sub-optimal composite service in terms of QoS and energy is obtained by applying a group teaching optimization method. The simulation scenarios show that the GT-EQCA algorithm significantly outperforms four baseline approaches in terms of composition time, while achieving sub-optimal composition in terms of QoS. Furthermore, the proposed approach proves to be highly efficient in terms of energy conservation, thereby increasing service availability. The following chapter presents the second contribution of this thesis, which addresses the service composition problem in dynamic environments. This contribution overcomes the limitations of existing approaches, including the one proposed in this chapter, by integrating mobility into the composition process to better manage its impact on QoS attributes and energy efficiency.

Chapter IV

Mobility and energy efficient service composition algorithm with QoS guarantee for large scale Cyber–Physical–Social Systems

1 Introduction

In the context of large-scale CPSS, service composition faces several challenges due to the dynamism of environments, the mobility of users and services, along with the resource-constrained devices that host them. During the past two decades, several approaches have been proposed to address the service composition problem by considering, separately or jointly, QoS, energy, or mobility issues. QoS-aware service composition approaches generally ensure high-quality compositions but often neglect energy efficiency, leading to premature service unavailability. Energy-aware approaches aim to extend the service lifetime by reducing energy consumption but do not meet users' QoS requirements. Some recent work combine QoS and energy constraints but ignore mobility, which is a decision factor in dynamic and large-scale CPSS environments. The few studies that consider mobility often rely on random movement models that fail to capture realistic mobility patterns, reducing the reliability of composition in real world scenarios. The first contribution of this thesis partially addressed these limitations by integrating QoS and energy-awareness into the service selection process using a group teaching optimization method. However, the GT-EQCA algorithm did not consider user mobility or realistic mobility patterns, which remain essential challenges for reliable service composition in dynamic CPPS environments

To address this limitation, this chapter introduces the learning-based swarm optimization-aware service composition algorithm (LS-SCA) that simultaneously accounts for the user's mobility, energy constraints, and QoS requirements during the service composition process. The subsequent parts of this chapter are organized as follows. Section 2 introduces the mobility and energy models, along with the problem of mobility-aware

service composition considering energy and QoS issues. Section 3 provides a motivational example to illustrate how the user's mobility impacts the energy consumption and QoS of services within the composition process. Section 4 describes the two-phase learning-based swarm optimization method exploited in this work. Section 5 details the proposed learning-based swarm optimization QoS-aware service composition (LS-SCA) approach along with an application scenario. Section 6 evaluates the performance of the proposed approach with respect to six recent baselines, whereas Section 7 summarizes the finding of this chapter.

2 Models and problem description

This section introduces the fundamental concepts related to the service composition issue, along with the mobility and energy consumption models used in this chapter. Furthermore, a new utility function is defined to evaluate services considering QoS, energy, and mobility aspects.

2.1 CPSS device

A CPSS device D refers to a fundamental unit characterized by the ability to integrate digital, physical, and social components. This device is defined as a 3-tuple (Id_D, S_D, E_{Res}) where:

- Id_D denotes the unique identifier of the CPSS device.
- S_D represents the set of CPSS services provided by the device. These services encompass a variety of functionalities (data sensing, processing, communication, actuation, user interaction, etc.) and capabilities that contribute to the device's role within a CPSS framework.
- E_{Res} refers to the residual energy level of the CPSS device, representing the amount of energy available to perform tasks in a CPSS environment. This parameter is essential to evaluate whether the device can continue to operate efficiently or if its energy may limit its ability to participate in the service composition process.

2.2 CPSS service

A CPSS service s_j is expressed as a 5-tuple $(I, O, D, QoS, EC_{Unit})$ where each component plays a significant role in defining the proprieties and performance of the service within the CPSS.

- I represents the size of the input parameters required by the CPSS service s_j. This size determines the amount of data processed by the service, which can affect the computation time.

- O denotes the size of the output parameters generated by the service s_j. The output size determines the bandwidth required for data transmission and the storage capacity needed to retain the service's results.
- D is the device that hosts the CPSS service s_j, which can be a physical sensor, a computer, or a virtual machine. The association between a service and its hosting device determines the service's execution environment that directly impacts its performance based on the device's processing power, network connectivity, and battery energy level.
- QoS is the quality of the CPSS service s_j represented by a vector of k criteria $QoS(s_j) = (q_{1,j} \cdots q_{q,j} \cdots q_{k,j})$, where $q_{q,j}$ refers to the value taken by the q^{th} QoS criterion that can be positive Q^+ (e.g., availability) or negative Q^- (e.g., cost).
- EC_{Unit} denotes the energy consumption of the CPSS service s_j during its execution. This parameter is crucial to assess the energy efficiency of the service.

2.3 Composite service

A composite service C within a CPSS is defined as a 4-tuple (SC, ST, QoS, EC) where :

- SC represents the set of CPSS concrete services that constitute the composite service C. These services are individual executable components that together form the functionality of the composite service.
- $ST = \{R(cs_i, cs_j)\}$ is the structure of the composition plan, where R refers to the composition pattern that links two services cs_i and cs_j, including loop $\odot$, switch $\oplus$, sequence $\rightarrow$, and parallel $\otimes$.
- QoS refers to the quality vector of the composite service C, such as $QoS(C) = (Q_1 \cdots Q_q \cdots Q_k)$ where Q_q is the value taken by the q^{th} criterion. This value is computed by aggregating the q^{th} values of the CPSS concrete services that belong to the composite service C. According to the type of QoS criterion and the structure of the composition plan, there are four aggregation functions: summation, product, minimum, and maximum [Khanouche et al., 2019b, Alrifai et al., 2012].
- EC represents the energy consumption of the composite service C which is calculated as the summation of the energy consumed by CPSS concrete service belonging to the composition.

2.4 Mobility model

Managing the user's mobility is crucial during the composition process, since the service-related data may change according to the user's location. The Small World in Motion (SWIM) mobility model [Mei and Stefa, 2009] is employed in this chapter to generate

realistic user's mobility traces. This model captures social behavior and interactions between users, enabling a more accurate simulation of movement patterns. Additionally, the SWIM model has been widely used in previous studies and has demonstrated reliable performance compared to other mobility models [Leguay et al., 2006]. The SWIM mobility model characterizes human movement based on two principals : (*i*) people usually visit popular locations near their homes more frequently and rarely go to less popular places far away from their homes; (*ii*) people spend most of their time at these popular locations, such as workplace or frequently visited public area. In this model, each user U is identified by its home location, which serves as the central reference point for generating mobility patterns. The environment is divided into several destination points. Each destination C is assigned a weight $w(C)$ that reflects its popularity and is inversely proportional to its distance between the destination from the user's home location. More specifically, let U be the user, h_u the user's home location, and C be a potential destination cell. Since h_u is a point and C represents a cell, the distance is measured from h_u to the center of the cell C. The weight $w(C)$ that the user U assigns to the destination cell C is calculated as follows:

$$w(C) = \alpha \cdot distance(h_u, C) + (1 - \alpha) \cdot seen(C) \tag{IV.1}$$

where $distance(h_u, C)$ is the distance between the user's home location h_u and the cell C, while $seen(C)$ measures the popularity of the cell C that represents the number of users observed by the user U during previous visits to this cell. This number is initially set to 0 and is updated each time the user U reaches a destination within the cell C. The parameter α takes a value within the range $[0, 1]$ and is used to determine whether users prefer visiting popular locations or those nearby. Once a destination is chosen, the user moves along a straight path toward this destination at a constant speed proportional to the travel distance.

2.5 Mobility-aware service composition

The mobility-aware service composition in CPSS aims to select the most appropriate concrete service that accounts for the user's mobility traces to form a composite service. The latter must not only satisfy the user's functional requirements but also ensure high QoS and minimal energy consumption, while taking into account the user's mobility. Dealing with the mobility issue in the composition process includes two key steps: (*i*) calculating a mobility-related metric, called hereafter aptitude M, for all possible connections between the mobile user and the devices hosting services within the user's coverage area; (*ii*) selecting services that have the highest aptitude values with respect to the mobile user's device.

The aptitude value $M_i(s_j)$ is defined as the expected time during which a user U_i who invoked the service s_j stays within the coverage area of this service. The value of $M_i(s_j)$ can be calculated as follows:

$$M_i(s_j) = \frac{D_{i,j}}{v_i} \tag{IV.2}$$

where v_i is the velocity of the user U_i and $D_{i,j}$ refers to the expected distance that the user U_i will travel within the signal range of the device offering the service s_j (see Figure IV.1). The value of $D_{i,j}$ is calculated as follows:

$$D_{i,j} = \sqrt{R_j^2 - (d_{i,j})^2 + 2R_j \cdot d_{i,j} \cdot cos\theta} \tag{IV.3}$$

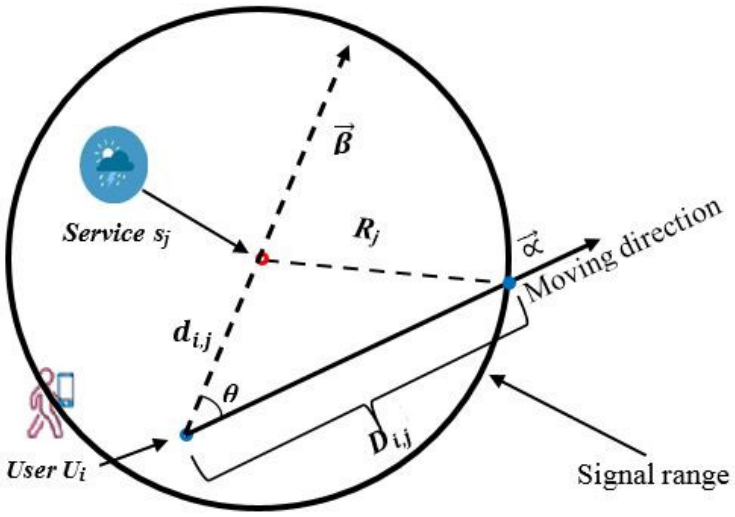


Figure IV.1: The aptitude value M used in the LS-SCA approach.

where R_j is the distance between the service s_j and the user destination, $d_{i,j}$ represents the distance between the user U_i and the service s_j, whereas θ is the angle between the direction vectors $\overrightarrow{\alpha}$ and $\overrightarrow{\beta}$. The vector $\overrightarrow{\alpha}$ indicates the direction of the user's movement as determined by the SWIM mobility model, and $\overrightarrow{\beta}$ points out the direction from the user U_i to the device providing the service s_j. Let (x, y) and (a, b) be the positions of a user U_i and a service s_j, respectively. The vector $\overrightarrow{\beta} = (a - x, b - y)$ and the angle θ can therefore be calculated as follows:

$$\theta = \arccos \frac{\overrightarrow{\alpha} . \overrightarrow{\beta}}{|\overrightarrow{\alpha}| . |\overrightarrow{\beta}|} \tag{IV.4}$$

2.6 Energy consumption model

As users move from one location to another in a CPSS environment, their devices (e.g., smartphones, tablets) often need to switch networks or access points. This transition can lead to a loss of connectivity or network handovers that can potentially impact the energy consumption of devices. Furthermore, the services chosen for the composition process are usually provided by devices with low-energy batteries. These services may become unavailable due to the battery overuse, leading to composition failures. The

service composition process should thus take into account the energy issue to ensure the continuity of the composition.

The energy consumed during the invocation of the composite service C can be calculated as follows:

$$E(C) = \sum_{s_j \in C} E_{global}(s_j) \tag{IV.5}$$

where $E_{global}(s_j)$ is the total energy consumed during the invocation of the candidate service s_j belonging to the composition C. This energy consumption is formally calculated as follows:

$$E_{global}(s_j) = E_{run}(s_j) + E_{inv}(s_j) \tag{IV.6}$$

where $E_{run}(s_j)$ and $E_{inv}(s_j)$ refer, respectively, to the energy consumed during the running and the invocation of the concrete service s_j.

2.6.1 Energy consumption of a service running

The energy consumed during the running of the concrete service s_i is calculated as:

$$E_{run}(s_j) = EC_{unit}(s_j) \cdot ExecTime(s_j) \tag{IV.7}$$

where $EC_{unit}(s_j)$ and $ExecTime(s_j)$ represent, respectively, the power consumption rate during service execution and the execution duration of the service s_j.

2.6.2 Energy consumption of a service invocation

The energy consumed by a mobile device during the invocation of the service s_j includes the energy consumption of data transmission and the energy consumption during the standby state. Formally:

$$E_{inv}(s_j) = EC_{up} + EC_{idle} + EC_{down} \tag{IV.8}$$

- EC_{up} represents the energy consumed to send the input parameters of the invocation request, calculated as follows :

  $$EC_{up} = p_u \cdot t_u \tag{IV.9}$$

  with p_u refers to the power spent by the mobile device to upload the service request and t_u is the upload duration computed as follows:

  $$t_u = \frac{D_u}{V_u} \tag{IV.10}$$

  such as D_u is the data size of the service request and V_u is the transmission speed between the user and the device that hosts the service.

- EC_{idle} refers to the energy consumed by the mobile device in the standby state while waiting for the execution of the service. Formally:

$$EC_{idle} = p_s \cdot t_s \tag{IV.11}$$

  where p_s is the power of the mobile device in standby state during processing of the service request and t_s represents the response time of the service.
- EC_{down} is the energy consumption of the mobile device when receiving the execution result of the service invocation, which is computed as follows:

$$EC_{down} = p_d \cdot t_d \tag{IV.12}$$

  such as p_d is the power of the mobile device when the user downloads the execution result from the device hosting the service, and t_d is the download duration, calculated as follows:

$$t_d = \frac{D_d}{V_d} \tag{IV.13}$$

  where D_d represents the size of the data returned from the execution of the service and V_d is the transmission speed between the mobile user and the device hosting the service.

2.7 Utility function

A composite service C is evaluated using three utility values depending on QoS, energy, and mobility.

2.7.1 QoS utility value

To evaluate a composite service C considering the QoS criteria, a utility value $U_{QoS}(C)$ is computed as follows:

$$U_{QoS}(C) = \sum_{q=1}^{k} \omega_q \cdot {Q_q}'(C) \tag{IV.14}$$

where ω_q refers to the user's preference assigned to the q^{th} QoS criterion in a specific application domain and ${Q_q}'(C)$ represents the normalized value of the q^{th} QoS criterion. This value is computed using Equation (III.5) presented in Subsection 2.4 of Chapter 3.

2.7.2 Energy utility value

To ensure service availability, the service composition process should select the most energy-efficient services by considering the residual energy of the devices and the energy cost of the services. The energy utility value of a composite service C is calculated as the ratio between the energy consumption of the service invocation and the energy level of the devices that host the services that belong to the composition. Formally:

$$U_{energy}(C) = \frac{E(C)}{E_{Res}(C)} \tag{IV.15}$$

where $E(C)$ represents the energy consumption of the composite service invocation and $E_{Res}(C)$ denotes the residual energy of the devices hosting the services belonging to the composition C, calculated as follows:

$$E_{Res}(C) = \sum_{s_j \in C} E_{Res}(s_j) \tag{IV.16}$$

where $E_{Res}(s_j)$ is the residual battery energy of the device that provides the concrete service s_j. Note that a lower $U_{energy}(C)$ means a better energy efficiency of the composite service.

2.7.3 Mobility utility value

To evaluate a composite service C considering mobility criteria, a mobility utility value $U_{mobility}(C)$ is calculated as follows:

$$U_{mobility}(C) = min_{s_j \in C}(M_i(s_j)) \tag{IV.17}$$

where $M_i(s_j)$ represents the aptitude value calculated using formula IV.2.

2.8 Global utility value

The *Tchebycheff norm* is a mathematical measure that aggregates multiple criteria into a single value [Steuer and Choo, 1983, Wierzbicki, 1986]. This norm is used in this chapter to assess composite services considering three criteria: QoS, energy, and mobility. This assessment is based on the concept of *ideal service* that refers to the vector of optimal utility values for each criterion. The *Tchebycheff norm* calculates the maximum distance between a given composite service and the ideal composition considering these three criteria, enabling comparison and ranking of composite services based on their QoS, energy, and mobility.

Definition 1. *Let S be the set of composite services, and let $c_1 \cdots c_n$ be the utility values associated with the n criteria. The ideal composite service $C^* = (\ c_1^* \cdots c_n^*)$ is a vector such as:*

$$c_i^* = max_{C \in S}(c_i(C)), \forall i \in \{1 \cdots n\} \tag{IV.18}$$

where $c_i(C)$ is the utility value of the criterion i associated with the composition C.

For a given criterion, the difference between the ideal composite service C^* and the value of a composition C refers to the *regret* of choosing this composite service as a near-to-optimal composition.

Definition 2. *Let $C^* = (c_1^* \cdots c_n^*)$ be the ideal composite service considering n criteria in the context of service composition and let $(c_1 \cdots c_n)$ be the utility values associated with these criteria. Let $C \in S$ be a solution for the service composition problem. The regret $R(C, i)$ of a composite service C with respect to the i^{th} criterion is calculated as follows:*

$$R_i(C) = c_i^* - c_i(C) \tag{IV.19}$$

where i represents one of the criteria considered in this chapter (QoS, energy, and mobility), c_i^* is the ideal utility value associated with the i^{th} criteria, and $c_i(C)$ is the utility value of the i^{th} criteria associated with the composite service C.

The *min-max regret* method can be used to identify the composition that minimizes the maximum regret with respect to the considered criteria.

Definition 3. *Let e and e' be two solutions, and $R_1 \cdots R_n$ the functions measuring regret according to the n criteria. The solution e is preferred over the solution e' considering the* ***min-max regret*** *method if and only if:*

$$max(\{R_i(e) \mid i \in \{1 \cdots n\}\}) < max(\{R_i(e') \mid i \in \{1 \cdots n\}\}) \tag{IV.20}$$

Let C and C' be two feasible compositions and $R_1, \cdots, R_n$ the utility values that represent the regret on energy, QoS, and mobility. In this chapter, the *Tchebycheff norm* of composition C, also called the max-regret, represents the global utility value of composition C that is computed as follows:

$$U(C) = max(\{R_i(C) \mid i \in \{1 \cdots n\}\}) \tag{IV.21}$$

The composition C is preferred over the composition C' when considering the Tchebycheff method if and only if:

$$U(C) < U(C') \tag{IV.22}$$

Consider the example given in Table IV.1, where three compositions C_1, C_2 and C_3 are characterized by their utility values in terms of QoS, energy, and mobility, respectively, referred to as U_{QoS}, U_{Energy}, $U_{Mobility}$. For instance, the composition C_1 has high utility values considering QoS and energy, but it has the lowest mobility utility, whereas the composition C_3 is the most balanced composition since it has no high utility value for any of the three criteria or a poor value for any of these criteria.

The values R_{QoS}, R_{Energy}, and $R_{Mobility}$ for a composition C_i represent the regret of each criterion that is equal to the distance between the utility vector of the ideal composition $C^* = (0.85, 0.90, 0.60)$ and the utility vector of the composition C_i. The global utility value of a composition C_i is computed by aggregating its individual criteria using the maximum regret value among them. The most balanced composition is one with the minimum max-regret value. In this example, the composition C_3 is the most balanced as it achieves the minimum max-regret which is equal to 0.15, making this composition the preferred choice in all three criteria.

3 Motivation scenario

In this section, a motivation example is provided to illustrate how the user's mobility can impact the energy consumption and the QoS of services involved in a composition

Table IV.1: Example of Tchebycheff method considering three feasible compositions.

Compositions	C_1	C_2	C_3
QoS	0.4	0.85	0.7
$Energy$	0.9	0.30	0.75
$Mobility$	0.6	0.35	0.5
R_{QoS}	0.45	0	0.15
R_{Energy}	0	0.6	0.15
$R_{Mobility}$	0	0.25	0.1
$\max(R_{QoS}, R_{Energy}, R_{Mobility})$	0.45	0.6	0.15

process. Let Simon be a professor of medicine who must perform a surgical operation at a hospital center. To provide a detailed overview of the patient's critical situation before the beginning of the intervention, Simon invokes a teleconference service along his way using his smartphone, allowing him to communicate in real time with the assistant surgeons already in the operating room.

Simon's path goes from the university located in zone A and passes through zone B before arriving at the hospital in zone C (see Figure IV.2). Along this path, three mobile network operators ($OP1$, $OP2$, and $OP3$) provide teleconference services, each in a specific zone : $OP1$ in zone A, $OP2$ in zone B, and $OP3$ in zone C. In each zone, the operator offers three different teleconference services (cs_1, cs_2, cs_3), which together constitute an abstract service for that zone. Each teleconference service is defined by two QoS criteria: response time RT and cost C (see Table IV.2).

Table IV.2: QoS criteria of teleconference services in the motivation example.

Services	Zone A		Zone B		Zone C	
	RT	C	RT	C	RT	C
cs_1	20	5	21	2	24	3
cs_2	25	1	17	4	20	1
cs_3	13	2	15	7	18	1

In conventional service composition approaches that do not account for mobility, the

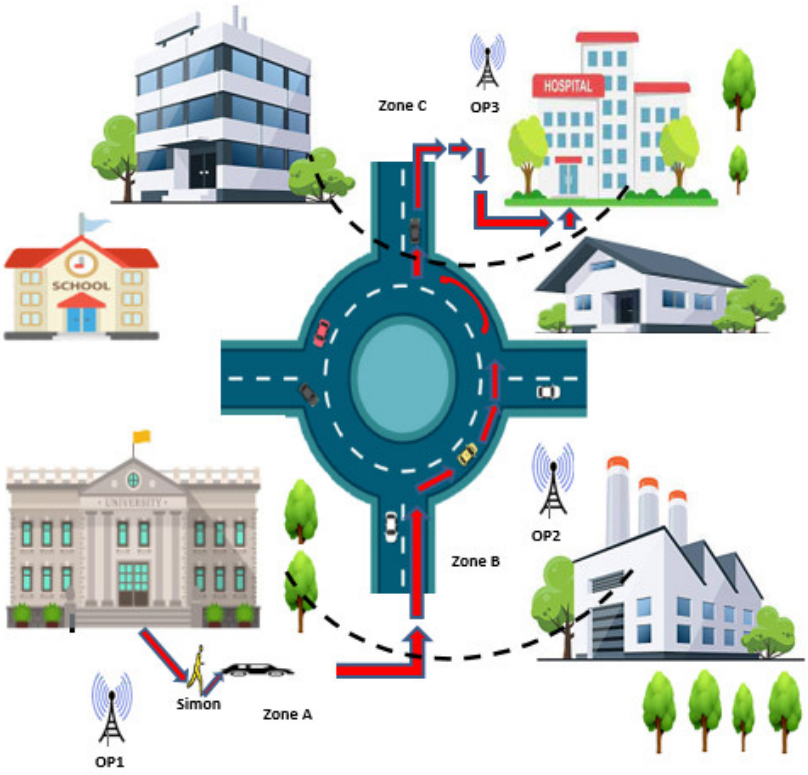


Figure IV.2: Motivation scenario of the LS-SCA approach.

service selection is only based on QoS criteria. In this case, the selected composite service is $C = < cs_3, cs_1, cs_3 >$ provided, respectively, in zones A, B and C. This combination has the highest QoS utility value, F(C) = 0.8295, calculated using Equation (III.4) presented in Section 2.4 of Chapter III, where the weights assigned to QoS attributes are $\omega_{RT} = 0.5$ and $\omega_C = 0.5$.

In the case where mobility is considered during the composition process, the service selection must be dynamic to ensure optimal performance within each zone along the user's trajectory. Due to the variation in network coverage among the three mobile operators, the signal strength fluctuates from a zone to another, resulting in services with different QoS values and energy consumption. Accordingly, it is crucial to select the operator that provides the most reliable and stable coverage throughout all areas of Simon's route to avoid dropped calls or communication interruptions. Additionally, mobility-aware selection improves energy efficiency, as the Simons's device avoids frequent signal searches, thus extending the lifetime of the device battery. Table IV.3 shows the energy consumption of services provided by operators in zones A, B and C.

To determine the best composite service, the energy and QoS are considered to calculate the overall utility as follows:

$$U(C) = U_{energy}(C) \cdot U_{QoS}(C) \tag{IV.23}$$

where $U_{energy}(C)$ and $U_{QoS}(C)$ represent, respectively, the energy utility and the QoS

Table IV.3: Energy consumption of teleconference services in the motivation example.

Services	Zone A	Zone B	Zone C
cs_1	1	3	5
cs_2	3	1	3
cs_3	5	3	1

utility of the composite service C.

As explained above, when mobility is not considered during the composition process, the composite service $C = < cs_3, cs_1, cs_3 >$ achieves a utility value $U(C) = 0.3318$ using Equation (IV.23). However, when a mobility-aware service composition is applied, the selected composite service becomes $C = < cs_1, cs_2, cs_3 >$, achieving a higher utility value of 0.5398, which represents an improvement of approximately 62%.

4 Two-phase learning-based swarm optimizer for large-scale optimization

The two-phase learning-based swarm optimizer (TPLSO) is a novel reinforced competition-based learning strategy for particle swarm optimization inspired by nature and cooperative learning behavior in human society [Lan et al., 2020]. This algorithm evolves in several iterations, each carried out through two phases: (*i*) the mass learning phase and (*ii*) the elite learning phase.

4.1 Mass learning

A swarm $P_1(t)$ in iteration t contains NP particles, where NP is the size of the swarm. These NP particles are divided into $\frac{NP}{K}$ study groups. The particles in each group may have different exploration and exploitation abilities to traverse the objective space. Consequently, particles within a group should be treated differently during the evolution process. In this phase, three particles are randomly selected from the swarm $P_1(t)$ for competitions. The particle with the best fitness is called *winner* W, whereas the remaining two particles designed by $L1$ and $L2$ are called *losers*. The winner is directly passed to the elite learning phase. The loser $L1$ updates its position by learning from the winner W according to the formula (IV.25), whereas the loser $L2$ updates its position by learning from the winner W and the first loser $L1$ according to the formula (IV.24). Formally:

$$V_{L_2} = R_1\ V_{L_2}\ + R_2(X_W\ -\ X_{L_2})\ +\ \phi\ R_3\ (X_{L_1}\ -\ X_{L_2}) \tag{IV.24}$$

$$V_{L_1} = R_1 \; V_{L_1} \; + R_2(X_W \; - \; X_{L_1}) \; + \phi \; R_3 \; (\bar{X} \; - \; X_{L_1}) \tag{IV.25}$$

$$X_{L_i} = X_{L_i} \; + V_{L_i}; i = 1, 2 \tag{IV.26}$$

where X_W represents the position of the winner W, whereas X_{L_i} and V_{L_i} refer to the position and the velocity of the loser L_i. $R1$, $R2$, and $R3$ are three random numbers within the range $[0, 1]$. The parameter ϕ is within the interval $[0, 1]$ that controls the influence of X_{L_1} or $\bar{X}$ where $\bar{X}$ is the mean position of the group to which the particle belongs or the mean position of the entire swarm.

4.2 Elite learning

In this phase, the particles in $P_1(t)$ are sorted in ascending order with respect to their fitness values. The top N particles are selected to form a new swarm P_h, whereas the remaining particles are passed directly to the next iteration without position update. Note that P_h consists of particles that have better fitness values than the remaining particles in the swarm $P_1(t)$. These particles are usually more informative and play a critical role in the learning process of other particles and are likely to be closer to the optimal solution. Each particle j in P_h updates its position X_j and velocity V_j by learning from two randomly selected particles Xr_1 and Xr_2 that have a better fitness value. Formally:

$$V_j = R_1 \; V_j \; + R_2(X_{r_1} \; - \; X_j) \; + \phi \; R_3 \; (X_{r_2} \; - \; X_j) \tag{IV.27}$$

$$X_j = X_j \; + V_j \tag{IV.28}$$

where j, r_1 and r_2 represent three particle indexes. Note that $r_1 < r_2 < j$ means that particle j is lower than particle r_2 in terms of fitness value, which in turn is lower than particle r_1. Thus, X_j is worse than X_{r_2}, which is worse than X_{r_1}.

5 The proposed LS-SCA approach

Several existing service composition approaches do not simultaneously address the challenges posed by user mobility, energy limitations, and QoS requirements in large-scale CPSS. These challenges become more critical when services are hosted on mobile devices with limited resources and dynamic availability. To overcome these limitations, this thesis proposes a learning-based swarm optimization QoS-aware service composition algorithm (LS-SCA) [Hameche et al., 2024b]. The proposed approach begins with the use of the SWIM mobility model to generate realistic user's mobility traces. An energy consumption model is then introduced to improve energy efficiency by reducing excessive battery usage, which could decrease service availability and lead to composition failures. Finally, the TPLSO method is applied to find a sub-optimal service composition that satisfies global QoS constraints while maximizing overall utility in terms of mobility, energy efficiency, and QoS.

To address the service composition problem in large-scale CPSS, each composition is modeled as a particle within a swarm, and the TPLSO algorithm is used to guide the search process. In this context, the particle $X_p = \{x^1_{j,p}, \cdots, x^i_{j,p}, \cdots, x^D_{j,p}\}$ represents the p^{th} composite service $C_p = \{x^1_{j,p}, \cdots, x^i_{j,p}, \cdots, x^m_{j,p}\}$, where $x^i_{j,p}$ is the index of the j^{th} candidate service in the i^{th} abstract service of the p^{th} composition. The mapping between the TPLSO algorithm and the QoS-aware service composition problem considering mobility is shown in Table IV.4.

Table IV.4: TPLSO algorithm versus mobility and QoS-aware service composition problem.

TPLSO algorithm	**Mobility and QoS-aware service composition**
Swarm	Population of compositions
Particle	Composite service
Particle with the best position	Best composition in terms of QoS utility
Position vector values	Indexes of candidate services
Mass learning	Improvement of compositions
Elite learning	Learning from the b composite services
Fitness function	Utility value in terms of QoS, energy, and mobility

5.1 LS-SCA algorithm phases

The Learning-based Swarm optimization QoS-aware Service Composition Algorithm (LS-SCA) begins by initializing several parameters : the population size NC, the sub-population size K, the maximum number of iterations that acts as the stopping condition TC, and a control parameter ϕ. To solve the service composition problem taking into account QoS, energy, and mobility, the swarm $P(t)$ contains NC composite services in iteration t that are initially chosen and then iteratively updated during the evolution process. The LS-SCA algorithm operates in two phases: (i) the *mass learning* phase where the compositions of the population are improved through a collaborative process and (ii) *the elite learning* phase where each composition is improved by learning from the N best compositions resulting from the previous phase (see Figure IV.3).

5.1.1 Mass learning phase

In iteration t, the population $P_1(t)$ is divided into $\frac{NC}{K}$ sub-populations, where each sub-population contains K composite services. The compositions within each sub-population

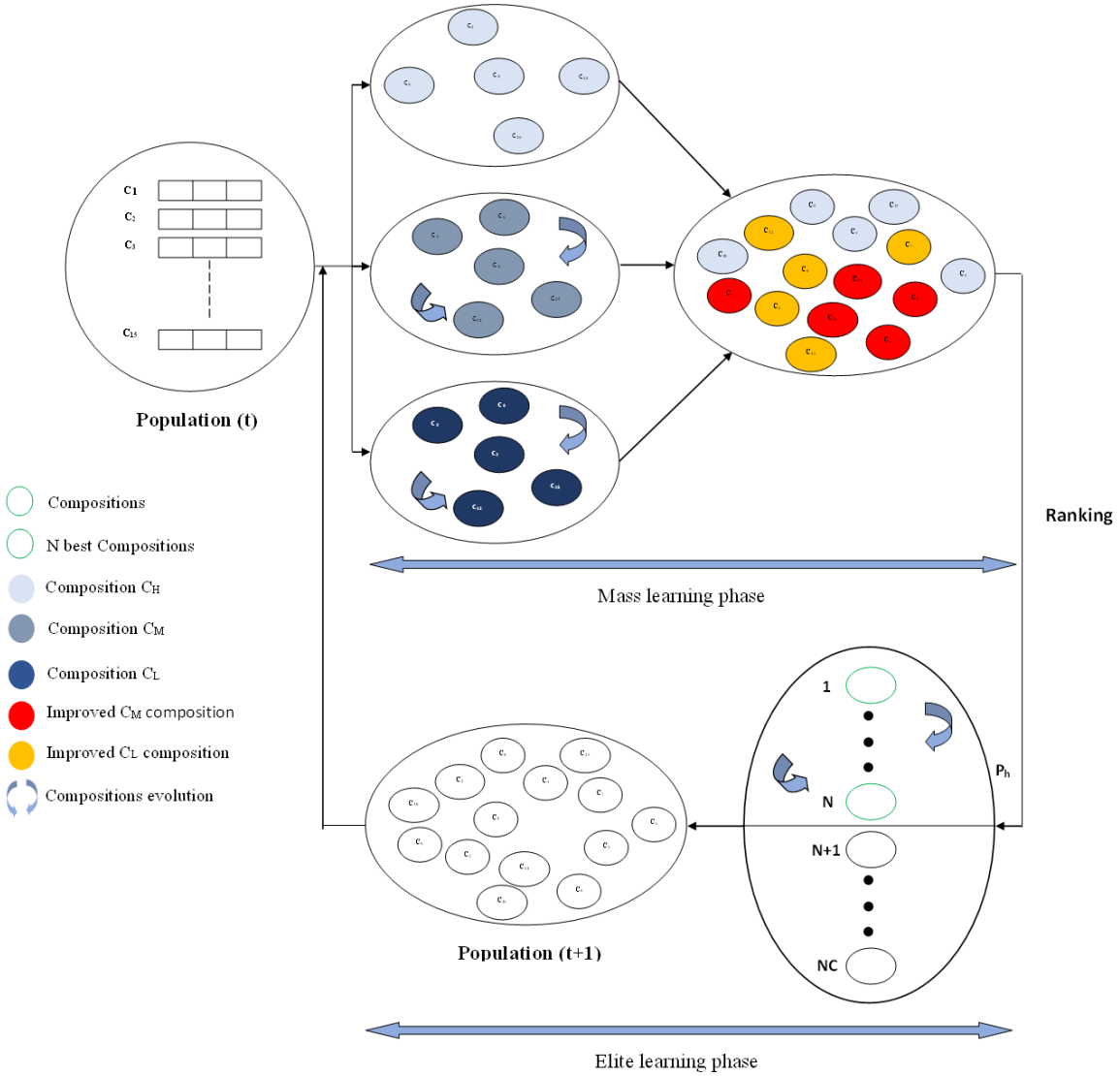


Figure IV.3: Phases of the LS-SCA algorithm.

are improved through collaborative and competitive strategies. More specifically, in each iteration, the composite services are ranked according to a global utility value that considers QoS, energy, and mobility criteria. Three composite services are randomly selected from $P_1(t)$ and compared considering their utility values. The composition C_H with the highest utility value is passed directly to the elite learning phase since it could be closest to the best composition of the population. The compositions C_L and C_M having the lowest and middle utility values, respectively, are updated using the following formulas:

$$V_L = sig(R_1\ V_L + R_2(C_H \ominus C_L) + \phi\ R_3\ (C_M \ominus C_L)) \tag{IV.29}$$

$$V_M = sig(R_1\ V_M + R_2(C_H \ominus C_M) + \phi\ R_3\ (\bar{C} \ominus C_M)) \tag{IV.30}$$

$$C_L = C_L \oplus V_L \tag{IV.31}$$

$$C_M = C_M \oplus V_M \tag{IV.32}$$

where $\bar{C}$ is the average utility value of the population, V_M and V_L are the velocities of the middle and lowest composite services, respectively. R_1, R_2, and R_3 are three random

values within the range $[0, 1]$. ϕ is a control parameter chosen from the interval $[0, 1]$ to balance the influence of C_M and C_L or $\bar{C}$. The operators $\oplus$, $\ominus$ and sig are used in the improvement strategy to guide search space exploration (see Subsection 5.2). After this update, both compositions C_M and C_L are added to the population for the next phase of the algorithm. The steps of the mass learning phase of the LS-SCA algorithm are summarized in Algorithm 4.

Algorithm 4 Mass learning phase of the LS-SCA algorithm.

Inputs: NC : The size of the population P.
K : The sub-population size.
TC : The terminal condition.
P_0 : The initial population.

Outputs: A new population of composite services.

while (TC is not satisfied) **do**
  $P_1 \leftarrow P_0$;
  Calculate the utility values of compositions in P_1;
  Divide the population into $\frac{NC}{K}$ sub-populations;
  for each sub-population **do**
    Find the compositions C_H, C_M and C_L;
    switch (Composition)
    case C_H**:**
      The composition C_H is passed to the next phase;
    case C_M**:**
      Update the velocity of C_M using formula (IV.30);
      Update the composition C_M using formula (IV.32);
      Add the composition C_M to the next phase;
    case C_L**:**
      Update the velocity of C_L using formula (IV.29);
      Update the composition C_L using formula (IV.31);
      Add the composition C_L to the next phase;
    end switch
  end for
end while

5.1.2 Elite learning phase

The composite services in $P_1(t)$ are sorted in ascending order of their utility values. The top N composite services are selected to form a new population P_h, while the remaining compositions are passed to the next iteration of the population $P(t+1)$. The P_h size is set to $\frac{NC}{2}$, where NC represents the original population size. In this phase, each composite service C_j in P_h is improved by learning from two composite services C_p and C_q that are

randomly selected among the three best compositions in P_h. Formally:

$$V_j = sig(R_1\ V_j + R_2(C_p \ominus C_j) + \phi\ R_3\ (C_q \ominus C_j)) \tag{IV.33}$$

$$C_j = C_j \oplus V_j \tag{IV.34}$$

where j, p, and q are, respectively, the composite services indexes of C_j, C_p and C_q. R_1, R_2, and R_3 are three random values within the range $[0, 1]$. ϕ is a parameter within the interval $[0, 1]$ that controls the influence of C_q. Note that $F(C_p) < F(C_q) < F(C_j)$ means that the composition C_j is worse than the composition C_q in terms of utility value, which in turn is worse than the composition C_p. The algorithm 5 outlines the steps of the elite learning phase of the LS-SCA algorithm.

Algorithm 5 Elite learning phase of the LS-SCA algorithm.

Inputs: NC : The size of the population P_1.
N : The size of the population P_h.
K : The sub-population size.
TC : The terminal condition.
P_1 : The population resulting from the mass learning phase.

Outputs: A new population of composite services.

```
while (TC is not satisfied) do
    P2 ← P1;
    Sort the compositions in ascending order of their utility values;
    Select the N best compositions from P2 to form the Ph population;
    for j = 3 to N do
        Select two compositions indexes p and q within [0, j − 1]
        if F(Cp) > F(Cq) then
            Swap (p, q);
        end if
        Update the velocity of Cj using formula (IV.33);
        Update the composition Cj using formula (IV.34);
    end for
end while
```

5.2 Operators definition in the LS-SCA algorithm

Several service composition approaches use arithmetic operators such as addition $(+)$, subtraction $(-)$, and multiplication $(*)$ in their improvement strategy (e.g., [Deng et al., 2016b], [Deng et al., 2016a]). However, the use of these operators could limit the exploration of the search space within the context of service composition. To overcome this limitation, logical operators $\oplus$ and $\ominus$ are introduced in this thesis to improve the exploration capabilities of the LS-SCA algorithm. A composite service is represented as an index vector of concrete services that are part of the composition. For example,

the composite service $C_i = (6, 3, 8, 1, 5)$ means that the concrete service with index 8 is selected from the third abstract service in the composition plan.

5.2.1 Subtraction operator $\ominus$

The result of the operation $C_i \ominus C_j$ is obtained as follows:

- In the case where the concrete services of the composition C_i are different from those of the composition C_j, the result will be equal to 1.
- If the concrete services are the same in the two compositions C_i and C_j, the result will be equal to 0.

5.2.2 Addition operator $\oplus$

The result of $C_i \oplus V_i$ is used to determine the location of the changed concrete services. The result of $C_i \oplus V_i$ is obtained as follows:

- If the search velocity V_i of the composition C_i is equal to 0, the result will be the same as C_i.
- In the case where the search velocity V_i of the composition C_i is equal to 1, the result is the index of a candidate service randomly selected from a service set having a utility value better than that of the composition C_i.

5.2.3 Sigmoid operator *sig*

The function $V = sig(X)$, where $V = (v_1 \cdots v_n)$, $X = (x_1 \cdots x_n)$, is defined as follows:

$$v_i = \begin{cases} 1 & \text{if } rand(0,1) < sigmoid(x_i) \\ 0 & \text{if } rand(0,1) \geq sigmoid(x_i) \end{cases} \tag{IV.35}$$

where the sigmoid function is defined as:

$$sigmoid(x_i) = \frac{1}{1 + \exp(-x_i)} \tag{IV.36}$$

The *sigmoid* function is applied to equations (IV.29), (IV.30), and (IV.33) to improve the accuracy of the velocities V_M, V_L and V_i.

5.3 An example of the LS-SCA application scenario

In the context of a smart city, Emma, a busy professional, takes advantage of an application within a CPSS environment. This application is designed to optimize her daily tasks that include: locating a cafeteria for an informal meeting booking, a co-working space for the afternoon planning, an itinerary for a training session, and ordering dinner from restaurants. This sequence of tasks can be seen as a service composition process, where

several functionalities from various providers are integrated to create a seamless and integrated user's experience. For the meeting organization, the application adjusts Emma's route considering time constraints to ensure her punctuality. Emma goes to a cafeteria adapted to her requirements according to the application's recommendation. After the meeting, Emma seeks for a co-working space. Taking into account Emma's mobility, the application recommends spaces close to her intended routes to minimize travel time and reduce the energy consumption of her device. For the training session, the application analyzes the sports infrastructure while considering Emma's mobility and suggests the best route to the gym. Finally, when ordering dinner, Emma would like to order a takeaway meal from a restaurant. The application selects restaurants that account for Emma's mobility, the restaurant service quality, and the energy efficiency to ensure stable connectivity with the delivery service. The application then proposes several restaurants that meet Emma's requirements. The aim is to provide personalized solutions to improve Emma' daily life.

To achieve this goal, four abstract services must be composed: a meeting cafeteria locator service, a co-working space booking service, a training session itinerary planner service, and a restaurant take-out meal ordering service (see Figure IV.4). Each abstract service can be carried out through multiple concrete services that have the same functionality but differ in the values of their QoS attributes, energy features, and mobility's aptitude. In this example, there are five concrete services for each abstract service that belongs to the composition (see Table IV.5), resulting in 5^4 possible combinations of concrete services to be evaluated during the composition. Each concrete service is characterized by three QoS attributes (response time: RT, availability: AV and cost: C), several energy-related features (Input data: D_u, Output data: D_d, Upload power: p_u, Download power: p_d, Standby power: p_s, Energy consumption per unit time: EC_{unit}, Upload transmission speed: V_u, Download transmission speed: V_d) and the mobility criterion (the aptitude value that represents the average time spent by a user within the coverage area of the service). The proposed LS-SCA approach aims to find the best combination of concrete services that meets the user's QoS requirements and minimizes energy consumption as much as possible, while taking into account the user's mobility.

Figure IV.5 shows the flowchart of an LS-SCA execution example. A population of six compositions is randomly generated $P1 = \{ C_1, C_2, C_3, C_4, C_5, C_6 \}$, where each composition C_i $(i = 1 \cdots 6)$ is evaluated using a global utility value $U(C_i)$ that represents the minimum max-regret value. Let $C_3 =< 4, 2, 3, 4 >$ be the composition with the best utility value, such as 4, 2, 3, and 4 refer to the indexes of concrete services that are selected from the abstract services AS_1, AS_2, AS_3 and AS_4 respectively. In the Mass learning phase, three compositions are randomly selected from the population and compared according to their utility values. In the considered example, the compositions C_3, C_4 and C_5 are first chosen with $U(C_3) < U(C_4) < U(C_5)$, followed by the compositions C_1, C_2 and C_6 with $U(C_1) < U(C_2) < U(C_6)$. The compositions C_3 and C_1 have the best utility value and form the first C_H sub-population. The compositions C_4 and C_2 have an

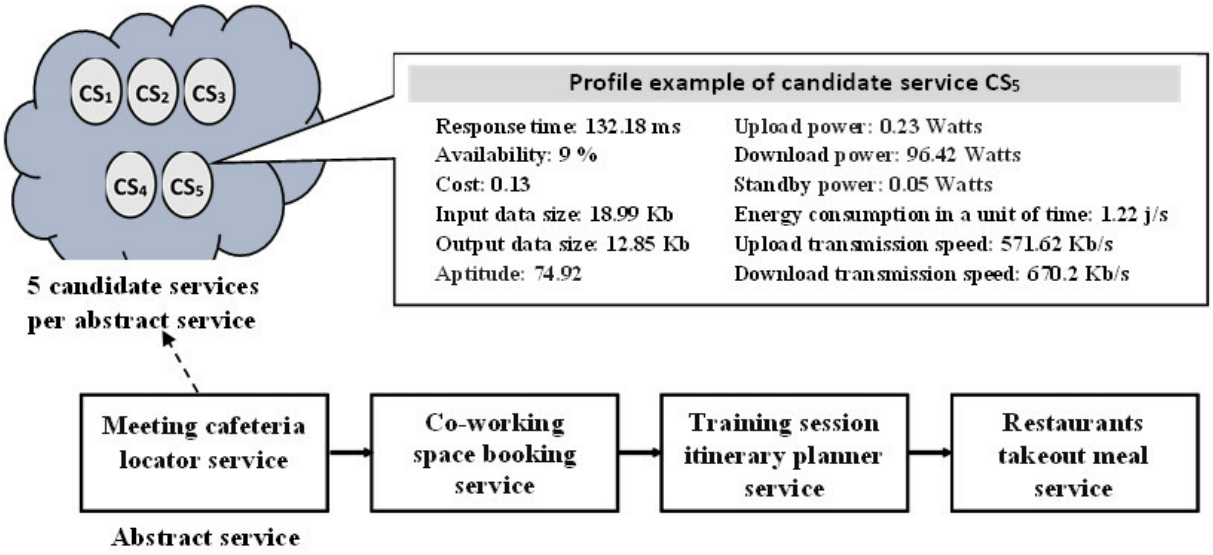


Figure IV.4: Service composition scenario of the LS-SCA approach.

average utility value and form the second C_M sub-population. The compositions C_5 and C_6, having the lowest utility value, constitute the third C_L sub-population. The compositions C_3 and C_1 of C_H sub-population are directly passed to the next phase, whereas the compositions belonging to the C_L and C_M sub-populations are updated according to formulas (IV.31) and (IV.32) respectively. The compositions C_4, C_2 and C_5 are retained as their new utility values (0.04, 0.21, and 0.01) are better than their initial ones. However, the composition C_6 keeps its initial value (0.41) as it is better than the new one (0.42). In the elite learning phase, the compositions are sorted in ascending order of their global utility values. The top $(NC/2)$ compositions are selected to form the new population P_h. The update process starts with the third composition of the P_h population by learning from two compositions randomly selected among those with better utility values. Thus, in the example, only C_3 is updated using formulas (IV.33) and (IV.34). This process is repeated until the stopping condition TC is satisfied.

6 Performance evaluation

This section evaluates and compares the performance of the LS-SCA approach with those of six among the most relevant and representative service composition approaches from the literature, according to different simulation scenarios.

6.1 Simulation parameters and dataset

We consider different large-scale service composition scenarios to evaluate the performance of the LS-SCA approach. These scenarios are carried out using MATLAB software on a 64-bit Windows OS running on an Intel Core (TM) i5-5005U PCU at a frequency of 2.20 GHz with 8 GB of RAM.

Table IV.5: Concrete services for each abstract service of the composition scenario.

AS	CS	Device/ provider	QoS attributes			Energy features					Mobility
			RT	AV	C	D_u	D_d	V_d	p_d	EC_{unit}	Aptitude
Cafeteria	cs_1^1	ConnectCup-Coffe	196	32,8	0,83	14,9	11,11	494,63	88,16	1,72	96,89
	cs_2^1	ChillChaiCorner	102,62	15,3	0,91	11,78	15,33	894,33	95,78	1,27	114,51
	cs_3^1	EasyMeetCoffee	154	14,4	0,71	19,22	18,11	991,57	96,78	0,48	119,27
	cs_4^1	CasualMeetCof-fee	227,8	14,6	1	12,67	10,39	627,43	88,43	0,83	71,26
	cs_5^1	RelaxedCorner	132,18	9	0,13	18,99	12,85	670,2	96,42	1,22	74,92
Co-Working	cs_1^2	ProSpaceWork	115	24,6	0,83	17,21	16,92	991,57	96,77	0,59	119,27
	cs_2^2	CoSpaceWork	160,64	7,9	0,59	17,31	15,82	470,12	91,96	0,81	119,5
	cs_3^2	FlexSpaceWork	176,48	2,4	0,97	14,57	16,8	29,25	93,92	1,23	130,6
	cs_4^2	CityHive	392,5	6,3	0,91	18,41	10,8	494,63	88,16	0,19	96,89
	cs_5^2	AgileOffice Hub	252,35	5,3	0,85	10,45	13,55	290,25	93,92	0,67	130,6
Training	cs_1^3	FitJourneyPro	269,83	4,5	0,85	14,28	17,7	627,43	88,43	1,1	71,26
	cs_2^3	GymFlow	167	24,6	0,85	19,23	17,26	714,48	91,09	0,68	99,87
	cs_3^3	FitRoutes	484,5	5,9	0,78	16,65	10,46	967,33	96,1	1,4	53,78
	cs_4^3	ActiveCommute	252	20,6	1	14,28	14,48	894,33	95,78	0,44	114,51
	cs_5^3	WelnessMapPro	213	17,1	0,96	12,12	18,94	906,64	95,4	0,47	85,5
Restaurants	cs_1^4	QuickEats	216,6	1,8	0,94	12,27	19,39	576,58	96,29	1,57	95,1
	cs_2^4	RapidGrains	370,5	10,9	0,86	17,77	14,22	576,58	96,29	1,86	95,10
	cs_3^4	SwiftSavorGrill	1074,3	3,1	0,14	16,54	14,61	490,61	99,24	1,09	123,32
	cs_4^4	QuickCrisp	329,8	6	0,51	13,04	13,03	576,58	96,29	0,16	95,1
	cs_5^4	PangeaPizza	386,44	6	0,27	10,05	16,75	576,58	96,29	0,71	95,1

The performance of the LSS-CA approach are assessed using two real-world datasets: the QWS dataset [Al-Masri and Mahmoud, 2008] and the WSC-2009 dataset [Kona et al., 2009]. The QWS dataset contains nine QoS parameters for 2507 services, whereas the WSC- 2009 dataset contains six QoS parameters for 36351 concrete services. The simulation scenarios are based on five QoS parameters: availability, reliability, success, throughput, and response time. For these scenarios, the number of abstract services

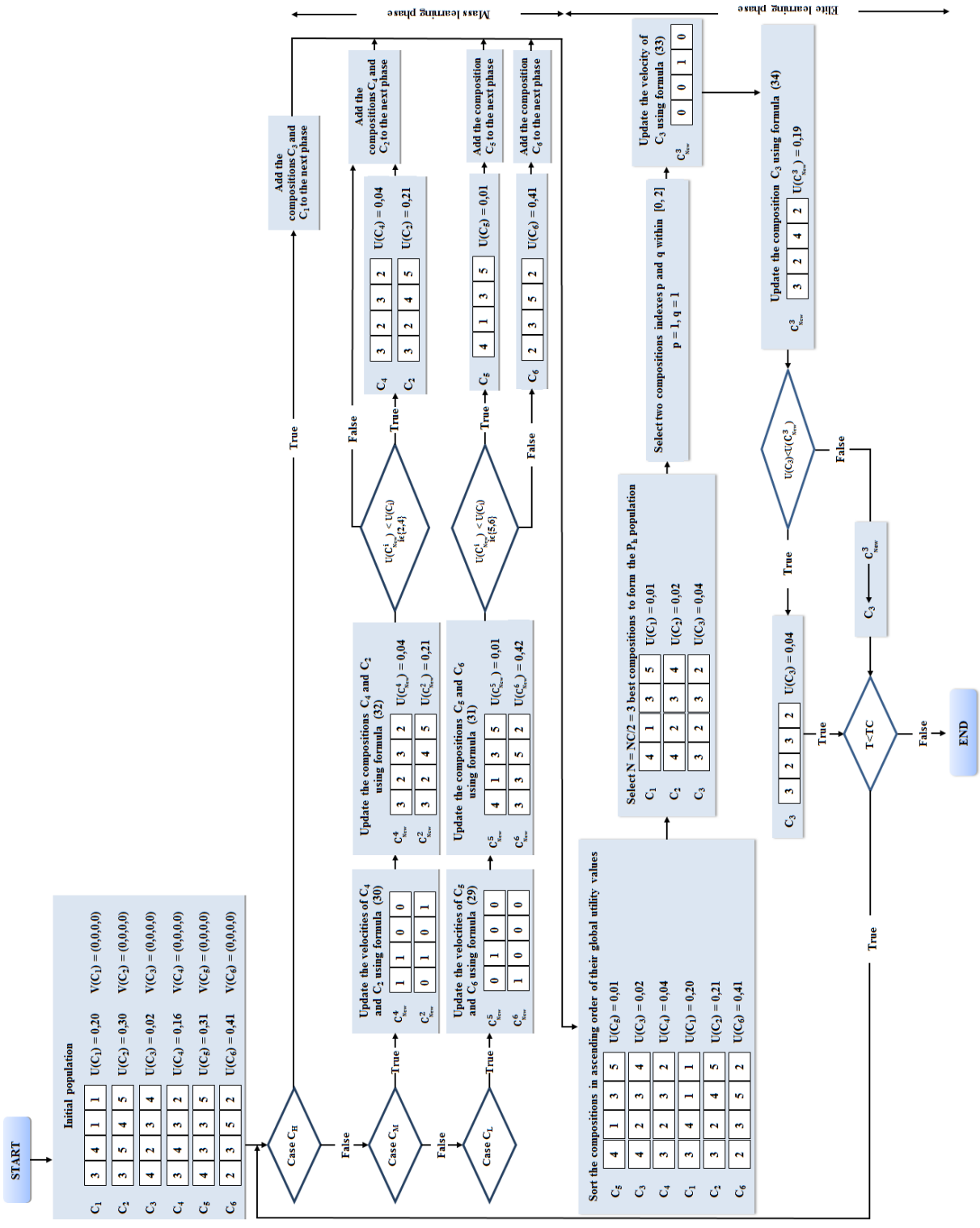


Figure IV.5: Flowchart of an LS-SCA execution example.

in each composition is set to 5, where each abstract service contains between 2000 and 10000 concrete services randomly chosen from the QWS and WSC-2009 datasets. As each device in a CPSS environment is equipped with a battery and can provide a set of services, the realistic energy model introduced in [Furthmüller and Waldhorst, 2012] is used to define the energy features of the services belonging to the QWS and WSC-2009 datasets. More precisely, a CPSS environment is simulated with 10 devices and

one mobile user's device. The battery of each device has an initial charge $C_{initial}$ chosen uniformly from the interval $[0.7C_{max}, C_{max}]$, where $C_{max} = 1500\ mA.h$ is the maximum battery charge. Candidate services are provided on different devices, while the input and output data sizes are randomly distributed between $10KB$ and $100KB$ for each service. Data transmission speeds between the mobile user and the devices hosting services are generated according to a uniform distribution within the interval $[10\ Kb/s, 1000\ Kb/s]$. The energy consumption of a service at each invocation is chosen uniformly in the interval $[10^2\ mA.s, 10^4\ mA.s]$ and its residual energy is updated after each invocation.

6.2 Baselines and performance metrics

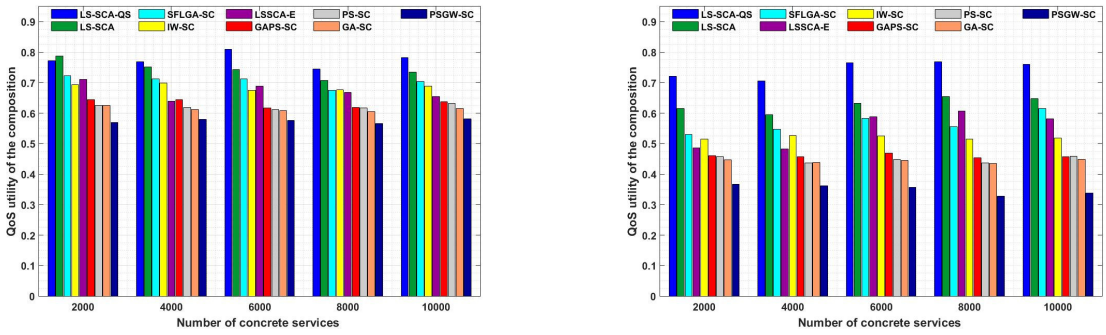
We compare the performance of the LS-SCA algorithm to those of six baselines approaches: the Particle Swarm optimisation and Gray Wolf optimizer-based Service Composition (PSGW-SC) approach [Sun et al., 2019b], the Genetic Algorithm-based Service Composition (GA-SC) approach [Deng et al., 2017b], the Particle Swarm optimization-based Service Composition (PS-SC) approach [Khanam et al., 2018], the Shuffled Frog Leaping and Genetic Algorithm-based Service Composition (SFLGA-SC) approach [Ibrahim et al., 2020], the Genetic algorithm and Particle Swarm optimization-based Service Composition (GAPS-SC) approach [Gao and Liu, 2020], and the modified Invasive Weed optimization-based Service Composition (IW-SC) approach [Jatoth et al., 2019]. The population size is set to 15 and the maximum number of iterations to 100 for all algorithms. The control parameter ϕ and the sub-population size k are set to 5 and 0.15, respectively, for the LS-SCA algorithm. The crossover and mutation probabilities are equal to 0.7 and 0.3, respectively, for the GA-SC and GAPS-SC algorithms. The inertia weight is equal to 0.5, whereas the values of C_1 and C_2 are equal to 2 for the PS-SC, PSGW-SC and GAPS-SC algorithms. The following performance metrics are used to evaluate the LS-SCA algorithm:

- *Composition time* that refers to the time required by the algorithms to find the sub-optimal composition.
- *Energy consumption of the composite service* that represents the energy consumption of the concrete services chosen during the composition process.
- *Utility value of the composite service* that measures the QoS utility value of the best composition obtained with the algorithms.
- *Availability of the composite service* that refers to the degree to which the composite service remains accessible and operational during a given period of time.
- *Expected duration of the composite service* that refers to the time during which a user remains within the coverage area of the services belonging to the composite service, reflecting the user's mobility.

6.3 Simulation results and comparison

6.3.1 Utility value of the composition

This simulation aims to compare the performance of the LS-SCA, SFLGA-SC, IW-SC, PS-SC, GA-SC, GAPS-SC, and PSGW-SC algorithms in terms of composition QoS utility with respect to the variation of the number of concrete services. As we can see in Figure IV.6(a), the utility value of the composition obtained in the case of the LS-SCA algorithm with QWS dataset varies between 0.73 and 0.79 when the number of concrete services varies from 2000 to 10000. This utility value is much better than those obtained with the SFLGA-SC, IW-SC, GAPS-SC, PS-SC, GA-SC, and PSGW-SC approaches, which vary from 0.70 to 0.73, 0.67 to 0.69, 0.61 to 0.64, 0.61 to 0.62, 0.60 to 0.62, and 0.56 to 0.58, respectively. Similarly, as shown in Figure IV.6(b), the utility value of the composition obtained with the WSC-2009 dataset in the case of the LS-SCA algorithm ranges from 0.60 to 0.66 when the number of concrete services increases from 2000 to 10000. This utility value is better than those achieved by the SFLGA-SC, IW-SC, GAPS-SC, PS-SC, GA-SC, and PSGW-SC approaches, which vary from 0.53 to 58, 0.51 to 0.53, 0.45 to 0.47, 0.43 to 0.46, 0.44 to 0.45, and 0.33 to 0.37, respectively. Note that the LS-SCA algorithm provides a composition utility very close to that obtained with the LS-SCA-QS variant where only the QoS is taken into account in the selection process. This outcome is due to the improved exploration and exploitation strategies used in the LS-SCA algorithm that allow selecting candidate services with high QoS to be part of the composition.



(a) QWS Dataset

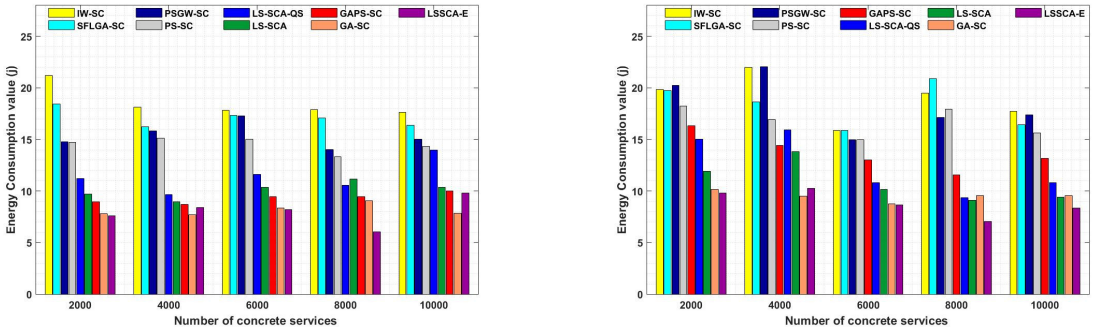
(b) WSC-2009 Dataset

Figure IV.6: Impact of the concrete services' number on the QoS utility of the composition.

6.3.2 Energy consumption of the composition

This simulation scenario measures and compares the energy consumption of the LS-SCA algorithm with those of the SFLGA-SC, IW-SC, PS-SC, GA-SC, GAPS-SC, and PSGW-SC algorithms by varying the number of concrete services. As illustrated in Figure IV.7(a), when using the QWS dataset and considering 10000 concrete services for every abstract service of the composition, the IW-SC, SFLGA-SC, PSGW-SC, and PS-SC algorithms

consume about 70.03%, 57.79%, 44.73%, and 37.89% more energy compared to the LS-SCA algorithm. Similarly, Figure IV.7(b) shows that for the WSC-2009 dataset, the IW-SC, PSGW-SC, SFLGA-SC, PS-SC and GAPS-SC algorithms consume almost 88.92%, 74.76%, 84.87%, 66.56%, and 40.13% more energy compared to the LS-SCA algorithm when 10000 concrete services are considered for every abstract service of the composition. This is due to the realistic energy model exploited in the LS-SCA algorithm that allows selecting concrete services with the highest energy efficiency during the composition process. However, Figure IV.7(a) shows that using the QWS dataset, the GA-SC and GAPS-SC algorithms, along with the LS-SCA-E variant, consume less energy than the LS-SCA algorithm, while Figure IV.7(b) shows that the GA-SC algorithm and the LS-SCA-E variant have low energy consumption compared to the LS-SCA algorithm using the WSC-2009 dataset. This can be explained by the fact that these approaches focus only on minimizing energy consumption during the selection process, thus decreasing the energy consumption of the composition. Note that the LS-SCA-E variant, GA-SC and GAPS-SC approaches produce a low composition QoS utility compared to the LS-SCA algorithm because these approaches do not account for QoS during composition, leading to the selection of services that are not efficient in terms of QoS (see Figure IV.6).



(a) Dataset QWS.

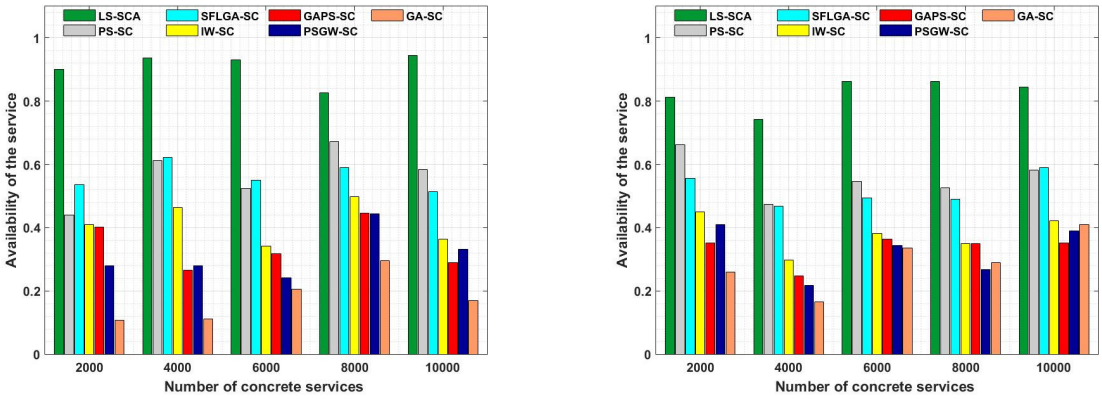
(b) Dataset WSC-2009

Figure IV.7: Impact of the concrete services' number on the energy consumption of the composition.

6.3.3 Availability of the composition

Figure IV.8 shows the result obtained when measuring the impact of the concrete services' number on the availability of the composition. As we can see in Figure IV.8(a), using the QWS dataset, the PS-SC, SFLGA-SC, IW-SC, GAPS-SC, PSGW-SC, and GA-SC algorithms offer almost 44.35%, 44.78%, 59.89%, 63.31%, 68.67%, and 77.86% less service availability compared to the LS-SCA algorithm when 10000 concrete services are considered for every abstract service of the composition. Similarly, Figure IV.8(b) shows that with the WSC-2009 dataset, the PS-SC, SFLGA-SC, IW-SC, GAPS-SC, PSGW-SC, and GA-SC algorithms offer approximately 31.09%, 30.27%, 50.10%, 58.37%, 53.93%, and

51.64% less service availability compared to the LS-SCA algorithm when 10000 concrete services are considered for every abstract service of the composition. The high availability of service achieved by the LS-SCA algorithm can be attributed to its selection mechanism that considers energy efficiency. This mechanism uses the aptitude concept, based on the expected moving distance and the user's velocity, to ensure that the selected composite services remain accessible during the user movement, leading to high service availability. The results of this scenario demonstrate that the LSSCA algorithm is more efficient in maintaining high service availability compared to existing approaches, especially when dealing with a large-scale service-oriented environment when mobility is a critical factor.



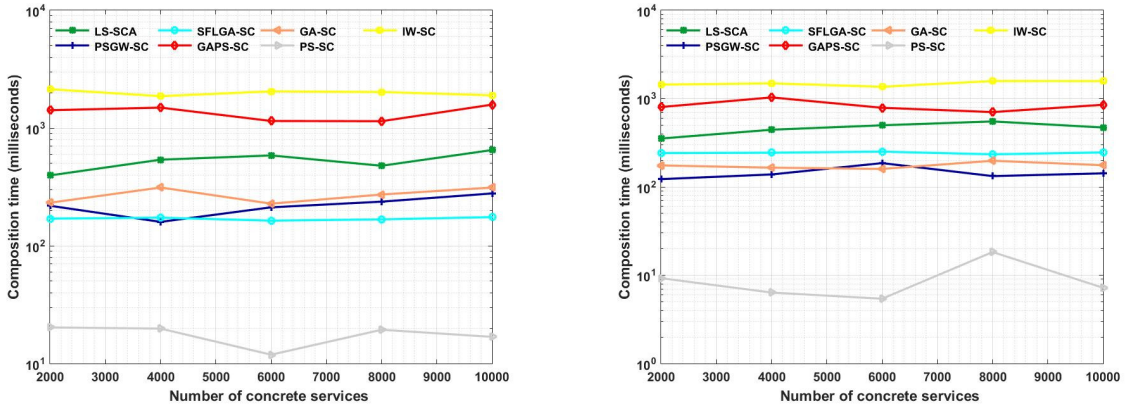
(a) Dataset QWS.

(b) Dataset WSC-2009

Figure IV.8: Impact of the concrete services' number on the availability of the composition.

6.3.4 Composition time

This simulation scenario compares the composition times obtained with the LS-SCA, SFLGA-SC, IW-SC, PS-SC, GA-SC, GAPS-SC, and PSGW-SC algorithms when the number of concrete services varies from 2000 to 10000. Figures IV.9(a) and IV.9(b) show that when using the QWS and WSC-2009 datasets, the composition time of the LS-SCA algorithm is lower than that obtained in the case of the IW-SC and GAPS-SC algorithms, even when the number of concrete services increases. This is due to the fact that the LS-SCA algorithm has good exploitation capabilities and can quickly find the best services while maintaining the population diversity during the learning phase. However, the LS-SCA algorithm has a higher composition time compared to that obtained with the SFLGA-SC, PS-SC, GA-SC, and PSGW-SC algorithms. This is mainly due to the fact that the LS-SCA algorithm requires more time to deeply explore the composition search space compared to the baseline algorithms. This additional computational effort allows the LS-SCA approach to achieve a better composition in terms of QoS utility, energy consumption, and availability (as demonstrated in Figures IV.6, IV.7, and IV.8).



(a) Dataset QWS.

(b) Dataset WSC-2009

Figure IV.9: Impact of the concrete services' number on the composition time.

6.4 Mobility assessment

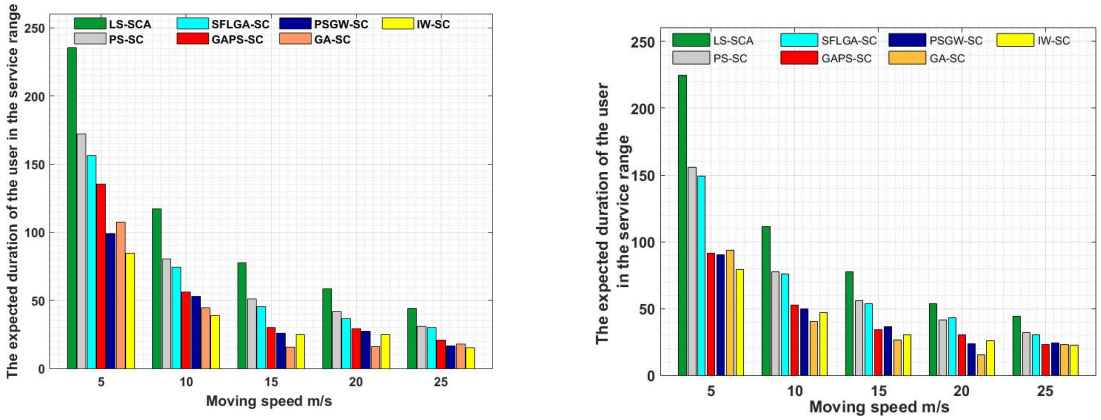
During the invocation of composite services, users may move continuously, leading therefore to a change in their device's power. This change in power can affect the energy consumption and availability of composite services. Accordingly, accounting for the user's movement and velocity when optimizing the composition process is a crucial issue.

6.4.1 The expected duration of the user

In this simulation, we investigate the impact of the user's speed on the expected time during which the user remains within the coverage area of the composite service. From Figure IV.10, we can see that the expected duration generated by the algorithms obviously decreases with the user's speed increase when using the QWS and WSC-2009 datasets. The LS-SCA algorithm achieves significantly higher performance compared to the PS-SC, SFLGA-SC, IW-SC, GAPS-SC, PSGW-SC, and GA-SC algorithms. This is due to the fact that, unlike the baseline approaches that select concrete services according only to their QoS values or energy, the selection process used in the LS-SCA algorithm is carried out considering the user's mobility, energy, and QoS values of services. In conclusion, by simultaneously considering QoS, energy consumption, and user's mobility, the LS-SCA algorithm outperforms all other baseline algorithms in maintaining service availability during user's movement.

6.4.2 Energy consumption

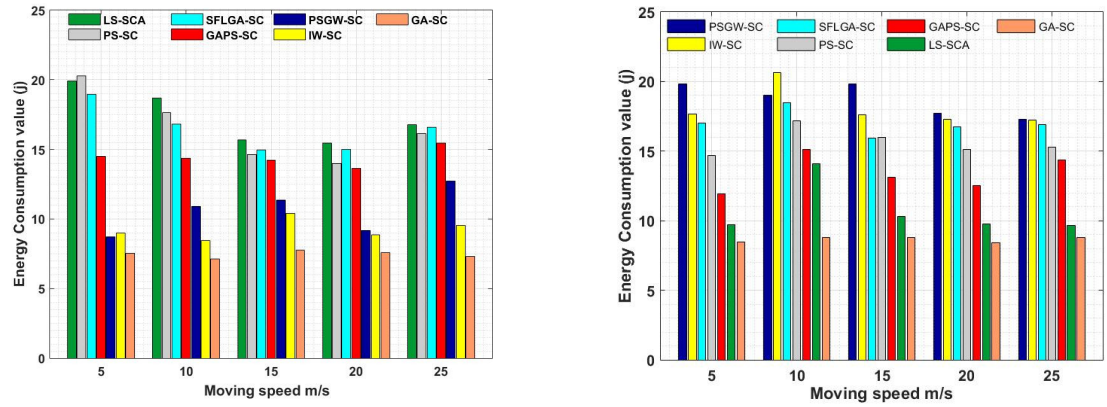
This simulation scenario assesses the impact of the user's speed on the energy consumption of composite services. As observed in Figures IV.11(a) and IV.11(b), the energy consumption of compositions generated by the IW-SC, SFLGA-SC, PS-SC, and PSGW-SC algorithms using the QWS and WSC-2009 datasets is significantly higher than that obtained with the LS-SCA algorithm. This finding is because the LS-SCA algorithm



(a) Dataset QWS. (b) Dataset WSC-2009

Figure IV.10: Impact of the users' moving speed on the expected duration of the user in the service range.

selects services while taking into account the user's speed and dynamically adjusting the data transmission locations. By performing data transmission between services and users in areas with stronger signal strength, the LS-SCA algorithm effectively reduces the energy consumption of the composition. Furthermore, Figure IV.11(a) shows that the GA-SC and GAPS-SC approaches consume slightly less energy compared to the LS-SCA algorithm because they select services only according to their energy. However, this strategy leads to a decrease in QoS utility and service availability. Nevertheless, Figure IV.11(b) illustrates that only the GA-SC algorithm consumes slightly less energy compared to the LS-SCA algorithm. This outcome demonstrates that the LS-SCA approach is adaptable and efficient in different CPSS deployment environments by considering the user's mobility, while balancing the QoS and energy consumption.

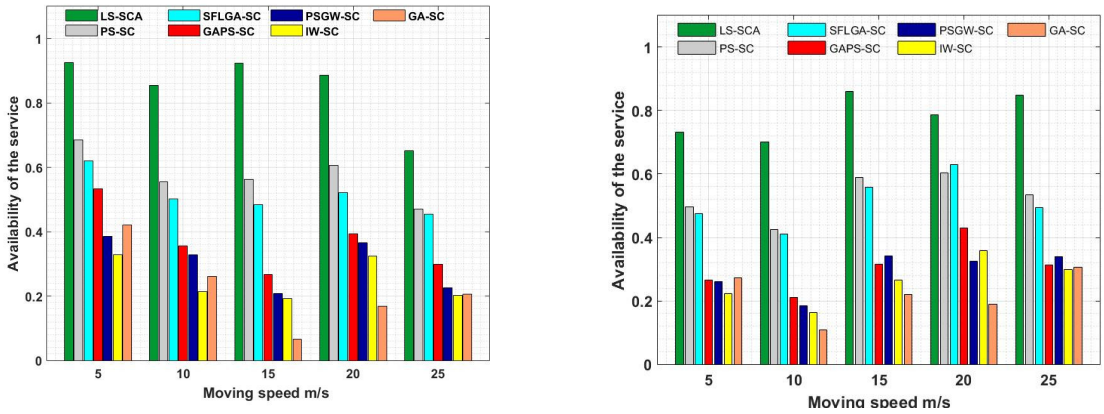


(a) Dataset QWS. (b) Dataset WSC-2009

Figure IV.11: Impact of the users' moving speed on the energy consumption of the composition.

6.4.3 Availability of composite service

The effect of user's speed on the availability of the composite services is investigated in this simulation scenario. As shown in Figure IV.12, with the QWS and WSC-2009 datasets, the availability of services for the LS-SCA, PS-SC, SFLGA-SC, IW-SC, GAPS-SC, PSGW-SC, and GA-SC algorithms decreases as the moving speed increases. However, the LS-SCA approach offers better service availability than all of the baselines due to the consideration of the user's mobility and energy consumption during the service selection process. On the one hand, taking into account mobility during the composition process allows the LSSCA algorithm to select the service offering the best aptitude, i.e., the largest expected time during which a user remains within the coverage area of the service. On the other hand, accounting for the energy criterion in the composition allows the LSSCA algorithm to select the most efficient services in terms of energy consumption, ensuring their availability as long as possible. Overall, the LS-SCA algorithm provides a more efficient and reliable service composition process by balancing the user's mobility and energy consumption of services. This results in an optimized service that provides benefits to both users and service providers.



(a) Dataset QWS.

(b) Dataset WSC-2009

Figure IV.12: Impact of the users' moving speed on the availability of the composition.

7 Conclusion

A novel approach is proposed in this chapter to address the service composition problem in the CPSS environment, taking into account QoS, energy consumption, and mobility. The introduced mobility and energy models allow for significant improvements in the performance and efficiency of the proposed approach. The mobility model based on the aptitude concept enables a dynamic adaptation and optimization of the service composition process based on the user's path and location. Furthermore, the used energy model allows the selection of the most relevant services in terms of energy consumption, leading to significant energy savings. This model performs data transmission between services

and users located in areas with high signal strength, which optimizes energy consumption and increases the availability of service. The simulation results show that the LS-SCA approach outperforms six baseline approaches by almost 23% in terms of QoS utility values, 28.72% in terms of energy consumption, and 40% in terms of service availability. This outcome is due to the balance achieved between exploration and exploitation during the mass learning and elite learning phases of the proposed approach. More precisely, the process of regrouping compositions during the mass learning phase enhances the diversity and allows a larger exploration of the optimal composition, increasing the probability of finding best compositions. In addition, mutual learning between compositions in the elite learning phase further improves their utility values, accelerating the convergence to optimal compositions. Besides, the LSSCA approach provides a more efficient and reliable service composition process by balancing both mobility, QoS, and energy consumption of services compared to existing baselines. These findings underline the potential of the proposed approach in practical real-world scenarios.

General conclusion & perspectives

1 Contribution synthesis

In this thesis, we have addressed the problem of service composition with QoS guarantees in large-scale Cyber-Physical-Social Systems (CPSS) and the Internet of Things (IoT). As these environments are interconnected and highly dynamic, the need to address adaptive service composition approaches has become crucial to ensure optimal QoS (e.g., response time, availability, reliability), balanced energy consumption, and seamless adaptation to the user mobility. However, existing approaches often fail to simultaneously consider energy efficiency, user mobility, and QoS constraints, leading to limited compositions in terms of service availability and resource utilization.

To handle these challenges, two contributions are proposed in this thesis. The first contribution introduces the Group Teaching-based Energy-efficient and QoS-aware service Composition Algorithm (GT-EQCA) designed for IoT environments where services are hosted on energy-constrained devices. This approach is proposed to overcome the limitations of most existing service composition approaches that do not simultaneously take into account the energy and QoS of services, leading to (i) a decrease in the composition's QoS and/or (ii) an unbalanced energy consumption between services, thus reducing the composition's lifetime. Given that energy is a critical resource in IoT applications, the proposed algorithm aims to achieve energy efficiency by ensuring a trade-off between energy consumption and the user's QoS requirements. To achieve this, the service selection problem under QoS constraints is formulated as a combinatorial optimization problem and solved using the lexicographic optimization technique, which is particularly well-suited for multi-objective optimization problems where a predefined priority order exists among objectives. Unlike traditional approaches that consider all available services in the composition process, the GT-EQCA approach selects only the *top-k* most relevant IoT services based on relative Pareto dominance. The latter is computed based on the energy consumption and QoS attributes of services, while considering user preferences. By restricting the search to the *top-k* services, this approach reduces computational complexity, while improving at the same time the QoS utility of the resulting composition. Once the *top-k* services are selected, the composition process is performed using the Group Teaching Optimization (GTO) algorithm to obtain a near-to-optimal composite service, i.e., a composition

that maximizes QoS utility while satisfying global QoS constraints. Unlike most existing bio-inspired optimization techniques, the GTO method does not require manual tuning of control parameters, making it highly adaptable and efficient for large-scale service composition. Simulation scenarios demonstrate that the GT-EQCA approach outperforms four baseline algorithms by almost 76% in terms of composition time, 88.8% in terms of energy consumption, and 28.4% in terms of QoS utility values, thus improving service availability and prolonging composition lifetime. These results highlight the robustness and scalability of the proposed approach in large-scale IoT environments by balancing QoS requirements and energy consumption compared to existing approaches.

The second contribution deals with the complementarity challenge that was not fully addressed in many existing approaches, and in the first contribution, particularly the absence of user's mobility consideration during the composition process. This contribution proposes the Learning-based Swarm optimization-aware Service Composition Algorithm (LS-SCA) to solve the service composition problem in Cyber-Physical-Social Systems (CPSSs). In CPSS environments, services are subject to continuous mobility that directly impacts their availability and quality, leading to significant challenges to achieve stability and reliability of the service composition. To address these issues, the LS-SCA approach simultaneously integrates the mobility aspect, energy efficiency, and QoS constraints. First, the Small World in Motion (SWIM) mobility model is employed to generate realistic user's mobility traces. Second, since excessive battery consumption of service-hosting devices can reduce runtime and cause composition failures, services involved in the composition are efficiently managed and replaced by considering the energy to ensure the continuity of the composition. To support this decision-making process, an energy consumption model is introduced to evaluate the use of services and their impact on overall energy efficiency, thus increasing the availability of services. Third, the two-phase learning-based swarm optimizer (TPLSO) method is used in the LS-SCA algorithm to find the near-to-optimal composite service that satisfies the global QoS constraints while considering energy and mobility. Unlike most bio-inspired service composition approaches, which improve the overall composition population through a given number of iterations, the TPLSO method improves only a subset of promising compositions into two phases, which reduces the computation time and maintains the composition quality. The experimental results demonstrate that the LS-SCA approach outperforms six baseline algorithms, achieving an improvement of approximately 23% in the QoS utility values, a reduction of $28,72\%$ in energy consumption, and an increase of 40% in service availability. The LSSCA approach ensures an efficient service composition process by balancing mobility, QoS constraints, and energy consumption of services compared to the existing baselines. These findings highlight the potential of the proposed approach in real-world scenarios.

2 Future search directions

In this thesis, we not only highlight the key challenges associated with QoS-aware service composition, but also propose effective strategies to address them by leveraging recent advances. The following subsections outline promising directions for future research:

2.1 Real-world validation in mobile and dynamic environments

An interesting perspective is the deployment of the approaches proposed in this thesis on mobile devices with varying computing capacities, battery limits, and network conditions. Such a deployment would provide a better understanding of the algorithms' adaptability and effectiveness in heterogeneous environments, ensuring their applicability in real-world CPSS scenarios. Beyond a simple deployment, this perspective opens the way to extend the proposed service composition approaches by considering the mobility of multiple entities, including users, services, and devices. This extension could specifically explore the integration of complementary models, such as group mobility or heterogeneous device mobility. Group mobility refers to entities that move collectively, such as connected vehicles or medical emergency teams, sharing similar trajectories. This correlated mobility offers the opportunity to anticipate their movements, pre-position services along their routes, and optimize resource allocation collectively to reduce reconfiguration costs and improve QoS. Predictive models such as the Reference Point Group Mobility Model (RPGM) can be used to predict group trajectories and proactively allocate or place services in proximity to the moving group. Meanwhile, heterogeneous device mobility involves a variety of entities such as smartphones, IoT sensors, drones, and vehicles, with different speeds, computing capacities, energy resources, and connectivity. The objective is to maintain continuous QoS and provide energy-efficient services despite these differences by classifying devices according to their capacity, predicting their trajectories using models such as Kalman filters, and dynamically adapting the orchestration and migration of services to the most suitable computing resource. One promising direction would be the design of adaptive composition algorithms that combine realistic mobility models with continuous monitoring of service availability, QoS variations, and energy consumption. These algorithms could then rely on predictive techniques, such as mobility prediction, to anticipate changes in user or service locations and proactively reconfigure the composition to minimize interruptions and service failure.

2.2 Enhancing stability and reliability via predictive learning

Further research could focus on integrating predictive analytics and reinforcement learning techniques to anticipate mobility traces and proactively adjust the service composition process. By incorporating intelligent decision-making mechanisms, the proposed approaches could dynamically adapt to fluctuations in service availability, user preferences, and environmental changes, thus improving the stability and reliability of compositions.

To develop this perspective, predictive models (e.g., Markov chains, trajectory clustering, or machine-learning-based predictors) could be driven by historical mobility traces and QoS variations to anticipate future service behavior, allowing a reconfiguration of compositions before any failure. Reinforcement learning could then be applied to continuously refine these decisions by learning from previous successes and failures of the composition.

2.3 Toward autonomous microservice composition

Another promising future research should investigate methods for autonomous microservice composition, where the system can dynamically select, orchestrate, and adapt microservices according to changes in user requirements, context, and environment. Microservices are a software architectural style in which applications are built as a set of small, loosely coupled, and independently deployable services. Unlike traditional service-oriented architectures, microservices allow developers to update, upgrade, and maintain individual components without affecting the overall application. This fine-grained modularity improves agility, scalability, and failure tolerance, making microservices useful for the creation of scalable and adaptive applications. Such approaches would rely on sophisticated techniques such as self-learning, reinforcement learning, and contextual learning to continuously refine composition approaches without human intervention. From this perspective, microservices could become proactive entities able to monitor their own performance and make reconfiguration in real-time to ensure both QoS satisfaction and resource efficiency. This evolution requires decentralized orchestration mechanisms, where multiple autonomous agents collaborate without relying on a central entity.

LIST OF PUBLICATIONS

The research conducted in this thesis has led to two publications:

Hameche, S., Khanouche, M. E., Chibani, A., & Tari, A. (2024). A group teaching optimization-based approach for energy and qos-aware internet of things services composition. *Journal of Network and Systems Management*, 32(1), 4.

Hameche, S., Khanouche, M. E., & Tari, A. (2024). Mobility and energy efficient services composition algorithm with QoS guarantee for large scale Cyber–Physical–Social Systems. *Expert Systems with Applications*, 249, 123683.

Bibliography

[Al-Masri and Mahmoud, 2008] Al-Masri, E. and Mahmoud, Q. H. (2008). Investigating web services on the world wide web. In *Proceedings of the 17th Int. Conf. on World Wide Web*, pages 795–804, New York, USA. ACM.

[Albino et al., 2015] Albino, V., Berardi, U., and Dangelico, R. M. (2015). Smart cities: Definitions, dimensions, performance, and initiatives. *Journal of urban technology*, 22(1):3–21.

[Alizadeh et al., 2020] Alizadeh, P., Osmani, A., Khanouche, M. E., Chibani, A., and Amirat, Y. (2020). Reinforcement learning for interactive qos-aware services composition. *IEEE Systems Journal*, 15(1):1098–1108.

[Aljohani and Alenazi, 2020] Aljohani, S. and Alenazi, M. (2020). Evaluation of wsn's resilience to challenges in smart cities. *Int. J. Comput. Commun. Eng*, 9:193–206.

[Alrifai et al., 2012] Alrifai, M., Risse, T., and Nejdl, W. (2012). A hybrid approach for efficient web service composition with end-to-end qos constraints. *ACM Transactions on the Web (TWEB)*, 6(2):7:1–7:31.

[Angelidou, 2015] Angelidou, M. (2015). Smart cities: A conjuncture of four forces. *Cities*, 47:95–106.

[Anthopoulos, 2015] Anthopoulos, L. G. (2015). Understanding the smart city domain: A literature review. *Transforming city governments for successful smart cities*, pages 9–21.

[Aoudia, 2022] Aoudia, I. (2022). *Composition adaptative de services pour l'Internet des objets*. PhD thesis, Université de mohamed kheider biskra.

[Ardagna and Pernici, 2007] Ardagna, D. and Pernici, B. (2007). Adaptive service composition in flexible processes. *IEEE Transactions on software engineering*, 33(6):369–384.

[Arsanjani, 2004] Arsanjani, A. (2004). Service-oriented modeling and architecture. *IBM developer works*, 1:15.

[Asghari et al., 2018] Asghari, P., Rahmani, A. M., and Javadi, H. H. S. (2018). Service composition approaches in iot: A systematic review. *Journal of Network and Computer Applications*, 120:61–77.

[Ashari et al., 2020] Ashari, I. F. et al. (2020). Implementation of cyber-physical-social system based on service oriented architecture in smart tourism. *Journal of Applied Informatics and Computing*, 4(1):66–73.

[Atzori et al., 2010] Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer networks*, 54(15):2787–2805.

[Barros et al., 2006] Barros, A., Dumas, M., and Oaks, P. (2006). Standards for web service choreography and orchestration: Status and perspectives. In *Business Process Management Workshops: BPM 2005 International Workshops, BPI, BPD, ENEI, BPRM, WSCOBPM, BPS, Nancy, France, September 5, 2005. Revised Selected Papers 3*, pages 61–74. Springer.

[Bei et al., 2024] Bei, L., Wenlin, L., Xin, S., and Xibin, X. (2024). An improved aco based service composition algorithm in multi-cloud networks. *Journal of Cloud Computing*, 13(1):17.

[Benatallah et al., 2005] Benatallah, B., Dijkman, R. M., Dumas, M., and Maamar, Z. (2005). Service-composition: concepts, techniques, tools and trends. In *Service-Oriented Software System Engineering: Challenges and Practices*, pages 48–67. IGI Global.

[Benghozi et al., 2008] Benghozi, P.-J., Bureau, S., and Massit-Folea, F. (2008). L'internet des objets. quels enjeux pour les européens?

[Bondavalli et al., 2016] Bondavalli, A., Bouchenak, S., and Kopetz, H. (2016). *Cyber-physical systems of systems: foundations–a conceptual model and some derivations: the AMADEOS legacy*, volume 10099. Springer.

[Borgia, 2014] Borgia, E. (2014). The internet of things vision: Key features, applications and open issues. *Computer Communications*, 54:1–31.

[Camtepe and Yener, 2007] Camtepe, S. A. and Yener, B. (2007). Modeling and detection of complex attacks. In *2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops-SecureComm 2007*, pages 234–243. IEEE.

[Casati and Shan, 2002] Casati, F. and Shan, M.-C. (2002). Event-based interaction management for composite e-services in eflow. *Information Systems Frontiers*, 4:19–31.

[Cassandras, 2016] Cassandras, C. G. (2016). Smart cities as cyber-physical social systems. *Engineering*, 2(2):156–158.

[Chaganti et al., 2021] Chaganti, R., Gupta, D., and Vemprala, N. (2021). Intelligent network layer for cyber-physical systems security. *International Journal of Smart Security Technologies (IJSST)*, 8(2):42–58.

[Chattopadhyay and Banerjee, 2020] Chattopadhyay, S. and Banerjee, A. (2020). Qos-aware automatic web service composition with multiple objectives. *ACM Transactions on the Web (TWEB)*, 14(3):1–38.

[Chen et al., 2019] Chen, Y., Huang, J., Lin, C., and Shen, X. (2019). Multi-objective service composition with qos dependencies. *IEEE Transactions on Cloud Computing*, 7(2):537–552.

[Cho et al., 2015] Cho, J.-H., Ko, H.-G., and Ko, I.-Y. (2015). Adaptive service selection according to the service density in multiple qos aspects. *IEEE Trans. Services Comput.*, 9(6):883–894.

[Comuzzi and Pernici, 2005] Comuzzi, M. and Pernici, B. (2005). An architecture for flexible web service qos negotiation. In *Ninth IEEE International EDOC Enterprise Computing Conference (EDOC'05)*, pages 70–79. IEEE.

[Dahan et al., 2021a] Dahan, F., Binsaeedan, W., Altaf, M., Al-Asaly, M. S., and Hassan, M. M. (2021a). An efficient hybrid metaheuristic algorithm for qos-aware cloud service composition problem. *IEEE Access*, 9:95208–95217.

[Dahan et al., 2021b] Dahan, F., El Hindi, K., Ghoneim, A., and Alsalman, H. (2021b). An enhanced ant colony optimization based algorithm to solve qos-aware web service composition. *IEEE Access*, 9:34098–34111.

[De et al., 2017] De, S., Zhou, Y., Larizgoitia Abad, I., and Moessner, K. (2017). Cyber–physical–social frameworks for urban big data systems: A survey. *Applied Sciences*, 7(10):1017.

[Deng et al., 2016a] Deng, S., Huang, L., Hu, D., Zhao, J. L., and Wu, Z. (2016a). Mobility-enabled service selection for composite services. *IEEE Transactions on Services Computing*, 9(3):394–407.

[Deng et al., 2016b] Deng, S., Huang, L., Hu, D., Zhao, J. L., and Wu, Z. (2016b). Mobility-enabled service selection for composite services. *IEEE Transactions on Services Computing*, 9(3):394–407.

[Deng et al., 2018] Deng, S., Huang, L., Li, Y., Zhou, H., Wu, Z., Cao, X., Kataev, M. Y., and Li, L. (2018). Toward risk reduction for mobile service composition. *IEEE transactions on cybernetics*, 46(8):1807–1816.

[Deng et al., 2017a] Deng, S., Huang, L., Taheri, J., Yin, J., Zhou, M., and Zomaya, A. Y. (Mar. 2017a). Mobility-aware service composition in mobile communities. *IEEE Trans. Syst., Man, and Cybern.: Syst.*, 47(3):555–568.

[Deng et al., 2016c] Deng, S., Huang, L., Wu, H., and Wu, Z. (2016c). Constraints-driven service composition in mobile cloud computing. In *2016 IEEE International Conference on Web Services (ICWS)*, pages 228–235. IEEE.

[Deng et al., 2017b] Deng, S., Wu, H., Tan, W., Xiang, Z., and Wu, Z. (2017b). Mobile service selection for composition: an energy consumption perspective. *IEEE Transactions on Automation Science and Engineering*, 14(3):1478–1490.

[Ding and Jiang, 2018] Ding, K. and Jiang, P. (2018). Incorporating social sensors, cyber-physical system nodes, and smart products for personalized production in a social manufacturing environment. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 232(13):2323–2338.

[Dorsemaine et al., 2015] Dorsemaine, B., Gaulier, J.-P., Wary, J.-P., Kheir, N., and Urien, P. (2015). Internet of things: a definition & taxonomy. In *2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies*, pages 72–77. IEEE.

[Dressler, 2018] Dressler, F. (2018). Cyber physical social systems: Towards deeply integrated hybridized systems. In *2018 International Conference on Computing, Networking and Communications (ICNC)*, pages 420–424. IEEE.

[Du et al., 2018] Du, R., Santi, P., Xiao, M., Vasilakos, A. V., and Fischione, C. (2018). The sensable city: A survey on the deployment and management for smart city monitoring. *IEEE Communications Surveys & Tutorials*, 21(2):1533–1560.

[Faratin et al., 1998] Faratin, P., Sierra, C., and Jennings, N. R. (1998). Negotiation decision functions for autonomous agents. *Robotics and Autonomous Systems*, 24(3-4):159–182.

[Fishburn, 1974] Fishburn, P. C. (1974). Exceptional paper-lexicographic orders, utilities and decision rules: A survey. *Management science*, 20(11):1442–1471.

[Furthmüller and Waldhorst, 2012] Furthmüller, J. and Waldhorst, O. P. (2012). Energy-aware resource sharing with mobile devices. *Computer Networks*, 56(7):1920–1934.

[Gao and Liu, 2020] Gao, Y. and Liu, B. (2020). Energy efficient and delay aware service selection in mobile edge computing. In *2020 IEEE 20th International Conference on Communication Technology (ICCT)*, pages 1546–1550. IEEE.

[Gati et al., 2021] Gati, N. J., Yang, L. T., Feng, J., Nie, X., Ren, Z., and Tarus, S. K. (2021). Differentially private data fusion and deep learning framework for cyber–physical–social systems: State-of-the-art and perspectives. *Information Fusion*, 76:298–314.

[Geebelen et al., 2014] Geebelen, D., Geebelen, K., Truyen, E., Michiels, S., Suykens, J. A., Vandewalle, J., and Joosen, W. (2014). Qos prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset. *Information Sciences*, 268:397–424.

[Gharaibeh et al., 2017] Gharaibeh, A., Salahuddin, M. A., Hussini, S. J., Khreishah, A., Khalil, I., Guizani, M., and Al-Fuqaha, A. (2017). Smart cities: A survey on data management, security, and enabling technologies. *IEEE Communications Surveys & Tutorials*, 19(4):2456–2501.

[Gharib et al., 2017] Gharib, M., Lollini, P., and Bondavalli, A. (2017). Towards an approach for analyzing trust in cyber-physical-social systems. In *2017 12th System of Systems Engineering Conference (SoSE)*, pages 1–6. IEEE.

[Goldman et al., 2012] Goldman, A., Ngoko, Y., and Milojicic, D. (2012). An analytical approach for predicting qos of web services choreographies. In *Proceedings of the 10th International Workshop on Middleware for Grids, Clouds and e-Science*, pages 1–6.

[Guinard et al., 2010] Guinard, D., Trifa, V., Karnouskos, S., Spiess, P., and Savio, D. (2010). Interacting with the soa-based internet of things: Discovery, query, selection, and on-demand provisioning of web services. *IEEE transactions on Services Computing*, 3(3):223–235.

[Guo et al., 2015] Guo, B., Yu, Z., and Zhou, X. (2015). A data-centric framework for cyber-physical-social systems. *It Professional*, 17(6):4–7.

[Guzel and Ozdemir, 2022] Guzel, M. and Ozdemir, S. (2022). Fair and energy-aware iot service composition under qos constraints. *The Journal of Supercomputing*, pages 1–28.

[Halfaoui et al., 2015] Halfaoui, A., Hadjila, F., and Didi, F. (2015). Qos-aware web services selection based on fuzzy dominance. In *IFIP Int. Conf. on Computer Science and its Applications*, pages 291–300, Cham. Springer.

[Hameche et al., 2024a] Hameche, S., Khanouche, M. E., Chibani, A., and Tari, A. (2024a). A group teaching optimization-based approach for energy and qos-aware internet of things services composition. *Journal of Network and Systems Management*, 32(1):4.

[Hameche et al., 2024b] Hameche, S., Khanouche, M. E., and Tari, A. (2024b). Mobility and energy efficient services composition algorithm with qos guarantee for large scale cyber–physical–social systems. *Expert Systems with Applications*, 249:123683.

[Hamzei and Navimipour, 2018] Hamzei, M. and Navimipour, N. J. (2018). Toward efficient service composition techniques in the internet of things. *IEEE Internet of Things Journal*, 5(5):3774–3787.

[Han et al., 2009] Han, Y., Wang, J., and Zhang, P. (2009). Business-oriented service modeling: A case study. *Simulation Modelling Practice and Theory*, 17(8):1413–1429.

[Haque et al., 2014] Haque, S. A., Aziz, S. M., and Rahman, M. (2014). Review of cyber-physical system in healthcare. *international journal of distributed sensor networks*, 10(4):217415.

[Hossain et al., 2016] Hossain, M. S., Moniruzzaman, M., Muhammad, G., Ghoneim, A., and Alamri, A. (2016). Big data-driven service composition using parallel clustered particle swarm optimization in mobile environment. *IEEE Transactions on Services Computing*, 9(5):806–817.

[Huang et al., 2009] Huang, A. F., Lan, C.-W., and Yang, S. J. (2009). An optimal qos-based web service selection scheme. *Information Sciences*, 179(19):3309–3322.

[Huang et al., 2016] Huang, C., Marshall, J., Wang, D., and Dong, M. (2016). Towards reliable social sensing in cyber-physical-social systems. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1796–1802. IEEE.

[Huhns and Singh, 2005] Huhns, M. N. and Singh, M. P. (2005). Service-oriented computing: Key concepts and principles. *IEEE Internet computing*, 9(1):75–81.

[Hwang and Yoon, 1981] Hwang, C.-L. and Yoon, K. (1981). Multiple criteria decision making. *Lecture notes in economics and mathematical systems*, 186:58–191.

[Ibrahim et al., 2020] Ibrahim, G. J., Rashid, T. A., and Akinsolu, M. O. (2020). An energy efficient service composition mechanism using a hybrid meta-heuristic algorithm in a mobile cloud environment. *Journal of Parallel and Distributed Computing*, 143:77–87.

[Jatoth et al., 2019] Jatoth, C., Gangadharan, G., and Fiore, U. (2019). Optimal fitness aware cloud service composition using modified invasive weed optimization. *Swarm and evolutionary computation*, 44:1073–1091.

[Jiang et al., 2016] Jiang, P., Ding, K., and Leng, J. (2016). Towards a cyber-physical-social-connected and service-oriented manufacturing paradigm: Social manufacturing. *Manufacturing Letters*, 7:15–21.

[Jin et al., 2022] Jin, H., Lv, S., Yang, Z., and Liu, Y. (2022). Eagle strategy using uniform mutation and modified whale optimization algorithm for qos-aware cloud service composition. *Applied Soft Computing*, 114:108053.

[Khalaf and Leymann, 2003] Khalaf, R. and Leymann, F. (2003). On web services aggregation. In *Technologies for E-Services: 4th International Workshop, TES 2003, Berlin, Germany, September 7-8, 2003. Proceedings 4*, pages 1–13. Springer.

[Khanam et al., 2018] Khanam, R., Kumar, R. R., and Kumar, C. (2018). Qos based cloud service composition with optimal set of services using pso. In *2018 4th international conference on recent advances in information technology (RAIT)*, pages 1–6. IEEE.

[Khanouche et al., 2016] Khanouche, M. E., Amirat, Y., Chibani, A., Kerkar, M., and Yachir, A. (2016). Energy-centered and qos-aware services selection for internet of things. *IEEE Trans. Automation Science and Engineering*, 13(3):1256–1269.

[Khanouche et al., 2020a] Khanouche, M. E., Atmani, N., and Cherifi, A. (2020a). Improved teaching learning-based qos-aware services composition for internet of things. *IEEE Systems Journal*, 14(3):4155–4164.

[Khanouche et al., 2019a] Khanouche, M. E., Attal, F., Amirat, Y., Chibani, A., and Kerkar, M. (2019a). Clustering-based and qos-aware services composition algorithm for ambient intelligence. *Inf. Sciences*, 482:419–439.

[Khanouche et al., 2019b] Khanouche, M. E., Attal, F., Amirat, Y., Chibani, A., and Kerkar, M. (2019b). Clustering-based and qos-aware services composition algorithm for ambient intelligence. *Information Sciences*, 482:419–439.

[Khanouche et al., 2020b] Khanouche, M. E., Gadouche, H., Farah, Z., and Tari, A. (2020b). Flexible qos-aware services composition for service computing environments. *Computer Networks*, 166:106982.

[Kim et al., 2020] Kim, M. J., Cho, M. E., and Jun, H. J. (2020). Developing design solutions for smart homes through user-centered scenarios. *Frontiers in psychology*, 11:335.

[Kogut and Singh, 1988] Kogut, B. and Singh, H. (1988). The effect of national culture on the choice of entry mode. *Journal of international business studies*, 19:411–432.

[Kona et al., 2009] Kona, S., Bansal, A., Blake, M. B., Bleul, S., and Weise, T. (2009). Wsc-2009: a quality of service-oriented web services challenge. In *2009 ieee conference on commerce and enterprise computing*, pages 487–490. IEEE.

[Kouicem et al., 2022] Kouicem, A., Khanouche, M. E., and Tari, A. (2022). Novel bat algorithm for qos-aware services composition in large scale internet of things. *Cluster Computing*, 25(5):3683–3697.

[Lan et al., 2020] Lan, R., Zhu, Y., Lu, H., Liu, Z., and Luo, X. (2020). A two-phase learning-based swarm optimizer for large-scale optimization. *IEEE Transactions on Cybernetics*. 10.1109/TCYB.2020.2968400.

[Leguay et al., 2006] Leguay, J., Lindgren, A., Scott, J., Riedman, T., Crowcroft, J., and Hui, P. (2006). Crawdad trace upmc/content/imote/cambridge (v. 2006–11–17). *Content/Imote/Cambridge*, 2006:11–17.

[Li et al., 2022] Li, J., Ren, H., Li, C., and Chen, H. (2022). A novel and efficient salp swarm algorithm for large-scale qos-aware service composition selection. *Computing*, 104(9):2031–2051.

[Li and Zhu, 2023] Li, J. and Zhu, S. (2023). Service composition considering energy consumption of users and transferring files in a multicloud environment. *Journal of Cloud Computing*, 12(1):1–12.

[Liang et al., 2021] Liang, H., Wen, X., Liu, Y., Zhang, H., Zhang, L., and Wang, L. (2021). Logistics-involved qos-aware service composition in cloud manufacturing with deep reinforcement learning. *Robotics and Computer-Integrated Manufacturing*, 67:101991.

[Liu et al., 2004] Liu, Y., Ngu, A. H., and Zeng, L. Z. (2004). Qos computation and policing in dynamic web service selection. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 66–73.

[Liu et al., 2011] Liu, Z., Yang, D.-s., Wen, D., Zhang, W.-m., and Mao, W. (2011). Cyber-physical-social systems for command and control. *IEEE Intelligent Systems*, 26(4):92–96.

[Loeckx, 1981] Loeckx, J. (1981). Lectu re notes in economics and mathematical systems.

[MacKenzie et al., 2006] MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F., Metz, R., and Hamilton, B. A. (2006). Reference model for service oriented architecture 1.0. *OASIS standard*, 12(S18):1–31.

[Mahmoud et al., 2015] Mahmoud, R., Yousuf, T., Aloul, F., and Zualkernan, I. (2015). Internet of things (iot) security: Current status, challenges and prospective measures. In *2015 10th international conference for internet technology and secured transactions (ICITST)*, pages 336–341. IEEE.

[Maximilien and Singh, 2004] Maximilien, E. M. and Singh, M. P. (2004). A framework and ontology for dynamic web services selection. *IEEE Internet Computing*, 8(5):84–93.

[Mei and Stefa, 2009] Mei, A. and Stefa, J. (2009). Swim: A simple model to generate small mobile worlds. In *IEEE INFOCOM 2009*, pages 2106–2113. IEEE.

[Morello et al., 2017] Morello, R., Mukhopadhyay, S. C., Liu, Z., Slomovitz, D., and Samantaray, S. R. (2017). Advances on sensing technologies for smart cities and power grids: A review. *IEEE Sensors Journal*, 17(23):7596–7610.

[Murakami, 2012] Murakami, K. J. (2012). Cpss (cyber-physical-social system) initiative-beyond cps (cyber-physical system) for a better future. In *Keynote Speech, the First Japan-Egypt Conference on Electronics Communication and Computers JEC-ECC*.

[Nandury and Begum, 2015] Nandury, S. V. and Begum, B. A. (2015). Smart wsn-based ubiquitous architecture for smart cities. In *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 2366–2373. IEEE.

[Ngoko et al., 2013] Ngoko, Y., Goldman, A., and Milojicic, D. (2013). Service selection in web service compositions optimizing energy consumption and service response time. *Journal of Internet Services and Applications*, 4(1):19.

[Nisiotis et al., 2020] Nisiotis, L., Alboul, L., and Beer, M. (2020). A prototype that fuses virtual reality, robots, and social networks to create a new cyber–physical–social eco-society system for cultural heritage. *Sustainability*, 12(2):645.

[Orriëns et al., 2003] Orriëns, B., Yang, J., and Papazoglou, M. P. (2003). Model driven service composition. In *Service-Oriented Computing-ICSOC 2003: First International Conference, Trento, Italy, December 15-18, 2003. Proceedings 1*, pages 75–90. Springer.

[O'Sullivan et al., 2002] O'Sullivan, J., Edmond, D., and Ter Hofstede, A. (2002). What's in a service? *Distributed and Parallel Databases*, 12:117–133.

[Papazoglou, 2003] Papazoglou, M. P. (2003). Service-oriented computing: Concepts, characteristics and directions. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering, 2003. WISE 2003.*, pages 3–12. IEEE.

[Papazoglou and Van Den Heuvel, 2007] Papazoglou, M. P. and Van Den Heuvel, W.-J. (2007). Service oriented architectures: approaches, technologies and research issues. *The VLDB journal*, 16:389–415.

[Peltz, 2003] Peltz, C. (2003). Web services orchestration and choreography. *Computer*, 36(10):46–52.

[Peng et al., 2025] Peng, G., Wen, Y., Liu, J., Kang, G., Zhang, B., and Zhou, M. (2025). Energy-aware cloud manufacturing service selection and scheduling optimization. *International Journal of Computer Integrated Manufacturing*, 38(3):309–334.

[Peng et al., 2020] Peng, Q., Xia, Y., Zhou, M., Luo, X., Wang, S., Wang, Y., Wu, C., Pang, S., and Lin, M. (2020). Reliability-aware and deadline-constrained mobile service composition over opportunistic networks. *IEEE Transactions on Automation Science and Engineering*, 18(3):1012–1025.

[Porkodi and Bhuvaneswari, 2014] Porkodi, R. and Bhuvaneswari, V. (2014). The internet of things (iot) applications and communication enabling technology standards: An overview. In *2014 International conference on intelligent computing applications*, pages 324–329. IEEE.

[Rahman, 2017] Rahman, S. M. (2017). Cyber-physical-social system between a humanoid robot and a virtual human through a shared platform for adaptive agent ecology. *IEEE/CAA Journal of Automatica Sinica*, 5(1):190–203.

[Rahmani et al., 2018] Rahmani, A. M., Gia, T. N., Negash, B., Anzanpour, A., Azimi, I., Jiang, M., and Liljeberg, P. (2018). Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach. *Future Generation Computer Systems*, 78:641–658.

[Sarkar et al., 2014] Sarkar, C., SN, A. U. N., Prasad, R. V., Rahim, A., Neisse, R., and Baldini, G. (2014). Diat: A scalable distributed architecture for iot. *IEEE Internet of Things journal*, 2(3):230–239.

[Sefati and Navimipour, 2021] Sefati, S. and Navimipour, N. J. (2021). A qos-aware service composition mechanism in the internet of things using a hidden-markov-model-based optimization algorithm. *IEEE Internet of Things Journal*, 8(20):15620–15627.

[Seghir, 2021] Seghir, F. (2021). A genetic algorithm with an elitism replacement method for solving the nonfunctional web service composition under fuzzy qos parameters. In *2021 International Conference on Artificial Intelligence and Mechatronics Systems (AIMS)*, pages 1–7. IEEE.

[Sharif et al., 2025] Sharif, R. H., Masdari, M., Ghaffari, A., and Gharehchopogh, F. S. (2025). A chaotic-based artificial rabbit optimization and dandelion optimizer for qos-aware web service composition in mobile edge computing. *Neural Computing and Applications*, pages 1–60.

[Sharma et al., 2020] Sharma, T., Bambenek, J. C., and Bashir, M. (2020). Preserving privacy in cyber-physical-social systems: An anonymity and access control approach. *ISSN 1613-0073*.

[Sheng et al., 2014] Sheng, Q. Z., Qiao, X., Vasilakos, A. V., Szabo, C., Bourne, S., and Xu, X. (2014). Web services composition: A decade's overview. *Information Sciences*, 280:218–238.

[Sheth et al., 2013] Sheth, A., Anantharam, P., and Henson, C. (2013). Physical-cyber-social computing: An early 21st century approach. *IEEE Intelligent Systems*, 28(1):78–82.

[Smirnov et al., 2015] Smirnov, A., Kashevnik, A., and Ponomarev, A. (2015). Multi-level self-organization in cyber-physical-social systems: Smart home cleaning scenario. *Procedia Cirp*, 30:329–334.

[Smirnov et al., 2014] Smirnov, A., Levashova, T., Shilov, N., and Sandkuhl, K. (2014). Ontology for cyber-physical-social systems self-organisation. In *Proceedings of 16th Conference of Open Innovations Association FRUCT*, pages 101–107. IEEE.

[Sowe et al., 2016] Sowe, S. K., Simmon, E., Zettsu, K., De Vaulx, F., and Bojanova, I. (2016). Cyber-physical-human systems: Putting people in the loop. *IT professional*, 18(1):10–13.

[Steuer and Choo, 1983] Steuer, R. E. and Choo, E.-U. (1983). An interactive weighted tchebycheff procedure for multiple objective programming. *Mathematical programming*, 26:326–344.

[Su et al., 2017] Su, Z., Qi, Q., Xu, Q., Guo, S., and Wang, X. (2017). Incentive scheme for cyber physical social systems based on user behaviors. *IEEE Transactions on Emerging Topics in Computing*, 8(1):92–103.

[Sun et al., 2019a] Sun, M., Zhou, Z., Wang, J., Du, C., and Gaaloul, W. (2019a). Energy-efficient iot service composition for concurrent timed applications. *Future Gener. Comput. Syst.*, 100:1017–1030.

[Sun et al., 2019b] Sun, M., Zhou, Z., Zhang, W., and Hung, P. C. (2019b). Iot service composition for concurrent timed applications. In *2019 IEEE International Conference on Web Services (ICWS)*, pages 50–54. IEEE.

[Sun and Zhao, 2012] Sun, S. X. and Zhao, J. (2012). A decomposition-based approach for service composition with global qos guarantees. *Inf. Sciences*, 199:138–153.

[Sun and Ansari, 2016] Sun, X. and Ansari, N. (2016). Edgeiot: Mobile edge computing for the internet of things. *IEEE Communications Magazine*, 54(12):22–29.

[Tian et al., 2019] Tian, S., Yang, W., Le Grange, J. M., Wang, P., Huang, W., and Ye, Z. (2019). Smart healthcare: making medical care more intelligent. *Global Health Journal*, 3(3):62–65.

[Tong et al., 2020] Tong, E., Chen, L., and Li, H. (Sept.-Oct. 2020). Energy-aware service selection and adaptation in wireless sensor networks with qos guarantee. *IEEE Trans. Services Comput.*, 5(13):829–842.

[Wang, 2010] Wang, F.-Y. (2010). The emergence of intelligent enterprises: From cps to cpss. *IEEE Intelligent Systems*, 25(4):85–88.

[Wang et al., 2020a] Wang, H., Hu, X., Yu, Q., Gu, M., Zhao, W., Yan, J., and Hong, T. (2020a). Integrating reinforcement learning and skyline computing for adaptive service composition. *Information Sciences*, 519:141–160.

[Wang et al., 2020b] Wang, S., Guo, Y., Li, Y., and Hsu, C.-H. (2020b). Cultural distance for service composition in cyber–physical–social systems. *Future Generation Computer Systems*, 108:1049–1057.

[Wang et al., 2017] Wang, S., Zhou, A., Yang, M., Sun, L., Hsu, C.-H., and Yang, F. (2017). Service composition in cyber-physical-social systems. *IEEE Transactions on Emerging Topics in Computing*, 8(1):82–91.

[Want, 2006] Want, R. (2006). An introduction to rfid technology. *IEEE pervasive computing*, 5(1):25–33.

[Wierzbicki, 1986] Wierzbicki, A. P. (1986). On the completeness and constructiveness of parametric characterizations to vector optimization problems. *Operations-Research-Spektrum*, 8(2):73–87.

[Wu et al., 2019] Wu, H., Deng, S., Li, W., Yin, J., Li, X., Feng, Z., and Zomaya, A. Y. (2019). Mobility-aware service selection in mobile edge computing systems. In *2019 IEEE International Conference on Web Services (ICWS)*, pages 201–208. IEEE.

[Xiong et al., 2015] Xiong, G., Zhu, F., Liu, X., Dong, X., Huang, W., Chen, S., and Zhao, K. (2015). Cyber-physical-social system in intelligent transportation. *IEEE/CAA Journal of Automatica Sinica*, 2(3):320–333.

[Xu et al., 2019] Xu, X., Sheng, Q. Z., Wang, Z., Yao, L., et al. (Mar.-Apr. 2019). Novel artificial bee colony algorithms for qos-aware service selection. *IEEE Trans. Services Comput.*, 12(2):247–261.

[Xue and Yu, 2017] Xue, Y. and Yu, X. (2017). Beyond smart grid—cyber–physical–social system in energy future [point of view]. *Proceedings of the IEEE*, 105(12):2290–2292.

[Yaacoub et al., 2020] Yaacoub, J.-P. A., Salman, O., Noura, H. N., Kaaniche, N., Chehab, A., and Malli, M. (2020). Cyber-physical systems security: Limitations, issues and future trends. *Microprocessors and microsystems*, 77:103201.

[Yachir, 2014] Yachir, A. (2014). *Composition dynamique de services sensibles au contexte dans les systèmes intelligents ambiants*. PhD thesis, Université Paris-Est; Université des sciences et de la technologie Houari ….

[Yang et al., 2020] Yang, Y., Yang, B., Wang, S., Jin, T., and Li, S. (2020). An enhanced multi-objective grey wolf optimizer for service composition in cloud manufacturing. *Applied Soft Computing*, 87:106003.

[Yang and Li, 2014] Yang, Z. and Li, D. (2014). Iot information service composition driven by user requirement. In *2014 IEEE 17th international conference on computational science and engineering*, pages 1509–1513. IEEE.

[Yilma et al., 2019] Yilma, B. A., Naudet, Y., and Panetto, H. (2019). Introduction to personalisation in cyber-physical-social systems. In *On the Move to Meaningful Internet Systems: OTM 2018 Workshops: Confederated International Workshops: EI2N, FBM, ICSP, and Meta4eS 2018, Valletta, Malta, October 22–26, 2018, Revised Selected Papers*, pages 25–35. Springer.

[Yilma et al., 2021] Yilma, B. A., Panetto, H., and Naudet, Y. (2021). Systemic formalisation of cyber-physical-social system (cpss): A systematic literature review. *Computers in Industry*, 129:103458.

[Yu and Bouguettaya, 2011] Yu, Q. and Bouguettaya, A. (2011). Efficient service skyline computation for composite service selection. *IEEE Transactions on Knowledge and Data Engineering*, 25(4):776–789.

[Yu et al., 2007] Yu, T., Zhang, Y., and Lin, K.-J. (2007). Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Transactions on the Web (TWEB)*, 1(1):6–es.

[Yuan et al., 2019] Yuan, Y., Zhang, W., Zhang, X., and Zhai, H. (2019). Dynamic service selection based on adaptive global qos constraints decomposition. *Symmetry*, 11(3):403.

[Zeng et al., 2020] Zeng, J., Yang, L. T., Lin, M., Ning, H., and Ma, J. (2020). A survey: Cyber-physical-social systems and their system-level design methodology. *Future Generation Computer Systems*, 105:1028–1042.

[Zeng et al., 2004] Zeng, L., Benatallah, B., Ngu, A. H., Dumas, M., Kalagnanam, J., and Chang, H. (2004). Qos-aware middleware for web services composition. *IEEE Transactions on software engineering*, 30(5):311–327.

[Zhang and Jin, 2020] Zhang, Y. and Jin, Z. (2020). Group teaching optimization algorithm: A novel metaheuristic method for solving global optimization problems. *Expert Systems with Applications*, 148:113246.

[Zhang et al., 2015] Zhang, Y., Qiu, M., Tsai, C.-W., Hassan, M. M., and Alamri, A. (2015). Health-cps: Healthcare cyber-physical system assisted by cloud and big data. *IEEE Systems Journal*, 11(1):88–95.

[Zhang et al., 2019] Zhang, Y., Tao, F., Liu, Y., Zhang, P., Cheng, Y., and Zuo, Y. (2019). Long/short-term utility aware optimal selection of manufacturing service composition toward industrial internet platforms. *IEEE Transactions on Industrial Informatics*, 15(6):3712–3722.

[Zhao et al., 2017] Zhao, D., Zhou, Z., Ning, K., Duan, Y., and Zhang, L.-J. (2017). An energy-aware service composition mechanismss in service-oriented wireless sensor networks. In *2017 IEEE Int. Conf. on Internet of Things (ICIOT)*, pages 89–96, Honolulu, USA. IEEE.

[Zribi, 2014] Zribi, S. (2014). *La gouvernance SOA pour une approche de conception de Système d'Information de Médiation: réconciliation non-fonctionnelle de services pour mettre en œuvre les processus métier*. PhD thesis, Ecole des Mines d'Albi-Carmaux.

Abstract. This thesis addresses the problem of QoS-aware service composition in large-scale Cyber-Physical-Social Systems (CPSS) and the Internet of Things (IoT). These environments are highly dynamic and interconnected, which makes ensuring optimal Quality of Service (QoS), balanced energy consumption, and adaptation to user mobility particularly challenging. Existing approaches often fail to account for these dimensions simultaneously, resulting in reduced service availability and lower composition quality. To overcome these limitations, the first contribution introduces the Group Teaching-based Energy-efficient and QoS-aware Composition Algorithm (GT-EQCA) for IoT environments. By formulating the problem as a multi-objective combinatorial optimization and selecting only the top-k most relevant services, the GT-EQCA algorithm reduces the computation time while maintaining a high QoS level. It employs the Group Teaching Optimization (GTO) algorithm, which avoids the need for hard parameter tuning and ensures scalability. The experimental results show improvements of up to 76% in composition time, 88% in energy efficiency, and 28% in QoS utility. However, the mobility aspect is not addressed in this algorithm. To deal with this limitation, the second contribution proposes the Learning-based Swarm optimization-aware Service Composition Algorithm (LS-SCA) by jointly considering mobility, energy, and QoS during the composition process. Using a realistic mobility model and a two-phase learning-based swarm optimizer, the LS-SCA algorithm reduces computation time while improving composition quality. The results show 23% higher QoS utility, 28% less energy consumption, and 40% higher service availability compared to the literature baselines. This thesis proposes adaptive, energy-efficient, and mobility-aware algorithms for scalable and reliable service composition in dynamic IoT and CPSS environments.
Keywords : Service Composition, Quality of Service (QoS), Energy Efficiency, Mobility, Internet of Things (IoT), Cyber-Physical-Social Systems (CPSS), Metaheuristics.

Résumé. Cette thèse aborde le problème de la composition de services en tenant compte de la qualité de service (QoS) dans les systèmes cyber-physiques-sociaux (CPSS) à grande échelle et l'Internet des objets (IoT). Ces environnements sont très dynamiques et interconnectés, ce qui rend la garantie d'une QoS optimale, d'une consommation d'énergie équilibrée et d'une adaptation à la mobilité des utilisateurs un défi majeur. Les approches existantes ne tiennent pas simultanément compte de ces dimensions, conduisant ainsi à une disponibilité réduite des services et une faible qualité de la composition. Pour pallier ces limitations, la première contribution introduit l'algorithme GT-EQCA (Group Teaching-based Energy-efficient and QoS-aware Composition Algorithm) pour les environnements IoT. En formulant la composition de services comme un problème d'optimisation combinatoire multi-objectifs et en sélectionnant uniquement les k services les plus pertinents, l'algorithme GT-EQCA réduit le temps de calcul tout en maintenant un niveau de QoS élevé. Il utilise l'algorithme d'optimisation de l'enseignement en groupe (GTO), qui ne requiert pas un hyper paramétrages et garantit la scalabilité. Les résultats expérimentaux montrent des améliorations allant jusqu'à 76 % en termes de temps de composition, 88 % d'efficacité énergétique et 28 % d'utilité de la QoS. Cependant, l'aspect mobilité n'est pas pris en compte dans cet algorithme. Pour remédier à problème, la deuxième contribution propose l'algorithme LS-SCA (Learning-based Swarm optimization-aware Service Composition Algorithm), qui prend en compte conjointement la mobilité, l'énergie et la QoS pendant le processus de composition. En utilisant un modèle de mobilité réaliste et une optimisation par essaim d'apprentissage en deux phases, l'algorithme LS-SCA réduit le temps de calcul tout en améliorant la qualité de la composition. Les résultats montrent une utilité de QoS supérieure de 23 %, une consommation d'énergie inférieure de 28 % et une disponibilité des services supérieure de 40 % par rapport aux approches de la littérature. Cette thèse propose des algorithmes adaptatifs, efficaces sur le plan énergétique et sensibles à la mobilité pour une composition de services évolutive et fiable dans des environnements IoT et CPSS dynamiques.
Mots-clés : Composition de services, qualité de service (QoS), efficacité énergétique, mobilité, Internet des objets (IoT), systèmes cyber-physiques-sociaux (CPSS), métaheuristiques.

الملخص. تعالج هذه الأطروحة مشكلة تركيب الخدمات المراعية لجودة الخدمة في الأنظمة السيبرانية-الفيزيائية-الاجتماعية (CPSS) واسعة النطاق وإنترنت الأشياء IoT. تتميز هذه البيئات بدرجة عالية من الديناميكية والاتصال المتبادل، مما يجعل ضمان جودة الخدمة (QoS) المثلى، والاستهلاك المتوازن للطاقة، والتأقلم مع حركية المستخدمين تحديًا استثنائيًا. غالبًا ما تفشل المناهج الحالية في مراعاة هذه الأبعاد في آن واحد، مما يؤدي إلى انخفاض توافر الخدمة ومحدودية جودة التركيب. للتغلب على هذه العوائق، تقترح المساهمة الأولى خوارزمية التركيب القائمة على التدريس الجماعي والموفرة للطاقة والمراعية لجودة الخدمة (GT-EQCA) الموجهة لبيئات إنترنت الأشياء. من خلال صياغة المشكلة على هيئة تحسين توافقي متعدد الأهداف واختيار الخدمات الأكثر ملائمة، فإن خوارزمية GT-EQCA تعمل على تقليل وقت الحساب مع الحفاظ على مستوى عالٍ من جودة الخدمة. معتمدة على خوارزمية تحسين التدريس الجماعي (GTO)، التي تتجنب ضبط الإعدادات الصعبة وتؤمن قابلية التوسع. تُظهر النتائج التجريبية تحسينات تصل إلى 76% من حيث وقت التركيب، 88% من حيث كفاءة الطاقة، و 28% من حيث منفعة جودة الخدمة. لكن هذه الخوارزمية لا تتطرق لعامل الحركية. لمعالجة هذا العائق، تقترح المساهمة الثانية خوارزمية تركيب الخدمات القائمة على التعلم والمراعية للتحسين الجماعي (LS-SCA) من خلال النظر بشكل مشترك في عوامل الحركة والطاقة وجودة الخدمة أثناء عملية التركيب. باستخدام نموذج تنقل واقعي وخوارزمية تحسين السرب القائمة على التعلم من مرحلتين، فإن خوارزمية LS-SCA تقلل من وقت الحساب مع تحسين جودة التكوين. تظهر النتائج زيادة بنسبة 23% من فائدة جودة الخدمة، وانخفاض بنسبة 28% في استهلاك الطاقة، وزيادة بنسبة 40% في توفر الخدمة مقارنة بالخوارزميات المرجعية الموجودة في الأدبيات. تقترح هذه الأطروحة خوارزميات قابلة للتكيف وموفرة للطاقة ومراعية للحركة من أجل تركيب خدمات قابلة للتوسع وموثوقة في بيئات إنترنت الأشياء وبيئات CPSS الديناميكية.
الكلمات المفتاحية: تكوين الخدمة، جودة الخدمة (QoS)، كفاءة الطاقة، الحركية، إنترنت الأشياء (IoT)، الأنظمة السيبرانية-الفيزيائية-الاجتماعية (CPSS)، الخوارزميات فوق الإرشادية.