Democratic and Popular Republic of Algeria
Ministry of Higher Education and Scientific Research
University A.MIRA-BEJAIA

Faculty of Exact Sciences
Department of Computer Science
Affiliated Research Laboratory or Unit LIMED

THESIS

TO OBTAIN THE DIPLOMA OF

PHD

Domain: Mathematics and Computer Science Branch: Computer science
Specialty: Data science

Presented by
CHERIFI Asma

Topic

A parallel perspective for user-centered and QoS-aware service composition in the Internet of Things

Defended on: 14 January 2026 **In front of the Jury composed of:**

First and last name	**Degree**		
Mr. Mohand YAZID	Prof.	Univ. of Bejaia, Algeria	President
Mr. Achour ACHROUFENE	MCA	Univ. of Bejaia, Algeria	Examiner
Ms. Frédérique BIENNIER	Prof.	INSA Lyon, France	Examiner
Mr. Zoubeyr FARAH	MCA	Univ. of Bejaia, Algeria	Reporter
Mr. Mohamed Essaid KHANOUCHE	MC	Univ. Claude Bernard Lyon1, France	Co-Reporter

Academic year: 2025/2026

Acknowledgement

Praise and gratitude to Allah, for granting me patience and guidance necessary to complete this thesis. First, I am deeply thankful to my supervisor, Mr. Mohamed Essaid KHANOUCHE, for it invaluable guidance, encouragement, his permanent availability, his advice and the fact that he has always followed this work with great interest.

I would also like to thank my thesis co-supervisor Mr. Zoubeyr FARAH for his encouragement and sympathetic support.

I would also like to thank the members of the jury, for agreeing to assess my research work.

Finally, I dedicate this thesis to my family, and in particular to my parents and my husband, for their patience and confidence in me. Your constant encouragement has given me the strength to go all the way.

Abstract

The rapid growth of the Internet of Things (IoT) has led to the proliferation of services, making Quality of Service (QoS)-aware service composition a critical challenge. This thesis addresses this issue through three complementary contributions. The first contribution presents a systematic literature review of QoS-aware service composition approaches, introducing a two-layered taxonomy that distinguishes between plan-based and autonomous approaches. This review identifies key limitations in the state-of-the-art, such as the lack of semantic matching consideration, the assumption of a prior existence of an abstract composition plan, limited scalability, and the use of fixed population sizes. These findings highlight the need for more efficient and adaptive approaches. To overcome these issues, the second contribution proposes the parallel differential evolution-based approach with population size reduction for QoS-aware services composition (PDE-QSC). By evolving two parallel subpopulations with distinct strategies and adaptively reducing population size, the PDE-QSC approach improves the composition quality and computation time compared to five baseline approaches. However, this approach still relies on the existence of an abstract plan and does not account for the semantic matching aspect. The third contribution addresses these remaining limitations by introducing two database concepts-driven approaches for autonomous QoS-aware semantic service composition (HCFDSSC and ESFDSSC). The proposed approaches leverage functional dependency theory to ensure semantic feasibility, reduce the search space, achieve fault tolerance in the case of service failures, and generate high-quality compositions. This thesis advances QoS-aware service composition in IoT by linking plan-based optimization and autonomous semantic approaches, opening new perspectives for scalable, adaptive, and resilient service systems.

Keywords: Quality of Service (QoS), Service selection, Quality of Semantic Matching (QoSM), Autonomous service composition, Multi-population Differential Evolution, Population size reduction, Functional dependencies, Semantic services composition.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Context and motivation

Contents

1.1 Problem statement

The Internet of Things (IoT) is a network of interconnected smart devices with sensing, processing, and communication capabilities, offering diverse services in a wide range of application domains (e.g., smart homes, healthcare, smart cities, and industrial automation) [1, 2]. The continuous and rapid growth of the number of IoT devices results in a large number of IoT services. The latter are self-descriptive, open-access, reusable, and fully independent elements that represent the functionalities of IoT devices [3]. In this context, the Object as a Service (OaaS) paradigm is raised, where IoT devices (objects) are abstracted as on-demand and modular services. Practically, IoT services can be accessed, invoked, and seamlessly composed within the OaaS paradigm to create more sophisticated functionalities [4]. Figure 1.1 shows the architecture of an OaaS [5]. In the object layer, physical IoT devices or software components deliver data using their functionalities such as sensing, actuating, and computing (e.g., a smart bulb device). In the abstraction layer, each object is represented as a service, and standardized interfaces are provided, enabling interaction with the objects (e.g., a smart bulb service). In the service layer, the functionality of the needed service is described. This service can discover and interact with objects via the abstraction layer (e.g., energy monitoring service). In the user layer, the user accesses the needed service through the user's interfaces (e.g., a mobile application).

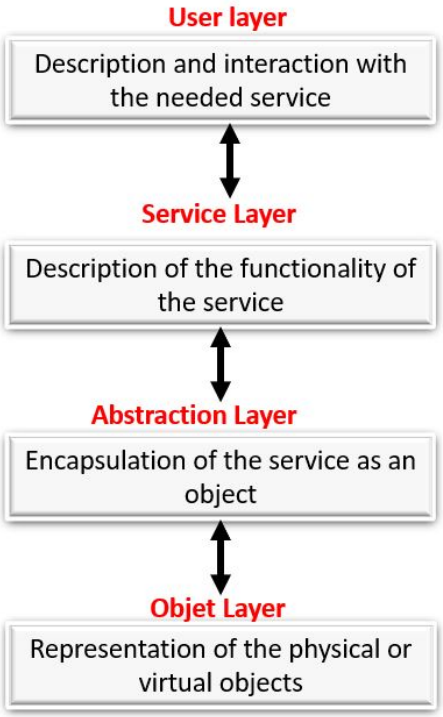


Figure 1.1: Architecture of an Object as a Service (OaaS)

.

A service is characterized by functional properties linked to *semantic features* and non-functional properties related generally to *quality of service* (QoS). The functional properties describe what a service can actually do, which consist of input data required to perform the service functionality and output data resulting from the service execution [6]. Figure 1.2 illustrates the functional properties of a *smart thermostat* service in an innovative office ecosystem [7]. This service receives the employee's office number, favorite temperature settings, and scheduled times as inputs to provide the number of air conditioners to switch on, along with their respective temperature settings as outputs.

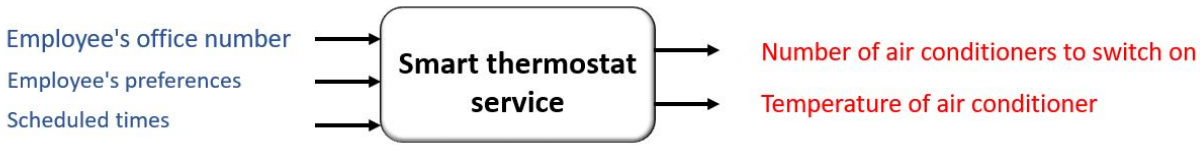


Figure 1.2: Functional properties of a smart thermostat service.

The QoS aspect can be seen as the ability of a service to match specific user's constraints and can be used to differentiate services offering similar functionalities. Table 1.1 defines the most commonly used QoS properties in the literature, including availability, price, response time, reliability, throughput, and reputation [8]. The QoS properties are either published by service providers (e.g., price), collected from historical service performance (e.g., response

time), or captured from user's feedback (e.g., reputation) [9]. A QoS property can be positive or negative. A positive property (e.g., availability) has a positive impact on QoS, i.e., the more the value of the property increases, the more the QoS increases. Conversely, a negative property has a negative influence on QoS, i.e., the more the value of the property increases, the more QoS decreases [10]. The value of a positive QoS property needs to be maximized, whereas the value of a negative QoS property needs to be minimized [11]. In most practical cases, a service providing a basic functionality, known as a basic service, may not meet the user's functional requirements. It is then necessary to combine several services to create a *composite service* with an added-value functionality [12]. For instance, humidity and temperature sensing services can be combined to create a weather monitoring composite service [13].

Table 1.1: The most commonly used QoS properties in the literature.

QoS property	**Definition**
Availability	The probability of a service being accessible, calculated as the ratio of successful invocations over the total number of invocations
Price	The amount that the user must pay to a service provider for the execution of its offering service
Response time	The estimated elapsed time between submitting a request and the delivery of results, in seconds or milliseconds
Reliability	The probability that a service performs its intended functionality effectively without failure within the specified response time
Throughput	The total number of completed invocations of a service over a given period of time
Reputation	The collective users' feedback on service credibility and performance

This thesis deals with the QoS-aware service composition issue. The nature of this problem depends on how the user's functional requirements are specified, leading to two main approaches: plan-based QoS-aware service composition and autonomous QoS-aware service composition. The user's functional requirements can be represented as an *abstract composition plan* where a set of functionalities called *service classes* (also known as tasks [14] or abstract services [15]) are orchestrated to fulfill the user's requirements. This abstract composition plan can be constructed either manually or automatically, and each service class within the plan can be performed by multiple functionally equivalent *concrete services* with different QoS property values. Consequently, selecting the most appropriate concrete service for each service class of the abstract composition plan, to create a composite service satisfying the user's requirements and optimizing multi-conflicting QoS properties, is a core research topic known as *plan-based QoS-aware service composition*. To assess the overall QoS of a composite service, the QoS properties of its constituent concrete services are aggregated based on the composition's structure. A composite service can be designed according to four

composition structures: sequential, loop, conditional, and parallel (see Figure 1.3) [16].

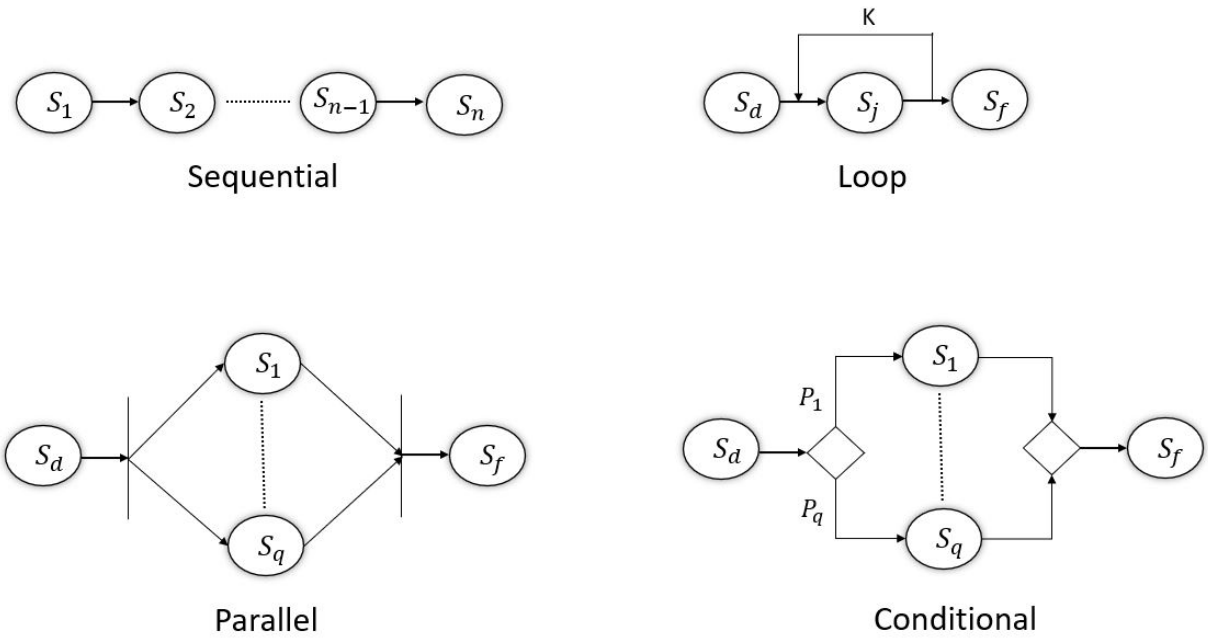


Figure 1.3: Typical composition structures.

In a sequential structure, a succession of services $\{S_1, ..., S_n\}$ are executed one after another in a predefined order. In a loop structure, a single service S_j is executed k times. In a parallel structure, several services $\{S_1, ..., S_q\}$ are executed simultaneously. In a conditional structure, only one service S_i $(1 \leq i \leq q)$ is selected for execution with probability p_i among a set of q services. According to the composition structure and the nature of the QoS property, the aggregation function applied to compute the overall QoS of the composite service could be summation $\sum$, product $\prod$, minimum min, maximum max or average avg (see Table 1.2) [8, 17, 16]. Note that m is the number of concrete services belonging to the composition, whereas qos_A, qos_P, qos_{Rt}, qos_R, qos_{Th} and qos_{Rp} represents the QoS values for the availability, price, response time, reliability, throughput and reputation criteria, respectively.

A service composition must not only optimize the overall QoS but also comply with the user's global QoS constraints that define the upper and lower bounds for the acceptable overall QoS values of the composition. The service composition problem is an NP-hard challenge since the computation time required to obtain the optimal composition in terms of QoS increases exponentially with the number of services, especially in large-scale IoT service environments [18]. More precisely, the optimal composition in terms of QoS that satisfies the user's global QoS requirements can be determined by evaluating all possible combinations of concrete services. Unfortunately, the composition search space grows exponentially with the number of services, resulting in an excessive computation time. Figure 1.4 illustrates this

Table 1.2: The aggregation functions of QoS properties.

	Aggregation function			
QoS property	**Sequential**	**Loop**	**Parallel**	**Conditional**
Availability	$\prod_{i=1}^{m} qos_A(S_i)$	$\prod_{i=1}^{m} qos_A(S_i)$	$max_{i=1}^{m} qos_A(S_i)$	$\sum_{i=1}^{m}(qos_A(S_i)*p_i)$
Price	$\sum_{i=1}^{m} qos_P(S_i)$	$K*\sum_{i=1}^{m} qos_P(S_i)$	$\sum_{i=1}^{m} qos_P(S_i)$	$\sum_{i=1}^{m}(qos_P(S_i)*p_i)$
Response time	$\sum_{i=1}^{m} qos_{Rt}(S_i)$	$K*\sum_{i=1}^{m} qos_{Rt}(S_i)$	$max_{i=1}^{m} qos_{Rt}(S_i)$	$\sum_{i=1}^{m}(qos_{Rt}(S_i) * p_i)$
Reliability	$\prod_{i=1}^{m} qos_R(S_i)$	$\prod_{i=1}^{m} qos_R(S_i)$	$max_{i=1}^{m} qos_R(S_i)$	$\sum_{i=1}^{m}(qos_R(S_i)*p_i)$
Throughput	$\min_{i=1}^{m} qos_{Th}(S_i)$	$\min_{i=1}^{m} qos_{Th}(S_i)$	$\min_{i=1}^{m} qos_{Th}(S_i)$	$qos_{Th}(S_i)$
Reputation	$avg_{i=1}^{m} qos_{Rp}(S_i)$	$avg_{i=1}^{m} qos_{Rp}(S_i)$	$avg_{i=1}^{m} qos_{Rp}(S_i)$	$\sum_{i=1}^{m}(qos_R(S_i)*p_i)$

problem using an abstract composition plan of m service classes, each one is performed by n concrete services. Solving this problem consists of determining the composite service with the highest QoS among all the n^m possible compositions. Thus, a practical and promising approach is to find suboptimal compositions in terms of QoS, while taking a reasonable composition time by reducing the composition search space.

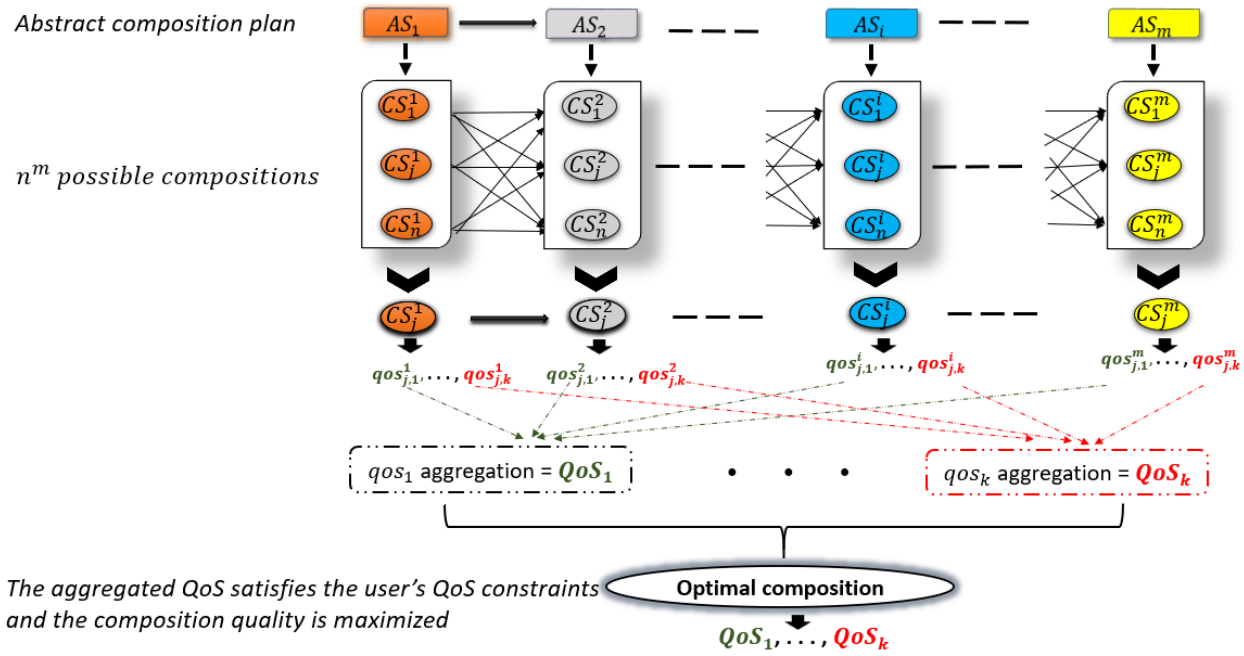


Figure 1.4: QoS-aware service composition problem.

Due to the high increase in the number of IoT concrete services, the manual design of an abstract composition plan has become complex and practically infeasible in several scenarios. Unlike the plan-based QoS-aware service composition problem where a predefined abstract composition plan is provided [19, 20, 21, 22], the user's functional requirements in

autonomous QoS-aware service composition are specified as a set of provided inputs and desired outputs. The composition process ensures functional correctness by connecting concrete services within a composition through the matching of their input and output parameters, even when these parameters do not match perfectly at the syntactic level [23]. To address such mismatches, an ontology-based semantic description is used to perform semantic matching between the input and output parameters of the concrete services belonging to the composition [24]. An ontology consists of a shared vocabulary that describes the input and output concepts of services and defines a relationship that could links these concepts. The Web Ontology Language for Services (OWL-S) [25], and Semantic Annotations for Web Services Description Language (SAWSDL) [26] are among the most widely adopted standard description languages. To ensure functional correctness in service composition, the semantic matching is usually assessed using the *quality of semantic matching (QoSM)* metric [27]. This raises the *autonomous QoS-aware service composition* issue, where the aim is to satisfy the user's requirements by finding the suboptimal composition that optimizes the QoSM and the QoS simultaneously [28]. The autonomous QoS-aware service composition involves the automatic selection and orchestration of a sequence of concrete services from a service repository such that the inputs of each service semantically match the outputs of the preceding one. The resulting service composition is expected to produce the output set required by the user's request from the input set while optimizing QoS criteria and providing a high level QoSM [29]. A service repository that hosts a wide variety of concrete services, each offering a specific functionality, plays a central role in this process and is described by a set of QoS properties, which can be combined to deliver more sophisticated functionalities.

A wide range of research work [29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70] have been conducted in the literature to deal with the NP-hardness of the service composition problem by searching for the composition that best meet the user's constraints in terms of QoS. Moreover, these works integrate the semantic into the process of finding the suboptimal composition that optimizes both QoS and QoSM metrics. Based on their reliance on an existing abstract composition plan, research work on QoS-aware service composition can be classified into two main categories: *plan-based approaches* and *autonomous approaches*. Plan-based QoS-aware service composition approaches assume assume that a predefined abstract composition plan is available to guide the selection of concrete services. However, this assumption often fails to reflect the real-world service composition scenarios especially in dynamic and large-scale IoT environments where the manual construction of such a plan is highly complex and impractical given the substantial number of concrete services. Furthermore, this assumption overlooks the consideration of semantic information that plays a crucial role in describing, understanding, and composing services which can not be

achieved through a syntactic matching. In contrast, the autonomous QoS-aware service composition approaches do not rely on any predefined abstract composition plan, which makes the service composition problem significantly more complex to solve but more adaptable in real-world service composition scenarios. Additionally, according to their exploration method of the search space, the plan-based QoS-aware service composition approaches can be classified into: *sequential exploration-based approaches* and *parallel exploration-based approaches*. Similarly, based on the methodology used to identify semantically correct compositions, the autonomous QoS-aware service composition approaches are divided into: *all matching-based approaches* and *random matching-based approaches*.

1.2 Motivations

In this thesis, we focus on addressing the plan-based and autonomous QoS-aware service composition problems in IoT environments by proposing two approaches that belong to different categories. The first approach belongs to the plan-based QoS-aware service composition category, whereas the second one falls under the category of autonomous QoS-aware service composition. The objective is to find effective methods to address the plan-based and autonomous QoS-aware service composition issues. In the plan-based QoS-aware service composition category, the existence of functionally equivalent concrete services with different QoS properties raises the issue of selecting the most appropriate services to build the suboptimal composition that best satisfies the user's global QoS constraints, while minimizing the composition time. In the autonomous QoS-aware service composition category, the challenge lies in finding the optimal combination of concrete services that best aligns with the user's functional requirements when no predefined abstract service is provided while minimizing QoS and QoSM. The research conducted in the context of this thesis was driven by the three motivations given hereafter.

The QoS-aware service composition has been widely studied in the literature. Most existing surveys on QoS-aware service composition [8, 71, 72, 73, 74, 75, 76, 77, 78] focus exclusively on one category, either plan-based or autonomous approaches, rather than providing a holistic view of both categories. In contrast, the works that review both categories [79, 80, 81] emphasize specific technical aspects, such as composition representations or QoS utility functions, without offering an integrated perspective that captures the broader methodological and semantic challenges involved in QoS-aware service composition. For example, the literature review in [73] proposed a classification of plan-based QoS-aware service composition approaches accounting for the method employed to address the QoS uncertainty. A survey on autonomous QoS-aware service composition approaches is introduced in [79] where the

studied approaches are categorized according to research concerns, including service description and service matching. A survey on the computational intelligence-based optimization methods used in plan-based and autonomous QoS-aware service composition approaches is proposed in [81]. The reviewed approaches are compared considering technical aspects of the used computational intelligence-based optimization methods such as the composition representation, the used utility function, and the employed operators. In [80], the plan-based and autonomous QoS-aware service composition approaches were analyzed according to technical aspects such as functional correctness, the solving algorithm, and target platforms. Unlike existing literature reviews, it is essential to conduct a comprehensive and systematic review of the most relevant plan-based and autonomous QoS-aware service composition approaches proposed in the last decade from various critical perspectives. Such a review not only bridges the knowledge gap in understanding these approaches but also serves as a foundation for researchers to develop robust and more efficient QoS-aware service composition approaches.

The rapid growth of IoT devices offering functionally equivalent services with different QoS levels has made the QoS-aware service composition issue one of the major challenges in the service computing area. This issue involves finding the optimal composition that best meets the user's functional requirements while optimizing multiple conflicting QoS properties. This challenge is NP-hard and requires exponential time complexity for an exact resolution [18]. Computational intelligence-based methods have been extensively utilized to solve the plan-based QoS-aware service composition problem since they allow finding a suboptimal composition in a reasonable computation time. Depending on the strategy used to explore the composition search space, computational intelligence-based service composition approaches can be classified into two categories: sequential exploration-based approaches [30, 31, 32, 33, 34, 35, 36, 37, 38, 40, 41, 42, 43, 44, 45, 46, 47] and parallel exploration-based approaches [48, 49, 50, 51, 52, 53, 54, 55]. In the first category, a single composition population with a fixed size is used to explore the search space through a sequential evolution process, resulting in (*i*) a low convergence speed that increases the composition time and/or (*ii*) a local optimum resulting in a low or non-optimal composition quality. The composition time represents the duration required to find the suboptimal composition, whereas the composition quality represents the QoS utility value of the composition. In the second category, the search space is explored using a parallel evolution process by dividing a composition population of fixed size into multiple composition sub-populations. Parallel exploration-based approaches are employed to preserve population diversity, enhance convergence speed (i.e., reduce composition time) and/or avoid local optima (i.e., achieve high composition quality) of sequential exploration-based approaches. However, some parallel approaches [52] lack the interaction between the composition sub-populations, which can lead to a decrease in the composition quality. Despite the high efficiency of parallel exploration-based optimization

methods in dealing with high-dimensional problems, their use in service composition is still in early stage and should be deeply investigated to improve the QoS-aware composition process. Furthermore, the performance of computational intelligence-based service composition approaches [30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55] in terms of composition time and/or composition quality may decrease when exploring a population of fixed size. The population size has a crucial impact on the performance of these optimization methods in terms of computation time and solution quality [82, 83]. A small population size accelerates the convergence to the suboptimal solution and therefore reduces the computation time but limits the search space exploration, which may decrease the solution quality. Conversely, a large population size allows for a deep exploration of the search space and thus improves the quality of the solution, but may increase at the same time the computation time due to the exploration of many unpromising search sub-spaces. From this perspective, several population resizing strategies have been proposed in the literature to determine the appropriate population size for computational intelligence-based optimization methods, with the objective of achieving a good trade-off between computation time and solution quality [84, 85, 86, 87, 88, 89, 90, 91]. Computational intelligence-based methods with a population resizing strategy have been proven to be efficient in finding a suboptimal solution in a reasonable computation time. However, to the best of our knowledge, these methods have not been employed in the context of service composition and should be investigated more and more to address the QoS-aware service composition issue. To address the challenges of high composition time and/or low composition quality, it is essential to design a scalable and a parallel exploration-based service composition algorithms capable to deliver a suboptimal composition with high QoS while significantly reducing the time required for the composition process.

To address the limitations of plan-based QoS-aware service composition approaches, in particular their lack of semantic consideration, it is essential to overcome the assumption that an abstract composition plan must already exist. Removing this assumption enables the composition process to better reflect real-world scenarios, where the user's request must be handled despite the diversity of concrete services and the dynamic nature of IoT environments. In particular, this allows the creation of compositions that ensure semantic matching among the selected concrete services while simultaneously optimizing QoS and QoSM metrics. As a result, the service composition approach aligns more accurately with the user's requirements. Several works have been devoted to address the autonomous QoS-aware service composition issue using two categories of approaches: all matching-based approaches and random matching-based ones. To find any functionally executable composition, all matching-based approaches generate all possible matching of concrete services belonging to the service repository [29, 56, 57, 58, 59]. An exact method is then employed to find the optimal compo-

sition considering QoS properties. However, these approaches often deal with only one QoS criterion, and generating all possible matching of services increases the composition time exponentially. Random matching-based approaches solve the autonomous QoS-aware service composition using computational intelligence-based algorithms where the initial population and the evolution process are based on randomness [60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70]. This category of approaches repeatedly uses a decoding algorithm [62] to ensure the semantic feasibility of compositions, which may increase the composition time and decrease the composition quality. In addition, most of these computational intelligence-based algorithms do not optimize QoS and QoSM metrics simultaneously [29, 56, 57, 58, 59, 60, 61, 62, 64, 66, 67, 68, 69, 70]. Furthermore, the stochastic nature of red IoT environments can make the compositions obtained with the two classes of approaches no longer executable, requiring a re-composition process to find new suboptimal composition in the case of service failure. Therefore, it is important to design an autonomous service composition approach that overcomes these shortcomings and returns a feasible composition with optimized QoS and QoSM metrics in a reasonable composition time.

1.3 Objectives and contributions

The objective of this thesis is to develop novel and efficient approaches to solve the QoS-aware service composition problem involving two main aspects. In the context of plan-based QoS-aware service composition, the aim is to select the most appropriate concrete service in each abstract service so as to find the best composition in terms of QoS that satisfies the user's global QoS constraints. In the case of autonomous QoS-aware service composition problem, the objective is to find the suboptimal combination of concrete services from a service repository that satisfies the user's functional requirement while simultaneously optimizing the QoS and QoSM criteria. The proposed approaches are designed to overcome the specific limitations highlighted in section 1.2.

The first objective of this thesis is to understand the problem of QoS-aware service composition by critically analyzing existing studies in this field. The goal is to guide researchers towards innovative resolution methods that focus on strengths and overcome the limitations identified in previous work. Unlike existing surveys on service composition [8, 71, 72, 73, 79, 80, 81, 74, 75, 76, 77, 78] that focus either on plan-based or autonomous QoS-aware service composition approaches, or on specific technical aspects such as service composition representations and QoS utility functions, the first contribution of this thesis [92] is to conduct a comprehensive review that covers both categories, with a particular emphasis on their resolution strategies and validation methodologies. To guide this review, the following five research questions (RQ) are addressed: $RQ1-$ What are the primary classification criteria

for QoS-aware service composition approaches? $RQ2-$ Can secondary criteria refine the initial classification categories? $RQ3-$ What methods are usually employed to solve the QoS-aware service composition problem? $RQ4-$ What performance metrics are commonly used to evaluate these approaches? $RQ5-$ What datasets are used to evaluate the performance of the investigated approaches? The main objectives of this contribution are outlined as follows: (*i*) a two-layered taxonomy is proposed to classify existing QoS-aware service composition approaches, taking into account the assumption of a prior existence of a composition plan, the search space exploration strategy, and the methodology used to identify functionally feasible compositions; (*ii*) a comparative analysis of the reviewed approaches is introduced by considering the most relevant criteria in service composition area, such as solving method, population size reduction, degree of scalability, utility value combining the QoS and QoSM properties, failure recovery, and automatic generation of an abstract composition plan; (*iii*) an analysis of the resolution and validation methodologies employed by the studied approaches is conducted to answer the five above mentioned research questions; (*iv*) a discussion of the studied approaches is given to identify potential research gaps and future directions in the context of QoS-aware service composition. This review provides the research community with a deeper understanding and pragmatic insights on existing studies in service composition. It identifies key challenges and research directions, particularly the investigation of parallel search methods for solving the plan-based QoS-aware service composition problem, improving the service failure handling, and considering QoS and QoSM metrics in dealing with the autonomous QoS-aware service composition issue.

The second objective of this thesis is to solve the plan-based QoS-aware service composition problem using parallel-exploration based method. The importance of addressing such a problem lies in ensuring scalable, efficient and high QoS user-centered approach that can meet complex user's requirements and faces the dynamic nature of IoT environments. In the second contribution of this thesis, we propose a Parallel Differential Evolution-based approach with population size reduction for QoS-aware service composition (PDE-QSC) [93]. Most of existing plan-based QoS-aware service composition approaches [30, 31, 32, 33, 34, 35, 36, 37, 38, 40, 41, 42, 43, 44, 45, 46, 47] rely on a sequential exploration of the composition search space using a population with a fixed size, resulting in a high composition time and/or a limited composition quality. To address these limitations, the proposed PDE-QSC approach uses a parallel exploration strategy of the composition search space where the composition population is divided into two composition sub-populations with smaller size. Compared to existing approaches based on sequential exploration, this parallel exploration-based strategy reduces the composition time while ensuring a high composition quality. The composition sub-populations are then combined to form a single population enabling interaction between sub-populations and, thus increasing the population diversity, resulting

in a much higher composition quality than evolutionary-based approaches. In addition, a population size reduction strategy is introduced to continuously decrease the size of the composition population by removing unpromising compositions in terms of QoS. Unlike existing computational intelligence-based approaches [30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55] that use a fixed population size, the second contribution of this thesis allows decreasing the composition time and improving the composition quality. Several simulation scenarios are considered to evaluate the performance of the PDE-QSC approach and show its superiority over five baseline approaches in terms of composition quality and composition time.

The third objective of this thesis is to develop an approach that overcomes the limitations of the first contribution, specifically by eliminating the assumption of prior-existence of an abstract composition plan composition plan and by incorporating semantic information within the service composition process. The third contribution of this thesis consists of two functional dependencies-based failure recovery approaches for autonomous QoS-aware service composition (HCFDSSC and ESFDSSC) [94]. A key innovation of these algorithms lies in using functional dependencies, a foundational concept from relational database theory that to the best of our knowledge has not been exploited in this context. While some semantic service composition approaches [95, 96] have superficially drawn on the concept of functional dependencies to model input-output relations of ontology concepts to perform service discovery or minimize the number of services in a composition plan, our work explicitly adopts functional dependencies in the database-theoretic sense to model services themselves, capture intrinsic semantic constraints, and ensure semantic correctness of the composition. More precisely, our algorithms do so while simultaneously integrating QoS considerations, allowing us to generate abstract service composition plans that are semantically valid and optimized concerning non-functional requirements. This dual focus on semantic consistency and QoS optimization, grounded in formal database principles, introduces a novel paradigm that clearly distinguishes our work from all existing methods. The two proposed approaches belong to the class of all matching-based approaches. Unlike the existing approaches [29, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 69, 70] that operate on the overall service repository, the HCFDSSC and ESFDSSC approaches reduce the problem's complexity by scaling down the service repository to finer levels by grouping the concrete services having the same inputs and outputs to form a set of abstract services. After that, the potential matching between the resulting abstract services is generated using the attribute closure algorithm [97, 98]. The latter ensures the semantic matching by finding a set of feasible (i.e., functionally correct) abstract composition plans such that the execution of each plan can produce the set of outputs specified by the user's request. Finally, two different methods are employed to perform the parallel service selection and composition process. The first method uses a parallel improved

hill climbing (HC) algorithm [99], while the second employs a parallel exhaustive search (ES) algorithm [100], resulting in the two proposed approaches: HCFDSSC and ESFDSSC. These methods are carried out on the resulting feasible abstract composition plans to find multiple suboptimal compositions that are compared to obtain the best composition in terms of QoS and QoSM. In contrast to most matching-based approaches [29, 56, 58, 59] and random matching-based approaches [60, 61, 62, 63, 64, 69, 70] that do not deal with the composition failure issue, the parallel service selection and composition methods used in HCFDSSC and ESFDSSC approaches allow addressing the service failure problem through replacing the unsuccessful composition by one of the alternative suboptimal compositions.

1.4 Organization of the thesis

The remainder of this thesis is organized as follows:

Chapter 2 develops the first contribution, a comprehensive review of the literature on QoS-aware service composition approaches. First, a novel taxonomy that offers a granular classification of the last decade QoS-aware service composition approaches is introduced based on their reliance on a predefined abstract composition plan. Then, a comparison of the reviewed approaches is provided considering the most relevant criteria in service composition. Finally, an analysis and discussion is conducted based on the resolution and validation methodologies used in the investigated approaches.

Chapter 3 presents the second contribution, the PDE-QSC approach that overcomes the limitations of existing plan-based QoS-aware service composition approaches. The service composition model and the problem statement are defined in the first part of this chapter. The proposed PDE-QSC approach that relies on a parallel exploration method and population resizing strategy is presented in the next part of the chapter. Finally, the performance of the PDE-QSC approach in terms of composition quality and composition time are evaluated and compared with those of five baseline approaches by considering three different scenarios. Furthermore, a statistical analysis is conducted to demonstrate the efficiency of the proposed PDE-QSC approach.

Chapter 4 details the third contribution, the HCFDSSC and ESFDSSC approaches that are proposed to deal with the shortcomings of the existing autonomous QoS-aware service composition approaches. First, the most important concepts related to the autonomous QoS-aware service composition issue are defined. Then, the steps of the HCFDSSC and ESFDSSC approaches are detailed, where each one performs in three phases: the pre-processing, the construction of feasible abstract plans and the service selection and composition. Finally, a

set of experiments is conducted to evaluate the HCFDSSC and ESFDSSC approaches and compare their performance with existing baseline approaches in terms of computational time, QoSM, QoS, and utility value combining both metrics.

Chapter 5 summarizes the findings of this thesis and outlines future research perspectives. We first provide a synthesis of the contributions. Then, we discuss key challenges in service composition, including concurrent QoS-aware service composition, autonomous microservice composition for next-generation systems, blockchain-assisted approaches for reliable IoT environments, and handling uncertain data. To address these challenges, we suggest exploiting multi-request optimization methods, natural language processing, artificial intelligence methods, blockchain technology, and machine learning techniques.

Chapter 2

Literature review

Contents

2.1 Introduction

The QoS-aware service composition has been the subject of several studies in the literature based on various scenarios. In this chapter, we present a literature review of the most significant QoS-aware service composition approaches [92]. The first part describes the research methodology employed to conduct this literature review. The second part proposes a novel two-layered taxonomy that classifies existing QoS-aware service composition approaches based on three key aspects: the assumption of a predefined composition plan, the resolution strategy used to explore the search space, and the methodology for identifying functionally feasible compositions. The third part conducts a comparison of the reviewed approaches considering the most relevant criteria in the service composition area such as solving method, search space exploration strategy, population size reduction, degree of scalability, methodology used to ensure semantic correctness of the composition, utility value combining QoS and QoSM criteria, failure recovery and automatic generation of an abstract composition plan. After that, an analysis of the QoS-aware service composition approaches

is performed based on the employed resolution technique and validation methodology. Finally, a critical discussion of the studied approaches is given to illustrate their limitations and highlight the motivations of the approaches proposed in this thesis.

2.2 Background

A *concrete service* is a software component characterized by a set of input and output concepts that define its functional properties, as well as a vector of QoS attributes that specify its non-functional properties [101]. Several services can be combined to provide an added-value functionality resulting in a *composite service*. A huge number of concrete services, offering similar functionalities but different QoS attributes, are deployed in IoT environments. The challenge in service composition lies in identifying the optimal combination of services to best fulfill the user's requirements. The latter can be expressed in distinct forms resulting in two formulations of the QoS-aware service composition problem: plan-based QoS-aware service composition and autonomous QoS-aware service composition.

2.2.1 Plan-based QoS-aware service composition problem

In plan-based QoS-aware service composition, the user's functional requirements are formalized as an abstract composition plan that orchestrates a set of functionalities known as abstract services. The user's non-functional requirements are expressed as global QoS constraints that specify upper and lower bounds on the aggregated QoS values of the resulting composition. Each functionality of the abstract composition plan can be executed by a set of concrete services sharing the same input and output concepts. In this case, the composition process involves selecting a concrete service from each abstract service to ensure that the aggregated QoS values of the resulting composition meet the global constraints and the composition quality is maximized (see Figure 2.1). The aggregation of the QoS values depends on the nature of the QoS attributes and the structure of the abstract composition plan (see [17] for more details).

2.2.2 Autonomous QoS-aware service composition problem

In autonomous QoS-aware service composition, the user's functional requirements are specified as a set of given inputs and desired outputs, which constitute the basis for the construction of the abstract composition plan. Consequently, the QoS-aware service composition issue becomes more challenging, requiring the identification of the most optimized combination of concrete services from a service repository. This selection must not only ensure a high QoS level but also check the semantic matching between the inputs and outputs of the constituent services, i.e., providing functionally correct composition (see Figure 2.2). The

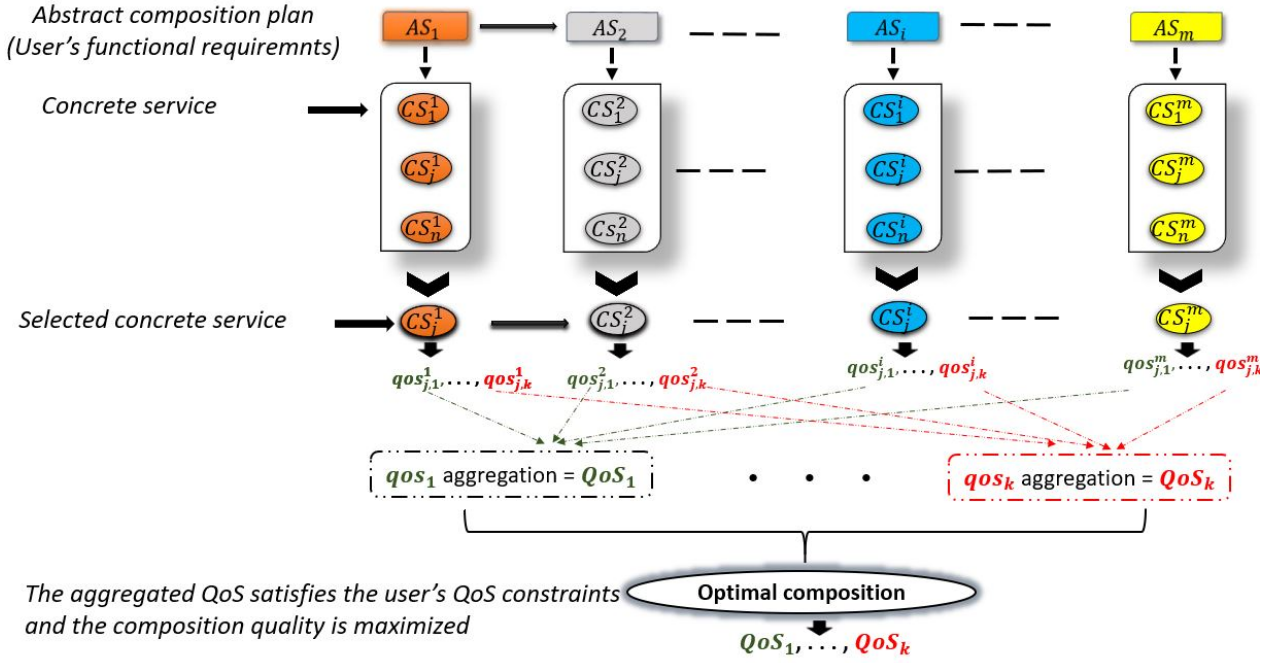


Figure 2.1: Plan-based QoS-aware service composition.

semantic matching of concrete services and composite services is evaluated using the QoSM metric, which is determined by the degree of semantic similarity between input and output concepts based on a given ontology that describes the semantic relationships between these concepts [65].

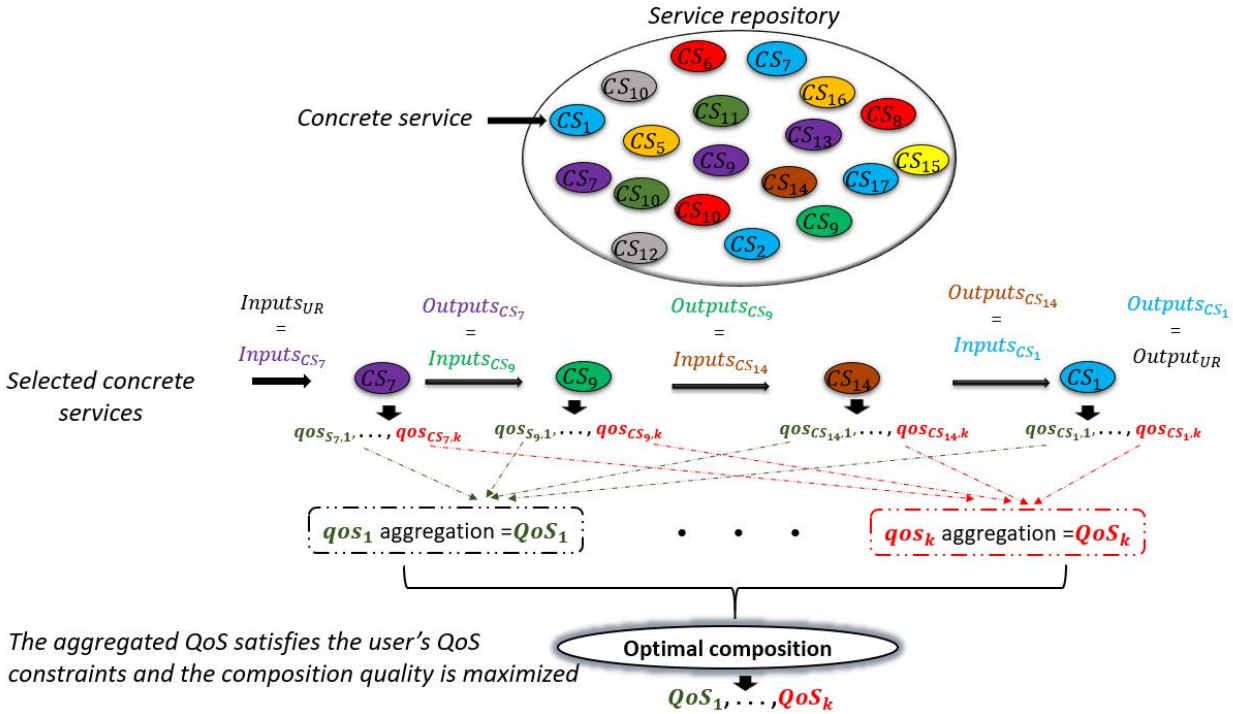


Figure 2.2: Autonomous QoS-aware service composition.

2.3 Research methodology

The first contribution of this thesis seeks to review the QoS-aware service composition approaches that have appeared in the last decade [92]. The initial stage involves formulating research questions to outline the scope and objectives of this review. Subsequently, a search query is constructed, incorporating rigorous inclusion and exclusion criteria to identify relevant papers for the analysis.

2.3.1 Research questions

To address the problem of QoS-aware service composition, we answer the five underlying research questions:

- $RQ1$: What are the primary classification criteria for QoS-aware service composition approaches?
  At a primary level, QoS-aware service composition approaches can be categorized based on the assumption of a pre-existing abstract composition plan. These approaches are divided into two main categories: plan-based QoS-aware service composition approaches and autonomous QoS-aware service composition approaches.

- $RQ2$: Can secondary criteria refine the initial classification categories?
  The approaches within each of the above-mentioned categories can be further categorized at a secondary level based on the strategy used to explore the composition search space and the techniques applied to ensure the semantic correctness of compositions.

- $RQ3$: What methods are commonly employed to solve the QoS-aware service composition problem?
  This thesis explores plan-based QoS-aware service composition approaches, with a focus on computational intelligence-based optimization methods, and investigates autonomous QoS-aware service composition approaches that employ computational intelligence optimization algorithms or exact methods.

- $RQ4$: What performance metrics are commonly used to evaluate these approaches?
  Frequently employed metrics for assessing QoS-aware service composition approaches include composition time, QoS utility, and utility value combining QoS and QoSM metrics.

- $RQ5$: What datasets are used to evaluate the performance of the investigated approaches?
  The performance assessment of QoS-aware service composition approaches typically relies on experimental datasets and real-world datasets.

2.3.2 Search query

2.3.2.1 Research selection

To carry out the literature search, we have used keywords specific to the QoS-aware service composition problem, including Internet of Things/Web services, workflow/ plan-based service composition, semantic/automated/fully automated service composition, QoS, QoSM, service computing, service selection, semantic similarity, and ontology. This study investigates 42 recent papers selected from leading editorial journals and prominent international conferences, accessed through major digital libraries and publishing platforms, including IEEE Xplore, ACM Digital Library, ScienceDirect, and SpringerLink (see Figure 2.3).

2.3.2.2 Selection criteria

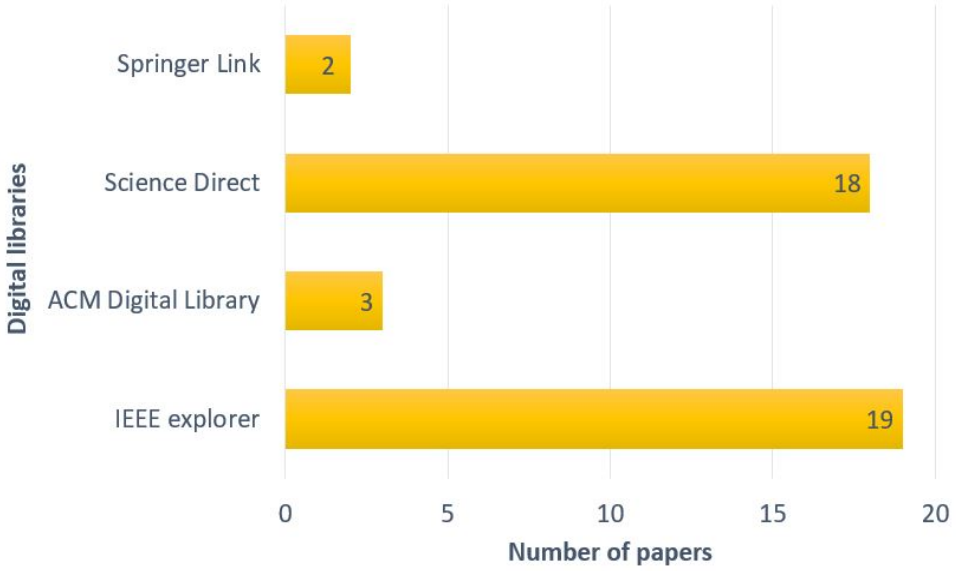
The inclusion criteria that were considered for choosing papers to be analyzed are summarized as follows:

- Papers that solve the plan-based QoS-aware service composition problem using computational intelligence-based optimization algorithms.
- Papers that deal with the QoS-aware service composition in a fully automated way and without a predefined abstract composition plan.
- Papers addressing the QoS-aware service composition problem while considering other relevant challenges such as QoS fluctuation, mobility-aware, data-intensive services, etc.
- Papers published online from 2015 to 2024.

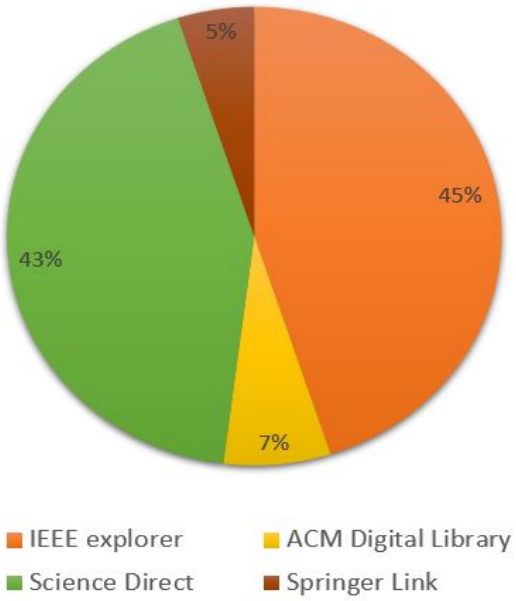
The exclusion criteria that were used to discard papers from this study are presented as follows:

- Papers that deal with the plan-based QoS-aware service composition problem using exact methods, graph-based algorithms, and decomposition-based methods.
- Papers related to the service discovery problem that consists of finding the concrete services having the best semantic matching of input and output concepts.
- Papers dealing only with semantic service composition issue, while neglecting the QoS aspect.

The publication type of the different sub-categories of QoS-aware service composition approaches is presented in Figure 2.4.



(a) Number of investigated papers.



(b) Ratio of investigated papers.

Figure 2.3: The reviewed papers versus digital libraries.

2.4 Taxonomy of QoS-aware service composition

Figure 2.5 shows the two-layered taxonomy proposed in this thesis where QoS-aware service composition approaches are categorized based on their reliance on an abstract composi-

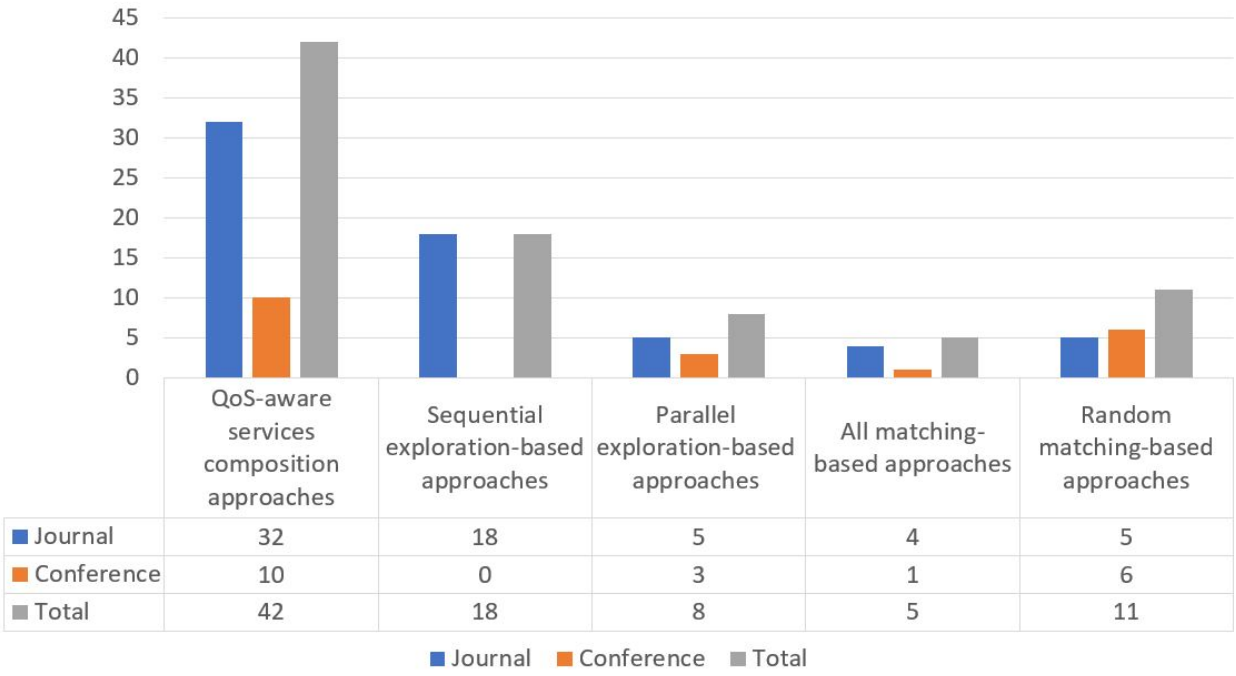


Figure 2.4: QoS-aware service composition approaches versus publication type.

tion plan into: 1) plan-based QoS-aware service composition approaches and 2) autonomous QoS-aware service composition approaches. Plan-based approaches are further divided into two sub-categories according to their composition search strategy: (*i*) sequential exploration-based approaches and (*ii*) parallel exploration-based approaches. Autonomous QoS-aware approaches are divided based on the methodology for identifying the semantic correctness of compositions into two sub-categories: (*i*) all matching-based approaches and (*ii*) random matching-based approaches.

2.4.1 Plan-based QoS-aware service composition approaches

The plan-based QoS-aware service composition approaches assume the existence of an abstract composition plan that aligns with the user's functional request. The primary objective is to maximize the overall QoS utility by selecting the most suitable concrete service from a set of functionally equivalent services for each abstract service, ensuring that the resulting composition best meets the user's global QoS constraints. The two sub-categories of plan-based QoS-aware service composition approaches are described hereafter based on their employed resolution method.

2.4.1.1 Sequential exploration-based approaches

This sub-category of approaches uses mainly computational intelligence-based optimization methods that begin with a randomly generated population of compositions that are

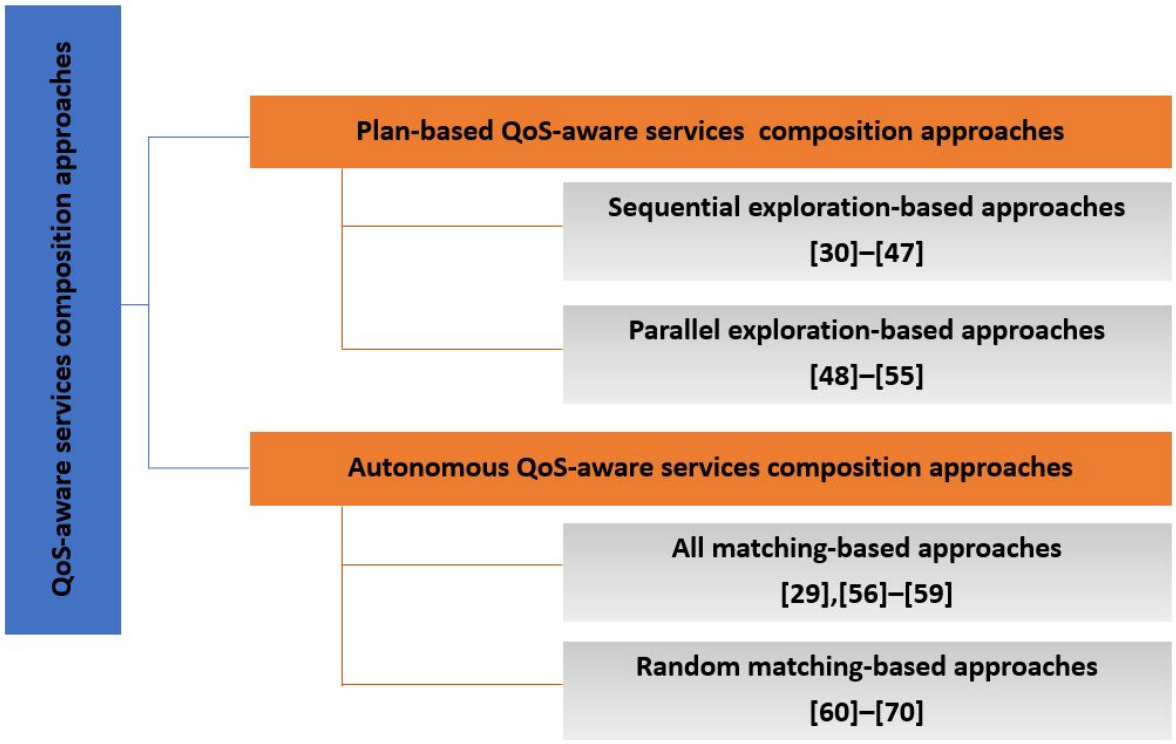


Figure 2.5: Taxonomy of existing QoS-aware service composition approaches.

iteratively improved to achieve a suboptimal composition. Sequential exploration-based approaches include: evolutionary optimization methods, swarm intelligence-based optimization methods, human-inspired optimization methods, hybrid optimization methods, and other computational intelligence-based optimization methods (see Figure 2.6).

Evolutionary optimization methods are population-based methods that mimic biological evolution for solving optimization problems [102]. They evolve based on the theory of natural selection and survival of the fittest [103]. To improve individuals for a given objective, several operators can be applied including selection, mutation, and crossover. Evolutionary optimization methods have been widely used by this sub-category of approaches [36, 39, 47]. For instance, the multi-objective QoS-aware service composition problem is first defined considering the weighted Tchebycheff distance function instead of the weighted sum utility function to enable the evaluation of composition quality [39]. The weighted Tchebycheff distance function is one of the most common scalarization methods employed in multi-objective optimization problems [104], where a smaller value indicates better quality. Subsequently, a fuzzy preference model is introduced, incorporating fuzzy linguistic terms to specify the user's preferences regarding different QoS properties. A new method is then presented to transform the fuzzy preference terms into numeric weights. Finally, two evolutionary algorithms are proposed to solve the QoS-aware service composition problem: a single evolutionary algorithm and a hybrid evolutionary algorithm. In the single evolutionary algorithm, a population

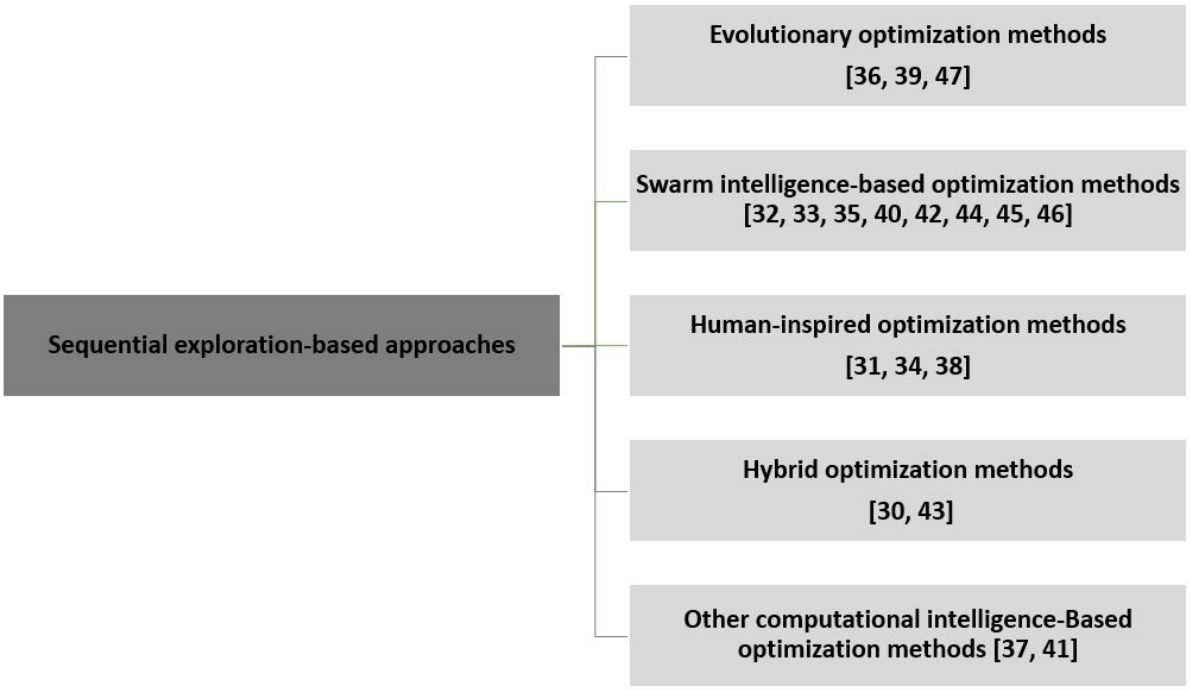


Figure 2.6: Taxonomy of sequential exploration-based service composition approaches.

of randomly generated compositions is maintained as well as their corresponding QoS vectors. Three types of operators are used on the current composition population to produce the next generation of compositions. The compositions having better QoS values based on both weighted Tchebycheff distance and the constraint violation degree are retained for the next iteration. This evolution process is repeated until reaching a predefined number of iterations. In the hybrid evolution algorithm, the same operators used in the single evolution algorithm are used on the current population of compositions to generate the evolved composition population. The current and evolved composition populations are then combined and a Pareto ranking process, called non-dominated sorting strategy [105] is used to preserve the better composition for the next population until reaching the predefined number of iterations.
Although both proposed algorithms are effective in finding high quality compositions, the use of the Pareto ranking strategy increases the composition time as the problem search space increases, making these algorithms not adaptive for the large-scale service composition problem.

Swarm intelligence-based optimization methods are nature-inspired algorithms that imitate the collaborative social behavior exhibited by animals or insects (fish, wolves, bees, etc.) to solve optimization problems [106]. These methods have been largely employed to address the QoS-aware service composition issue [32, 33, 35, 40, 42, 44, 45, 46]. For example, a fuzzy multi-objective artificial bee colony approach for QoS-aware service composition (FMOQWSCP) is proposed in [32]. This approach addresses the uncertainty of concrete services' QoS properties by representing them using a fuzzy model. The FMOQWSCP is carried

out in five phases: 1) *initialization*, 2) *employed bees foraging*, 3) *onlooker bees foraging*, 4) *scout bees foraging* , and 5) *external archive update*. In the initialization phase, the population of compositions is randomly generated where each composition, representing a food source, is initialized as a vector of concrete service indices that are randomly selected from their respective abstract services. Afterwards, the aggregated fuzzy QoS values of each composition are calculated and its corresponding fuzzy QoS utility value is computed according to the defined fuzzy model. An external archive is also initialized with fuzzy non-dominated compositions. The following phases are repeated until the stopping criterion is satisfied. In the employed bees foraging phase, a new composition is generated for each existing composition using a defined neighborhood process. A fuzzy non-dominated sorting procedure is then applied to determine whether the new composition replaces the old one based on fuzzy dominance. In the onlooker bees foraging phase, the binary tournament selection operator [105] is used to choose the compositions to update using the neighborhood search, and they replace the old ones if they are fuzzy dominant. In the scout bees foraging phase, compositions that have not improved after a predefined number of iterations are replaced by new ones, and their aggregated QoS vectors are recalculated. In the external archive update phase, a fuzzy non-dominated sorting procedure is applied to the merged compositions of the current population and the external archive. The Pareto-dominant compositions are then identified and the external archive is updated with respect to its size. The optimal composition is then selected in the external archive using a fuzzy multi-criteria decision-making method.
The FMOQWSCP approach deals with the uncertainty QoS properties by representing them as trapezoidal fuzzy numbers instead of fixed values. However, the composition time exponentially increases with both the number of concrete services and the number of QoS properties making this approach not suitable for large-scale environments.

Human-inspired optimization methods are population-based methods that mimic the learning process of humans in a social context, where their knowledge are enhanced based on experience or observation. Human-inspired optimization methods have been applied by sequential exploration-based approaches [31, 34, 38]. For instance, an improved teaching learning-based QoS-aware service composition approach (ITLQCA) is proposed in [31] where the service composition is formulated as a teaching and learning process. The ITLQCA approach is performed in four phases: 1) *initialization*, 2) *teaching*, 3) *learning*, and 4) *self-learning*. In the initialization phase, the algorithm parameters are chosen such as the population size and the maximum number of iterations. The initial population of compositions is then randomly generated and the QoS utility value of each composition is computed. The teaching phase aims at improving each composition by using the best and the mean compositions in terms of QoS in the population. In the learning phase, each composition is improved according to its neighboring compositions or randomly selected compositions from

the whole population. In the self-learning phase, each composition is improved according to its historical information. The QoS utility value of each modified composition in each phase is compared to the QoS utility value of the earlier composition and the composition having the highest utility value is kept for the next evolution. These three phases are repeated for a given number of iterations to find the near-to-optimal composition in terms of QoS.
The ITLQCA approach is characterized by a few tuning parameters and a high exploration capability of the search space. This allows to find a very close to the optimal composition while keeping a reasonable computation time.

Hybrid optimization methods combine several computational intelligence-based methods to exploit their respective strengths to build stronger methods [107]. Hybrid optimization methods have been used to solve the QoS-aware service composition problem [30, 43]. For instance, a hybrid eagle strategy with a whale optimization approach (ESWOA) is proposed in [30] to address the QoS-aware service composition problem. The ESWOA approach operates in three phases: 1) *initialization*, 2) *exploration*, and 3) *exploitation*. In the initialization phase, the population of whales is randomly generated where each whale represents a feasible composition. The compositions are modeled as an m-dimensional vector containing the indices of their corresponding concrete services. The QoS utility values of compositions are calculated, and the composition having the highest utility value is designed as the best composition (i.e., the leader whale). During the exploration phase, each composition evolves by modifying the indices of its concrete services according to a defined probability. The utility value of the new composition is then calculated and compared to that of the old one to determine whether the new composition can replace the old one. The best composition is also updated if its utility value is lower than that of the new composition. The exploitation phase is performed if a random number R_n generated within the range $[0, 1]$ is greater than a fixed parameter F_p. The compositions are evolved according to two different exploitation mechanisms: the shrinking encircling mechanism and spiral updating mechanism. A random number is used to decide which exploitation mechanism will be used for each composition. In the shrinking encircling mechanism, each composition is updated according to the best composition in the population, whereas in the spiral updating mechanism, each composition is updated following the best composition in an helix shaped path. The QoS utility value of the modified composition is then calculated. The old composition as well as the best composition are updated if the utility value of the new composition is higher than their respective utility values. The exploration and exploitation phases are repeated until a maximum number of iterations is reached. Finally, the best composition in terms of QoS is returned.
This method achieves a good balance between exploration and exploitation of the search space, preventing thus premature convergence. However, the proposed algorithm demonstrated only moderate scalability, which may mean that its performances may decrease in a

large-scale search space.

Other computational intelligence-based optimization methods have been used in the context of service composition [37, 41]. For example, an invasive weed-based approach is proposed in [37]. This approach is carried out in tree main phases: 1) *pre-processing*, 2) *optimization*, and 3) *heuristic search*. In the pre-processing phase, the QoS properties of each concrete service in each abstract service are ranked according to their normalized values. Then, local utility values for these concrete services are computed based on the mean, standard deviation and skewness [108] of the ranks of the QoS properties. These utility values serve as a fundamental indicator to prune concrete services and keep the services having the highest utility values. In the optimization phase, the improved invasive weed algorithm is applied to find the best composition. This algorithm operates in four steps. In the first step, the initial population of compositions is randomly generated. In the second step, each composition is evolved in accordance to the best and worst compositions in the population. In the third step, the compositions resulting from the second step are evolved based on their standard deviations and those of the initial compositions. In the heuristic search phase, the utility values of compositions are calculated based on the mean, standard deviation, and skewness of their associability scores. More specifically, the associability score of a composition represents the number of connections formed between its concrete services delivered by the same provider. The composition having the highest utility value is returned as the best composition. This procedure is repeated until the maximum number of iterations is reached. The filtering process used by this approach guarantees a promising search space of concrete services with high utility in terms of QoS. This contributes to the generation of a suboptimal composition satisfying the user's global constraints. However, the heavy calculations used in the filtering and optimization processes increase the composition time, which limits the scalability of this approach.

2.4.1.2 Parallel exploration-based approaches

The parallel exploration-based approaches decrease the composition time through the simultaneous evolution of composition sub-populations. More precisely, a population of randomly generated compositions is divided into sub-populations, that are iteratively improved in parallel to find the suboptimal composition. As a result, the exploration of a composition population becomes faster since the required time approximates the composition time of sequential exploration-based approaches divided by the number of sub-populations (see Figure 2.7). Parallel exploration-based approaches include: evolutionary optimization methods, swarm intelligence optimization methods, and hybrid optimization methods (see Figure 2.8).

Evolutionary optimization methods have been applied by parallel exploration-based approaches [49, 52, 54]. For example, a multi-population genetic algorithm is introduced in [52]

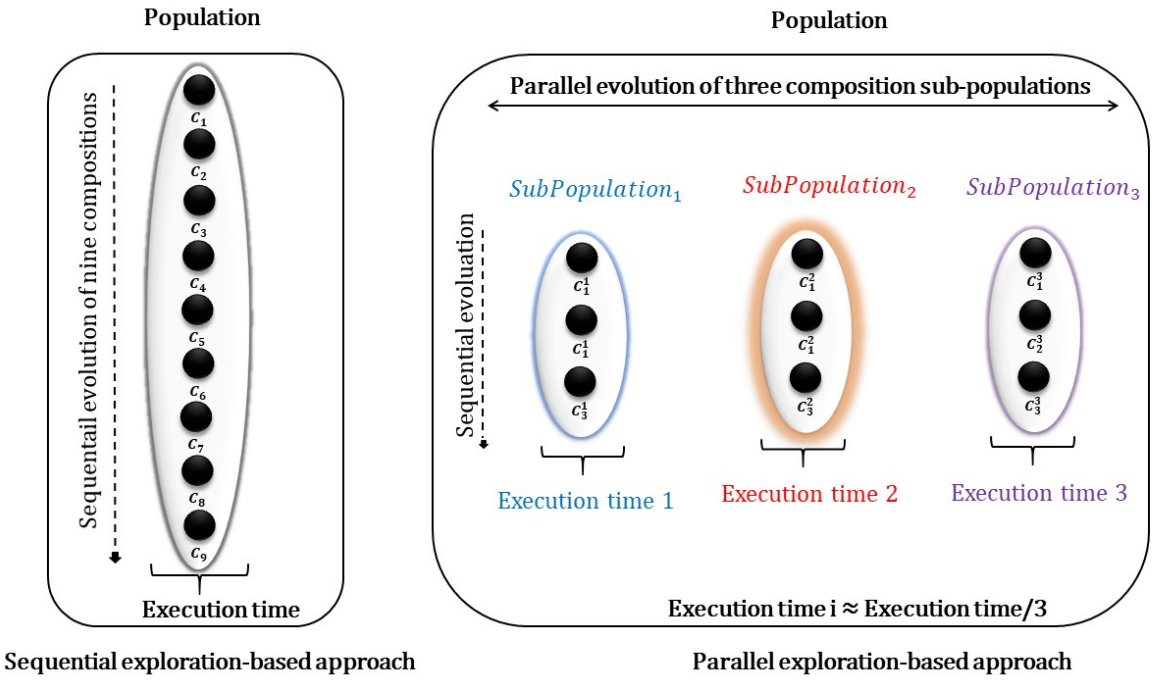


Figure 2.7: Sequential exploration-based approach versus parallel exploration-based approach.

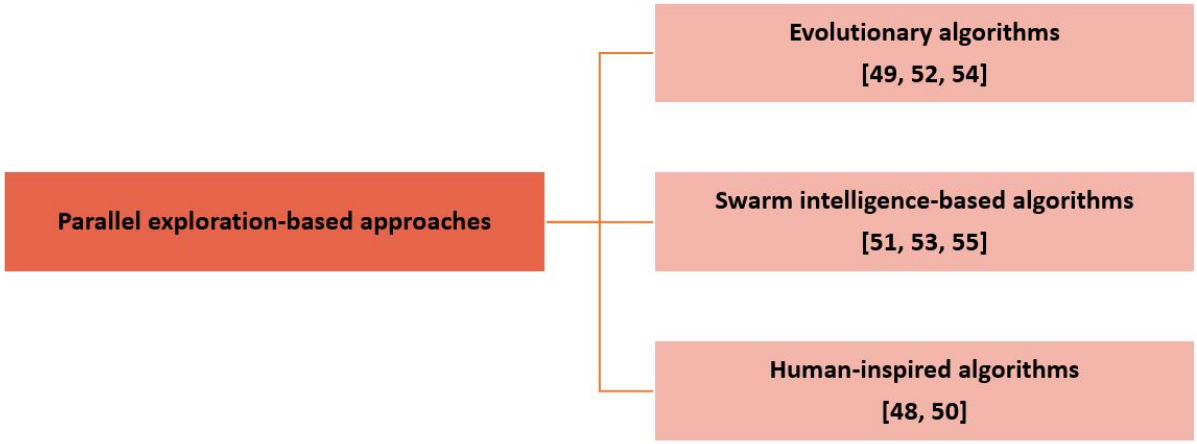


Figure 2.8: Taxonomy of parallel exploration-based servie composition approaches.

to deal with the plan-based QoS-aware service composition problem. In the beginning, the initial parameters of the algorithm are initialized such as the population size, the number of sub-populations, the size of each sub-population, the mutation and crossover probabilities, along with the maximum number of iterations. Then, the initial population of compositions is randomly generated, where chromosomes are used to represent compositions, each one is defined as an array of integers whose size is equal to the number of abstract services in the composition plan. The integers in this array refer to the indices of the randomly selected

concrete services in their corresponding abstract services. The utility values of the initial compositions are calculated based on their global aggregated QoS values and the user's preferences associated with QoS properties. After that, the initial composition population is then divided into distinct sub-populations by randomly assigning compositions according to their specified sizes. The roulette wheel selection strategy [109] is employed to select compositions for evolution based on their utility values, giving higher selection probability to compositions with greater utility. The roulette wheel selection associates a probability for each composition based on the ratio of its utility value and the total utility values of the remaining compositions in the population. The selected compositions are modified using a crossover operator according to the crossover probability, and then updated by a mutation operator based on the mutation probability. The evolved compositions replace the old ones when their utility values are better than those of the old compositions. This process is repeated in parallel for each sub-population until reaching the maximum number of iterations to return the best composition in terms of QoS.
The parallel evolution of multiple sub-populations in the proposed multi-population genetic algorithm allows for a considerable decrease in the composition time. However, the lack of interaction between the composition sub-populations reduces the population diversity, leading to a low composition quality.

Swarm intelligence-based optimization methods have been used in the sub-category of parallel exploration-based approaches to address the service composition problem [51, 53, 55]. In [51], a clustering-based shuffled frog leaping algorithm (SFLA) is proposed to solve the QoS-aware service composition issue in a large-scale environment. The SFLA algorithm performs in two main phases: (*i*) *concrete service filtering* and (*ii*) *service composition*. The objective of the concrete service filtering phase is to reduce the overall composition time by discarding low quality concrete services in each abstract service. To do so, the problem of QoS-aware service composition is modeled as an automatic partitioning problem and solved using the k-means clustering method [110, 111] where a data point represents a concrete service described by a vector of QoS properties. In this context, the k-means method is used to create multiple clusters, each one groups concrete services with similar QoS properties. Subsequently, only the cluster with the highest QoS level, within each abstract service, is retrieved to perform the service composition phase. The latter is carried out in four steps: 1) *initialization*, 2) *partitioning*, 3) *local exploration*, and 4) *shuffling*. In the initialization step, the algorithm settings are initialized and the QoS-aware service composition is modeled as a shuffled frog leaping optimization problem where a frog represents a service composition. More precisely, the population of compositions is randomly generated based on the high quality concrete services resulting from the concrete services filtering phase. The upper limit of global iteration, the upper limit of local iteration, and the number of sub-populations

($nbMemeplex$) are also initialized. The weighted sum utility function is used to compute the global quality of each composition based on the QoS property values of its included concrete services and the user's preferences. In the partitioning step, the composition population is divided into $nbMemeplex$ sub-populations and the compositions are assigned to the sub-populations according to their utility values. More specifically, the best composition in terms of QoS is assigned to the first sub-population, the second best composition to the second sub-population, the $nbMemeplex^{th}$ best composition to the $nbMemeplex^{th}$ sub-population, and the process continues in a round-robin manner (assigning the $(nbMemeplex + 1)^{th}$ back to the first sub-population, and so on). In the local exploration phase, the compositions of each sub-population are improved based on the best and worst composition in their related sub-population, as well as the best composition in the population. This process continues iteratively until reaching the upper limit of local iterations. The updated compositions replace the old ones when their utility values are better. In the shuffling step, the resulting compositions of the sub-populations are grouped and sorted in ascending order of their utility values. The partitioning, local exploration, and shuffling steps are repeated until the upper limit of global iteration is reached. The composition with the best utility value in terms of QoS and that satisfies the global QoS constraint is returned as the suboptimal composition. The SFLA algorithm achieves promising results in terms of composition time and global utility thanks to the combination of the clustering method and the parallelism process. However, the random choice of the number of sub-populations and the upper limit of local iterations can have a negative impact on the performance of the algorithm.

Hybrid optimization methods have been utilized to deal with the plan-based QoS-aware service composition problem [48, 50]. For example, a hybrid multi-population artificial bee colony algorithm (HMABC) integrated with various differential evolution operator strategies is proposed in [48]. The HMABC operates in five phases: (*i*) *initialization*, (*ii*) *sub-populations performance*, (*iii*) *adaptive parameter setting*, (*iv*) *sub-populations size adjustment*, and (*v*) *sub-populations recombination*. In the initialization phase, a food source represents a service composition, and a population of compositions is randomly generated. The total number of iterations, the parameters of the mutation strategies, the number of sub-populations, and their respective sizes are also initialized. In sub-populations performance phase, the composition population is divided into four sub-populations each one is embedded with a specific differential evolution strategy. These sub-populations evolve in parallel by performing the artificial bee colony algorithm steps including employed bees, onlookers, and scouts' steps. In the employed bee step, a specific differential evolution strategy is used to improve the current compositions in each sub-population. The new compositions replace the old ones if they have higher utility values in terms of QoS. In the onlooker bee step, the compositions of each sub-population resulting from the previous step are further evolved

using a specific differential evolution strategy to refine their quality. In the scouts step, the compositions that have not been improved over a defined number of iterations are modified to avoid local optima and improve the exploration capability of the algorithm. In the adaptive parameter setting phase, the parameters related to each differential evolution strategy are self-adapted to find the most appropriate values to further enhance the performance of each sub-population. In the sub-populations size adjustment phase, the sizes of the sub-populations are redefined based on the performance of their associated evolution strategy in the latest iterations. More precisely, a large size is attributed to the sub-populations that had better performance to better exploit the promising regions. In the sub-populations recombination phase, the sub-populations are grouped to form a single population that is divided again and the composition process is repeated until achieving the maximum number of iterations and the suboptimal composition is returned.
The parallel evolution of multiple sub-populations, combined with the distinct evolution strategies employed by the HMABC algorithm, ensures a good balance between exploration and exploitation, providing high QoS composition. In addition, the adaptive sub-population size adjustment allows to reduce the composition time by focusing only on promising regions. However, the algorithm's performance and its scalability are strongly related to the choice of the number of sub-populations as well as their initial size.

2.4.2 Autonomous QoS-aware service composition approaches

The autonomous QoS-aware service composition approaches find the suboptimal composition without relying on a predefined abstract composition plan. They automatically compose concrete services by matching their input and output parameters to find the suboptimal composition that simultaneously optimizes the QoSM and the QoS metrics. The two sub-categories of autonomous QoS-aware service composition approaches (all matching-based and random matching-based approaches) are presented in the following based on their employed resolution method.

2.4.2.1 All matching-based approaches

This sub-category of approaches find semantically correct compositions that could be generated in the service repository. This is done by computing all possible matching between the inputs and outputs of services while accounting for a given semantic model. An exact method is then usually used to determine the most appropriate composition in terms of QoS. The all matching-based approaches include: graph-based methods, database search methods, and Pareto front-based methods (see Figure 2.9).

Graph-based methods use graph theory to model and solve problems. These methods

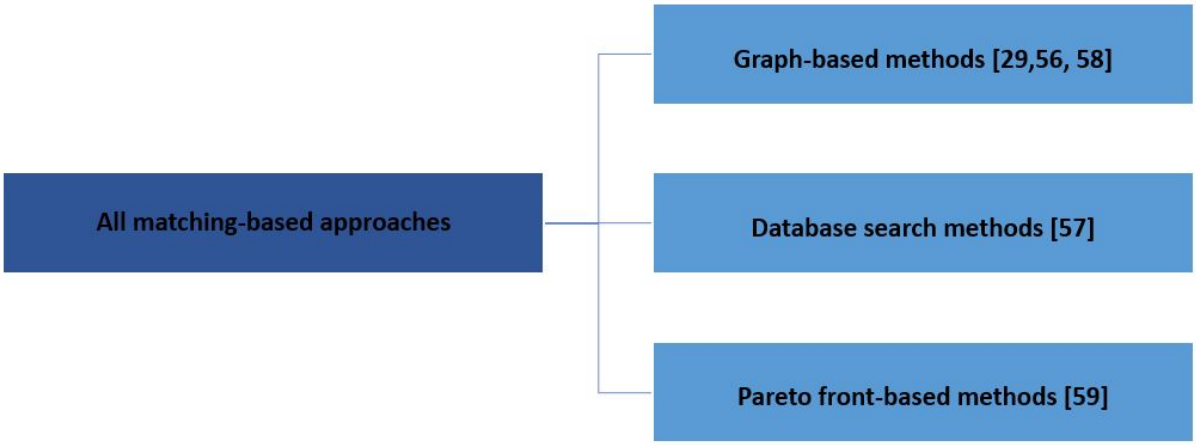


Figure 2.9: Taxonomy of all matching-based service composition approaches.

express and analyze the relationship between data using a graphical structure composed of nodes and connecting edges. Graph-based methods have been widely exploited by a sub-category of all matching-based approaches [29, 56, 58]. For example, a directed graph model is used in [29] to formulate the autonomous QoS-aware service composition as a scheduling problem with AND/OR constraints. A directed graph is first constructed based on the concrete services available in the service repository, where the vertices represent both services and data (i.e., input and output concepts) and edges express two types of connections. An edge from a concrete service to a data node implies that this data is an output of this service, whereas an edge form a data node to a concrete service means that this data is required as input for the service's execution. To maintain the graph, fictitious start and end vertices are added. The QoS values of concrete services are presented as weights on their corresponding vertices. To fulfill the user's request defined by a set of provided inputs and desired outputs, the constructed graph is interpreted as a scheduling problem with AND/OR constraints, where AND-constrained jobs represent concrete services and OR-constrained jobs represent input and output concepts. Indeed, AND-constrained jobs can only be executed when all their preceding jobs are completed, meaning that concrete services can be executed only when their required inputs are available. In contrast, OR-constraints jobs can be activated as soon as at least one of their preceding jobs has been completed. This corresponds to input and output concepts because they become available when they are produced by any concrete service. A Dijkstra algorithm is finally used to solve the service composition issue defined as a job scheduling problem in a polynomial time providing an optimized global execution time or throughput.

The proposed algorithm modeled and solved the autonomous QoS-aware service composition issue as a simple path problem. However, only the execution time and throughput properties are considered. The study also showed that the proposed algorithm cannot be extended to

other QoS properties, either in a single-objective or multi-objective context.

Database search methods have been employed in the context of autonomous QoS-aware service composition. In [57], a relational database approach is introduced to provide the top K compositions for the autonomous QoS-aware service composition according to two main phases: (*i*) *compositions' generation* and (*ii*) *composition query*. In compositions' generation phase, all semantically feasible compositions that could be obtained from the concrete services in the repository are generated and stored in a relational database. To generate these compositions four rules are applied. First, when an output of a concrete service cs_1 serves as the input of a concrete service cs_2, the concrete service cs_1 is considered as an input service of the concrete service cs_2 and there is a composition connecting them. Second, the insertion of a concrete service to a composition that already includes that concrete service is avoided. Third, the inputs of a concrete service in a newly constructed composition can be provided either by the inputs of this composition or by the outputs of concrete services preceding this service. Fourth, the outputs of the concrete services belonging to a composition make up the outputs of this composition. This process ends when no more composition can be generated. Subsequently, the weighted sum utility function is used to calculate the global QoS of each composition. In the composition query phase, all compositions generated from the aforementioned phase and their utility values in terms of QoS are stored in the relational database. An SQL query is then constructed to express the user's requirements and search in the database for the composition that fulfills these requirements. The latter includes a set of required inputs, a set of desired outputs, and global QoS constraints. The SQL query is processed in three steps. Semantically feasible compositions that meet the user's input and output requirements are first identified from the relational database. Then, based on their utility values in terms of QoS, the returned compositions are ranked and filtered according to the specified global QoS constraints. Finally, the top K compositions that satisfy these constraints are returned.
With the dynamic nature of IoT environments, new concrete services may be added, while others may fail or disappear. The proposed approach returns the top K compositions that provide backup solutions for users in the case of service disappearance or failure. However, to find the optimal compositions that meet the user's requirement, all possible compositions have to be generated, resulting in an exponential composition time and low scalability.

Pareto front-based methods have been used to address the autonomous QoS-aware service composition problem. The multi-objective autonomous QoS-aware service composition problem is addressed in [59] by proposing an optimal algorithm that find the Pareto optimal composition satisfying global QoS constraints. The proposed algorithm performs in three main phases: (*i*) *pre-processing*, (*ii*) *dependency graph construction*, and (*iii*) *Pareto front*

construction. The pre-processing aims to reduce the overall composition time by decreasing the size of the search space. To achieve this, a clustering algorithm [112] is first used to group functionally equivalent concrete services into clusters that refer to abstract services. For each abstract service, a skyline set is constructed by keeping only non-dominated concrete services based on their QoS property values. Subsequently, each abstract service is modeled by one representative service associated with multiple QoS tuples, corresponding to the QoS vectors of the concrete services in its skyline service set. In the graph construction phase, a directed graph is built [113], starting from fictitious node including the user's request inputs. Each node in the graph corresponds to a representative service, whereas edge between two consecutive nodes means that the outputs of the preceding service serve as inputs for the succeeding service. A fictitious end node is also added to collect the final outputs of the graph. In the Pareto front construction phase, the dependency graph is transformed into a layered graph based on the matching between the inputs and outputs of the nodes. The Pareto optimal compositions are identified by traversing the dependency graph in a backward direction. The last layer contains the final abstract services in the dependency graph. The second last layer includes the abstract services whose outputs fully satisfy the input requirements of the abstract services in the last layer. The preceding layers are constructed iteratively using the same process until processing all the abstract services in the dependency graph. To find the Pareto optimal compositions, the non-dominated concrete services are first identified in each abstract service of the last layer. Each of these services is then combined with the non-dominated concrete services of the corresponding abstract services in the preceding layer, and the cumulative QoS property values are calculated accordingly. Dominated sub-compositions are discarded whereas non-dominated ones are retained. This process is repeated iteratively until reaching the starting node that contains the user's inputs and the set of non-dominated compositions is returned. Note that the layered graph and the Pareto optimal compositions are constructed simultaneously to ensure the satisfaction of both functional and non-functional requirements.
The proposed approach guarantees the generation of Pareto optimal compositions. However, this method can have exponential time complexity when applied to large-scale environments, despite avoiding the evaluation of all possible compositions.

2.4.2.2 Random matching-based approaches

This sub-category of approaches randomly selects a population of compositions from the service repository. The semantic correctness of each composition is then checked using a decoding algorithm to ensure that the generated compositions meet the user's functional requirements. Finally, the compositions that satisfy the user's functional requirements are employed as the initial population, and a computational intelligence-based algorithm is applied to identify the suboptimal composition. Random matching-based approaches include:

evolutionary and hybrid optimization methods (see Figure 2.10).

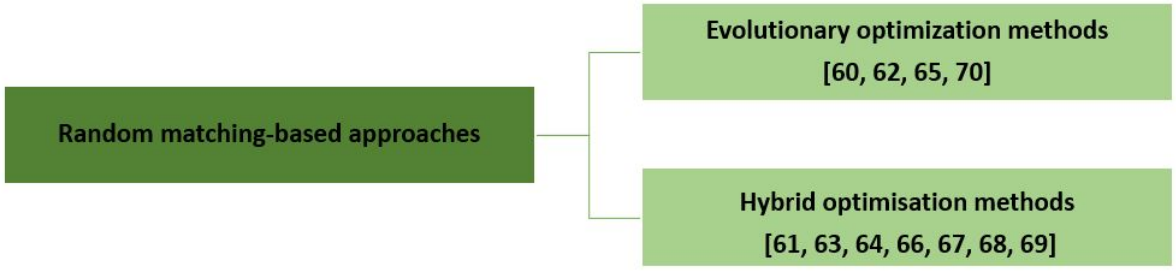


Figure 2.10: Taxonomy of random matching-based service composition approaches.

Evolutionary optimization methods have been widely applied in this sub-category [60, 62, 65, 70]. For example, a two stages genetic algorithm (2StageGA) is introduced in [65] to solve the autonomous QoS-aware service compositions issue with stochastic service failure. This algorithm performs in two phases: (*i*) *design* and (*ii*) *execution*. In the design phase, the algorithm takes as inputs the user's request expressed as a set of input and output concepts, a service repository containing concrete services, and an ontology describing the functional features of the concrete services. Subsequently, a population of compositions is generated by randomly selecting concrete services from the service repository. The semantic feasibility of each composition is checked using a decoding algorithm [62] that determines an execution order for the concrete services belonging to the composition and retains only the sequence or the sub-sequence that satisfies the user's functional requirements. The utility value of each feasible composition is then calculated by jointly taking into account QoS and QoSM metrics. An order crossover operator [114] and a one-point swap mutation are employed to further improve the quality of the current population. In the order crossover operator, two offspring compositions are generated from a pair of parent compositions. Each offspring composition preserves a sub-sequence of concrete services from one parent, while another parent fulfills the remaining services. In the mutation operator, a single offspring is produced by swapping two concrete services within a parent composition. An offspring composition replaces the old one when its utility value in terms of QoS and QoSM is better. An adaptive number of iterations is calculated to determine the starting time of the execution phase. In this phase, the same genetic operators are employed to further improve the compositions. The robustness of these compositions is then evaluated using a utility approximation function that calculates the expected quality of the compositions in the case of a stochastic service failure by considering different service failure scenarios. More precisely, the robustness of each composition is calculated based on the best repaired solution for each composition with respect to QoS and QoSM metrics. After that, according to the population size, the most robust compositions are kept to perform a new evolution in the next iteration. The 2StageGA algorithm repeats these phases until reaching a maximum number of iterations.

The 2StageGA algorithm can find robust compositions that handle the stochastic service failure which is a challenging issue in the field of service composition. However, to determine whether the user's functional requirements have been met, the decoding algorithm is used each time a composition is generated resulting in a relatively high composition time.

Hybrid optimization methods have been used in the context of autonomous QoS-aware service composition issue [61, 63, 64, 66, 67, 68, 69]. For example, a QoS-aware service composition approach using the *data-intensive service composition* concept is proposed in [67]. This approach not only considers QoS but also addresses the data distributed across different locations, which need to be transferred and manipulated between concrete services. Subsequently, a hybrid algorithm based on the non-dominated sorting genetic algorithm (NSGA-II) and a local search is proposed to find the Pareto optimal composition for the distributed data-intensive autonomous QoS-aware service composition problem. The proposed algorithm starts by randomly selecting concrete services from the service repository to generate the initial population of compositions, where the feasibility of the compositions is ensured by employing a decoding algorithm. The general idea of this decoding algorithm is to trace all inputs that have not yet been fulfilled, by traversing the composition backward from the last concrete services to the first, until all required inputs are satisfied. As a result, each composition is reduced to contain only the concrete services that have matching links between their inputs and outputs. The utility values of these compositions are then calculated based on their QoS property values, the expected communication time, and the cost of transferring and accessing data between their concrete services. Subsequently, the composition population is divided into different non-dominated groups, such as the first group contains all the non-dominated compositions, the second includes the compositions dominated only by those in the first group, the third contains the compositions dominated only by those in the second group, and so on. Thereafter, offspring compositions are generated using the distance guided crossover operator [61] between pairs of compositions. This crossover incorporates service locations by preserving the longest sequence of concrete services that appears in both parent compositions. Afterwards, a mutation operator is applied to randomly select and swap two concrete services in each offspring composition. The resulting offspring compositions are then merged with the current population, and the non-dominating groups are recalculated. After that, a subset of compositions is selected for improvement using a local search method based on the concept of the communication link dominance. The latter defines a new dominance rule based on the cost and execution time of data transfer between concrete services to identify neighborhood compositions. Finally, compositions are compared based on their non-dominated groups to select the better one for the new evolution cycle. This process is repeated until a maximum number of iterations is reached.
The proposed hybrid NSGA-II with a local search method deals with many challenges in the

context of multi-objective QoS-aware service composition issue and finds the Pareto optimal compositions. However, in large-scale scenarios, the use of the Pareto dominance concept to find the suboptimal composition becomes computationally expensive.

2.5 Comparative Analysis

An analytical review of the two categories of QoS-aware service composition approaches is given in the following.

Tables 2.1 and 2.2 present a comparison of sequential exploitation-based approaches and parallel exploration-based approaches, respectively, considering three criteria: solving method, population size reduction, and scalability degree. The solving method is the computational intelligence-based optimization algorithm used to find the suboptimal composition in the context of QoS-aware service composition problem. The population size reduction indicates whether or not a method is used to reduce the size of the composition population through the iterations. The scalability measures the effectiveness of the service composition approach in terms of composition time and/or composition quality when the number of concrete and abstract services increases. Three scalability degrees are considered in this thesis: a low number of concrete services and abstract services means a low scalability degree, a low number of concrete services and a high number of abstract services, or vice versa, results in a medium scalability degree, and a high number of concrete services and abstract services refers to a high scalability degree.

Table 2.1 shows that swarm optimization methods are the most widely used computational intelligence-based techniques within sequential exploration-based approaches. Additionally, these approaches do not employ a population size reduction strategy, and some studies [39, 45] have a low scalability degree.

Table 2.2 shows that parallel exploration-based approaches exhibit a balanced use of various computational intelligence-based solving methods and none of these approaches incorporates a population size reduction strategy. Furthermore, parallel exploration-based approaches generally have a medium to high scalability degree.

Tables 2.1 and 2.2 show that, within the plan-based QoS-aware service composition category, parallel exploration-based approaches have been less explored than sequential ones. In addition, most sequential exploration-based approaches are moderately scalable, whereas parallel approaches achieve higher scalability, making them more suitable for large-scale service composition scenarios. However, neither approach type employs a population size reduction method, which constitutes a promising direction for further investigation.

Tables 2.3 and 2.4 give a comparison of all matching-based approaches and random matching-based approaches, respectively, with respect to the following criteria: ($C1$) solving

Table 2.1: Sequential exploration-based QoS-aware service composition approaches.

	Comparison criterion		
Approach	Solving method	Population size reduction	Scalability
[30]	Eagle strategy with whale optimization algorithm	✗	Medium
[31]	Improved teaching learning-based algorithm	✗	High
[32]	Fuzzy multi-objective artificial bee colony	✗	Medium
[33]	Krill herd algorithm	✗	Medium
[34]	Improved gale shapley algorithm	✗	Medium
[35]	Modified cuckoo search algorithm	✗	Medium
[36]	Improved genetic algorithm	✗	Medium
[37]	Improved invasive weed algorithm	✗	High
[38]	Improved social learning algorithm	✗	Medium
[39]	Fuzzy linguistic-based algorithm	✗	Low
[40]	Improved multi-objective grey wolf optimizer	✗	Medium
[41]	Multi-verse algorithm	✗	High
[42]	Hidden markov-based artificial bee colony	✗	High
[43]	Hybrid eagle strategy	✗	High
[44]	Improved harris hawks algorithm	✗	High
[45]	Improved particle swarm algorithm	✗	low
[46]	Improved bat algorithm	✗	Medium
[47]	Genetic algorithm	✗	High

Table 2.2: Parallel exploration-based QoS-aware service composition approaches.

	Comparison criterion		
Approach	Solving method	Population size reduction	Scalability
[48]	Improved artificial bee colony	✗	High
[49]	Hybrid genetic algorithm	✗	Not mentioned
[50]	Improved artificial bee colony	✗	Medium
[51]	Shuffled frog-leaping algorithm	✗	Medium
[52]	Improved genetic algorithm	✗	Not mentioned
[53]	Hybrid shuffled frog-leaping algorithm	✗	Medium
[54]	Improved genetic algorithm	✗	High
[55]	Hybrid monarch butterfly	✗	High

method, ($C2$) utility combining QoS and QoSM metrics, ($C3$) failure recovery, and ($C4$) automatic building of abstract composition plan. The solving method refers to the algorithm

employed by the approach to find the composition that satisfies the user's requirements. The utility combining QoS and QoSM indicates whether the composition approach takes into account QoS and QoSM metrics in the calculation of the overall quality of a composition. The failure recovery specifies whether the approach addresses the problem of service failure or not. Automatic building of an abstract composition plan specifies if the approach automatically generates an abstract composition plan during the composition process.

Table 2.3: All matching-based QoS-aware service composition approaches.

	Comparison criterion			
Approach	($C1$)	($C2$)	($C3$)	($C4$)
[29]	An extension of the Dijkstra algorithm	✗	✗	✗
[56]	Dijkstra-based label-setting algorithm	✗	✗	✗
[57]	SQL query in relational database	✗	✓	✗
[58]	The shortest bidirectional breadth-first and Dijkstra algorithms	✗	✗	✗
[59]	Pareto front-based method	✗	✗	✗

Table 2.4: Random matching-based QoS-aware service composition approaches.

	Comparison criterion			
Approach	($C1$)	($C2$)	($C3$)	($C4$)
[60]	Improved NSGA- Ⅱ algorithm	✗	✗	✗
[61]	Hybrid genetic algorithm	✗	✗	✗
[62]	Improved genetic algorithm	✗	✗	✗
[63]	Memetic estimation of the distribution-based algorithm	✓	✗	✗
[64]	Memetic algorithm	✗	✗	✗
[65]	Two-stage genetic algorithm	✓	✓	✗
[66]	Genetic algorithm	✗	✗	✗
[67]	Hybrid non-dominated sorting genetic algorithm	✗	✗	✗
[68]	Memetic algorithms	✗	✗	✗
[69]	Hybrid non-dominated sorting genetic algorithm	✗	✗	✗
[70]	Evolutionary algorithm	✗	✗	✗

Table 2.3 shows that all matching-based approaches are infrequently used in the context of autonomous QoS-aware service composition. This is because of the significant computation time they require to ensure that all possible matches between concrete service input and output concepts are explored to identify the suboptimal composition. In addition, these

approaches do not simultaneously take into account QoS and QoS metrics when calculating the overall utility, which limits their efficiency. Furthermore, these approaches do not automatically generate an abstract composition plan as part of the composition process, and most of them [29, 56, 58, 59] do not address the service failure issue.

Table 2.4 highlights that random matching-based approaches are the most frequently used methods for addressing the autonomous QoS-aware service composition problem. This is due to the efficiency of random matching-based approaches in finding a good trad-off between composition quality and composition time. In addition, most of random matching-based approaches [61, 62, 64, 66, 67, 68, 69, 70] calculate the overall quality of composition by considering only QoS, while neglecting the QoSM metric. Moreover, only a few approaches [65] address the service failure problem and none of these approaches automatically generate abstract composition while performing the composition process.

Table 2.3 and 2.4 show that random matching-based approaches are more employed than all matching-based approaches for addressing the autonomous QoS-aware service composition problem. More precisely, all matching-based approaches outperform random matching-based approaches in terms of composition quality when exploring small search spaces, as they provide suboptimal composition within a reasonable composition time. However, in large-scale IoT environments, random matching-based approaches perform better despite a slight reduction in composition quality, offering a compromise between computation time and composition quality. In addition, most autonomous QoS-aware service composition approaches do not address service failure and do not simultaneously consider QoS and QoSM to calculate the overall quality of compositions although the fact that QoSM is a fundamental property in the expression of semantic. This analysis illustrates that investigating a hybrid method that combines the advantages of all matching-based approaches and random matching-based approaches is an interesting direction for future research.

2.6 Analysis and discussion

This section analyzes and discusses the QoS-aware service composition approaches investigated in this chapter.

2.6.1 Analysis

The studied QoS-aware service composition approaches are analyzed with respect to the five research questions outlined in subsection 2.3.1, which are presented as follows:

- *RQ*1: What are the primary classification criteria for QoS-aware service composition approaches?

Figure 2.11 provides a comparison of the studied QoS-aware service composition approaches according to the first layer taxonomy proposed in this thesis. We observe that 62% of the reviewed approaches represent plan-based QoS-aware service composition approaches, whereas the autonomous QoS-aware composition approaches cover only 38% of approaches. This demonstrates that the majority of existing approaches assume the existence of a predefined abstract composition plan representing the user's functional requirements. This assumption facilitates the resolution of the QoS-aware service composition problem. More precisely, unlike the autonomous QoS-aware service composition approaches that search to simultaneously satisfy the user's functional and non-functional needs, the plan-based approaches focus only on finding the composition that best matches the user's requirements in terms of QoS.

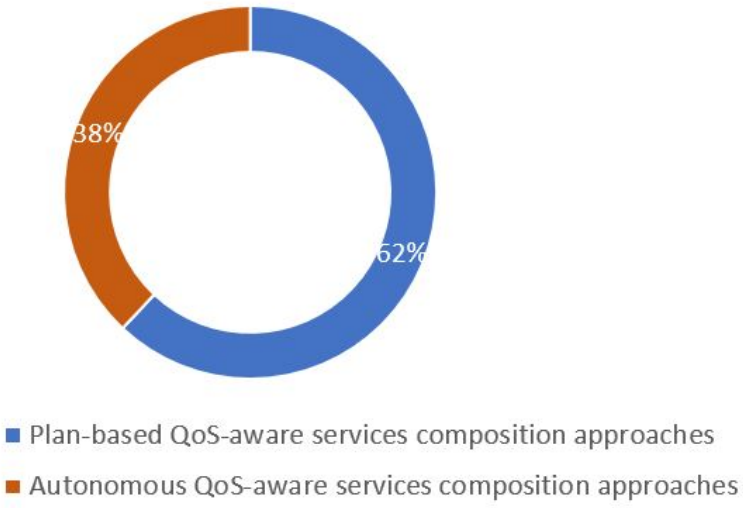
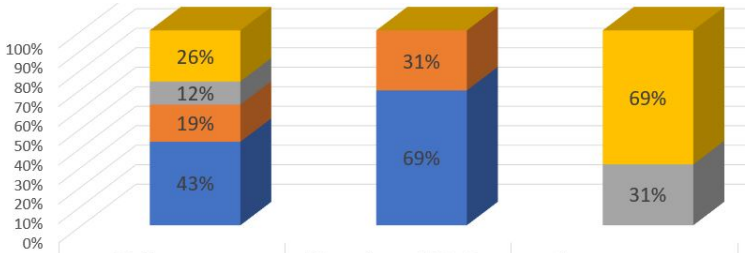


Figure 2.11: Ratio of QoS-aware service composition approaches with respect to $RQ1$

.

- $RQ2$: Can secondary criteria refine the initial classification categories?
  Figure 2.12 shows that 69% of the studied plan-based QoS-aware service composition approaches employ sequential exploration-based search methods, whereas only 31% of approaches use parallel exploration-based methods. For autonomous approaches, 69% rely on random matching-based strategies and 31% use all matching-based ones. These results indicate that sequential exploration-based search methods are the most widely used to solve the plan-based QoS-aware service composition problem, whereas the autonomous service composition is more frequently solved using random matching-based strategies.



	QoS-aware service composition approaches	Plan-based QoS-aware approaches	Autonomous QoS-aware approaches
Random matches-based approaches	11	0	11
All matches-based approaches	5	0	5
Parallel exploration-based approaches	8	8	0
Sequential exploration-based approaches	18	18	0

Figure 2.12: Ratio of QoS-aware service composition approaches regarding $RQ2$.

- $RQ3$: What methods are commonly employed to solve the QoS-aware service composition problem?
  Figure 2.13 shows that 88% of the reviewed QoS-aware service composition approaches rely on computational intelligence-based optimization methods to find suboptimal compositions, whereas only 12% of the approaches employ exact methods. More specifically, while all plan-based QoS-aware service composition approaches exploit computational intelligence-based optimization methods, autonomous QoS-aware service composition approaches exhibit a more diverse range of resolution strategies where 69% among these approaches use computational intelligence-based optimization methods and the remaining 31% employ exact methods. The computational intelligence-based methods are widely used to solve the QoS-aware service composition problem due to their efficiency in finding the suboptimal composition in a reasonable computation time. Conversely, exact methods are less frequently used due to their exponential time complexity especially in large-scale environments, despite finding the optimal composition.

- $RQ4$: What performance metrics are commonly used to evaluate these approaches?
  Figure 2.14 reveals that 83% of the analyzed QoS-aware service composition approaches assess their performance by measuring the composition time that represents the computation time required by an algorithm to determine a suboptimal composition. Additionally, 95% of the approaches prioritize the use of the overall QoS utility as the primary performance metric, whereas only 5% focus on an overall utility that integrates QoS and QoSM metrics to evaluate the overall quality of the composition. The overall QoS utility is calculated as a weighted sum of the QoS attribute values in the resulting

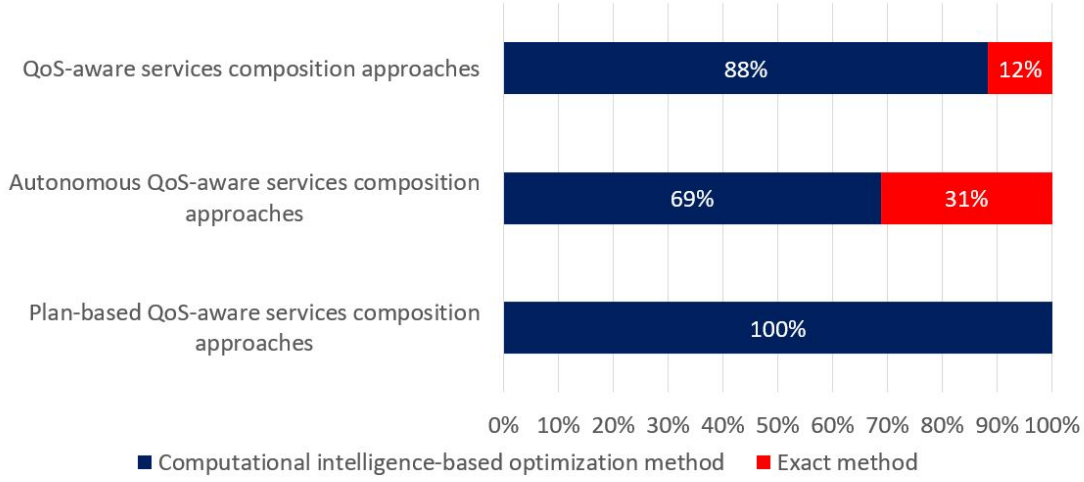


Figure 2.13: Ratio of QoS-aware service composition approaches considering $RQ3$.

suboptimal composition, where the weights reflect the user's preference for each QoS attribute. The overall utility metric combines the aggregated QoS and QoSM values of the suboptimal composition, each weighted by the corresponding user's preference.

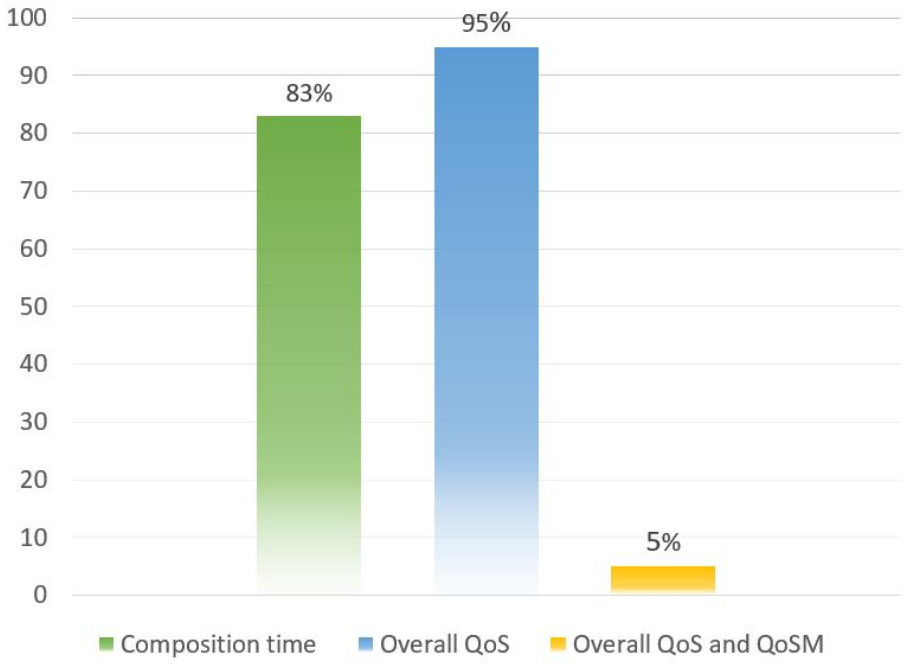
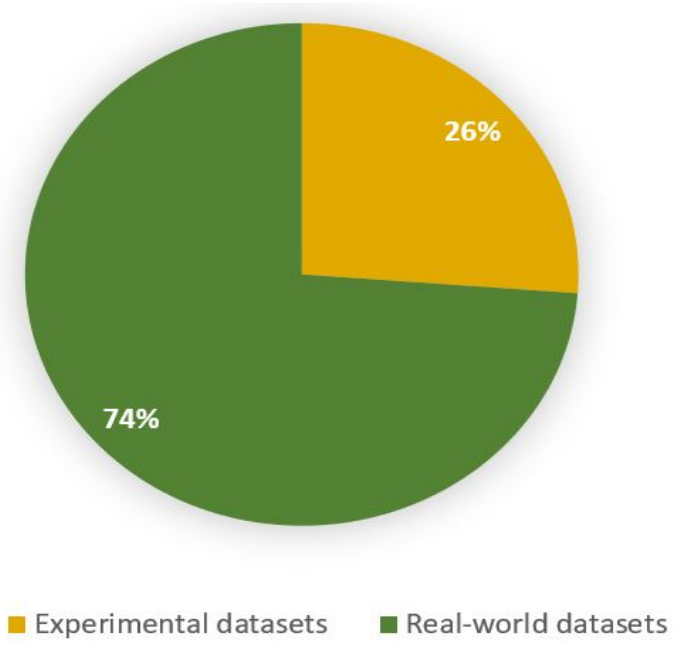


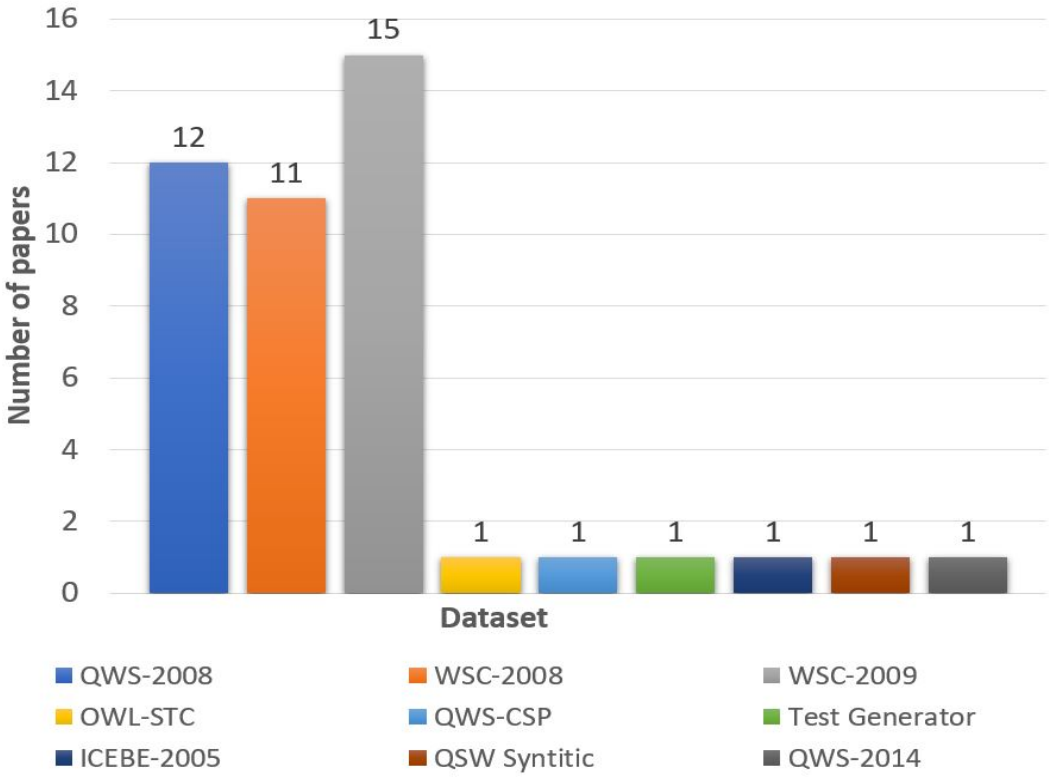
Figure 2.14: Ratio of QoS-aware service composition approaches accounting for $RQ4$.

- $RQ5$: What datasets are used to evaluate the performance of the investigated approaches?

Figure 2.15a shows that 26% of the reviewed QoS-aware service composition approaches use experimental datasets to assess their performance, while 74% of the approaches employ real-world datasets. More precisely, according to Figure 2.15b, 12 papers [30, 31, 32, 34, 36, 37, 41, 44, 46, 47, 52, 53] use the QWS-2008 dataset [115], 11 papers [60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70] employ the WSC-2008 dataset [116], and 15 papers [29, 56, 58, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70] exploit the WSC-2009 dataset [117]. The QWS-2008 dataset includes 2507 concrete services. The WSC-2008 and WSC-2009 datasets contain respectively 8 and 5 service repositories with their respective users' requests and ontologies. Each ontology is described as a tree of concepts that allows performing the semantic matching between any pairs of concrete services. The service repositories of the WSC-2008 dataset contains respectively 158, 558, 608, 1041, 1090, 2198, 4113 and 8119 concrete services, whereas the service repositories of the WSC-2009 dataset have 572, 4129, 8138, 8301, and 15211 services, respectively. Each service repository in the WSC-2008 and WSC-2009 datasets includes the concrete service names, their QoS attribute values, and their input and output concept descriptions. In addition, the studies [65, 33, 45, 57, 59, 42] use respectively the OWLS-TC [118], QWS-mobility [119], QWS-CSP [120], TestGenerator2009 [121], ICEBE-2005 [122], and QWS-2014 [123] datasets. The OWLS-TC dataset [118] consists of 5 equally sized service repositories containing a total of 946 concrete services along with their corresponding users' requests and ontologies. The QWS-mobility dataset [119] contains the QoS attribute values of 4500 real-world mobile concrete services recorded by 142 users every 15 minutes. The QWS-CSP dataset [120] provides historical values for 5 QoS properties across 100 cloud concrete services gathered during 6 months in 28 time intervals. The TestGenerator2009 [121] is a Java tool that allows the generation of datasets, each one containing a concrete service repository, a user request, and an ontology. Each concrete service is defined by a set of QoS attribute values along with its input and output concepts. The ICEBE-2005 dataset [122] contains 19 varying size service repositories in the range of 143 to 8356 concrete services. Each service repository holds the description of service inputs, outputs, and user requests. More specifically, the first repository includes 4 user's requests, whereas each of the remaining repositories contains 11 user's requests. The QWS-2014 dataset [123] contains a single repository of concrete services, where each service is defined by 6 QoS attribute values and each QoS attribute is described by its name, domain, type, measurement unit, and its possible sampling values. This analysis reveals that the QWS-2008 dataset is the most frequently employed in the plan-based QoS-aware service composition approaches, while the WSC-2008 and WSC-2009 datasets are predominantly used by autonomous QoS-aware service composition approaches.



(a) Ratio of experimental datasets versus real-world datasets.



(b) Number of investigated real-world datasets.

Figure 2.15: Ratio of QoS-aware service composition approaches regarding $RQ5$.

2.6.2 Critical discussion and research justification

This section identifies the main challenges and limitations of the various categories of the QoS-aware service composition approaches analyzed in section 2.5.

The plan-based QoS-aware service composition is a challenging issue in the area of IoT. It consists of selecting several concrete services from a predefined abstract composition plan and combining them to obtain a composite service that satisfies the user's global QoS constraints while handling multi-conflicting QoS properties. The difficulty encountered in the resolution of the plan-based QoS-aware service composition problem can be related to three main factors: the complexity of optimizing multi-conflicting QoS properties, user's constraints, and scalability issues, making this problem a subject of much research in the literature. Currently, computational intelligence-based service composition approaches allow to find a suboptimal combination of concrete services in a reasonable time. Depending on the strategy used to explore the composition search space, computational intelligence-based service composition approaches can be classified into two sub-categories: sequential exploration-based approaches [30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47] and parallel exploration-based approaches [48, 49, 50, 51, 53, 54, 55]. In the first sub-category, the population search space is explored through a sequential evolution process, resulting in (*i*) a low convergence speed that increases the composition time and/or (*ii*) a local optimum that leads to low or non-optimal composition quality. The second category of approaches is employed to preserve population diversity, enhance convergence speed (i.e., reduce composition time), and/or avoid local optima (i.e., achieve high composition quality) of sequential exploration-based approaches. The parallel exploration-based approaches explore the search space using a parallel evolution process. However, some of these approaches lack interactions between the composition sub-populations, which can lead to a decrease in composition quality. Furthermore, some parallel exploration-based approaches [52] are characterized by a limited interaction between the compositions of their sub-populations, which can reduce the population diversity and consequently decrease the composition quality. The population diversity reflects the variations among the concrete services that make up the compositions within the population. In addition, plan-based QoS-aware service composition approaches [30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55] explore a fixed-size population of composition, which can considerably influence their effectiveness in terms of composition quality and composition time. More precisely, the search space may include unpromising compositions in terms of QoS, which can have a negative impact on composition time and/or composition quality, especially in a large-scale search space. A large population size allows a deep exploration of the search space, leading to a high composition quality and an increase in the composition time. Conversely, a small population size leads to a low composition time and can decrease the composition quality.

Furthermore, plan-based QoS-aware service composition approaches rely on the existence of an abstract composition plan and neglect the semantic aspect of the composition process. In real-world scenarios, abstract composition plans are not given beforehand and their manual construction is impractical.

The absence of semantic considerations makes the service composition process syntactic, which can lead to inconsistencies in the interpretation of service functionalities or failures in the obtained results. To overcome this limitation of plan-based approaches, autonomous QoS-aware service composition approaches have been introduced, as they integrate both QoS and semantic matching, making the composition process more realistic and applicable in real-world IoT environments. These approaches are divided into two sub-categories: all matching-based approaches [29, 56, 57, 58, 59] and random matching-based approaches [60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70]. All matching-based approaches employ exact methods to find the optimal composition in terms of QoS. However, they require an exponential composition time as they explore all possible matching between concrete services in the service repository to find functionally correct compositions. Furthermore, these approaches focus on a single QoS attribute, overlooking a fundamental challenge in QoS-aware service composition, which is the need to optimize multi-conflicting QoS attributes simultaneously. Random matching-based approaches employ computational intelligence-based optimization methods to find suboptimal composition in a reasonable composition time. However, they rely on randomness in selecting compositions and employ a decoding algorithm [62] to ensure the semantic feasibility of evolved compositions. This process reduces the composition quality and considerably increases the composition time. Furthermore, all matching-based approaches [29, 56, 57, 58, 59] and most of random matching-based approaches [60, 61, 62, 64, 66, 67, 68, 69, 70] neglect addressing both QoS and QoSM in the calculation of the overall composition quality. This is inappropriate for addressing the problem of autonomous QoS-aware service composition, as QoSM metric is a key concept for measuring semantic quality.

2.7 Conclusion

The study conducted in this chapter shows that plan-based QoS-aware service composition has been explored more extensively than autonomous QoS-aware service composition. More precisely, within the plan-based category, sequential exploration-based approaches have received more attention than parallel ones, while in the autonomous category, random matching-based approaches are more studied than all matching-based approaches. Further analysis of the reviewed approaches demonstrates a preference for computational intelligence-based optimization methods over exact methods to deal with the composition problem. The performance of these approaches relies primarily on composition time, QoS utility value, and

utility value combining QoS and QoSM metrics. In addition, real-world datasets are more commonly used than experimental ones to assess the effectiveness of these approaches.

This review highlights key limitations in existing studies and provides a foundation for developing more efficient QoS-aware service composition approaches. To overcome these drawbacks, the objective of the following chapters is to present two service composition approaches proposed in this thesis. The first one addresses the plan-based QoS-aware service composition problem using a parallel exploration-based method, while the second one solves the autonomous QoS service composition problem using an all matching-based method.

Chapter 3

A parallel approach for user-centred QoS-aware service composition

Contents

3.1 Introduction

Two categories of computational intelligence-based approaches have been proposed in the literature to find suboptimal compositions within a reasonable computational time: sequential and parallel exploration-based approaches. However, most sequential exploration-based approaches suffer from high composition time and/or limited composition quality, whereas some parallel approaches lack sufficient diversity. Furthermore, the two categories of approaches operate on a fixed population size, which can lead to an increase in composition time and/or a decrease in composition quality.

To address these limitations, the first contribution of this thesis introduces a *Parallel Differential Evolution-based approach with population size reduction for QoS-aware service Composition* (PDE-QSC). This approach combines a parallel exploration of the composition search space with a population size reduction strategy that continuously decreases the size of the composition population by removing unpromising compositions in terms of QoS. In this

chapter, we first describe the service composition model and present the problem statement along with the adaptive multi-population differential evolution method exploited in the proposed approach. We then introduce the main idea of the PDE-QSC approach and detail its phases. Finally, we compare the performance of the PDE-QSC approach with those of five baseline approaches and perform a statistical analysis of the obtained results.

3.2 Background

3.2.1 Service composition model

3.2.1.1 Abstract composition plan

An abstract composition plan (ACP) is defined as:

$$ACP = Combin_{k=1}^{m} AS_k \tag{3.1}$$

where m is the number of functionalities required to satisfy the user's needs. Each functionality AS_k refers to an abstract service, and the $Combin$ operator represents the composition structure, which can take the following forms: loop, sequential, parallel, or conditional [16].

Example. Smart traffic management service

Consider a *smart traffic management service* designed for a smart city environment to optimize traffic flow, reduce congestion, and prioritize emergency vehicles (see Figure 3.1). In this case, the ACP consists of five abstract services connected in a sequential structure: traffic monitoring service, traffic data analysis service, notification service, traffic signal adjustments service, and route optimization service (see Figure 3.2). The functionality of each abstract service is described in Table 3.1.

3.2.1.2 Abstract service

An abstract service AS_k is defined as a set of n executable services offering the similar functionality with different quality levels. Formally, this can be expressed as:

$$AS_k = \{cs_j^k\}_{j=1}^{n} \tag{3.2}$$

Example. Table 3.2 shows executable services corresponding to each abstract service in the smart traffic management service.

Figure 3.1: Traffic flow visualization in a smart city.

Table 3.1: Abstract services of the smart traffic management service.

Abstract service	**Description of the functionality**
Traffic monitoring	Collect real-time traffic data.
Traffic data analysis	Detect congestion and accidents.
Notification	Notify passenger and emergency vehicles in the case of congestion traffic.
Traffic signal adjustments	Adjust traffic lights based on traffic data analysis to priorize emergency lanes if an accident occurs or optimize flow by favoring less congested routes.
Route optimization	Guides vehicles through optimized routes.

3.2.1.3 Concrete service

A concrete service cs_j^k is an executable service that implements the functionality of an abstract service AS_k. It is characterized by a quality vector consisting of f components:

$$QoS(cs_j^k) = (q_{j,1}^k, \cdots, q_{j,l}^k, \cdots, q_{j,f}^k) \tag{3.3}$$

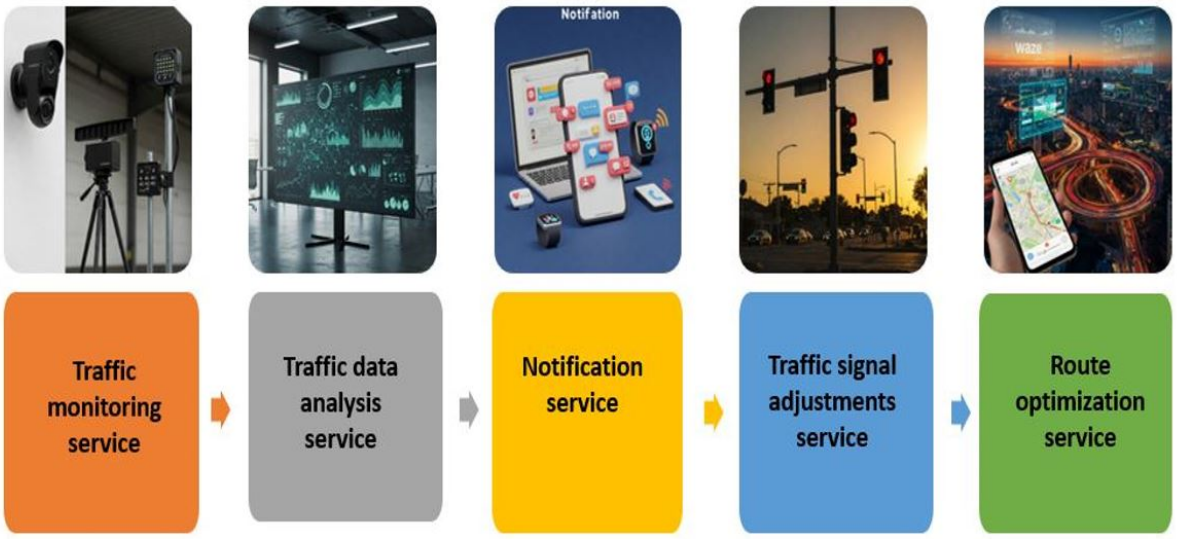


Figure 3.2: Abstract composition plan of the smart traffic management service.

Table 3.2: Executable services corresponding to each abstract service in the smart traffic management service.

Abstract service	**Executable services**
Traffic monitoring	CameraHD, sensor, and radar
Traffic data analysis	Autonomous traffic signal system, artificial intelligence-powered video analytics platform, traffic analyzer system
Notification	Official smart city alert application, advanced driver-assistance system, and smartphone
Traffic signal adjustments	Adaptive traffic signal control system, traffic management center, and urban application programming interface
Route optimization	Google Map, Waze, and MapFactor Navigator

where the component $q^k_{j,l}$ refers to the l^{th} QoS property that represents a non-functional aspect of the service. These properties can be positive (e.g., reliability) or negative (e.g., cost).

Example. The concrete service $cameraHD^1_1$ implements the abstract service for *traffic monitoring*. This concrete service is defined by a quality vector comprising four QoS properties:

- Latency: 50 milliseconds (ms).
- Cost: 0.02 dollars ($).

- Reliability: 99.8%.
- Availability: 99.7%.

Formally, the quality vector for this service is expressed as:

$$QoS(CameraHd_1^1) = (50, 0.02, 99.8, 99.7) \tag{3.4}$$

A normalization process is applied to ensure consistent evaluation and enable a fair comparison between services described by QoS properties with different scales. This process transforms all QoS values into a common numerical scale within the range $[0, 1]$ to eliminate discrepancies caused by heterogeneous units or value ranges. The values of the negative QoS properties are normalized according to the formula (3.5), whereas the positive values are normalized using the formula (3.6).

$$Norm(q_{j,l}^k) = \begin{cases} \frac{q_j^{k,max} - q_{j,l}^k}{q_j^{k,max} - q_j^{k,min}} & \text{if } q_j^{k,max} - q_j^{k,min} \neq 0 \ , \\ 1 & \text{if } q_j^{k,max} - q_j^{k,min} = 0 \end{cases} \tag{3.5}$$

$$Norm(q_{j,l}^k) = \begin{cases} \frac{q_{j,l}^k - q_j^{k,max}}{q_j^{k,max} - q_j^{k,min}} & \text{if } q_j^{k,max} - q_j^{k,min} \neq 0 \ , \\ 1 & \text{if } q_j^{k,max} - q_j^{k,min} = 0 \end{cases} \tag{3.6}$$

Note that $q_j^{k,min}$ and $q_j^{k,max}$ represent, respectively, the minimum and the maximum values that the l^{th} QoS property can take among all concrete services available in the k^{th} abstract service.

3.2.1.4 Composite service

A composite service $C = Combin_{k=1}^m cs_j^k$ is a combination of m concrete services where each concrete service cs_j^k is selected for each abstract service AS_k from the abstract composition plan ACP. The operator $Combin$ specifies the composition structure that defines how the concrete services are linked with each other. The quality of a composite service C is represented by a vector:

$$QoS(C) = \{Ag_{k=1}^m q_{j,1}^k, \cdots, Ag_{k=1}^m q_{j,l}^k, \cdots, Ag_{k=1}^m q_{j,f}^k\} \tag{3.7}$$

where $Ag_{k=1}^m q_{j,l}^k$ denotes the aggregated value of the l^{th} QoS property across all m selected concrete services. The aggregation method depends on the type of the QoS property (e.g., additive for cost) and the composition structure (see [50] for more details).

Example. A smart traffic management composite service could be represented as:
C = Radar → Traffic analyzer system → Adaptive traffic signal control system → Traffic Management Center → Waze.

3.2.1.5 Composition quality

The overall quality of the composite service C is measured using a QoS utility value $F(C)$ that aggregates the normalized values of all QoS properties based on the user's preferences. Formally:

$$F(C) = \sum_{l=1}^{f} w_l \times Norm(Ag_{k=1}^{m} q_{j,l}^{k}) \tag{3.8}$$

where f is the number of QoS properties, $w_l \in [0,1]$ is the user's preference weight with respect to the l^{th} QoS property with the constraint $\sum_{l=1}^{f} w_l = 1$. $Ag_{k=1}^{m} q_{j,l}^{k}$ refers to the aggregated value of the l^{th} QoS property across the m concrete services of the composition, and $Norm(.)$ is the normalization function applied to the aggregated values. This value is computed using formula (3.9) for negative QoS properties and employing formula (3.10) for positive QoS properties.

$$Norm(Ag_{k=1}^{m} q_{j,l}^{k}) = \begin{cases} \frac{Ag_{k=1}^{m} q_j^{k,max} - Ag_{k=1}^{m} q_{j,l}^{k}}{Ag_{k=1}^{m} q_j^{k,max} - Ag_{k=1}^{m} q_j^{k,min}} & \text{if } Ag_{k=1}^{m} q_j^{k,max} - Ag_{k=1}^{m} q_j^{k,min} \neq 0 \ , \\ 1 & \text{if } Ag_{k=1}^{m} q_j^{k,max} - Ag_{k=1}^{m} q_j^{k,min} = 0 \end{cases} \tag{3.9}$$

$$Norm(Ag_{k=1}^{m} q_{j,l}^{k}) = \begin{cases} \frac{Ag_{k=1}^{m} q_{j,l}^{k} - Ag_{k=1}^{m} q_j^{k,max}}{Ag_{k=1}^{m} q_j^{k,max} - Ag_{k=1}^{m} q_j^{k,min}} & \text{if } Ag_{k=1}^{m} q_j^{k,max} - Ag_{k=1}^{m} q_j^{k,min} \neq 0 \ , \\ 1 & \text{if } Ag_{k=1}^{m} q_j^{k,max} - Ag_{k=1}^{m} q_j^{k,min} = 0 \end{cases} \tag{3.10}$$

Note that $Ag_{k=1}^{m} q_j^{k,min}$ and $Ag_{k=1}^{m} q_j^{k,max}$ represent, respectively, the minimum and the maximum aggregated values that the l^{th} QoS property can take among all compositions composed of m the concrete services.

3.2.2 Formalization of the plan-based QoS-aware service composition problem

In the context of service composition, global constraints $GC = (gc_1, ..., gc_l, ..., gc_f)$ are boundaries that the QoS values of a composite service must satisfy. Each constraint gc_l defines a global limit imposed by the user on the value of the l^{th} QoS property of composite service C. The QoS-aware service composition issue can be formulated as an optimization problem decomposed into multiple parallel sub-problems, where the aim is to maximize the

utility value $F(C_{Best})$ in terms of QoS called also composition quality, while meeting the global constraints. Formally:

$$F(C_{Best}) = Max_{p=1}^{NbSubPop}(Max_{i=1}^{SubPopSize_p} F(C_{p,i})) \tag{3.11}$$

where $NbSubPop$ is the number of sub-populations that evolve in parallel, the overall population of compositions is represented as $Pop = \bigcup_{p=1}^{NbSubPop} SubPop_p$, where each sub-population $SubPop_p$ contains initially $SubPopSize_p$ compositions and $C_{p,i}$ refers to the i^{th} composition of the p^{th} sub-population. $F(C_{p,i})$ is the QoS utility value of the composition $C_{p,i}$ calculated as follows:

$$F(C_{p,i}) = \sum_{l=1}^{f} w_l \times Norm(Ag_{k=1}^{m} \sum_{j=1}^{n} q_{j,l}^{k} \times d_j^k) \tag{3.12}$$

where w_l is the user's preferences weight for the l^{th} QoS property, the aggregated QoS $Ag_{k=1}^{m} \sum_{j=1}^{n} q_{j,l}^{k}$ of the composite service $C_{p,i}$ satisfies the global constraint imposed for each QoS property. More precisely, $Ag_{k=1}^{m} \sum_{j=1}^{n} q_{j,l}^{k} \times d_j^k >= gc_l$ if the l^{th} QoS property is positive and $Ag_{k=1}^{m} \sum_{j=1}^{n} q_{j,l}^{k} \times d_j^k <= gc_l$ if the l^{th} QoS property is negative (see [124] for more details). The $d_j^k \in \{0, 1\}$ represents the binary decision variable indicating whether the j^{th} concrete service is selected for the k^{th} abstract service, such as:

$$\sum_{j=1}^{n} d_j^k = 1, \forall j \in [1..n] \tag{3.13}$$

3.2.3 Adaptive Multi-population Differential Evolution method with dynamic population size reduction

The Adaptive Multi-population Differential Evolution (AMDE) optimization method with dynamic population size reduction [85] is an improvement of Differential Evolution (DE) optimization method [125]. The overall idea of the AMDE method consists of dividing a population of individuals into sub-populations that evolve in parallel during a given number of iterations to find the suboptimal solution. More precisely, the individual population is divided into two sub-populations that are improved simultaneously using different evolution processes then combined to form a whole population at the end of each iteration. Furthermore, according to a population size reduction strategy, the population size is readjusted by filtering out the unpromising individuals before starting the next iteration. The sub-populations sizes are equal in the first iteration and recalculated in each iteration according to the number of improved individuals over the previous iterations. The AMDE optimization method is carried out in the following five phases:

3.2.3.1 Phase 1

A population of individuals $Pop = \{X_i\}_{i=1}^{PopSize}$ is randomly generated according to a uniform distribution, where $PopSize$ is the population size. Each individual X_i=($x_j^1, ..., x_j^k, ..., x_j^m$) represents a positions vector, where x_j^k ($1 \leq j \leq n$) refers to the j^{th} position of the i^{th} individual in the k^{th} dimension and m is the variable dimension. The individual having the highest fitness value is considered as the best individual $BestI$. Furthermore, the AMDE method's parameters are initialized in this phase, such as the maximum number of iterations $MaxIt$, the sub-populations sizes $SubPopSize_1$ and $SubPopSize_2$, the memory size $MSize$, the location parameter μ_{SF^2}, the crossover rate CR^2, the location parameter vector $\mu_{SF^1} = (\mu_{SF^1}(1), \cdots, \mu_{SF^1}(h), \cdots, \mu_{SF^1}(MSize))$ and the mean parameter vector $\mu_{CR^1} = (\mu_{CR^1}(1), \cdots, \mu_{CR^1}(h), \cdots, \mu_{CR^1}(MSize))$. The μ_{SF^1}, μ_{CR^1}, μ_{SF^2} and CR^2 are initialized to 0.5.

3.2.3.2 Phase 2

In every iteration, the population Pop is randomly divided into two sub-populations: memory-based sub-population $MBSubPop$ and memoryless sub-population $MLSubPop$. The sub-populations sizes $SubPopSize_1$ and $SubPopSize_2$ are equal in the first iteration, and they are recalculated in each iteration according to a sub-population resizing strategy.

3.2.3.3 Phase 3

In every iteration, the sub-populations $MBSubPop$ and $MLSubPop$ evolve according to a mutation strategy, a crossover strategy, a selection process, and an adaptive parameters tuning mechanism.

Evolution of the sub-population $MBSubPop$**:** in each iteration it and for every individual $X_i(it)$ in the sub-population $MBSubPop$, the *DE/current-to-pbest/1 mutation* strategy is first employed to generate a mutation vector $V_i(it)$ as follows:

$$V_i(it) = X_i(it) + SF_i^1(it)(X_{pbest}(it) - X_i(it)) + SF_i^1(it)(X_{r1}(it) - X_{r2}(it)) \tag{3.14}$$

where $SF_i^1(it)$ represents the scaling factor of the individual $X_i(it)$, $X_{r1}(it)$ is a randomly selected individual from the sub-population $MBSubPop$ and $X_{r2}(it)$ refers to an individual selected randomly from the union of the sub-population $MBSubPop$ and an external archive A (see [126] for more details). The $X_{pbest}(it)$ is an individual selected randomly among the p best individuals in terms of fitness value belonging to the sub-population $MBSubPop$. The number p is calculated as follows:

$$p = SubPopSize_1(it) \cdot rand \tag{3.15}$$

such as *rand* is a random number generated from the interval $[0, 1]$ and $SubPopSize_1(it)$ is the size of the sub-population $MBSubPop$.

A trial vector $U_i(it)$ is then calculated using the *DE binomial crossover* strategy between the original vector $X_i(it)$ and the resulting mutation vector $V_i(it)$. The k^{th} component ($k \in [1, m]$) of this vector is computed as follows:

$$U_i^k(it) = \begin{cases} V_i^k(it) & \text{if } r \leq CR_i^1(it) \text{ or } k = s \text{ ,} \\ X_i^k(it) & \text{otherwise ,} \end{cases} \tag{3.16}$$

where $CR_i^1(it)$ is the crossover rate of the individual $X_i(it)$, r and s are random numbers generated within intervals $[0, 1]$ and $[1, m]$ respectively.

A selection process is finally done to determine the best individual $X_i(it+1)$ to be retained for the next iteration. Accordingly, the fitness value $f(X_i(it))$ of the original vector $X_i(it)$ is compared with the fitness value $f(U_i(it))$ of the trial vector $U_i(it)$ and the individual having the highest fitness value is retained in the sub-population.

$$X_i(it+1) = \begin{cases} U_i(it) & \text{if } f(U_i(it)) > f(X_i(it)) \text{ ,} \\ X_i(it) & \text{otherwise ,} \end{cases} \tag{3.17}$$

The individual $X_i(it+1)$ replaces the best individual $BestI$ when its fitness value $f(X_i(it+1))$ is higher than that of the best individual $f(BestI)$.

Furthermore, the adaptive parameters tuning mechanism aims to calculate the scaling factor $SF_i^1(it)$ and the crossover rate $CR_i^1(it)$ of each individual $X_i(it)$ in every iteration it. More precisely, the scaling factor $SF_i^1(it)$ of the individual $X_i(it)$ is generated according to a cauchy distribution *randc* with a location parameter value $\mu_{SF^1}(h)$ and a scale parameter 0.1. Formally:

$$SF_i^1(it) = randc(\mu_{SF^1}(h), 0.1) \tag{3.18}$$

where h is a random number within the interval $[1, MSize]$ and $MSize$ represents the memory size that is computed using the following formula:

$$MSize = \frac{PopSize(1)}{2} \cdot 0.02. \tag{3.19}$$

Note that the scaling factor $SF_i^1(it)$ is regenerated when its value is lower than 0 and set to 1 in the case where its value is higher than 1.

The crossover rate $CR_i^1(it)$ of the individual $X_i(it)$ is generated according to a normal distribution *randn* with a mean value $\mu_{CR^1}(h)$ and a standard deviation 0.1 as follows:

$$CR_i^1(it) = randn(\mu_{CR^1}(h), 0.1) \tag{3.20}$$

where h is a random number in the interval $[1, MSize]$ and $MSize$ represents the memory size. It should be noted that the crossover rate $CR_i^1(it)$ is truncated to a value in the interval $[0, 1]$.

At the end of each iteration, the location parameter vector μ_{SF^1} and the mean parameter vector μ_{CR^1} are updated according to the historical memory-based process (see [84]).

Evolution of the sub-population $MLSubPop$**:** the *DE/current-to-pbest/1 mutation* strategy is first used in each iteration it to generate a mutation vector $V_i(it)$ for every individual $X_i(it)$ in the sub-population $MLSubPop$. Formally:

$$V_i(it) = X_i(it) + SF_i^2(it)(X_{pbest}(it) - X_i(it)) + SF_i^2(it)(X_{r1}(it) - X_{r2}(it)) \tag{3.21}$$

where $SF_i^2(it)$ refers to the scaling factor of the individual $X_i(it)$, $X_{r1}(it)$ and $X_{r2}(it)$ are individuals selected randomly from the sub-population $MLSubPop$. The X_{pbest} is an individual selected randomly among the best p individuals in terms of fitness value in the sub-population $MLSubPop$. The number p is calculated as follows:

$$p = SubPopSize_2(it) \cdot rand \tag{3.22}$$

where $rand$ is a random number generated within the interval $[0, 1]$ and $SubPopSize_2(it)$ is the size of the sub-population $MLSubPop$.

A trial vector $U_i(it)$ is then computed using the *DE exponential crossover* strategy between the original vector $X_i(it)$ and the obtained mutation vector $V_i(it)$. The k^{th} component $U_i^k(it)$ ($k \in [1, m]$) is computed as follows:

$$U_i^k(it) = \begin{cases} V_i^k(it) & \text{if } r \leq CR_i^2(it) \text{ and } l \leq m \text{ ,} \\ X_i^k(it) & \text{otherwise ,} \end{cases} \tag{3.23}$$

where $CR_i^2(it)$ is the crossover rate of the individual $X_i(it)$, r is a random number generated within the interval $[0, 1]$, l is initially set to 1 and incremented until reaching the dimension m. Moreover, k is a random number generated in the interval $[1, m]$ and incremented as $k = (k + 1)\ mod\ m$.

The selection process employed in the sub-population $MBSubPop$ is finally used to retain the best individuals $X_i(it+1)$ in the sub-population $MLSubPop$ for the evolution in the next iteration. This individual $X_i(it+1)$ substitutes the best one $BestI$ in the case where its fitness value $f(X_i(it + 1))$ is higher than that of the best individual $f(BestI)$.

In addition, the adaptive parameters tuning mechanism intends to compute a scaling factor $SF_i^2(it)$ and a crossover rate $CR_i^2(it)$ for every individual $X_i(it)$ in each iteration it. Indeed, the scaling factor $SF_i^2(it)$ is generated using formula (3.18) with the location

parameter value μ_{SF^2} and updated as follows:

$$SF_i^2(it) = \begin{cases} max(SF_i^2(it), abs(1-\sigma)) & \text{if } \sigma < 1 \ , \\ max(SF_i^2(it), abs(1-\frac{1}{\sigma})) & \text{otherwise} \ , \end{cases} \tag{3.24}$$

$$\sigma = \frac{f(X_{best}(it))}{f(X_{pbest}(it))} \tag{3.25}$$

where $f(X_{best}(it))$ and $f(X_{pbest}(it))$ are the fitness values of the best individual $X_{best}(it)$ in the sub-population $MLSubPop$ and a randomly selected individual $X_{pbest}(it)$ among the p best individuals in terms of fitness value belonging to the sub-population $MLSubPop$.

The crossover rate $CR_i^2(it)$ of the individual $X_i(it)$ is initialized to 0.5 and updated for the next iterations according to the following formula:

$$CR_i^2(it+1) = \frac{\sum_{k=1}^{m} \phi_k}{m} \tag{3.26}$$

$$\phi_k = \begin{cases} 1 & \text{if } U_i^k(it) = V_i^k(it) \\ 0 & \text{if } U_i^k(it) = X_i^k(it) \end{cases} \tag{3.27}$$

where m is the variable dimension and $\sum_{k=1}^{m} \phi_k$ represents the number of variables that were copied from the trial vector when using exponential crossover.

At the end of each iteration, the location parameter value μ_{SF^2} is updated according to the historical memory-based process (see [84]).

3.2.3.4 Phase 4

At the end of each iteration it, the two sub-populations $MBSubPop$ and $MLSubPop$ are combined to form a whole population. In some iterations, a subset of individuals RX belonging to the population do not take part in the evolution process due to the fact that the population size is larger than the sum of the sub-populations' sizes. Accordingly, the whole population is obtained in the iteration it as follows:

$$Pop(it) = MBSubPop \bigcup MLSubPop \bigcup RX \tag{3.28}$$

where RX is an empty set in the first iteration since the sub-populations have the same size. This set evolves according to the sizes of sub-populations in the next iterations.

3.2.3.5 Phase 5

At the end of each iteration it, the population size for the next iteration $PopSize(it+1)$ is first calculated using the *reduction strategy* introduced in [84]. This strategy linearly

decreases the population size as follows:

$$PopSize(it+1) = Round(\frac{S_{min} - S_{max}}{MaxIt} \cdot it \ + \ S_{max}) \tag{3.29}$$

where S_{max} (respectively S_{min}) is the maximum (respectively the minimum) population size, which is set as the initial $PopSize$ (respectively eight).

The worst individuals in terms of fitness value are then discarded from the population. The number of individuals $NbW(it)$ to be discarded is calculated as follows:

$$NbW(it) = PopSize(it) - PopSize(it+1) \tag{3.30}$$

The sub-population size for the next iteration $SubPopSize_y(it+1)$ ($y \in \{1,2\}$) is finally calculated based on the number of individuals that have been improved until the iteration it. Formally:

$$SubPopSize_y(it+1) = \frac{\sum_{t=1}^{it} SR_y(t)}{it} \tag{3.31}$$

where $SR_y(t)$ is the success rate of the y^{th} sub-population calculated as follows:

$$SR_y(t) = \sum_{i=1}^{SubPopSize_y} sr_i(t) \tag{3.32}$$

where $sr_i(t)$ is the success rate of the individual $X_i(t)$ computed as follows :

$$sr_i(t) = \begin{cases} 1 & \text{if } f(U_i(t)) > f(X_i(t)) \ , \\ 0 & \text{otherwise} \ , \end{cases} \tag{3.33}$$

Note that the current population size can be larger than the sum of sub-populations sizes. This means that some individuals in the population do not participate in the evolution process during the next iteration.

3.3 The proposed approach : PDE-QSC algorithm

3.3.1 Main idea of the PDE-QSC approach

In this thesis, a Parallel Differential Evolution-based approach with population size reduction for QoS-aware service Composition PDE-QSC is proposed [93]. This approach exploits the parallel differential evolution (PDE) optimization method [85] to find the suboptimal composition in a non-prohibitive computation time. In the PDE optimization method, the population of individuals is divided in each iteration into two sub-populations that are improved in parallel using different evolution processes and combined at the end of the iteration to form a single population. In addition, a population resizing strategy is used to readjust

the size of each sub-population by filtering out unpromising individuals for the next iteration. The sizes of the sub-populations are equal in the first iteration and then updated in subsequent iterations based on the number of individuals improved in the previous iterations.

In the context of service composition, each individual $X_i = (x_j^1, ..., x_j^k, ..., x_j^m)$ represents a composite service $C_i = (x_{CS_j^1}, ..., x_{CS_j^k}, ..., x_{CS_j^m})$, where $x_{CS_j^k}$ denotes the index of the j^{th} concrete service in the k^{th} abstract service. To compute the quality of the composite service C_i, each index is replaced by its corresponding concrete service and the composition C_i is then expressed as $C_i = Combin_{k=1}^{m} cs_j^k$. Note that the concrete services within each abstract service belonging to the composition are sorted in descending order of their utility values in terms of QoS. Table 3.3 shows the terminology correspondence between the PDE-QSC approach and the QoS-aware service composition problem.

Table 3.3: PDE-QSC approach versus QoS-aware service composition.

PDE-QSC approach	**QoS-aware service composition**
Individual	Composite service
Individual population	Population of composite services
Position	Index of a concrete service in a composition
Fitness value	Utility value in terms of QoS

3.3.2 The phases of the PDE-QSC approach

The PDE-QSC approach operates in five phases: (*i*) initialization, (*ii*) population partitioning, (*iii*) sub-populations evolution, (*iv*) sub-populations recombination and (*v*) population/sub-populations resizing (see Figure 3.3). In the initialization phase, the setting parameters of the algorithm are initialized and the initial population of compositions is randomly generated. The compositions are evaluated according to a utility function to find the best composition in terms of QoS. The population partitioning phase aims at randomly dividing the composition population into two composition sub-populations. In the sub-populations evolution phase, the two composition sub-populations are improved in parallel using different evolution processes and the resulting compositions are retained if they have better utility value in terms of QoS than the old ones. The sub-populations recombination phase aims at merging the retained compositions in each sub-population to form a single population. In the population/sub-populations resizing phase, a linear reduction strategy is employed to reduce the population size of the compositions and the sizes of the new sub-populations are calculated. The second to fifth phases are repeated for a predefined number of iterations to find the suboptimal composition. Figure 3.4 illustrates the flowchart of the PDE-QSC approach.

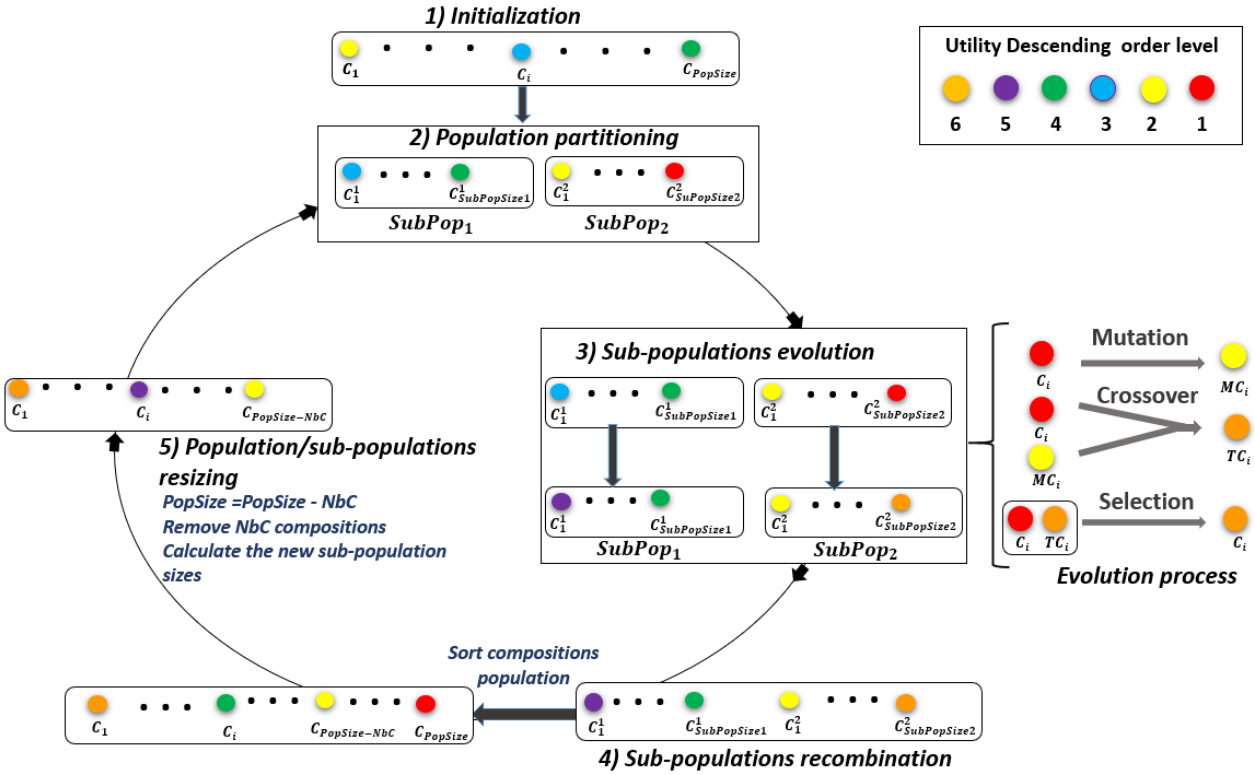


Figure 3.3: PDE-QSC approach phases.

3.3.2.1 Initialization

A population of $PopSize$ feasible compositions (i.e., those satisfying the global QoS constraints) $Pop = \{C_i\}_{i=1}^{PopSize}$ is randomly generated where the best composition $BestC$ is the feasible composition with the highest composition quality. The parameters of the approach are also initialized in this phase: the maximum number of iterations $MaxIt$, the sub-population sizes $SubPopSize_y$ ($y \in \{1, 2\}$), the memory size $Msize$, the location parameter vector $\mu_{SF^1} = (\mu_{SF^1}(1), \cdots, \mu_{SF^1}(h), \cdots, \mu_{SF^1}(MSize))$, the mean parameter vector $\mu_{CR^1} = (\mu_{CR^1}(1), \cdots, \mu_{CR^1}(h), \cdots, \mu_{CR^1}(MSize))$, the location parameter μ_{SF^2}, and the crossover rate CR^2.

3.3.2.2 Population partitioning

At iteration it, the composition population $Pop(it)$ is randomly divided into two sub-populations $SubPop_y(it)$ ($y \in \{1, 2\}$) such as:

$$SubPop_y(it) = \{C_i^y(it)\}_{i=1}^{SubPopSize_y(it)} \tag{3.34}$$

where $C_i^y(it)$ represents the i^{th} composite service in the y^{th} sub-population whose size is set to $\frac{PopSize(it)}{2}$ at the first iteration and recomputed at next iterations according to a sub-populations resizing strategy.

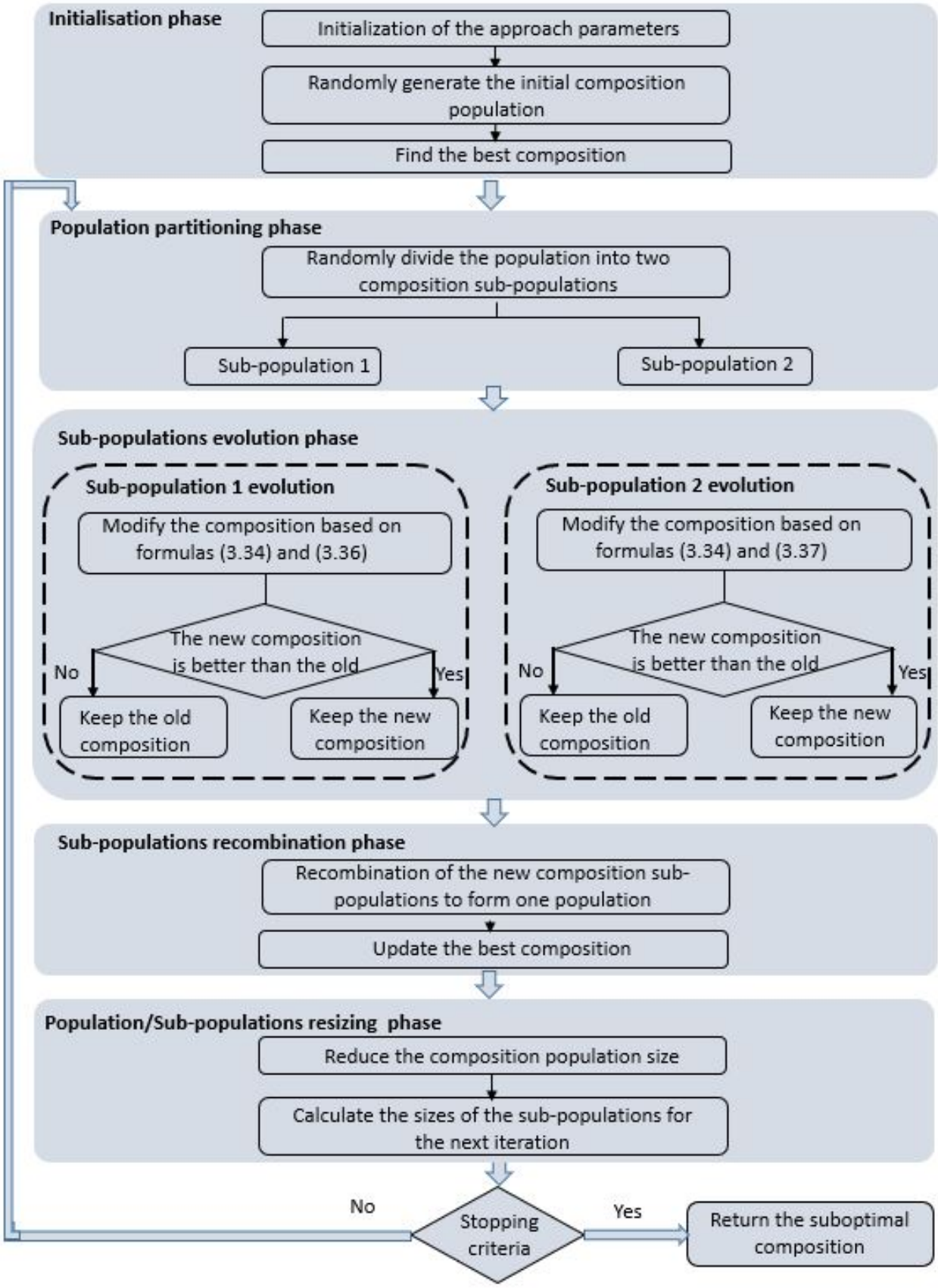


Figure 3.4: PDE-QSC approach flowchart.

3.3.2.3 Sub-populations evolution

Each composition sub-population $SubPop_y$ ($y \in \{1, 2\}$) goes through an evolution process including a mutation strategy, a crossover strategy, a selection process, and an adaptive parameter tuning mechanism.

Mutation strategy: at iteration it and for each composite service $C_i^y(it)$ of the sub-population $SubPop_y(it)$, a mutation composition $MC_i^y(it)$ is generated using the *DE/current-to-pbest/1 mutation* strategy:

$$MC_i^y(it) = C_i^y(it) + SF_i^y(it)(C_{pbest}^y(it) - C_i^y(it)) + SF_i^y(it)(C_{r1}^y(it) - C_{r2}^y(it)) \tag{3.35}$$

where $SF_i^y(it)$ is the scaling factor of the composite service $C_i^y(it)$ computed according to the adaptive parameter tuning mechanism and $C_{r1}^y(it)$ denotes a randomly selected composition from the sub-population $SubPop_y(it)$. The composite service $C_{r2}^y(it)$ is randomly chosen from (a) the union of the sub-population $SubPop_1(it)$ and an external archive A (see [126]) in the case of the first sub-population and (b) the sub-population $SubPop_2(it)$ in the case of the second sub-population. $C_{pbest}^y(it)$ denotes a composite service randomly selected among the p compositions of sub-population $SubPop_y(it)$ having the highest QoS utility value where p is calculated as follows:

$$p = \frac{PopSize(it)}{2} \cdot rand \tag{3.36}$$

where $rand$ is a randomly generated number in the interval $[0, 1]$ and $PopSize(it)$ is the size of the population at iteration it.

Crossover strategy: at iteration it, a trial composite service $TC_i^y(it)$ is generated for each resulting mutation composite service $MC_i^y(it)$ using two crossover strategies.

The *DE binomial crossover* strategy is applied to the sub-population $SubPop_1(it)$ using the following formula:

$$TC_i^{1k}(it) = \begin{cases} MC_i^{1k}(it) & \text{if } r \leq CR_i^1(it) \text{ or } k = s \ , \\ C_i^{1k}(it) & \text{otherwise} \end{cases} \tag{3.37}$$

where $CR_i^1(it)$ is the crossover rate of the composition $C_i^1(it)$ calculated according to the adaptive parameter tuning mechanism. The numbers r and s are randomly generated in the intervals $[0, 1]$ and $[1, m]$, respectively where m represents the number of abstract services in the composition plan. The number k is randomly generated in the interval $[1, m]$ and incremented as $k = (k + 1)\ mod\ m$.

The *DE exponential crossover* strategy is applied to the sub-population $SubPop_2(it)$ using the following formula:

$$TC_i^{2k}(it) = \begin{cases} MC_i^{2k}(it) & \text{if } r \leq CR_i^2(it) \text{ and } l \leq m \text{ ,} \\ C_i^{2k}(it) & \text{otherwise ,} \end{cases} \tag{3.38}$$

where $CR_i^2(it)$ is the crossover rate of the composition $C_i^2(it)$ calculated based on the adaptive parameter tuning mechanism, r is a random number within the interval $[0, 1]$, and l is initialized to 1 and incremented until reaching the number of abstract services m.

Selection process: At iteration it, the utility value $F(C_i^y(it))$ of the original composition $C_i^y(it)$ is compared with the utility value $F(TC_i^y(it))$ of the trial composition $TC_i^y(it)$ in the sub-population $SubPop_y$(it). The composition with the highest utility value $C_i^y(it+1)$ is then kept for the evolution process at the next iteration. Formally:

$$C_i^y(it+1) = \begin{cases} TC_i^y(it) & \text{if } F(TC_i^y(it)) > F(C_i^y(it)) \text{ ,} \\ C_i^y(it) & \text{otherwise} \end{cases} \tag{3.39}$$

The composition $C_i^y(it+1)$ replaces the best composition $BestC$ when its utility value $F(C_i^y(it+1))$ is higher than that of the best composition $F(BestC)$ and the global QoS constraints GC are satisfied for all QoS properties.

Adaptive parameter tuning mechanism: this mechanism aims to compute the scaling factor $SF_i^1(it)$ (respectively $SF_i^2(it)$) and the crossover rate $CR_i^1(it)$ (respectively $CR_i^2(it)$) corresponding to the composition $C_i^1(it)$ (respectively $C_i^2(it)$) in the sub-population $SubPop_1(it)$ (respectively $SubPop_2(it)$).

$SubPop_1$ **adaptive parameter tuning mechanism:** the scaling factor $SF_i^1(it)$ of the composition $C_i^1(it)$ is generated according to a Cauchy distribution $randc$ with a location parameter value $\mu_{SF^1}(h)$ and a scale parameter 0.1. Formally:

$$SF_i^1(it) = randc(\mu_{SF^1}(h), 0.1) \tag{3.40}$$

where h is a number randomly selected in the interval $[1, MSize]$ and $MSize$ denotes the memory size computed as follows:

$$MSize = \frac{PopSize(1)}{2} \cdot 0.02 \tag{3.41}$$

where $PopSize(1)$ is the population size at the first iteration. Note that the scaling factor $SF_i^1(it)$ is regenerated when its value becomes lower than 0 and set to 1 in the case where its value is greater than 1.

The crossover rate $CR_i^1(it)$ of the composition $C_i^1(it)$ is generated according to a normal distribution $randn$ with a mean value $\mu_{CR^1}(h)$ and a standard deviation 0.1. Formally:

$$CR_i^1(it) = randn(\mu_{CR^1}(h), 0.1) \tag{3.42}$$

Note that the crossover rate $CR_i^1(it)$ is truncated to a value in the interval $[0, 1]$.

At the end of iteration it, the location parameter vector μ_{SF^1} and the mean parameter vector μ_{CR^1} are updated according to the historical memory-based process used in [84].

$SubPop_2$ **adaptive parameter tunings mechanism:** the scaling factor $SF_i^2(it)$ is generated using formula (4.10) with the location parameter value μ_{SF^2} and updated as follows:

$$SF_i^2(it) = \begin{cases} max(SF_i^2(it), abs(1-\sigma)) & \text{if } \sigma < 1 \\ max(SF_i^2(it), abs(1-\frac{1}{\sigma})) & \text{otherwise} \end{cases} \tag{3.43}$$

$$\sigma = \frac{F(BestC^2(it))}{F(C_{pbest}^2(it))} \tag{3.44}$$

where $F(BestC^2(it))$ and $F(C_{pbest}^2(it))$ are the utility values of the best composition $BestC^2(it)$ in the sub-population $SubPop_2$ and a randomly selected composition C_{pbest}^2 among the p best compositions of the sub-population $SubPop_2$, respectively.

The crossover rate $CR_i^2(it)$ of the composition $C_i^2(it)$ is initialized to 0.5 and then updated in the next iteration according to the following formula:

$$CR_i^2(it+1) = \frac{\sum_{k=1}^{m} \phi_k}{m} \tag{3.45}$$

$$\phi_k = \begin{cases} 1 & \text{if } TC_i^{2^k}(it) = MC_i^{2^k}(it) \\ 0 & \text{if } TC_i^{2^k}(it) = C_i^{2^k}(it) \end{cases} \tag{3.46}$$

where $\sum_{k=1}^{m} \phi_k$ represents the number of concrete service indices copied from the trial composition when using exponential crossover.

At the end of iteration it, the location parameter value μ_{SF^2} is updated according to the historical memory-based process used in [84].

Algorithms 3.1 and 3.2 summarize the evolution steps of the sub-populations $SubPop_1(it)$ and $SubPop_2(it)$, respectively. Note that the success rate $SR_y(it)$ of sub-population $SubPop_y(it)$ and the success rate $sr_i(it)$ of composition $C_i^y(it)$ $(y \in \{1, 2\})$ are calculated using formulas (3.51) and (3.52), which are detailed later in this chapter.

3.3.2.4 Sub-populations recombination

The two composition sub-populations $SubPop_1(it)$ and $SubPop_2(it)$ are combined at the end of iteration it to form a single population. In some iterations, a subset of compositions SC do not take part of the evolution process due to the fact that the population size is greater

Algorithm 3.1 Evolution steps of the sub-population $SubPop_1(it)$ in the PDE-QSC approach

Inputs: $SubPop_1(it)$, μ_{SF^1}, μ_{CR^1}, $Msize$, GC.

Begin
for each composite service $C_i^1(it) \in SubPop_1(it)$ **do**
  Generate $SF_i^1(it)$ using formula (3.40).
  Apply DE/current-to-pbest/1 mutation strategy using formula (3.35).
  Generate $CR_i^1(it)$ using formula (3.42).
  Apply the DE binomial crossover strategy using formula (3.37).
  Apply the selection process using formula (3.39).
  Calculate the success rate $sr_i(it)$ of the composite service $C_i^1(it)$ using formula (3.52).
  if $F(C_i^1(it)) > F(BestC)$ and $C_i^1(it)$ satisfies GC **then**
    $C_i^1(it)$ replaces $BestC$.
  end if
end for
Update μ_{SF^1} and μ_{CR^1}.
Calculate the success rate $SR_1(it)$ of the sub-population $SubPop_1(it)$ using formula (3.51).
End

Outputs: A new composition sub-population, $SR_1(it)$ and a new $BestC$.

Algorithm 3.2 Evolution steps of the sub-population $SubPop_2(it)$ in the PDE-QSC approach

Inputs: $SubPop_2(it)$, μ_{SF2}, CR^2, GC.

Begin
for each composite service $C_i^2(it) \in SubPop_2(it)$ **do**
  Generate $SF_i^2(it)$ using formula (3.40).
  Update $SF_i^2(it)$ using formulas (3.43) and (3.44).
  Apply DE/current-to-pbest/1 mutation strategy using formula (3.35).
  Apply the DE exponential crossover strategy using formula (3.38).
  Apply the selection process using formula (3.39).
  Calculate the success rate $sr_i(it)$ of the composite service $C_i^1(it)$ using formula (3.52).
  Generate $CR_i^2(it+1)$ using formula (3.45) and (3.46).
  if $F(C_i^2(it)) > F(BestC)$ and $C_i^2(it)$ satisfies GC **then**
    $C_i^2(it)$ replaces $BestC$.
  end if
end for
Update μ_{SF^2}.
Calculate the success rate $SR_2(it)$ of the sub-population $SubPop_1(it)$ using (3.51).
End

Outputs: A new composition sub-population, $SR_2(it)$ and a new $BestC$.

than the sum of the composition sub-populations sizes. The resulting population obtained at iteration it is as follows:

$$Pop(it) = SubPop_1 \bigcup SubPop_2 \bigcup SC \tag{3.47}$$

where SC is an empty set at the first iteration since the composition sub-populations have the same size. This set is updated in the next iterations based on the sizes of sub-populations.

This combination allows for interaction between the compositions of the two sub-populations, which preserves the diversity of the population and improves the quality of the composition.

3.3.2.5 Population/ sub-population resizing

The PDE-QSC approach uses the linear reduction strategy introduced and validated experimentally in [84] to dynamically decrease the composition population size. The aim is to prevent an increase in the composition time generated by the exploration of unpromising compositions. More specifically, the population size $PopSize(it+1)$ at the next iteration is first computed as follows:

$$PopSize(it+1) = Round(\frac{S_{min} - S_{max}}{MaxIt} \cdot it \ + \ S_{max}) \tag{3.48}$$

where S_{max} (respectively S_{min}) denotes the maximum (respectively the minimum) population size, which is set as the initial $PopSize$ (respectively eight).

The compositions of the population are then sorted according to their QoS utility values and the $NbC(it)$ compositions with the lowest QoS utility value are removed from the population. The number of compositions $NbC(it)$ to be removed from the population is calculated as follows:

$$NbC(it) = PopSize(it) - PopSize(it+1) \tag{3.49}$$

At iteration $(it+1)$, a new sub-population size $SubPopSize_y(it+1)$ is calculated for each composition sub-population $SubPop_y(it)$, where $y \in \{1, 2\}$. This size is computed based on the cumulative number of improved compositions up to iteration it. According to this process, the PDE-QSC approach is able to explore a subset of compositions among a reduced population size, thus significantly decreasing the overall composition time. Formally:

$$SubPopSize_y(it+1) = \frac{\sum_{t=1}^{it} SR_y(t)}{it} \tag{3.50}$$

where $SR_y(t)$, denoting the success rate of the y^{th} composition sub-population, is calculated as follows:

$$SR_y(t) = \sum_{i=1}^{SubPopSize_y} sr_i(t) \tag{3.51}$$

where the success rate $sr_i(t)$ of the composition $C_i^y(t)$ is calculated as follows:

$$sr_i(t) = \begin{cases} 1 & \text{if } F(TC_i^y(t)) > F(C_i^y(t)) \ , \\ 0 & \text{otherwise} \ , \end{cases} \tag{3.52}$$

The resulting size of the compositions population may be greater than the sum of sub-populations sizes, meaning that some compositions will not participate in the evolution process at the next iteration. The population/sub-population resizing strategy used in the PDE-QSC approach is summarized in Algorithm 3.3.

Algorithm 3.3 Population/sub-population resizing strategy in the PDE-QSC approach.

Inputs: Pop, $Popsize$, S_{min}, S_{max}, it. $SR_y(y \in \{1,2\})$

1: **Begin**
2: Define a new population size using formula (3.48).
3: Calculate the number of compositions NbC using formula (3.49).
4: Sort the compositions based on their quality values.
5: Remove the NbC worst compositions from the population.
6: Calculate the new sub-population sizes $SubPopSize_y(it+1)$ $(y \in \{1,2\})$ at the next iteration using formulas (3.50)–(3.52).
7: **End**

Outputs: New composition population with a reduced size.

The PDE-QSC approach stops the evolution process when the maximal number of iterations $MaxIt$ is reached and then returns the composition with the highest QoS utility value that satisfies the global constraints as the suboptimal composition. The pseudo-code of the PDE-QSC approach is given in Algorithm 3.4.

Algorithm 3.4 PDE-QSC approach.

Inputs: Pop, $Popsize$, μ_{SF^1}, μ_{SF^1}, $Msize$, μ_{SF^2}, CR^2, $BestC$, $MaxIt$.

1: **Begin**
2: **while** $it < MaxIt$ **do**
3: Do the population partitioning using formula (3.34).
4: **for** each sub-Population $SubPop_y(it) \in Pop(it)$ **do**
5: **if** y=1 **then**
6: Perform the evolution process of the sub-population $SubPop_1(it)$ using Algorithm 3.1.
7: **else**
8: Perform the evolution process of the sub-population $SubPop_2(it)$ using Algorithm 3.2.
9: **end if**
10: **end for**
11: Perform the sub-populations recombination using (3.47).
12: Reduce the population size and calculate the new sub-population sizes using Algorithm 3.3.
13: **end while**
14: **End**

Outputs: Composition with suboptimal QoS.

3.4 Performance evaluation

Different simulation scenarios were conducted to assess the performance of the PDE-QSC approach using the real-world dataset WSC-2009 [117]. The scenarios were implemented using MATLAB R2014b on a machine running a 64-bit Windows operating system, 8 GB RAM, and an Intel Core i5 - 8250U CPU clocked at 2 GHz. The concrete services extracted from the WSC-2009 dataset to instantiate each abstract service belonging to the composition plan are assumed to be semantically equivalent. The WSC-2009 dataset includes five repositories containing 572, 4129, 8138, 8301, and 15211 concrete services. These repositories are merged to form a single repository with 36351 concrete services, each one is described by five QoS properties: response time, throughput, availability, reliability, and cost. As shown in [16], any non-sequential composition plan including loop, sequential, parallel and conditional structures can be transformed into a sequential composition structure. Therefore, for simplicity, the composition plans considered in the performance evaluation of the PDE-QSC approach are assumed to follow a sequential structure.

3.4.1 Baseline methods and performance metrics

The PDE-QSC approach performance is compared with those of five among the most relevant and recent service composition approaches: the hybrid shuffled frog-leaping-based service composition (HSFL-QSC) approach [53], the multi-population self-adaptive differential artificial bee colony for service composition (MSDABC-QSC) [48], the multi-population genetic algorithm-based service composition (MGA-QSC) approach [52], the improved invasive weed-based service composition (IIW-QSC) approach [37], and the novel bat algorithm for qos-aware service composition (NBA-QSC) [46]. Table 3.4 summarizes the baseline approaches and highlights the motivations for their inclusion in the evaluation.

Regarding performance evaluation, the following metrics are considered to assess and compare the effectiveness of the proposed PDE-QSC approach against the baseline approaches:

- *Composition time*: refers to the computation time required to obtain the composition with the suboptimal QoS utility.
- *Composition quality*: represents the QoS utility value of the best composite service calculated as the weighted sum of the QoS property values, (see formula (3.8)).

3.4.2 Simulation parameters

Three service composition scenarios are considered in this evaluation. In scenario 1, the objective is to identify the population size offering a good trade-off between composition quality and composition time, while scenarios 2 and 3 aim to assess the PDE-QSC approach

Table 3.4: Description of baseline approaches and the motivations of their selection.

Approach	Description	Motivation
HSFL-QSC [53]	The population is divided into sub-populations where the compositions evolve independently by exploiting the best and worst compositions through mutation and crossover operators. The composition sub-populations are then shuffled to enable information sharing.	Uses a parallel exploration method with a combination of sub-populations at each iteration, but without reducing the population size. This allows the assessment of the impact of population size reduction on composition time and composition quality.
MSDABC-QSC [48]	Four sub-populations evolve in parallel, each applying a specific differential evolution strategy during the employed bees, onlookers, and scouts phases. The parameters and size of the sub-populations are adjusted dynamically according to their performance. The composition sub-populations exchange information through a cyclic recombination process.	Exploits a method similar to that used by the PDE-QCS approach (parallel exploration, sub-populations combination, and adaptive adjustment), but does not employ a population size reduction method. This allows demonstrating the advantages of the exploration method used by the PDE-QCS approach and the impact of the population size reduction.
MGA-QSC [52]	The compositions are first encoded in chromosome form. This population is divided into several sub-populations, which evolve in parallel. The composition sub-populations are updated by combining roulette selection, genetic operators, and elitist selection. The evolution of each sub-population is repeated until reaching a maximum number of iterations.	The sub-populations evolve independently without being iteratively combined. This allows for illustrating the limitation of parallel exploration-based approaches without interaction between composition sub-populations.
IIW-QSC [37]	The IIW-QS approach combines the invasive weed optimization method and statistical filtering in the composition process. A filtering process is first applied to rank concrete services based on statistical utility values. An improved invasive weed optimization algorithm is then employed to improve the compositions using a sequential exploration-based strategy and standard deviation adjustments.	Represents sequential exploration-based approaches with a fixed-size population. This allows to highlight the benefits of the parallel exploration-based strategy and the dynamic population size reduction used by the PDE-QSC in terms of composition time and composition quality.
NBA-QSC [46]	The composition population first evolves according to the quantum or mechanical mechanism, depending on a stochastic decision process. The evolved compositions are then updated based on a local search mechanism. An adaptive learning mechanism is finally implemented to avoid local optima.	The comparison with the NBA-QC approach allows to demonstrate the efficiency of the PDE-QSC approach compared to recent sequential exploration-based methods in terms of trade-off between composition time and composition quality

compared to the five baseline approaches in terms of composition time and composition quality. Note that the values of the parameters μ_{SF^1}, μ_{CR^1}, μ_{SF^2}, CR^2, $SubPopSize$, and $MSize$ were set to the values defined in [85] where several experiments were conducted to find the most appropriate parameter values to achieve high performance. The number and size of sub-populations for the baseline approaches are either set based on the values given in their

original optimization method or equal to the values considered in the PDE-QSC approach when these values are not given. Table 3.5 shows the parameters common to all the baseline approaches along with the parameters specific to each of them. It should be noted that the values of the parameters specific to each baseline approach are set to the values defined in their original methods [37, 46, 48, 52, 53, 85].

Table 3.5: Parameters setting of the simulation scenarios.

Common parameters			
Parameter	**Scenario 1**	**Scenario 2**	**Scenario 3**
Number of concrete services	$300 - 1500$	300	$1000 - 7000$
Number of abstract services	5	$6 - 30$	5
Population size	$50 - 200$	50	50
Number of iterations	100		
Number of simulations	50		
Specific parameters			
PDE-QSC [85]	$\mu_{SF^1} = 0.5$, $\mu_{CR^1} = 0.5$, $\mu_{SF^2} = 0.5$, $CR^2 = 0.5$, Sub-population size $= PopSize/2$, $S_{min} = 8$, $S_{max} = PopSize$, and $MSize = PopSize(1)/2 \cdot 0.02$.		
HSFl-QSC [53]	$SubPopSize = PopSize/2$.		
MSDABC-QSC [48]	The mean value $\mu_{CR} = 0.5$, the location parameter $\mu_F = 0.5$, and the crossover probability $S_F = 0.5$.		
MGA-QSC [52]	The crossover probability $P_C = 0.9$ and the mutation probability $P_m = 0.1$		
IIW-QSC [37]	The minimum value of seeds $S_{min} = 1$, the maximum value of seeds $S_{max} = 15$, the initial standard deviation $SD_{initial} = 10$, and the final standard deviation $SD_{final} = 0.5$.		
NBA-QSC [46]	The probability of habitat selection $p \in [0.1, 0.9]$, the maximum inertia weight $w \in [0.4, 0.9]$, the compensation rates for Doppler effect in echoes $c \in [0.1, 0.9]$, the contraction–expansion coefficient, $\theta \in [0.5, 1]$, the frequency of updating the loudness and pulse emission rate $G = 10$, the loudness $A_0 \in [0, 2]$, pulse emission rate $r_0 \in [0, 1]$, and the frequency $f \in [0, 2]$.		

3.4.3 Comparison and discussion

The performance of the approaches corresponds to the average of 50 simulations carried out for each service composition scenario.

3.4.3.1 Scenario 1: Population size variation

The objective of this scenario is to investigate the impact of the population size on the composition quality and composition time of the different approaches. Figure 3.5 shows that for all approaches, the composition quality increases slightly with the population size

regardless of the number of concrete services. This is because a large population allows a more in-depth exploration of the search space, resulting in a higher composition quality. For example, in the case of a population of 50 compositions and 900 concrete services, the composition quality values achieved by the PDE-QSC, HSFL-QSC, MSDABC-QSC, MGA-QSC, IIW-QSC, and NBA-QSC approaches are 0.74, 0.65, 0.52, 0.52, 0.51, and 0.51, respectively. However, when the population size is set to 200 compositions and for the same number of concrete services, the composition quality values obtained with the PDE-QSC, HSFL-QSC, MSDABC-QSC, MGA-QSC, IIW-QSC, and NBA-QSC approaches are 0.74, 0.66, 0.54, 0.55, 0.55, and 0.55, respectively.

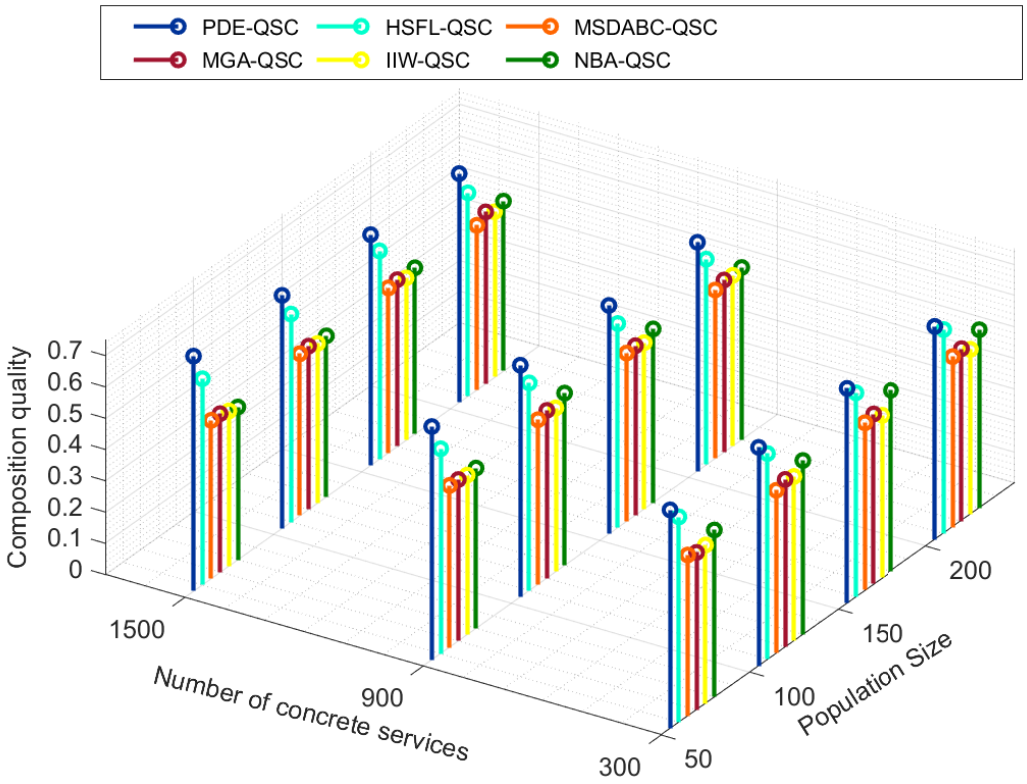


Figure 3.5: Composition quality versus population size for different numbers of concrete services.

Figure 3.6 shows that the composition times of the five approaches significantly increase with the population size, regardless of the number of concrete services. This is because as the population size increases, more execution time is required to explore additional compositions in the population, resulting in an increase in composition time. For instance, when the population size is set to 50 and the number of concrete services to 900, the composition times achieved by the PDE-QSC, HSFL-QSC, MSDABC-QSC, MGA-QSC, IIW-QSC, and NBA-QSC approaches are 140, 68, 1998, 107, 3344, and 230 ms, respectively. For the same number of concrete services and a population of 200 compositions, the composition times obtained with the PDE-QSC, HSFL-QSC, MSDABC-QSC, MGA-QSC, IIW-QSC, and NBA-

QSC approaches are 215, 88, 16595, 590, 13778, and 1044 ms, respectively.

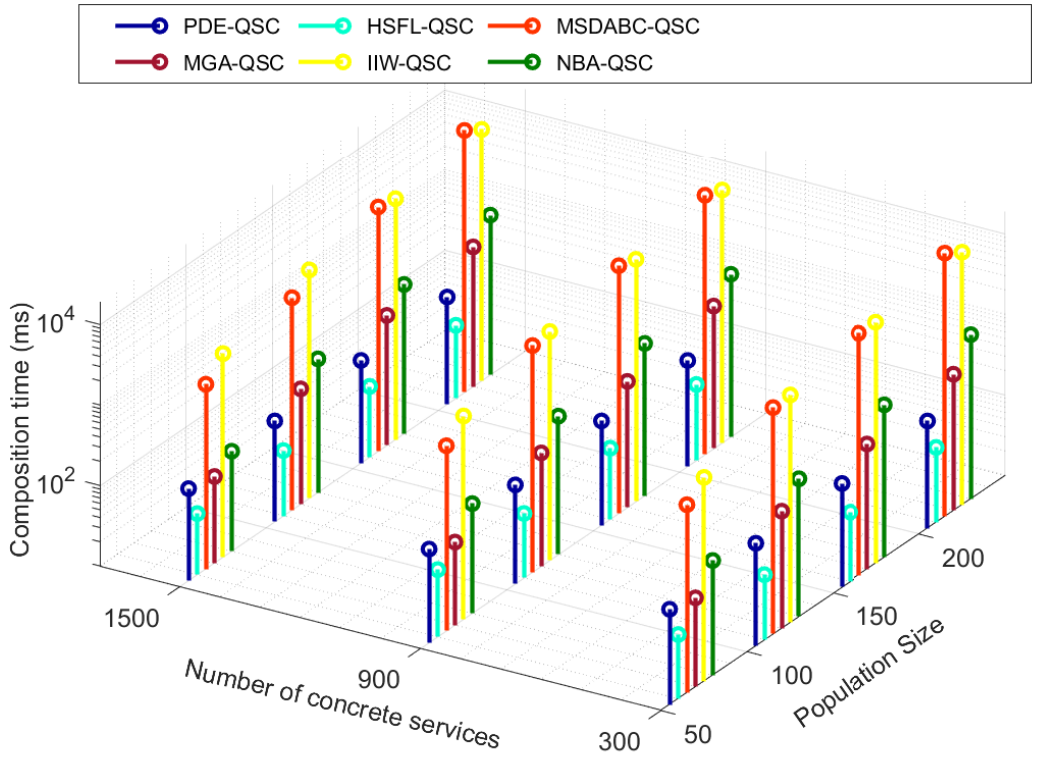


Figure 3.6: Composition time versus population size for different numbers of concrete services.

These simulations show that an increase in population size leads to both a high increase in composition time and a slight improvement in composition quality. However, the improvement in terms of composition quality with a large population size is not significant compared to the large increase in composition time. Based on this analysis, we conclude that 50 compositions is the appropriate population size for a good trade-off between composition time and composition quality.

3.4.3.2 Scenario 2: Variation of the number of concrete services

In this scenario, we analyze how the composition quality and composition time evolve with the number of concrete services for the different approaches. Figure 3.7 clearly shows that the PDE-QSC approach is more efficient than the HSFL-QSC, MSDABC-QSC, MGA-QSC, IIW-QSC, and NBA-QSC approaches in terms of composition quality. For instance, when the number of concrete services is equal to 7000, the benefit obtained by the PDE-QSC algorithm over the HSFL-QSC, MSDABC-QSC, MGA-QSC, IIW-QSC, and NBA-QSC approaches is 23%, 53%, 56%, 60%, 62%, respectively. This result can be explained by several factors related to the exploration of the composition search space. First, unlike the IIW-QSC and NBA-QSC approaches that use a sequential exploration-based strategy, the PDE-QSC approach employs a parallel exploration strategy that enables the evolution of

diverse sub-populations, preventing local optima and thus increasing the composition quality. Second, combining composition sub-populations enhances inter-population interactions and increases population diversity, which has a positive impact on the quality of the composition. Finally, the PDE-QSC approach employs a population size reduction strategy that retains only the best compositions in terms of QoS. This justifies the benefit of the PDE-QSC approach in terms of composition quality over the parallel exploration-based approaches, such as HSFL-QSC, MSDABC-QSC, and MGA-QSC. Specifically, the HSFL-QSC and MSDABC-QSC approaches rely on a parallel exploration-based strategy without population size reduction, while the MGA-QSC approach suffers from the lack of interactions between its evolved composition sub-populations, resulting in a decrease in compositional quality.

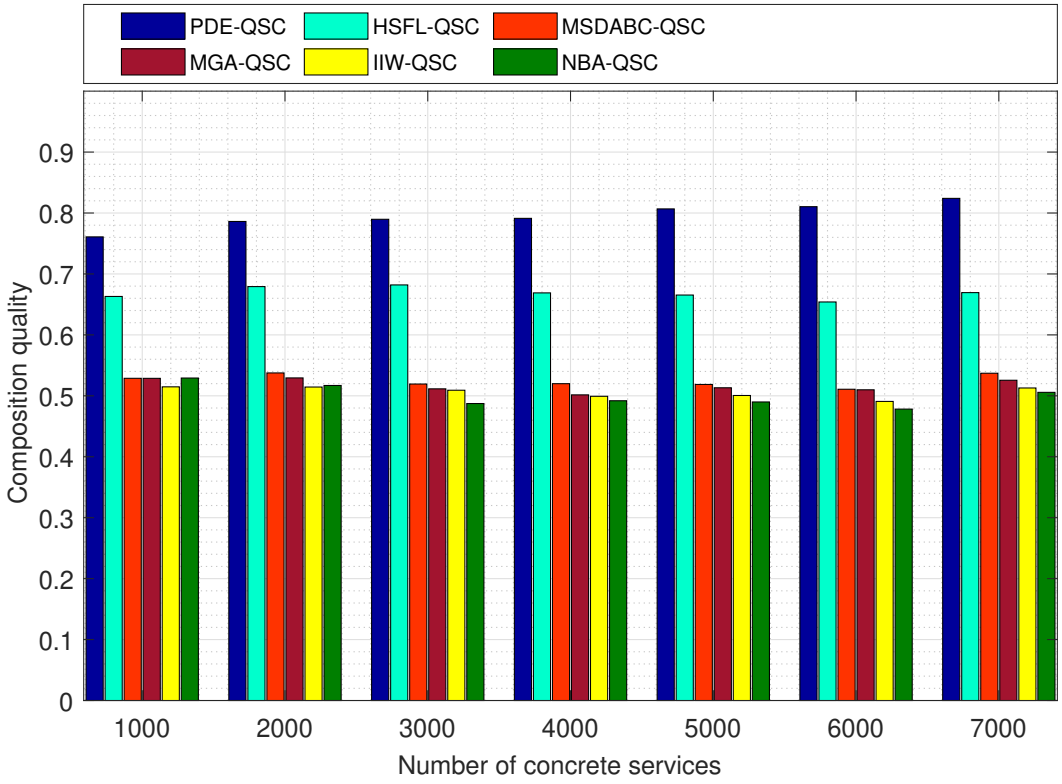


Figure 3.7: Composition quality versus number of concrete services.

As shown in Figure 3.8, the composition time achieved by the PDE-QSC approach is lower than those obtained with the IIW-QSC, NBA-QSC, MSDABC-QSC, and MGA-QSC approaches. In contrast to the IIW-QSC and NBA-QSC approaches that use sequential exploration, the PDE-QSC approach uses a parallel exploration strategy that significantly reduces the composition time. In addition, unlike the IIW-QSC, NBA-QSC, MSDABC-QSC, and MGA-QSC approaches that always explore a fixed number of compositions, the population/sub-populations size reduction strategy adopted in the PDE-QSC approach reduces the number of compositions to be explored, thus decreasing the composition time.

Figure 3.8 also shows that the composition time obtained with the PDE-QSC approach is slightly higher than that obtained with the HSFL-QSC approach, but still of a small order of magnitude (at most 118 milliseconds). This result is due to the fact that the HSFL-QSC approach selects only few compositions for the evolution in each sub-population, which results in a reduction of the composition time compared to the PDE-QSC approach. The latter explores all compositions in each sub-population, which results in a slight increase in the composition time. However, the PDE-QSC approach is more efficient than the HSFL-QSC approach in terms of composition quality and finds a very close-to-optimal composition in terms of QoS (see Figure 3.7). The average composition time taken by the PDE-QSC approach is compatible with composition in large-scale IoT environments as it ranges from 99 to 118 milliseconds for a number of concrete services varying from 1000 to 7000.

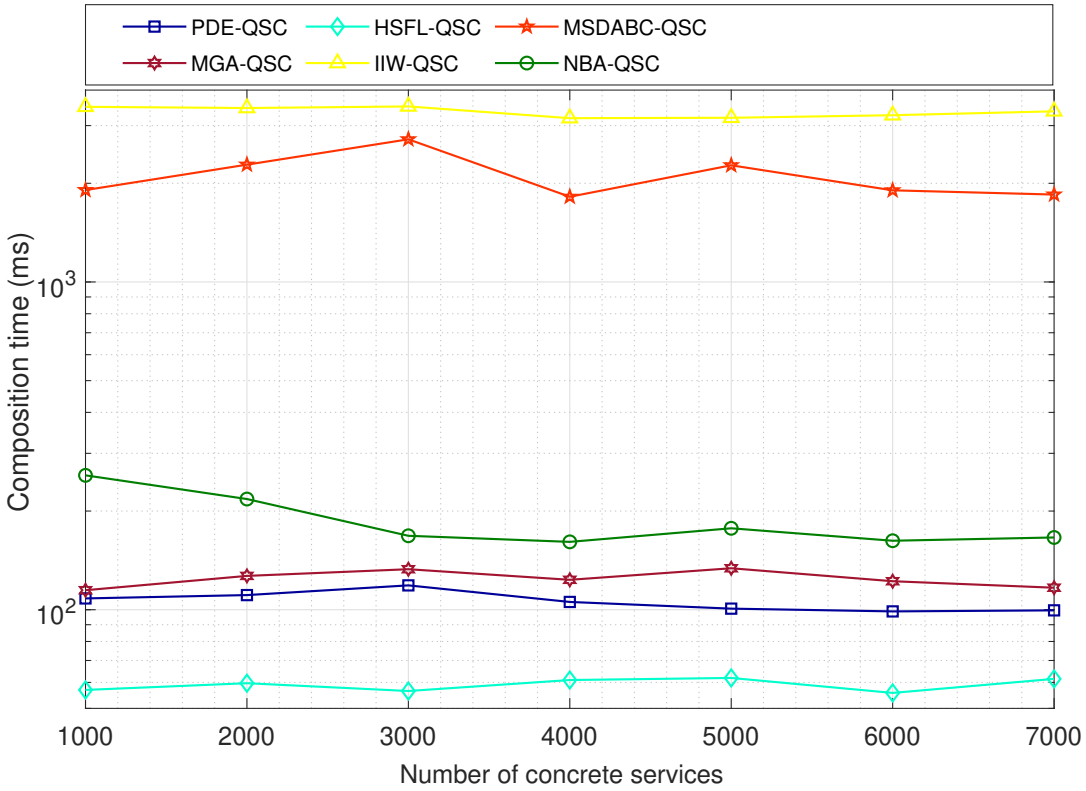


Figure 3.8: Composition time versus number of concrete services.

3.4.3.3 Scenario 3: Variation of the number of abstract services

In this scenario, we evaluate the impact of the number of abstract services on computation time and composition quality for the different approaches. As observed in Figure 3.9, when the number of abstract services increases, the composition quality decreases regardless of the considered approach; however, the PDE-QSC approach remains more efficient than the other baseline approaches. For example, when the number of service classes is set to 12, the performance improvement of the PDE-QSC approach over the HSFl-QSC, MSDABC-

QSC, MGA-QSC, IIW-QSC, and NBA-QSC approaches is about 18%, 54%, 51%, 53%, and 64%, respectively. The composition quality obtained with the PDE-QSC approach is twice as good as that obtained with any of the baseline approaches except the HSFL-QSC algorithm. It should be noted that the PDE-QSC approach remains better than the HSFL-QSC approach in terms of composition quality with a significant improvement regardless of the number of service classes. These results are mainly due to the parallel exploration strategy of composition sub-populations followed by their recombination, as well as to the population size reduction strategy. These strategies allow the PDE-QSC approach to explore in parallel smaller size populations with promising compositions in terms of QoS.

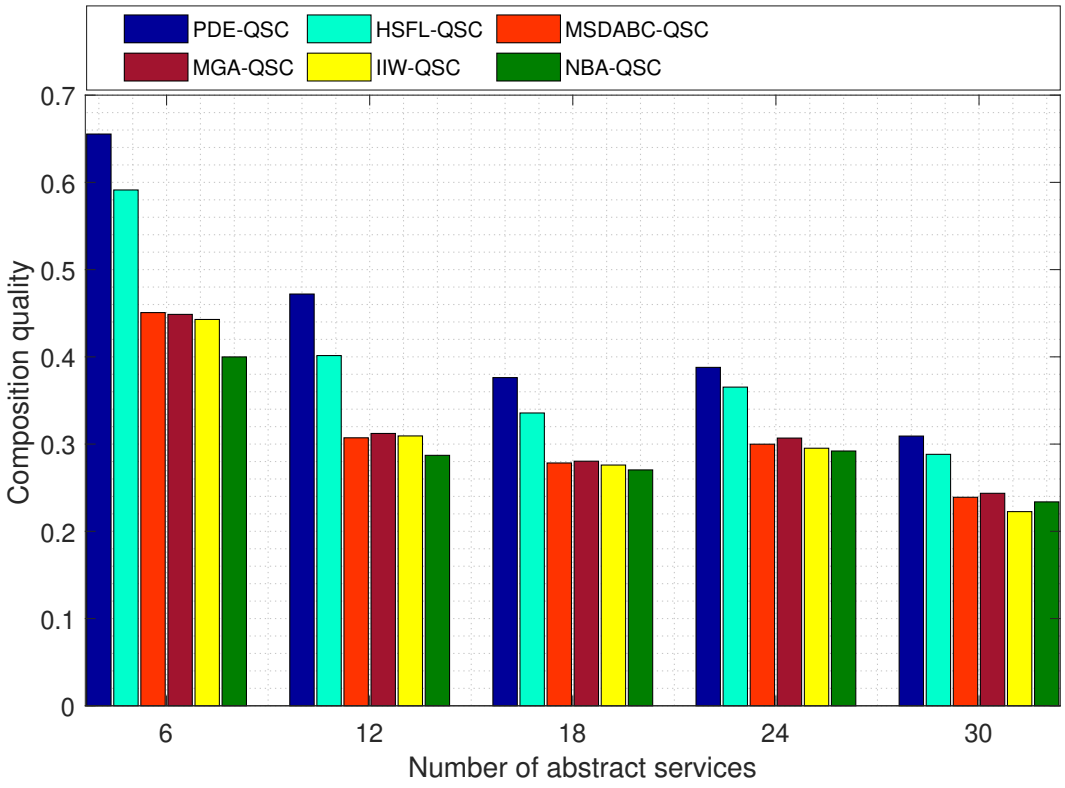


Figure 3.9: Composition quality versus number of abstract services.

Figure 3.10 shows that the composition time obtained with the PDE-QSC approach is much lower than those achieved by the MSDABC-QSC and IIW-QSC approaches. This result is because the PDE-QSC approach explores in parallel populations with reduced sizes, unlike the MSDABC-QSC or IIW-QSC approaches, which explore each time a fixed size population. Additionally, it can be observed that the composition times achieved by the PDE-QSC and MGA-QSC approaches are very close. This result is because the MGA-QSC approach explores the sub-populations of compositions without recombining them and redividing the population; this results in a significant reduction in composition time. Furthermore, the composition time obtained with the PDE-QSC approach is moderately higher than that obtained with the HSFL-QSC and NBA-QSC approaches. This is mainly because

the PDE-QSC approach requires additional computation time to explore all of compositions in each sub-population, leading to a deeper exploration of the composition search space and an increase in the composition time. However, by evolving a fixed subset of compositions (i.e., the number of evolved compositions remains constant and is significantly smaller than the sub-population sizes), the HSFL-QSC approach slightly reduces the composition time compared to the PDE-QSC approach, but results in a decrease in the composition quality in terms of QoS.

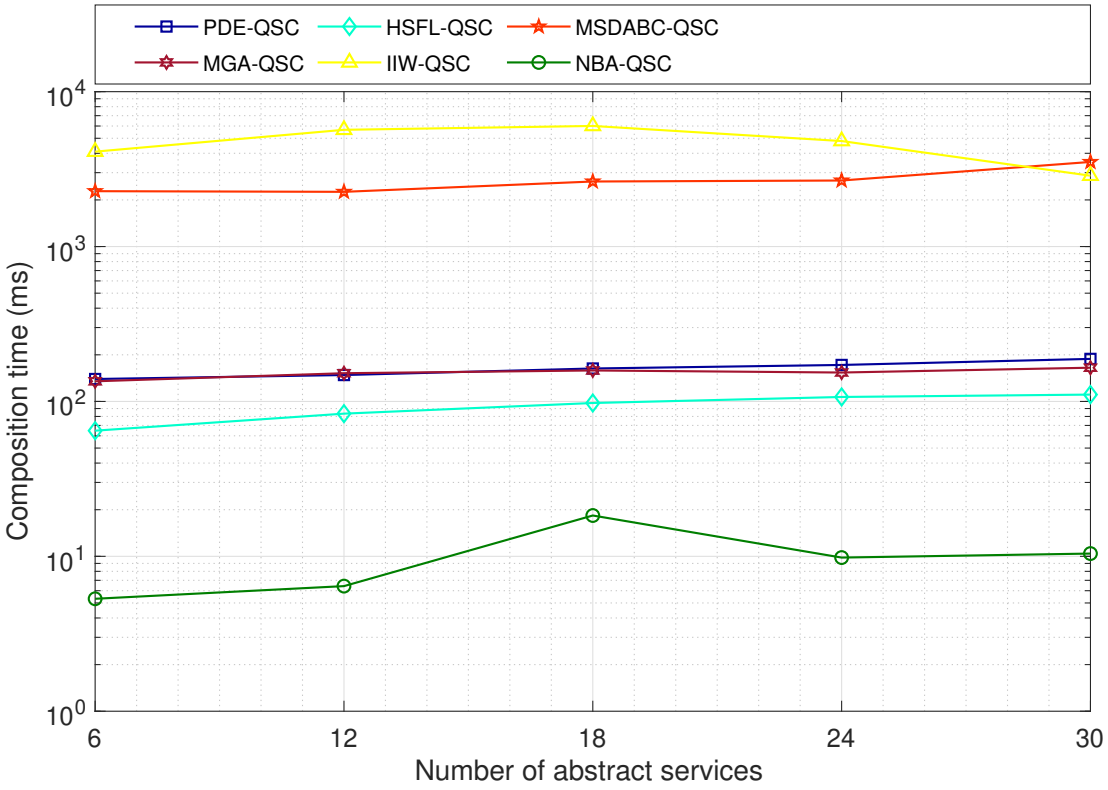


Figure 3.10: Composition time versus number of abstract services.

3.5 Statistical analysis

To ensure reliable statistical conclusions, the average results obtained from 50 simulations for each dataset were recorded for the PDE-QSC approach and all baseline approaches. The statistical significance of the performance differences between the PDE-QSC approach and the HSFL-QSC, MSDABC-QSC, MGA-QSC, IIW-QSC, and NBA-QSC approaches were then assessed using the non-parametric Wilcoxon rank sum test at a 95% confidence level. This test is largely employed in the context of service composition [50, 48, 54, 63] since it is well-suited for comparing computational intelligence-based optimization methods [127]. For each dataset, the approaches are ranked according to their composition quality and composition time, where the best performing approach has rank 1 and the worst performing one has rank

6 (see Tables 3.6 and 3.7). To summarize the comparison results across all datasets, the notation w/l is used in the last rows of the tables, accompanied by the symbols + and – placed next to the rank values. More precisely, for each pairwise comparison between the PDE-QSC approach and a baseline approach, the notation w refers to the number of datasets where the PDE-QSC approach outperforms the baseline approach in terms of composition quality or composition time. In these cases, the symbol + is appended to the rank values to denote a statistical superiority. Conversely, l represents the number of datasets where a baseline approach surpasses the PDE-QSC approach in terms of composition quality or composition time, where the symbol – is added to the rank values to indicate a statistical inferiority.

Table 3.6: Wilcoxon rank sum test on composition quality.

Dataset	**PDE-QSC**	**HSFL-QSC**	**MSDABC-QSC**	**MGA-QSC**	**IIW-QSC**	**NBA-QSC**
Dataset 1: 1000cs_5SC	1	2+	4+	5+	6+	3+
Dataset 2: 2000cs_5SC	1	2+	3+	4+	6+	5+
Dataset 3: 3000cs_5SC	1	2+	3+	4+	5+	6+
Dataset 4: 4000cs_5SC	1	2+	3+	4+	5+	6+
Dataset 5: 5000cs_5SC	1	2+	3+	4+	5+	6+
Dataset 6: 6000cs_5SC	1	2+	3+	4+	5+	6+
Dataset 7: 7000cs_5SC	1	2+	3+	4+	5+	6+
Dataset 8: 300cs_6SC	1	2+	3+	4+	5+	6+
Dataset 9: 300cs_12SC	1	2+	5+	3+	4+	6+
Dataset 10: 300cs_18SC	1	2+	4+	3+	5+	6+
Dataset 11: 300cs_24SC	1	2+	4+	3+	5+	6+
Dataset 12: 300cs_30SC	1	2+	4+	3+	5+	6+
Sum of ranks	12	24	42	45	62	67
Average of ranks	1	2	3.5	3.75	5.16	5.58
Final ranks	1	2	4	3	6	5
w/l		12/0	12/0	12/0	12/0	12/0

Table 3.6 shows that the PDE-QSC approach outperforms all baseline approaches in terms of composition quality. The PDE-QSC approach ranked first across all datasets with an average rank of 1. The result $(w/l) = 12/0$ indicates that the PDE-QSC approach achieved 12 wins and no ties or losses against all the compared approaches, where the HSFL-QSC approach is the best competitor, with an average rank of 2. These results confirm the discussions conducted in subsections 3.4.3.2 and 3.4.3.3. Particularly, the combination of the population size reduction strategy with an efficient parallel exploration-based method

allows the PDE-QSC approach to significantly achieve better composition quality in terms of QoS compared to parallel exploration-based approaches, such as the SFL-QSC, MSDABC-QSC, and MGA-QSC approaches. Table 3.6 also shows that the sequential exploration-based approaches (IIW-QSC and NBA-QSC) rank last with average ranks of 5.16 and 5.58, respectively. This demonstrates that the parallel exploration-based method employed by the PDE-QSC approach is more efficient than sequential exploration-based methods in finding the suboptimal composition.

Table 3.7: Wilcoxon rank sum test on composition time.

Dataset	**PDE-QSC**	**HSFL-QSC**	**MSDABC-QSC**	**MGA-QSC**	**IIW-QSC**	**NBA-QSC**
Dataset 1: 1000cs_5SC	2	1-	5+	3+	6+	4+
Dataset 2: 2000cs_5SC	2	1-	5+	3+	6+	4+
Dataset 3: 3000cs_5SC	2	1-	5+	3+	6+	4+
Dataset 4: 4000cs_5SC	2	1-	5+	3+	6+	4+
Dataset 5: 5000cs_5SC	2	1-	5+	3+	6+	4+
Dataset 6: 6000cs_5SC	2	1-	5+	3+	6+	4+
Dataset 7: 7000cs_5SC	2	1-	5+	3+	6+	4+
Dataset 8: 300cs_6SC	4	2-	5+	3-	6+	1-
Dataset 9: 300cs_12SC	3	2-	5+	4+	6+	1-
Dataset 10: 300cs_18SC	4	2-	5+	3-	6+	1-
Dataset 11: 300cs_24SC	4	2-	5+	3-	6+	1-
Dataset 12: 300cs_30SC	4	2-	6+	3-	5+	1-
Sum of ranks	33	17	61	37	71	33
Average of ranks	2.75	1.42	5.08	3.08	5.91	2.75
Final ranks	2	1	4	3	5	2
w/l		0/12	12/0	8/4	12/0	7/5

Table 3.7 shows that the PDE-QSC approach achieves a high performance in terms of composition time compared to the sequential exploration-based approaches (IIW-QSC and NAB-QSC) with $(w/l) = 12/0$ in all datasets. The comparison results further illustrate that the PEC-QSC approach remains competitive but performs less than some parallel exploration-based approaches for some datasets. More precisely, the PDE-QSC approach wins in 8 datasets and loses in 4 against the MGA-QSC approach with $(w/l) = 8/4$. This relative loss is due to the absence of sub-population recombination in the MGA-QSC approach, leading to a reduction in composition time. Additionally, the PDE-QSC approach wins in 7 datasets and loses in 5 with $(w/l) = 7/5$ compared to the NBA-QSC approach. The performance difference can be attributed to the fact that the NBA-QSC approach has

a rapid convergence to local optima, leading to a reduction in composition time. Furthermore, the PDE-QSC approach achieves the second highest average rank among the baseline approaches, being outperformed only by the HSFL-QSC approach. As explained in subsections 3.4.3.2 and 3.4.3.3, the PDE-QSC approach efficiently explores all compositions in each sub-population to find the composition with the highest quality. This process requires additional computation time compared to the HSFL-QSC approach, which partially explores the composition search space and consequently achieves a shorter composition time.

These statistical analysis confirm the findings reported in subsections 3.4.3.2 and 3.4.3.3, further demonstrating the advantages of the PDE-QSC approach over the baseline approaches. Specifically, the PDE-QSC approach guarantees finding a suboptimal composition with high quality within a reduced composition time when solving the QoS-aware service composition problem.

3.6 Conclusion

The Parallel Differential Evolution approach with population size reduction for QoS-aware service composition (PDE-QSC) proposed in this chapter efficiently addresses the plan-based QoS-aware service composition problem. The approach deals with the limitations of existing sequential and parallel exploration-based service composition approaches that usually suffer from limited composition quality and/or high composition time. The proposed PDE-QSC approach models and solves the QoS-aware service composition problem using a parallel optimization method, achieving high composition quality in a reduced composition time. The simulation scenarios conducted on several real-world datasets demonstrate that the PDE-QSC approach outperforms five baseline approaches in terms of composition quality and composition time. These superior results are mainly attributed to the efficiency of the adopted parallel exploration-based method that accelerates the search process without compromising quality and to the used population size reduction strategy that further decreases the composition time. The statistical significance of these results has been rigorously confirmed using the Wilcoxon rank sum test.

The proposed PDE-QSC approach, like many existing approaches in the literature, relies on the assumption of the prior existence of an abstract composition plan. To overcome this limitation, in the next chapter, we propose an autonomous QoS-aware service composition approach that eliminates the need for a predefined abstract plan and integrates the semantic aspect.

Chapter 4

Database concepts-driven failure recovery approaches for autonomous QoS-aware semantic service composition

Contents

4.1 Introduction

The autonomous QoS-aware service composition seeks to find suboptimal compositions that satisfy the user's functional and non-functional requirements without a prior existence of an abstract composition plan. Although several computational intelligence-based approaches have been proposed to optimize compositions from the service repository, most existing approaches either fail to simultaneously account for QoS and quality of semantic matching (QoSM) or rely on a complex process to ensure semantic matching, which increases computation time. Furthermore, these approaches often require a re-composition process in the case of service failure, further impacting the efficiency and reliability of the service composition.

The second contribution of this thesis introduces two functional dependencies-based failure recovery approaches for autonomous QoS- aware service composition (HCFDSSC and ESFDSSC) leveraging the functional dependency as a key concept from relational database theory. These approaches address the limitations of the existing approaches and overcome the shortcomings of the first contribution (the PDE-QSC approach), especially its reliance on the prior existence of an abstract composition plan and its lack of semantic consideration during the composition process composition plans. In this chapter, we first formalize and introduce the autonomous QoS-aware services composition problem. Then, we present the main idea and how the proposed HCFDSSC and ESFDSSC algorithms work. Finally, we conduct an experimental evaluation to assess the performance and robustness of the proposed approaches.

4.2 Background

4.2.1 Semantic service composition model

4.2.1.1 Semantic concrete service

A concrete service CS is semantically defined by a set of inputs I_{CS}, a set of outputs O_{CS}, and a vector of QoS attributes QoS_{CS}=$\{qos_1, \cdots, qos_k\}$, where each attribute can be positive (e.g., availability) or negative (e.g., cost) [17]. Formally:

$$CS = (I_{CS}, O_{CS}, QoS_{CS}) \tag{4.1}$$

Example. The concrete service *cameraHD* recording real-time data traffic is semantically defined as:
$CameraHD$ =({video feeds, disparity maps },{vehicles per hour/lane, percentage of road space occupied, unusual patterns},(50, 0.02, 99.8, 99.7)), where the considered QoS properties are respectively, latency, cost, reliability, and availability.

4.2.1.2 Semantic abstract service

An abstract service AS is semantically defined as a group of semantic concrete services G_{CS} performing the same functionality. This means that these concrete services share the same set of inputs I_{AS} and have similar outputs O_{AS}, but are different in terms of their QoS attributes. Formally:

$$AS = (I_{AS}, O_{AS}, G_{CS}) \tag{4.2}$$

Example. The traffic monitoring abstract service is semantically defined as:

Traffic monitoring service=({video feeds, disparity maps},{vehicles per hour/lane, percentage of road space occupied, unusual patterns},{cameraHD, sensor, radar}).

4.2.1.3 Semantic abstract composition plan

An abstract composition plan $ACP = (I_{ASP}, O_{ASP})$ is semantically defined as a sequence of semantic abstract services such that the outputs of each abstract service AS serve as inputs for the next one. Formally :

$$ACP = \bigwedge_{e=1}^{nb_{AS}} AS_e : O_{AS_e} \subseteq I_{AS_{e+1}}, \forall e \in [1, nb_{AS}] \tag{4.3}$$

where $\bigwedge$ is an operator that represents the sequence of abstract services in the composition plan ACP and nb_{AS} is the number of abstract services belonging to the semantic abstract composition plan ACP. The operator $\bigwedge$ refers to different composition structures [128].

Example. The abstract composition plan (ACP) for the smart traffic management can be semantically represented as the following sequence of abstract services:

$$ACP = \text{Traffic monitoring service} \xrightarrow[O^3_{TM}=I^3_{TDA}]{O^1_{TM}=I^1_{TDA},\ O^2_{TM}=I^2_{TDA},} \text{Traffic data analysis service} \xrightarrow[O^3_{TDA}=I^3_{N}]{O^1_{TDA}=I^1_{N},\ O^2_{TDA}=I^2_{N},} \text{Notification}$$
$$\xrightarrow{O^1_{N}=I^1_{TSA},\ O^2_{N}=I^2_{TSA},} \text{Traffic signal adjustments service} \xrightarrow{O^1_{TSA}=I^1_{RO},\ O^2_{TSA}=I^2_{RO},} \text{Route optimization service.}$$

The inputs, outputs, and set of concrete services associated with each abstract service belonging to the composition plan are summarized in Table 4.1.

4.2.1.4 Semantic composition

A semantic composition C is a sequence of concrete services such that the inputs of each concrete service semantically match with the outputs of its preceding service. Formally:

$$C = \bigwedge_{g=1}^{nb_{CS}} CS_g : O_{CS_g} \subseteq I_{CS_{g+1}}, \forall g \in [1, nb_{CS}] \tag{4.4}$$

where $\bigwedge$ is an operator representing the sequence of concrete services in the composition C. This sequence can integrate varied composition structures such as conditional, loop, sequential, and parallel [128]. The value nb_{CS} refers to the number of concrete services belonging to the composition C.

Table 4.1: Description of the smart traffic management abstract composition plan.

Abstract service	**Inputs**	**Outputs**	**Concrete services**
Traffic monitoring (TM)	• I^1_{TM}: Video feeds • I^2_{TM}: Disparity maps	• O^1_{TM}: Vehicles per hour/lane • O^2_{TM}: Percentage of road space occupied • O^3_{TM}: Unusual patterns	• CameraHD • Sensor • radar
Traffic data analysis (TDA)	• I^1_{TDA}: Vehicles per hour/lane • I^2_{TDA}: Percentage of road space occupied • I^3_{TDA}: Unusual patterns	• O^1_{TDA}: Congestion levels • O^2_{TDA}: Peak hours • O^3_{TDA}: Accidents detection	• Autonomous traffic signal system • Artificial intelligence-powered video analytics platform • Traffic analyzer system
Notification (N)	• I^1_N: Congestion levels • I^2_N: Peak hours • I^3_N:Accidents detection	• O^1_N: Emergency vehicle alerts • O^2_N: Driver notifications	• Official smart city alert application • Advanced driver-assistance system • smartphone
Traffic signal adjustments (TSA)	• I^1_{TSA}: Emergency vehicle alerts • I^2_{TSA}: Driver notifications	• O^1_{TSA}: Inclusion of emergency vehicle preemption • O^2_{TSA}: Adjustment of traffic signals	• Adaptive traffic signal control system • Traffic management center • Urban application programming interface
Route optimization (RO)	• I^1_{RO}: Inclusion of emergency vehicle preemption • I^2_{RO}: Adjustment of traffic signals	• O^1_{RO}: Traffic-optimized routes • O^2_{RO}: Arrival time estimates	• Google Map • Waze • MapFactor Navigator

Example. A smart traffic management semantic composition can be represented as the following sequence of concrete services:

$$C = Radar \xrightarrow[O^2_R=I^2_{TAS}\ O^3_R=I^3_{TAS}]{O^1_R=I^1_{TAS}} \text{Traffic analyzer system} \xrightarrow[O^2_{TAS}=I^2_{ATSCS}\ O^3_{TAS}=I^3_{ATSCS}]{O^1_{TAS}=I^1_{ATSCS}} \text{Adaptive traffic signal control system}$$

$\xrightarrow[O^2_{ATSCS}=I^2_{TMC}]{O^1_{ATSCS}=I^1_{TMC}}$Traffic management center$\xrightarrow[O^2_{TMC}=I^2_{W}]{O^1_{TMC}=I^1_{W}}$ $Waze$.

The inputs and outputs of each concrete service within this semantic composition are described in Table 4.2.

Table 4.2: Inputs and outputs of the concrete services included in a smart traffic management composition.

Concrete service	**Inputs**	**Outputs**
Radar (R)	• I^1_R: Video feeds • I^2_R: Disparity maps	• O^1_R:Vehicles per hour/lane • O^2_R:Percentage of road space occupied • O^3_R:Unusual patterns
Traffic analyzer system (TAS)	• I^1_{TAS}: Vehicles per hour/lane • I^2_{TAS}: Percentage of road space occupied • I^3_{TAS}: Unusual patterns	• O^1_{TAS}: Congestion levels • O^2_{TAS}: Peak hours • O^3_{TAS}: Accidents detection
Adaptive traffic signal control system ($ATSCS$)	• I^1_{ATSCS}: Congestion levels • I^2_{ATSCS}: Peak hours • I^3_{ATSCS}: Accidents detection	• O^1_{ATSCS}: Emergency vehicle alerts • O^2_{ATSCS}: Driver notifications
Traffic management center (TMC)	• I^1_{TMC}: Emergency vehicle alerts • I^1_{TMC}: Driver notifications	• O^1_{TMC}: Inclusion of emergency vehicle pre-emption • O^2_{TMC}: Adjustment of traffic signals
Waze (W)	• I^1_W: Inclusion of emergency vehicle pre-emption • I^2_W: Adjustment of traffic signals	• O^1_W:Traffic-optimized routes • O^2_W: Arrival time estimates

4.2.2 Quality of semantic matching

The *quality of semantic matching (QoSM)* associated with a composition is measured through two metrics: (*i*) the matching type (MT) and (*ii*) the semantic similarity (SIM) between concrete services belonging to this composition [63].

Let CS_1 and CS_2 be two successive concrete services belonging to the composition C, and let $concept_a$ and $concept_b$ be two concepts belonging to an ontology Θ such as the $concept_a$ is an output concept of the service CS_1 (i.e., $concept_a \in O_{CS_1}$) and the $concept_b$ is an input concept of the service CS_2 (i.e., $concept_b \in I_{CS_2}$). The degree of semantic matching between the services CS_1 and CS_2 is described using two types of matching: *exact* or *plugin* matching [129]. An exact matching between CS_1 and CS_2 occurs when an equivalence relationship exists between the two concepts, $concept_a \equiv concept_b$, whereas a plugin matching between CS_1 and CS_2 refers to an inclusion relationship between the two concepts, $concept_a \subseteq concept_b$. The matching type is then defined as follows:

$$MT = \begin{cases} 1 & \text{if } concept_a \equiv concept_b \\ p & \text{if } concept_a \subseteq concept_b \end{cases} \tag{4.5}$$

where $0 < p < 1$. In the case where multiple matching pairs exist between the outputs of the service CS_1 and the inputs of the service CS_2, the matching type MT is computed as the average of the matching types across all these pairs. The aggregated matching type $AggMT_C$ of the composition C is therefore calculated as follows:

$$AggMT_C = \prod_{l=1}^{nb_{CS}} MT_l \tag{4.6}$$

where nb_{CS} is the number of concrete services belonging to the composition C.

The semantic similarity SIM between the concept $concept_a$ of the service CS_1 and the concept $concept_b$ of the service CS_2 is computed using a similarity method based on conceptual distance [130]. Formally:

$$SIM(concept_a, concept_b) = \frac{2 \times D_{concept_{na}}}{D_{concept_a} + D_{concept_b}} \tag{4.7}$$

where $concept_{na}$ is the nearest common ancestor of $concept_a$ and $concept_b$. The terms $D_{concept_a}$, $D_{concept_b}$, and $D_{concept_{na}}$ denote the respective distances of $concept_a$, $concept_b$, and $concept_{na}$ from the root concept of the ontology Θ. Figure 4.1 shows an example of concept hierarchy in the ontology Θ, which is used to compute semantic similarity [130].

In the case of multiple matching pairs between the outputs of the service CS_1 and the inputs of the service CS_2 in the composition C, the semantic similarity SIM is calculated as the average of the semantic similarities over all these pairs. The aggregated semantic similarity $AggSIM_C$ of the composition C is then computed as follows:

$$AggSIM_C = \sum_{l=1}^{nb_{CS}} SIM_l \tag{4.8}$$

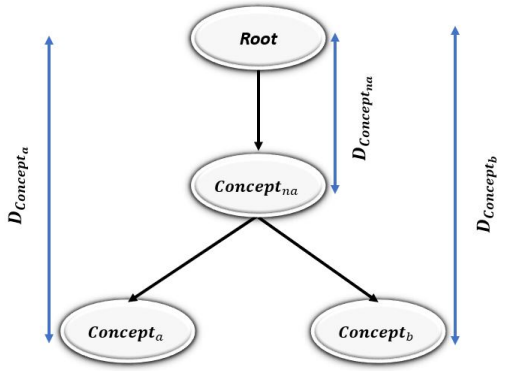


Figure 4.1: Concept hierarchy example.

where nb_{CS} is the number of concrete services belonging to the composition C.

4.2.3 Formalization of the autonomous QoS-aware service composition problem

The autonomous QoS-aware service composition problem can be described as a tuple $P = (UR, SR, \Theta)$. The *user request* $UR = (I_{UR}, O_{UR})$ expresses the functional constraints related to the user's needs as a set of provided inputs I_{UR} and a set of inquired outputs O_{UR}. The *service repository* $SR = \{CS_1, \cdots, CS_n\}$ is a collection of n concrete services that share a common ontology Θ formed by a set of concepts and the description of a semantic relationship between them. The ontology concepts include all inputs and outputs of concrete services belonging to the service repository SR along with the input and output concepts of the user's request. Formally:

$$\Theta = \{I_{UR} \cup O_{UR}\} \parallel \{I_{CS_i} \cup O_{CS_i}\} / \forall i \in [1, n] \tag{4.9}$$

where n is the number of concrete services in the service repository.

The overall quality of a composition C is measured using a *utility function*, which is calculated as a weighted sum of the QoS and QoSM metrics. Formally:

$$F(C) = \sum_{r=1}^{nbP} w_r \times QoS_r + \sum_{r=nbP+1}^{k} w_r \times (1 - QoS_r) + w_{(k+1)} \times AggMT_C + w_{(k+2)} \times AggSIM_C \tag{4.10}$$

where k is the number of QoS attributes, nbP is the number of positive QoS attributes, w_r ($\sum_{r=1}^{k+2} w_r = 1$) is a weight that represents the user's preference related to the r^{th} QoS attribute, and QoS_r ($1 < r < k$) is the normalized value of the r^{th} QoS attribute associated

with the composition C, which is calculated according to the composition structure (see [17] for more details).

The objective of the autonomous QoS service composition problem is to find a composition that maximizes the utility function computed using formula (4.10) while satisfying the user's functional constraints, i.e., the inputs and outputs of the obtained composition semantically matches the inputs and outputs of the user's request, respectively.

4.2.4 Relational database-related concepts

We introduce hereafter the key concepts of relational database model that are used in the approaches proposed in this chapter.

4.2.4.1 Database

A database $D = \{R_1, \cdots, R_t, \cdots, R_u\}$ is formally defined as a collection of u relations $R_1, \cdots, R_t, \cdots, R_u$ [131].

Example. Let $BusinessManagement$ be a database recording information related to companies, employees, and products such as:

$BusinessManagement = \{Companies, Employees, Products\}$.

4.2.4.2 Relation

A relation R is formally defined by a schema $R(A_1, \cdots, A_d, \cdots, A_w)$, where R denotes the name of the relation and $A_1, \cdots, A_d, \cdots, A_w$ represent its attributes [132].

Examples. Three relations are contained in the $BusinessManagement$ database where:

- The relation $Companies$ is defined by tree attributes: company identifier $CompId$, company name $CompName$, and Business Sector $BusSector$:

  $Companies(CompId, CompName, BusSector)$.

- The relation $Employees$ is defined by four attributes: employee identifier $EmpId$, employee name $EmplName$, affiliation department $Depart$, and telephone number $TelNumber$:

  $Employees(EmpId, EmplName, Depart, TelNumber)$.

- The relation $Products$ is defined by two attributes: product identifier $ProdId$ and price $Price$:

  $Products(ProdId, Price)$.

4.2.4.3 Attribute

An attribute A_d is a named characteristic of an entity within a database, which defines the data type and domain of possible values $\text{Dom}(A_d)$ that can be associated with an entity instance [133].

Examples. The attributes *BusSector*, *Depart*, and *Price* in the relations *Companies*, *Employees*, and *Product*, respectively, can take values from the following domains:

- $\text{Dom}(BusSector) = \{Healthcare, Electronics, Agriculture\}$.
- $\text{Dom}(Depart) = \{Marketing, HumanResources, Administration\}$.
- $\text{Dom}(Price) = \{100, 750, 1000\}$.

4.2.4.4 Tuple

A tuple t is an ordered list of values corresponding to the attributes in a relation, representing one data record within a relation. A tuple t in a relation R is formally defined as an ordered n-tuple [134]:

$$t = (v_1, .., v_d, ..., v_w) \ / \ v_d \in \text{Dom}(A_d) \ \forall \ d \in [1, w] \tag{4.11}$$

Examples. Let t_{C1}, t_{E2}, and t_{P3} be examples of tuples belonging to the relations *Companies*, *Employees*, and *Products*, respectively. Each tuple contains values for all attributes of the corresponding relation:

Companies		
CompId	CompName	BusSector
02	TechSolution	Technology

$t_{C1} = (02, \textit{TechSolution}, \textit{Technology})$

Employees			
EmpId	EmplName	Depart	TelNumber
20	Jak	Administration	0033946623197
34	Bill	Marketing	0033678647845

$t_{E2} = (34, \textit{Bill}, \textit{Marketing}, 0033678647845)$

Products	
ProdtId	Price
Sph768	750
Mask14	100
Dron3	1000

$t_{P3} = (Dron3, 1000)$

4.2.4.5 Projection

A projection $t[X]$ is a subset of attributes $X \subseteq \{A_1, ..., A_d, ..., A_w\}$ from a tuple t in a relation R [135].

Examples. Let $t_{C1}[CompId, BusSector]$, $t_{E2}[EmplName, Depar, TelNumber]$, and $t_{P3}[Price]$ be the projections of tuples t_{C1}, t_{E2}, and t_{P3} in the relations $Companies$, $Employees$ and $Products$, respectively:

- The projection of the tuple t_{C1} on the values corresponding to the attributes $BusSector$ in the relation $Companies$ is as follows:

  $t_{C1}[CompId, BusSector] = (02, Technology)$.

- The projection of the tuple t_{E2} on the values corresponding to the attributes $EmplName$ and $TelNumber$ in the relation $Employees$ is:

  $t_{E2}[EmplName, Depar, TelNumber] = (Bill, Marketing, 0033678647845)$.

- The projection of the tuple t_{P3} on the values corresponding to the attributes $ProdtId$ and $Price$ in the relation $Products$ is as follows:

  $t_{P3}[Price] = (1000)$.

4.2.4.6 Functional dependency

A functional dependency FD is a constraint between two sets of attributes, X and Y, within a relation R. This constraint specifies that for any two tuples t_1 and t_2 in R, if the values corresponding to the attributes in X are equal ($t_1[X] = t_2[X]$), then the values associated with to the attributes in Y must also be equal ($t_1[Y] = t_2[Y]$). In other words, the values of the attribute set Y are functionally determined by the values of the attribute set X [136].

Example. In a company, an employee is usually attached to a single department. This can be expressed through functional dependency $EmplName \rightarrow Depart$, which means that the attribute $EmplName$ determines the attribute $Depart$. In contrast, when several employees are attached to the same department, this situation can not be represented by the functional dependency $EmplName \rightarrow Depart$ as the department does not identify a single employee.

4.2.4.7 The attribute closure algorithm

Let $\mathcal{F}$ be a set of functional dependencies and X a set of attributes. The closure of the attribute set X under $\mathcal{F}$, defined as X^+, is the set of attributes that are determined by X using dependencies belonging to $\mathcal{F}$. The attribute closure algorithm that allows calculating the closure of an attribute set is summarized as follows [97, 98]:

Algorithm 4.1 Attribute closure algorithm.

Inputs: A set of attributes X and a set of functional dependencies $\mathcal{F}$.

1: **Begin**
2: $X^+ = \{X\}$.
3: Initialize a set $\mathcal{F}^{'}$ with the functional dependencies of $\mathcal{F}$.
4: Seek for a functional dependency $A \rightarrow B \in \mathcal{F}^{'}$ / $A \subseteq X^+$.
5: **if** the functional dependency $A \rightarrow B$ exists **then**
6: **if** $B \notin X^+$ **then**
7: $X^+ = X^+ \cup \{B\}$.
8: **end if**
9: $\mathcal{F}^{'} = \mathcal{F}^{'} - \{A \rightarrow B\}$.
10: Go to 4.
11: **end if**
12: **End**

Outputs: The closure X^+.

Example. Let $\mathcal{F} = \{M \rightarrow P;\ M \rightarrow K;\ PK \rightarrow M;\ P \rightarrow L\}$ be a set of functional dependencies. The attribute closure algorithm is applied to find M^+.

$M^+ = \{M\}$; $\mathcal{F}' = \{M \rightarrow P;\ M \rightarrow K;\ PK \rightarrow M;\ P \rightarrow L\}$;
$M \rightarrow P$ and $P \notin M^+ \Rightarrow M^+ = \{M,\ P\}$; $\mathcal{F}' = \{M \rightarrow K;\ PK \rightarrow M;\ P \rightarrow L\}$;
$M \rightarrow K$ and $K \notin M^+ \Rightarrow M^+ = \{M,\ P,\ K\}$; $\mathcal{F}' = \{PK \rightarrow M;\ P \rightarrow L\}$;
$PK \rightarrow M$ and $M \in M^+ \Rightarrow M^+ = \{M,\ P,\ K\}$; $\mathcal{F}' = \{P \rightarrow L\}$;
$P \rightarrow L$ and $L \notin M^+ \Rightarrow M^+ = \{M,\ P,\ K\ L\}$; $\mathcal{F}' = \{\}$.
There are no more dependencies in $\mathcal{F}'$, the attribute closure algorithm ends, and $M^+ = \{M,\ P,\ K\ L\}$.

4.2.5 Late Acceptance Hill Climbing algorithm

The Latte Acceptance Hill Climbing algorithm (LAHC) [99] is a local search-based optimization method that represents an improved version of the Hill climbing algorithm [137], incorporating a novel mechanism to avoid local optima. The main idea of this algorithm is to accept the non-improved solution where its fitness value is higher than that of the solution obtained from a fixed number of previous iterations. This algorithm is easy to implement, requires only one tuning parameter, and is highly effective for small-scale optimization problems. The LAHC algorithm performs in two phases: 1) *initialization* and 2) *evolution*.

4.2.5.1 Initialization phase

In this phase, the algorithm's parameters are set, including the move step $step_{size}$, the history length Lh, the counter of iterations with no improvement $Lidle$, and the maximum number of iterations $Maxit$. The initial solution $X_{Intial} = (x_1^b, ..., x_e^b, ..., x_{nb_{Dim}}^b)$ is randomly generated, where x_e^b refers to the b^{th} variable of the initial solution in the e^{th} dimension, nb_{Dim} is the variable dimension. Subsequently, the initial solution is used to initialize the current best solution, such as $BestX = X_{Intial}$ and the fitness value $f(BestX)$ of the current best solution is computed using a fitness function f. A fitness history list f_{List} of length Lh is initialized using the fitness value of the current best solution $f_{List} = (f(BestX)^1, f(BestX)^2, \cdots, f(BestX)^{Lh})$.

4.2.5.2 Evolution phase

During this phase, at each iteration, a neighboring solution is generated as follows by employing a move step $Step_{size}$ on the current best solution $BestX$:

$$X_N(it) = BestX(it) + step_{size} \times r \tag{4.12}$$

where the move step $step_{size}$ is a constant number and r is a randomly generated number in the interval $[1, nb_{Var}^e]$, nb_{Var}^e is the maximal value that a variable x_e^b can take. The fitness value of the neighboring solution $f(X_N)$ is then calculated and the solution is accepted when its fitness value exceeds either that of the current best solution or the value stored in the history list at the index corresponding to Lh iterations earlier. More specifically, the algorithm can accept solutions having a lower fitness value than the current best solution but higher than that of the solution found in several prior iterations. This process allows the temporary acceptance of no-improving solutions to avoid local optima using the following

formula:

$$BestX(it) = \begin{cases} X_N(it) & \text{if } f(X_N)(it) > f_{List}(Vb) \text{ or } f(C_N)(it) >= f(BestX)(it) \ , \\ BestX(it) & \text{otherwise} \ , \end{cases} \tag{4.13}$$

where Vb is an index corresponding to Lh iterations earlier calculated as follows:

$$Vb = it \mod Lh \tag{4.14}$$

After that, if the fitness value of the current best solution $f(BestX)$ is better than that of the solution recorded Lh iterations earlier, it replaces $f_{List}(Vb)$ in the history list at iteration Vb. The LAHC algorithm terminates when no further improvements can be made to the current best composition. This is determined by counting the number of no-improving iterations, denoted as I_{idle}. The algorithm stops automatically when I_{idle} reaches 2% of the maximum predefined number of iterations. The solution having the highest fitness value discovered during the evolution process is returned as the best solution.

4.2.6 Exhaustive search method

Exhaustive search [100] is a solving method that generates all possible solutions $X_{Set} = \{X_1, ..., X_p, ..., X_n\}$ for a given problem and evaluates them using a fitness function $f(X)$. This method guarantees finding the optimal solution X^* by selecting the best one in terms of fitness value. Formally:

$$X^* = \begin{cases} \arg\max_{X \in X_{Set}}(f(X)) & \text{if } f(X) \text{ is a maximization function} \ , \\ \arg\min_{X \in X_{Set}}(f(X)) & \text{otherwise} \ , \end{cases} \tag{4.15}$$

4.3 The proposed approaches : HCFDSSC and ESFDSSC algorithms

4.3.1 Main idea of the HCFDSSC and ESFDSSC approaches

Two functional dependencies-based failure recovery approaches for autonomous QoS-aware service composition (HCFDSSC and ESFDSSC) are proposed in this thesis [94]. The proposed approaches operate in three phases: pre-processing, feasible abstract plans construction, and parallel service selection and composition. To reduce the search space size and decreases the problem's complexity, the functionally equivalent concrete services are first grouped to form a set of abstract services. The *attribute closure algorithm* [97, 98] that ensures the semantic matching between functional dependencies is then employed to generate

the potential matching between the resulting abstract services. This is done by identifying all input and output concepts that can be derived from the set of user's request inputs. Several feasible abstract composition plans that meet the user's functional request are therefore automatically built. Finally, the HCFDSSC approach employs a parallel late acceptance hill climbing algorithm [99] while the ESFDSSC approach uses a parallel exhaustive search strategy [100] to obtain the suboptimal composition in terms of QoS and QoSM from the resulting abstract composition plans. Figure 4.2 presents an example of the HCFDSSC and ESFDSSC approaches applied to a service repository containing 22 concrete services. During the pre-processing phase, these services are grouped into 11 abstract services. The feasible abstract plan construction phase produces two abstract composition plans. Figure 4.3 depicts the flowchart illustrating the steps of the HCFDSSC and ESFDSSC approaches.

In the context of the semantic service composition, a set of attributes X and Y represent the input set I and the output set O, respectively. The notion of functional dependency $FD : X \rightarrow Y$ is used in this chapter to represent the user's request $UR = (I_{UR}, O_{UR})$ as $I_{UR} \rightarrow O_{UR}$ and each abstract service $AS = (I_{AS}, O_{AS})$ as $I_{AS} \rightarrow O_{AS}$. A set of functional dependencies $F = \{X_1 \rightarrow Y_1, \cdots, X_n \rightarrow Y_n\}$ represents the service repository $SR = \{I_{CS_1} \rightarrow O_{CS_2}, \cdots, I_{CS_n} \rightarrow O_{CS_n}\}$, whereas an abstract composition plan $ACP = (I_{ACP}, O_{ACP})$ is represented as a functional dependency $I_{ACP} \rightarrow O_{ACP}$. Table 4.3 shows the corresponding terms between the semantic service composition problem and the attribute closure algorithm.

Table 4.3: Attribute closure algorithm versus semantic service composition problem.

Attribute closure algorithm	**Semantic service composition**
Attribute	Input and output concepts
Tuple	The value of input and output concepts
Domain	All possible input and output concepts
Functional dependency	User's request, abstract service, and abstract composition plan
Set of functional dependencies	Service repository

4.3.2 The phases of approaches

4.3.2.1 Pre-processing

The size of the service repository $SR = \{CS_1, \cdots, CS_p, \cdots, CS_n\}$ is reduced in this phase, where n represents the number of concrete services and is significantly large ($n >> 1$). This reduction is achieved by grouping functionally equivalent concrete services to create abstract services. A set of concrete services $CS_1, \cdots, CS_m$ are considered functionally equivalent if they share the same inputs (i.e., $I_{CS_1} =, \cdots, = I_{CS_m}$) and the same outputs (i.e., $O_{CS_1} =, \cdots, = O_{CS_m}$). Such a set of services is represented as an abstract service AS_r. A

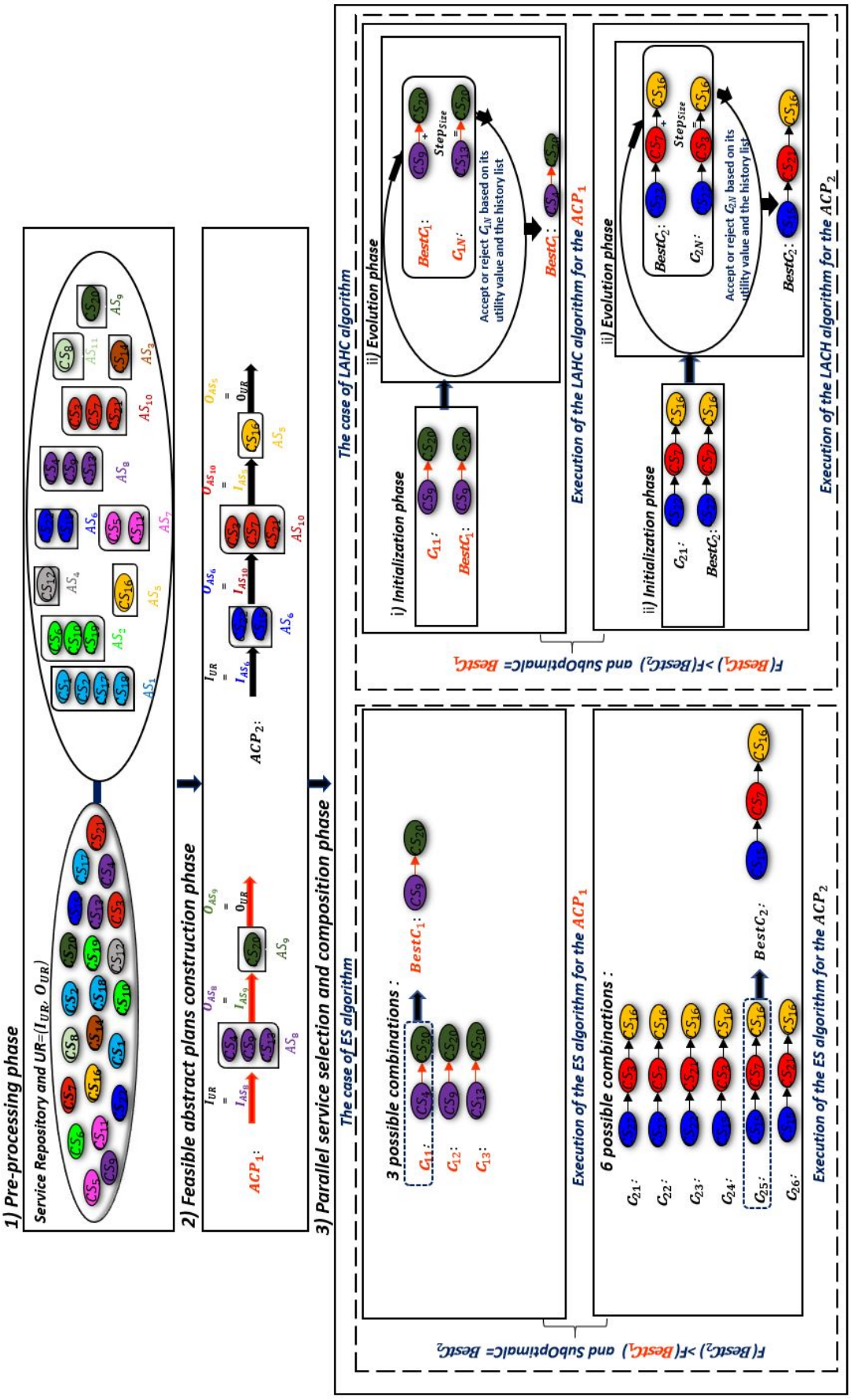


Figure 4.2: The phases of the HCFDSSC and ESFDSSC approaches.

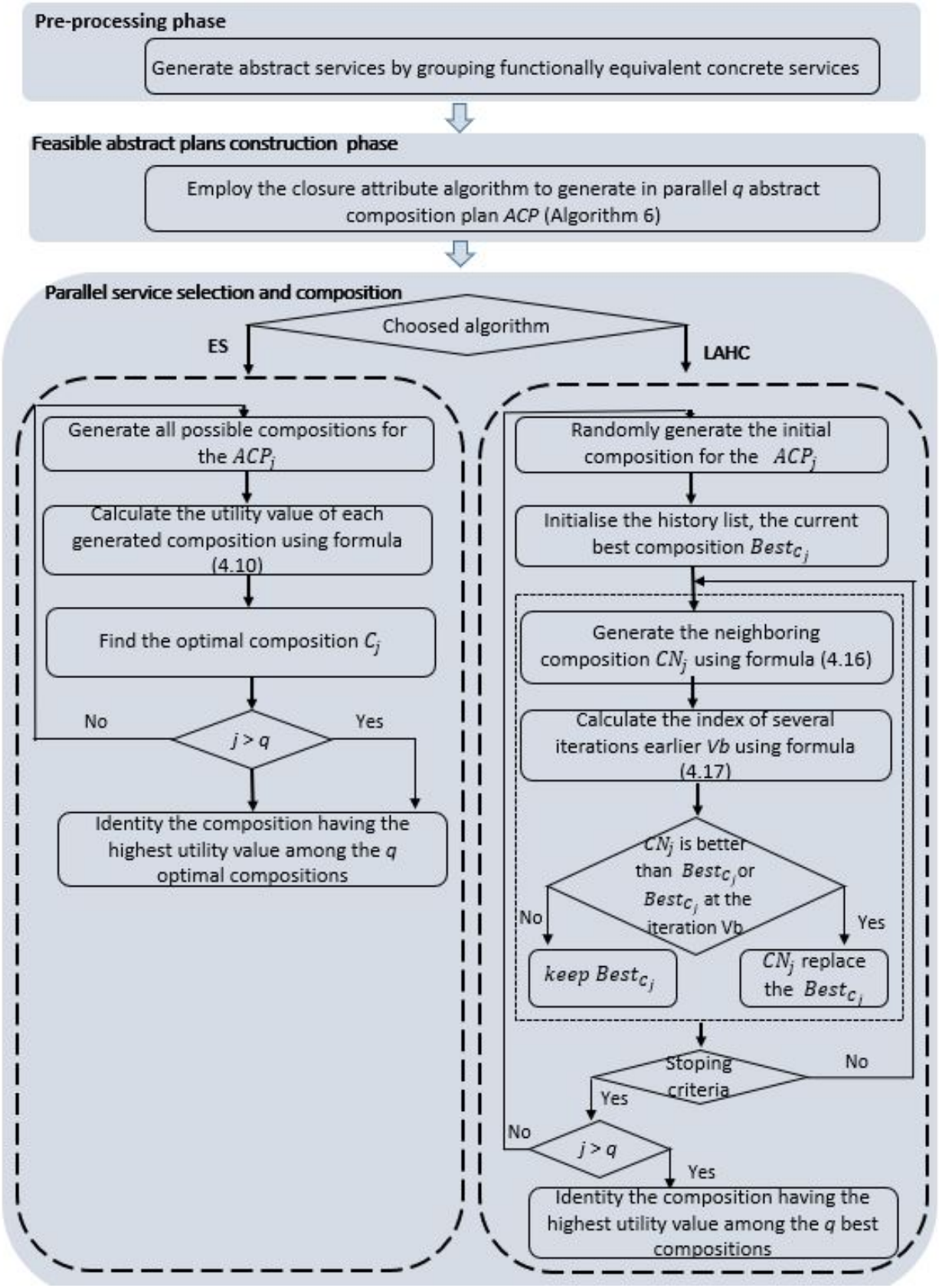


Figure 4.3: HCFDSSC and ESFDSSC approaches flowchart.

reduced service repository is then obtained as $SR = \{AS_1, \cdots, AS_r, \cdots, AS_z\}$, where z is

the size of the reduced service repository and $z << n$. The primary goal of reducing the repository size is to decrease the number of potential inputs-outputs matches to be evaluated between services, decreasing thus the overall computation time.

4.3.2.2 Feasible abstract plans construction

The attribute closure algorithm [97, 98] is exploited in this phase to build feasible abstract composition plans.Given a service repository $SR = \{AS_1, \cdots, AS_r, \cdots, AS_z\}$ and a user's request $UR = (I_{UR}, O_{UR})$, the attribute closure of I_{UR}, denoted I^+_{UR}, is calculated under the service repository SR to identify the sets of inputs and outputs that can be functionally derived from the inputs of the user's request. First, the closure I^+_{UR} is set to the user's request inputs (ligne 2 of Algorithm 4.2). The algorithm also initializes the set $SR^{'}$ that contains the abstract services in SR (ligne 3), and an empty set $\mathcal{S}$ (ligne 4). Then, an abstract service AS_h is identified from the set $SR^{'}$ whose inputs semantically match the concepts already included in the closure I^+_{UR} (ligne 5). In other words, the abstract service AS_h is functionally determined by the concepts contained in the closure I^+_{UR} according to the ontology Θ. After that, the outputs O_{AS_h} of the abstract service AS_h are added to the closure set I^+_{UR} (ligne 8), the abstract service is removed from the set $SR^{'}$ and added to the set $\mathcal{S}$ (lignes 10 to 11). This process is repeated until no abstract service can be added, i.e., when the inputs of the remaining abstract service are not included within the closure I^+_{UR}. To determine whether a feasible abstract composition plan in $SR^{'}$ meets the user's functional requirements, it is essential to check that the outputs of the user's request, O_{UR}, are included in the closure I^+_{UR} (ligne 13), i.e., the request outputs must semantically match, according to the ontology Θ, with one or more concepts that are part of the closure I^+_{UR}. When the above condition is satisfied, the potential semantic matches between the inputs and outputs of the abstract services in the set $\mathcal{S}$ are computed to produce a set of feasible abstract composition plans, denoted ACP (lignes 14 to 15). The steps of this phase are given in Algorithm 4.2.

4.3.2.3 Parallel service selection and composition

This phase aims to find the suboptimal composition $SubOptimalC$ having the highest utility value in terms of QoS and QoSM according to the ontology Θ in a reduced computation time. To achieve this, two approaches are proposed: 1) the latte acceptance hill climbing and functional dependencies-based autonomous QoS-aware service composition approach (HCFDSSC) that exploits the late acceptance hill climbing method (LAHC) [99], and 2) the exhaustive search and functional dependencies-based autonomous QoS-aware service composition approach (ESFDSSC) that uses the exhaustive search method (ES) [100].

The HCFDSSC approach exploits the LAHC method in parallel across several ab-

Algorithm 4.2 Feasible abstract plans construction.

Inputs: UR, SR, and Θ.

1: **Begin**
2: Initialize $I^{+}_{UR} = \{I_{UR}\}$.
3: Initialize a set $SR^{'}$ with the abstract services in SR.
4: Initialize a set $S = \emptyset$.
5: Seek for an abstract service $I_{AS_h} \rightarrow O_{AS_h} \in \mathcal{SR}^{'}$ where $I_{AS_h} \subseteq^{\Theta} I^{+}_{UR}$.
6: **if** the abstract service $I_{AS_h} \rightarrow O_{AS_h}$ exists **then**
7: **if** $O_{AS_h} \notin I^{+}_{UR}$. **then**
8: $I^{+}_{UR} = I^{+}_{UR} \cup \{O_{AS_h}\}$.
9: **end if**
10: $SR^{'} = SR^{'} - \{I_{AS_h} \rightarrow O_{AS_h}\}$.
11: $S = S \cup \{I_{AS_h} \rightarrow O_{AS_h}\}$.
12: Go to 5
13: **if** $O_{UR} \subseteq^{\Theta} I^{+}_{UR}$ **then**
14: Perform a semantic matching between the inputs and outputs of the AS contained in S.
15: Find a set of q feasible abstract composition plans $ACP = \{ACP_1, \cdots, ACP_q\}$.
16: **end if**
17: **end if**
18: **End**

Outputs: A set of q feasible abstract composition plans $ACP = \{ACP_1, \cdots, ACP_q\}$.

stract composition plans resulting from the previous phase, enabling the finding of multiple suboptimal compositions in a reasonable computation time (see Algorithm 4.3). The LAHC method is a greedy algorithm characterized by a simple implementation that requires few tuning parameters and a move acceptance strategy to avoid local optima [99]. In the context of service composition and for each abstract composition plan ACP_j $(1 \leq j \leq q)$, this phase starts with a randomly generated composition $C_{jIntial} = (pos^{jb}_1, \cdots, pos^{jb}_e, \cdots, pos^{jb}_{nb_{AS}})$ where pos^{jb}_e represents the index of the b^{th} concrete service of the e^{th} abstract service in the j^{th} abstract composition plan ACP_j. The following parameters are then initialized: the move step $step_{size}$, the history length Lh, the counter of no-improving iterations L_{idle}, and the maximum number of iterations Max_{it}. The current best composition is initialized as the initial composition $BestC_j = C_{jInitial}$, and its utility value $F(BestC_j)$ is calculated using formula (4.10) (lignes 7 to 8 of Algorithm 4.3). Then, a utility history list F_{List} of length Lh is initialized using the utility value of the current best composition $F_{List} = (F(BestC_j)^1, \cdots, F(BestC_j)^{Lh})$ (ligne 9). At each iteration it, the current best composition $BestC_j$ is iteratively updated by applying the move step $step_{size}$ to produce a neighboring solution C_{jN} using the following formula (ligne 11):

$$C_{jN}(it) = BestC_j(it) + step_{size} \times r \tag{4.16}$$

where the move step $step_{size}$ is a constant and r is a randomly generated number in the interval $[1, nb^{e}_{csj}]$, nb^{e}_{csj} is the number of concrete services in the e^{th} abstract service of the j^{th} abstract composition plan. This neighboring composition $C_{jN}(it)$ at iteration it is then

accepted or rejected for evolution in the next iteration based on the idea of the late acceptance (lignes 18 to 24). More precisely, the neighboring composition $C_{jN}(it)$ is accepted when its utility value $F(C_{jN})(it)$ is higher than that of the current best composition $F(BestC_j)(it)$ or the utility value stored in the history list at the index corresponding to Lh iterations earlier Vb, which is calculated as follows:

$$Vb = it \mod lh \tag{4.17}$$

Finally, the parallel service selection and composition phase of the HCFDSSC algorithm terminates when no further improvement is observed in the current best composition. This situation is detected by calculating the number of consecutive no-improving iterations I_{idle}. The algorithm stops automatically when I_{idle} achieves 2% of the maximum number of iterations and the composition with the highest utility value obtained during the evolution process is retained as the best composition. Subsequently, the function $BestUtilC$ is used to return the suboptimal composition $SubOptimalC$ by identifying the composition with the highest utility among those generated from the q abstract composition plans ACP (ligne 29).

The ESFDSSC approach uses the ES method that can provide all possible solutions for a given problem (see Algorithm 4.4). The choice of this method is motivated by its efficiency in finding the optimal solution in the case of small-scale problems. The ES method is used in parallel on the resulting abstract composition plans ACP_j $(1 \leq j \leq q)$ to find their corresponding best composition $BestC_j$ (lignes 3 to 8 of Algorithm 4.4). The utility values of these compositions are calculated using formula (4.10). Then, the function $BestUtilC$ is applied to produce the suboptimal composition $SubOptimalC$ by identifying the composition having the highest utility among the resulting best compositions (lignes 9). The parallel service selection and composition phase of the ESFDSSC approach is summarized in Algorithm 4.4.

4.3.3 Failure recovery ability of approaches

With the dynamic nature of the IoT environment, service failure is one of the most critical issues that can result from the disappearance or unavailability of concrete services existing in the service repository, and those belonging to the final sub-optimal composition, making the latter no longer acceptable. Consequently, efficient failure recovery mechanisms are required to maintain composition stability and accuracy while preserving minimal composition time. The parallel delivery of multiple compositions that functionally meet the same user's request is a promising approach that offers alternative compositions to replace the failed one in the case of service failure. Specifically, rather than relying on a single composition solution, the proposed HCFDSSC and ESFDSSC approaches simultaneously maintain multiple abstract

Algorithm 4.3 Parallel service selection and composition phase of the HCFDSSC approach.

Inputs: ACP, Θ, Lh, $step_{size}$, Max_{it}

Begin
$ListBestC = \emptyset$.
for $j = 1 : q$ **do**
  $it = 1$.
  $I_{idle} = 0$.
  Randomly generate $C_{jIntial}$.
  $BestC_j(it) = C_{jIntial}$.
  Calculate $F(BestC_j)(it)$ using formula (4.10).
  Initialize F_{List}.
  while $it <= Max_{it}$ **or** $I_{idle} <= Max_{it} \times 0.02$ **do**
    Generate $C_{jN}(it)$ using formula (4.16).
    Calculate $F(C_{jN})(it)$ using formula (4.10).
    if $F(C_{jN})(it) <= F(BestC_j)(it)$ **then**
      $I_{idle} = I_{idle} + 1$.
    else
      $I_{idle} = 0$.
    end if
    Calculate Vb using formula (4.17)
    if $F(C_{jN})(it) > F_{List}(Vb)$ **or** $F(C_{jN})(it) >= F(BestC_j)(it)$ **then**
      $BestC_j(it) = C_{jN}(it)$.
    end if
    if $F(BestC_j)(it) > F_{List}(Vb)$ **then**
      $F_{List}(Vb) = F(BestC_j(it))$.
    end if
    $it = it + 1$.
  end while
  $ListBestC = ListBestC \cup \{BestC_j\}$.
end for
$SubOptimalC = BestUtilC(ListBestC)$.
End

Outputs: The suboptimal composition $SubOptimalC$.

composition plans ACP_j $(1 \leq j \leq q)$ that fulfill the same user functional requirements, each accompanied by its corresponding solutions. This parallel maintenance of alternative plans effectively addresses the risk of composition failure. In this case, an alternative composition among the best compositions $ListBestC$ with the second-highest utility value is used to replace the suboptimal $SubOptimalC$ composition. This failure recovery mechanism does not involve any additional composition time and does not disrupt the end user.

4.4 Performance study

To assess the performance of the proposed HCFDSSC and ESFDSSC approaches, different simulation scenarios were realized using MATLAB R2019b on a machine running a 64-bit

Algorithm 4.4 Parallel service selection and composition phase of the ESFDSSC approach.

Inputs: ACP

1: **Begin**
2: $ListBestC = \emptyset$
3: **for** $j = 1 : q$ **do**
4: Execute the ES method to find all compositions.
5: Calculate the utility value of each resulting composition using formula (4.10).
6: $BestC_j = BestUtilC(ACP_j)$.
7: $ListBestC = ListBestC \cup \{BestC_j\}$.
8: **end for**
9: $SubOptimalC = BestUtilC(ListBestC)$.
10: **End**

Outputs: The suboptimal composition $SubOptimalC$.

Windows operating system, 8 GB RAM, and an Intel Core i5-8250U processor clocked at 2 GHz. The scenarios were conducted using WSC-2008 [116] and WSC-2009 [117] datasets that include service repositories of increasing size. Each dataset is associated with a user's request and an ontology that describes the inputs and outputs of concrete services using the OWL-S language. The service repositories include the descriptions of concrete services (i.e., their inputs, outputs, and names) along with associated QoS vectors containing four attributes: availability, cost, reliability, and response time.

4.4.1 Baseline methods and performance metrics

The performance of the HCFDSSC and ESFDSSC approaches are evaluated in comparison with the memetic algorithm-based indirect approach for service composition (MGA) [64]. The MGA approach is a population-based algorithm that has been widely used as a baseline in the domain of autonomous QoS-aware service composition. Its selection as a baseline approach is motivated by its well-established nature, comprehensive description in the literature, and reliance on standardized evaluation datasets and metrics. These characteristics facilitate the reproducibility of experiments and enable consistent benchmarking, allowing researchers to effectively assess and compare progress in the domain [65, 69, 60]. The datasets used for evaluation contain respectively 158, 558, 572, 608, 1090, and 2198 concrete services.
The performance metrics used for the comparison between the HCFDSSC, ESFDSSC, and MGA approaches are:

- *Utility value:* This metric represents the overall quality of a suboptimal composition including both QoS and QoSM criteria. This value is calculated as the weight sum using formula (4.10), where the equal weights are assigned to the QoS and QoSM criteria.
- *Computation time:* This metric reflects the execution time required by the HCFDSSC, ESFDSSC, and MGA approaches to find the suboptimal composition.

- *QoS utility:* This metric represents the overall QoS utility value of the suboptimal composition, which is computed using formula (4.10), where a zero weight is assigned to the QoSM criteria.

- *QoSM utility:* This metric refers to the overall QoSM utility value of the suboptimal composition, calculated using formula (4.10) where the weight assigned to the QoS criteria is assumed to be zero.

- *Failure recovery*: This metric reflects the ability of the approach to handle service failures, measured by the number of abstract composition plans generated by the HCFDSSC and ESFDSSC approaches for a given service repository.

4.4.2 Simulation parameters

Table 4.4 provides the values of the simulation parameters used in the evaluation scenarios.

Table 4.4: Parameters used in simulation scenarios.

Parameters	**HCFDSSC**	**ESFDSSC**	**MGA**
Population size	1	/	30
Number of simulations	10	/	10
Move step $step_{size}$	0.7	/	/
History length Lh	35	/	/

4.4.3 Comparison and discussion

4.4.3.1 Utility value

This simulation scenario aims to evaluate the performance of the proposed HCFDSSC and ESFDSSC approaches in finding a suboptimal composition compared to the MGA approach. Figure 4.4 demonstrates that the utility values achieved by the HCFDSSC and ESFDSSC approaches are consistently higher than that obtained by the MGA approach, across all used datasets. This result is because the HCFDSSC and ESFDSSC approaches guarantee the semantic satisfaction of the user's functional requirements through the use of the functional dependencies model that is primarily designed to capture and preserve the semantic of the data. In particular, the attribute closure algorithm used in the proposed approaches ensures obtaining feasible abstract composition plans that align with the user's functional requirements. Additionally, in the case of the HCFDSSC approach, the effective choice of a move step employed by the LAHC algorithm during the parallel service selection and composition

phase, enables the iterative discovery of compositions with higher utility values. Furthermore, the late acceptance method allows to deeply explore the search space, avoiding thus the local optima, i.e., the evolved compositions are accepted not only based on those obtained in the previous iteration but also rely on the best compositions found in several iterations earlier. In the case of the ESFDSSC approach, the employed parallel search method finds the optimal composition for each resulted abstract compositions plan and the composition having the height utility value among them is then returned. Unlike the HCFDSSC and ESFDSSC approaches that use deterministic methods to fulfill the user's functional request and avoid local optima, ensuring high-quality compositions, the MGA approach relies on a random matching and stochastic evolution. This explains why the utility values obtained by the MGA approach are significantly lower than those produced by the HCFDSSC and ESFDSSC approaches. Note that, the utility value obtained by the ESFDSSC approach is greater than or equal to that found by the HCFDSSC approach. This result is due to the fact that the ESFDSSC approach uses an exact method that allows finding the optimal composition for each abstract composition plan, whereas the HCFDSSC approach employs a greedy algorithm that finds a suboptimal composition.

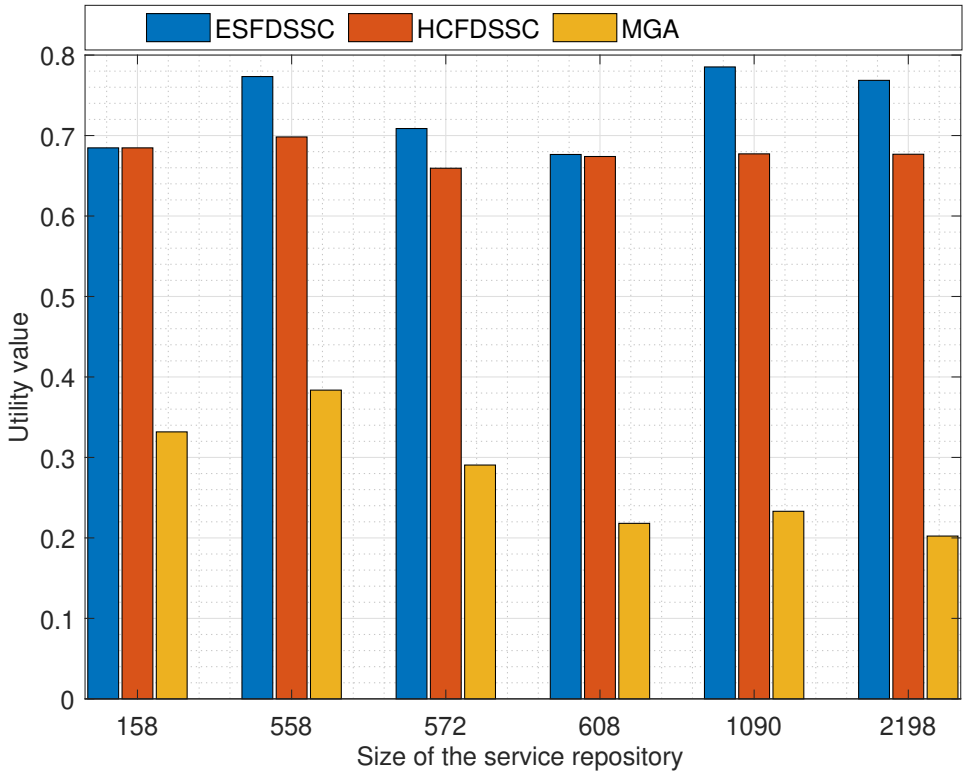


Figure 4.4: Utility value versus the service repository size.

4.4.3.2 Computation time

The objective of this simulation scenario is to assess the effectiveness of the HCFDSSC and ESFDSSC approaches in terms of computation time compared to the MGA approach. Figure 4.5 shows that the HCFDSSC and ESFDSSC approaches require much lower computation time than the MGA approach to find the suboptimal composition. This can be explained by two reasons. First, unlike the MGA approach that explores a large search space, the HCFDSSC and ESFDSSC approaches operate over a significantly smaller and more targeted search space. This efficiency is made possible by the pre-processing phase that reduces the size of the service repository, and through the attribute closure algorithm that provides multiple abstract service plans, each corresponding to a more refined search space. In addition, thanks to the first and second phases of the HCFDSSC and ESFDSSC approaches, the abstract services within the generated composition plans contain a small number of concrete services. As a result, the computation time required for the parallel service selection and composition phase has a negligible impact on the overall computation time, regardless of the employed algorithm (either the LAHC method or the ES method). It is worth noting that the exhaustive search used by the ESFDSSC approach is an exact method, i.e., its computation time increases when the number of concrete services in each abstract service of the composition is high, or when the abstract composition plans include a large number of the abstract services. This explains why, in most of datasets, the computation time obtained with the HCFDSSC approach is lower than that obtained with the ESFDSSC approach.

4.4.3.3 QoS utility

This simulation scenario is realized to evaluate the efficiency of the proposed HCFDSSC and ESFDSSC approaches in terms of QoS utility in comparison to the MGA approach. Figure 4.6 demonstrates that the QoS utility values of the compositions provided by the HCFDSSC and ESFDSSC approaches are significantly higher than that obtained with the MGA approach. Indeed, the first two phases used by the HCFDSSC and ESFDSSC approaches result in a search space containing only concrete services that have the potential to offer compositions with high QoS. Furthermore, the QoS utility values of the obtained compositions are influenced by the effectiveness of the method used in the parallel service selection and composition phase. This explains why the QoS utility values of the compositions obtained by the exact method used in the ESFDSSC approach are higher compared to that found by the greedy algorithm exploited in the HCFDSSC approach. This result highlights the advantages of the guided optimization process adopted in the proposed approaches over the random process adopted by the MGA algorithm that results in a lower QoS utility values.

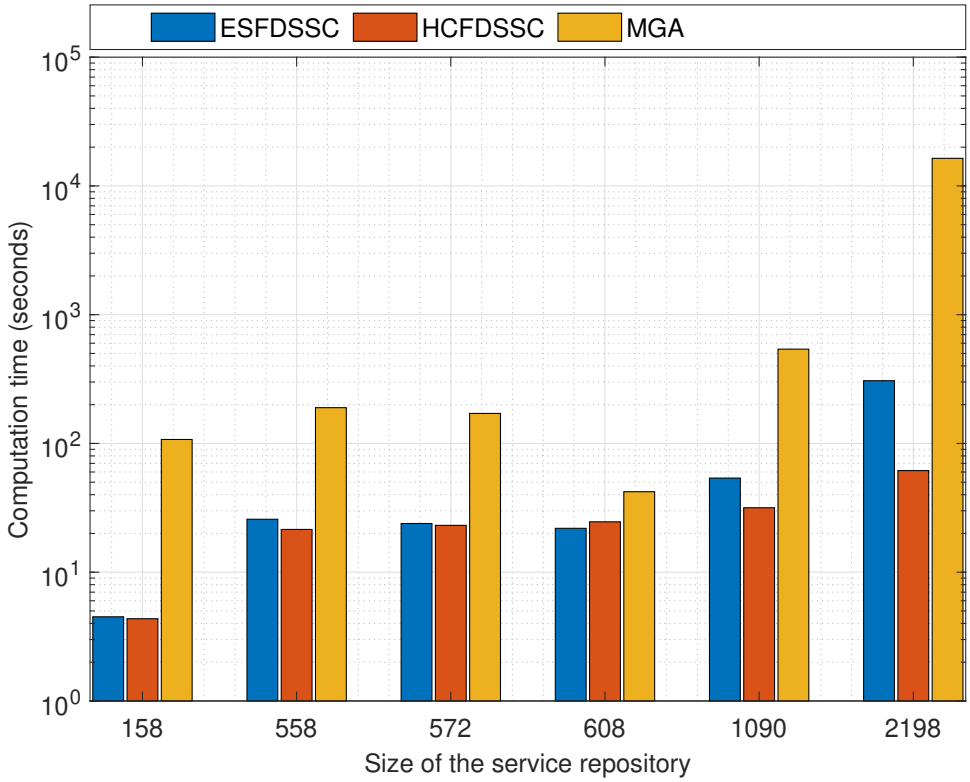


Figure 4.5: Computation time versus the service repository size.

4.4.3.4 QoSM utility

This simulation scenario is conducted to assess the performance of the proposed HCFDSSC and ESFDSSC approaches in terms of QoSM values compared to the MGA approach. Figure 4.7 shows that the QoSM utility values of compositions obtained by the HCFDSSC and ESFDSSC approaches are higher than that achieved by the MGA approach across most datasets. This result is because the attribute closure algorithm used in the HCFDSSC and ESFDSSC approaches guarantees that the evolved compositions satisfy the functional user's request, providing high QoSM values. Conversely, the MGA approach relies on a random process to determine whether a composition satisfies the functional user's request, leading to a fluctuation of the results in terms of QoSM values. Figure 4.7 also demonstrates that the QoSM utility value provided by the ESFDSSC approach is higher than that obtained with the HCFDSSC approach, except for the dataset having 608 concrete services. In fact, the decrease in terms of the QoSM value shown by the ESFDSSC approach is compensated by an increase in the QoS value (see Figure 4.6), resulting in an overall utility value that still better than that obtained in the case of the HCFDSSC approach.

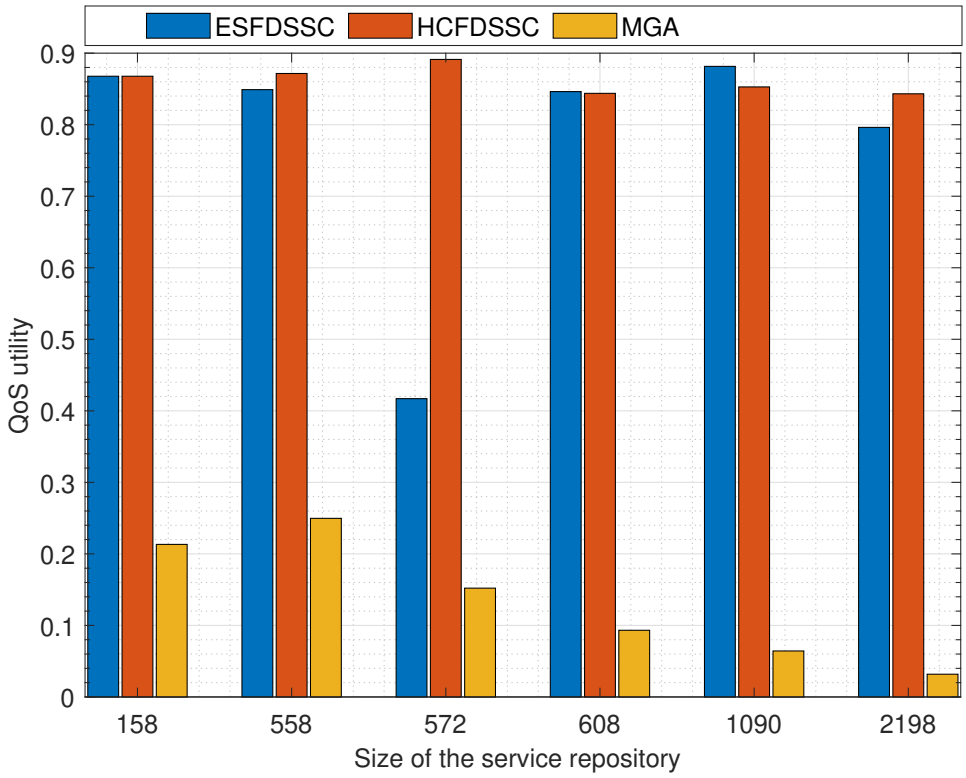


Figure 4.6: QoS utility versus the service repository size.

4.4.3.5 Failure recovery ability evaluation

This simulation scenario aims to demonstrate that the HCFDSSC and ESFDSSC approaches generate several abstract composition plans for the same service repository. Table 4.5 shows that the HCFDSSC and the ESFDSSC approaches produce multiple abstract composition plans for most datasets. As a result, backup compositions are stored to deal with service composition failure by directly replacing the suboptimal composition where no further execution time is needed. Note that, in some cases, as for the service repository having 158 and 608 concrete services, the HCFDSSC and the ESFDSSC approaches find one and only one abstract composition plan. This result is not related to the performance of the HCFDSSC and ESFDSSC approaches but depends on the existence of a possible semantic matching between the user's request and the abstract services available in the service repository.

4.5 Conclusion

This chapter proposes two functional dependencies-based failure recovery approaches for autonomous QoS-aware service composition (HCFDSSC and ESFDSSC). The HCFDSSC and ESFDSSC approaches operate on promising small-scale search spaces thanks to their pre-processing and feasible abstract plans construction phases that allow to reduce the com-

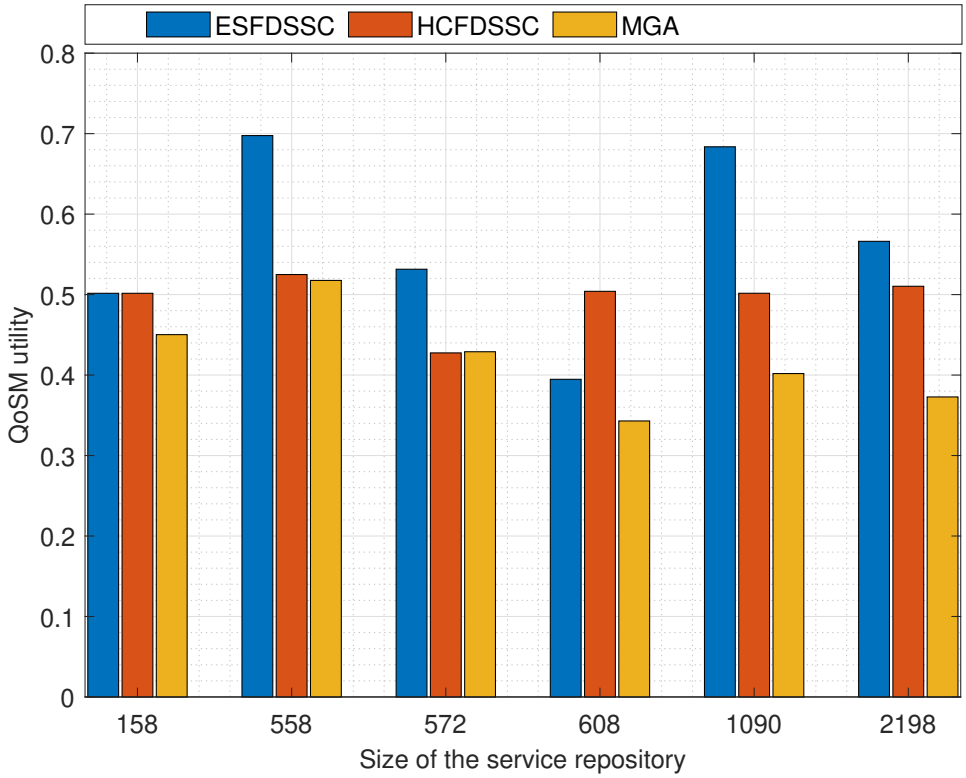


Figure 4.7: QoSM utility versus the service repository size.

putation time considerably. The performance study shows the efficiency of the proposed approaches in finding suboptimal composition with considerably high quality in terms of QoS, QoSM, and utility value combining both metrics while minimizing the computation time. The effectiveness in terms of utility value and QoS is due to the attribute closure algorithm that provides several abstract composition plan that semantically satisfy the functional user's request, whereas the employed LAHC algorithm and exhaustive search method ensure selecting high QoS suboptimal compositions. Additionally, the employed pre-processing phase and feasible abstract plans construction phases allow maintaining a reduced semantic search space with a small number of promising semantic concrete services, which explains

Table 4.5: Number of abstract composition plans generated by the HCFDSSC and the ESFDSSC approaches.

Dataset	**Number of abstract composition plans**
Dataset-1 (158)	1
Dataset-2 (558)	20
Dataset-3 (572)	21
Dataset-5 (608)	1
Dataset-6 (1090)	22
Dataset-7 (2198)	100

the low composition time of the proposed approaches. The number of abstract composition plans generated by the HCFDSSC and ESFDSSC approaches demonstrates their failure recovery ability, as they offer a substitution mechanism for directly replacing the suboptimal composition in the case of service failure.

Chapter 5

Conclusion and future search directions

Contents

5.1 Contribution synthesis

The QoS-aware service composition is a critical problem in the field of IoT, where the aim is to combine multiple services, each offering basic functionalities, to provide a more sophisticated service that fulfills user's complex requirements. This problem is characterized by several major challenges, including the simultaneous optimization of multi-conflicting QoS properties, the satisfaction of user-defined constraints, the consideration of semantic aspects, and the assumption of the prior existence of an abstract composition plan. These issues limit the large-scale applicability of existing approaches and require innovative algorithms that can address as many challenges as possible at the same time.

To address these challenges, three contributions are proposed in this thesis. The first contribution conducts a systematic literature review on autonomous and plan-based QoS-aware service composition approaches that have been proposed in the last decade. Existing surveys often focus on plan-based or autonomous approaches, limiting their scope to specific technical aspects. In contrast, the first contribution of this thesis introduced a comprehensive review with a two-layered taxonomy for a broader classification of QoS-aware service composition approaches. At the top layer of the taxonomy, the QoS-aware service composition approaches have been classified into two classes based on their reliance on a predefined abstract composition plan: (1) plan-based QoS-aware service composition approaches and (2) autonomous

QoS-aware service composition approaches. In the second layer, plan-based QoS-aware service composition approaches have been categorized into two sub-categories based on their underlying search space exploration strategy: (*i*) sequential exploration-based approaches and (*ii*) parallel exploration-based approaches, whereas autonomous QoS-aware service composition approaches are categorized into two sub-categories based on their employed strategy for finding semantically feasible compositions: (*i*) all matching-based approaches and (*ii*) random matching-based approaches. Additionally, a comparison of plan-based QoS-aware service composition approaches has been conducted considering criteria such as solving method, search space exploration methods, population size reduction strategy, and scalability degree. The autonomous QoS-aware service composition approaches have been compared based on their methodology for ensuring semantically correct compositions, solving method, utility value combining QoS and QoSM, failure recovery mechanism, and automatic generation of an abstract composition plan. An analysis and discussion of our findings is also conducted to address five research questions: *RQ*1− What are the primary classification criteria for QoS-aware service composition approaches? According to this question analysis, 62% of approaches are plan-based QoS-aware service composition approaches and 38% are autonomous QoS-aware service composition approaches. *RQ*2− Can secondary criteria refine the initial classification categories? Based on the analysis conducted for this question, 69% of the plan-based approaches rely on a sequential exploration-based search space strategy, while 31% of approaches use a parallel exploration-based strategy. In addition, 69% of the reviewed autonomous QoS-aware service composition approaches employ random matching strategy, whereas 31% adopt all matching strategy. *RQ*3− What methods are commonly employed to solve the QoS-aware service composition problem? Accounting for this question analysis, 88% of the investigated approaches have used computational intelligence-based optimization methods to solve the QoS-aware service composition problem, while only 12% exploited exact methods. *RQ*4− What performance metrics are commonly used to evaluate these approaches? With respect to this question analysis, the performance of investigated QoS-aware service composition approaches is validated using several metrics, including composition time, utility value in terms of QoS, and utility value combining QoS and QoSM. These metrics are respectively used by 83%, 95%, and 5% of the approaches reviewed in this thesis. *RQ*5− What datasets are used to assess the performance of the investigated approaches? According to the analysis conducted for this question, 26% of the studied QoS-aware service composition approaches assess their performance using experimental datasets, whereas 74% of approaches employ real-world datasets. Furthermore, the analysis highlights several shortcomings of the studied approaches such as the lack of semantic matching consideration, the assumption of prior existence of an abstract composition plan, increased composition time, limited composition quality, the use of a fixed population size, and the exclusion of the QoSM metric from the evaluation of the overall utility value of compositions. These findings highlight the need

for novel and efficient approaches, which motivates the subsequent contributions of this thesis.

Building on these insights, the second contribution introduces the Parallel Differential Evolution-based approach with population size reduction for QoS-aware Service Composition (PDE-QSC). This approach has been proposed to overcome the limitations of both sequential and parallel exploration-based approaches in terms of computation time and composition quality. As with most QoS-aware service composition approaches in the literature, this contribution assumes the prior existence of a predefined abstract composition plan. The PDE-QSC approach splits the initial population of compositions into two parallel-evolving sub-populations to reduce the composition time required to find the suboptimal composition, while simultaneously enhancing the composition quality. Each composition sub-population undergoes different evolutionary processes including mutation strategy, crossover strategy, selection process, and an adaptive parameter tuning mechanism. After evolving independently, the two sub-populations are combined into a single population to increase the diversity of the population, which prevents convergence to local optima and guarantees a wider exploration of the search space. To further improve performance in terms of composition time and overall QoS, the PDE-QSC approach incorporates a linear reduction strategy that adaptively decreases the population size by eliminating unpromising compositions. This strategy ensures a shorter computation time and allows the focus on the most potential compositions, contributing to a more efficient and effective composition process. Experimental results demonstrate that the PDE-QSC approach outperforms five baseline approaches by achieving a good balance between overall QoS and composition time. More specifically, the performance improvement of the PDE-QSC approach in terms of composition quality compared to sequential exploration-based approaches can be up to 64%, and up to 60% compared with parallel exploration-based ones while maintaining a reasonable composition time. This makes the proposed approach particularly suitable for large-scale IoT environments, where the number of concrete services and the complexity of the composition process may be extremely high.

The third contribution addresses complementary challenges that remained unresolved according to the insights gained from the first and second contributions, particularly the lack of semantic consideration and the assumption of the prior existence of an abstract composition plan. The third contribution introduces database concepts-driven approaches for autonomous QoS-aware service composition (HCFDSSC and ESFDSSC) to address the aforementioned limitations. Unlike existing all-matching and random matching semantic QoS-aware service composition approaches, the proposed approaches are designed to identify suboptimal compositions that meet both QoS and QoSM requirements, while significantly reducing composition time. The HCFDSSC and ESFDSSC approaches are carried out into three phases: *(i)* preprocessing, *(ii)* feasible abstract plans construction, and *(iii)* parallel service selection and

composition. Their main novelty lies in the use of functional dependencies, a fundamental concept from relational database theory, to model the services while capturing semantic constraints and ensuring the semantic feasibility of the compositions. The proposed approaches operate on a reduced semantic composition search space that is carefully constructed during the first two phases. The latter avoid additional composition time by focusing only on the most promising semantic concrete services, substantially decreasing the composition time required to find suboptimal compositions. The performance study demonstrates the effectiveness of the proposed approaches in finding suboptimal compositions on several evaluation criteria, including composition time, QoS, QoSM, and a utility value combining both metrics. For instance, the HCFDSSC approach outperforms the baseline approach by more than 84% in terms of combined utility value and by 90% in terms of composition time. Similarly, the ESFDSSC approach achieves improvement of over 103% and 87%, in terms of combined utility value and composition time, respectively. These performance gains are attributed to the reduction of the search space, the use of the functional dependencies model, and the parallel service selection and composition algorithms integrated within the HCFDSSC and ESFDSSC approaches. Another strength of the proposed approaches is their robustness in the presence of service failures where they can replace a suboptimal composition with an alternative solution, guaranteeing continuous service delivery without significant interruptions.

5.2 Future search directions

In this thesis, we not only outline the attractive challenges in the context of QoS-aware service composition, but also provide guidance on how to solve these challenges using recent and effective techniques. The future search directions are described as follows:

Towards multi-requesting optimization for concurrent QoS-aware service composition: The increasing number of users' requests with identical functional requirements (i.e., requests targeting the same abstract composition plan) highlights a critical yet rarely addressed research challenge in QoS-aware service composition, which is the effective handling of concurrent requests. Solving this problem comes up against several issues that negatively impact the composition quality and composition time. First, congestion problems arise due to competition for access to shared services, resulting in increased response time and latency. Second, the devices offering services are characterized by a limited energy capacity, and their simultaneous use by multiple requests can lead to failures during the composition process. Finally, the uncertainty of QoS properties increases due to the concurrent use of services and inter-blockage scenarios, which can be very frequent. Multitasking optimization methods [138] are recent methods that can effectively address the problem of QoS-aware

service composition with multiple concurrent requests especially since some methods include mechanisms that can detect and prevent the outlined challenges. In the context of service composition, multiple concurrent requests can be molded as a multiple tasks optimization problem that can be solved in parallel, where each task represents a user's request. Each request is associated with its own search space, a randomly generated population of compositions, and a utility function for their evaluation. The compositions of each request evolve simultaneously and independently from those of other concurrent requests. A global utility function, referred to as a skill factor, is then used to evaluate the compositions of each request with respect to the remaining requests. This evaluation allows for the use of shared knowledge across the compositions associated with each request, thus simultaneously finding the suboptimal composition for each request in a minimal composition time.

Autonomous QoS-aware microservice composition for next-generation systems: The ecosystems that have been developed as a single entity over time often become large and complex, making them difficult to maintain, scale, or adapt to evolving requirements. To address these challenges, many organizations are progressively migrating toward microservice architecture as part of modernization strategies to improve maintainability, scalability, and ensure high availability [139]. The microservice paradigm addresses the growing complexity of large-scale systems by decomposing the services into a set of small independent units, each focusing on a specific functionality [140]. Microservices are autonomous and decentralized units with diverse semantic descriptions, complicating service discovery and interoperability. With the exponential increase in the number of microservices, new challenges emerge, particularly in terms of semantic matching, discovery, and failure recovery. This creates a critical need for advanced methods to refine ontologies and improve semantic matching between services. In this context, Natural Language Processing (NLP) [141], and recently Large Language Models (LLMs) [142] offer an attractive solution for analyzing, interpreting, and inferring meaning from unstructured service descriptions, achieving more accurate semantic matching and improved discovery of microservices. Furthermore, the stochastic nature of IoT environments introduces additional challenges, including service failures, disappearances, and the emergence of new microservices. To address these challenges, an important line of research is to develop self-healing autonomous QoS-aware microservices composition approaches. A promising direction in this regard is the integration of digital twin technology [143] with artificial intelligence techniques. Digital twins create virtual representations of microservices and abstract composition plans, continuously supplied with real-time data. Based on the continuous monitoring of these data, artificial intelligence methods can be employed to detect anomalies, predict failure, and anticipate recovery solutions, enabling more resilient and autonomous microservice compositions.

Blockchain-assisted QoS-aware service composition for reliable IoT environments: IoT services are distributed software components located in different geographical areas, coordinating with each other via inputs and outputs to ensure the composition process. These interactions generate massive amounts of data that are vulnerable to loss, modification, or malicious attacks, requiring efficient and secure transfer mechanisms between services. Given the sensitive nature of these exchanges, integrating a secure mechanism into the autonomous QoS-aware service composition to ensure reliable data transfer is a critical challenge. To address this challenge, the Blockchain technology [144] has emerged as a promising solution to guarantee the security of data transfer during the service composition process. In this perspective, each transaction (i.e., input/output data transfer) is stored in a block, which is then added to the blockchain. Once validated, the block can no longer be altered, thus ensuring data reliability. To better illustrate this approach, let us consider an online payment composite service including three abstract services: authentication, payment, and notification. First, a suboptimal composition $C = \{CS_a, CS_p, CS_n\}$ is generated using a computational intelligence-based method (e.g., genetic algorithm). In this scenario, the service CS_a performs the authentication and produces output data O_{CS_a}. To ensure a reliable data transfer, a digital hash $Hash(O_{CS_a})$ is first generated for the output data of the authentication service CS_a. This hash is recorded in a blockchain smart contract and serves as a verifiable proof of the data's integrity. The output data O_{CS_a} is then transmitted to the payment service CS_p, which in turn generates a new digital hash for the received data and queries the blockchain to check that this hash matches the hash $Hash(O_{CS_a})$. Once the verification succeeds, the output data O_{CS_a} is accepted as valid input data I_{CS_p} for the execution of the payment service CS_p. Upon execution, this service produces its output data O_{CS_p}, generates the corresponding hash $Hash(O_{CS_p})$, and records this hash in the blockchain smart contract before transmitting the output data O_{CS_p} to the notification service CS_n. Following the same integrity verification mechanism, the notification service CS_n computes the hash of O_{CS_p}, compares it with the hash $Hash(O_{CS_p})$ stored in the blockchain smart contract, and if the verification succeeds, processes O_{CS_p} as valid input data I_{CS_n}. This data transfer mechanism allows for ensuring end-to-end data integrity through the execution of the service composition process.

Managing uncertain data in QoS-aware service composition: in IoT environments, numerous concrete services could be characterized by uncertain inputs and outputs, which introduces additional complexity to the autonomous service composition problem. This uncertainty can be attributed to several sources. First, the uncertainty may be caused by data variability. For instance, IoT sensors often produce fluctuating or incomplete data due to environmental changes. Second, uncertainty can result from the heterogeneity of data formats adopted by concrete services such as eXtensible Markup Language (XML), JavaScript

Object Notation (JSON), or binary formats, which complicates their interoperability. The lack of standardization further contributes to the uncertainty of input and output data. An important challenge lies therefore in developing robust approaches able to generate reliable compositions in the presence of uncertain input and output data, handle multi-conflicting QoS properties, and consider semantic information, while autonomously performing the service composition process. Several approaches can be investigated to solve this problem, in particular, the use of probabilistic models such as Bayesian networks [145] or Markov models [146] to capture the uncertainty of input and output data provided by semantic concrete services and predict the resulting service compositions. Furthermore, machine learning models such as meta-learning [147] can be used to predict missing or uncertain input and output values based on historical data.

List of publications

- **Asma Cherifi**, Mohamed Essaid Khanouche, and Zoubeyr Farah. "A parallel approach for user-centered QoS-aware services composition in the Internet of Things." *Engineering Applications of Artificial Intelligence*, vol.123, pp. 106277. Aug. 2023
- **Asma Cherifi**, Mohamed Essaid Khanouche, and Zoubeyr Farah. "A survey on centralized and decentralized QoS-aware services composition approaches in service community." *Under review by Computing*. Apr. 2025.
- **Asma Cherifi**, Mohamed Essaid Khanouche, and Zoubeyr Farah. "Database concepts-driven failure recovery algorithms for autonomous QoS-aware composition in service community." *Submitted to IEEE Transactions on Industrial Informatics*. Aug. 2025.
- Mohamed Essaid Khanouche, Nawel Atmani, and **Asma Cherifi**. "Improved Teaching Learning-Based QoS-Aware Services Composition for Internet of Things." *IEEE Systems Journal*, vol.14, no.3, pp. 806–817. Jan. 2020.
- Mohamed Essaid Khanouche, Atmani Nawel, **Asma Cherifi**, Chibani Abdelghani, Eric T Matson, and Yacine Amirat. "QoS-aware Agent Capabilities Composition in HARMS multi-agent systems." *Advances in Practical Applications of Survivable Agents and Multi-Agent Systems: The PAAMS Collection: 17th International Conference, PAAMS 2019*. Ávila, Spain, pp. 127–138. Jun. 2019.

Bibliography

[1] Maria Stoyanova, Yannis Nikoloudakis, Spyridon Panagiotakis, Evangelos Pallis, and Evangelos K Markakis. "A survey on the internet of things (IoT) forensics: challenges, approaches, and open issues," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1191–1221, Apr.-Jun 2020.

[2] Jianxin Wang, Ming K Lim, Chao Wang, and Ming-Lang Tseng. "The evolution of the Internet of Things (IoT) over the past 20 years," *Computers & Industrial Engineering*, vol. 155, pp. 107174, May 2021.

[3] Xiaoyun Li, Zibin Zheng, and Hong-Ning Dai. "When services computing meets blockchain: Challenges and opportunities," *Journal of Parallel and Distributed Computing*, vol. 150, pp. 1–14, Apr. 2021.

[4] Damian Arellanes and Kung-Kiu Lau. "Evaluating IoT service composition mechanisms for the scalability of IoT systems," *Future Generation Computer Systems*, vol. 108, pp. 827–848, Jul. 2020.

[5] Sylvain Cherrier and Yacine M Ghamri-Doudane. "The "object-as-a-service" paradigm." *2014 Global Information Infrastructure and Networking Symposium (GIIS)*. Montreal, QC, Canada, pp. 1–7. Sep. 2014.

[6] M Sathya, M Swarnamugi, P Dhavachelvan, and G Sureshkumar. "Evaluation of qos based web-service selection techniques for service composition," *International Journal of Software Engineering*, vol. 1, no. 5, pp. 73–90, 2010.

[7] Samir Awad, Abdelhamid Malki, Mimoun Malki, Mahmoud Barhamgi, and Djamal Benslimane. "Composing wot services with uncertain data," *Future Generation Computer Systems*, vol. 101, pp. 940–950, Dec. 2019.

[8] Vahideh Hayyolalam and Ali Asghar Pourhaji Kazem. "A systematic literature review on QoS-aware service composition and selection in cloud environment," *Journal of Network and Computer Applications*, vol. 110, pp. 52–74, May 2018.

[9] Yutu Liu, Anne H Ngu, and Liang Z Zeng. "QoS computation and policing in dynamic web service selection." *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*. New York NY USA, pp. 66–73. May 2004.

[10] Mohamed Essaid Khanouche, Yacine Amirat, Abdelghani Chibani, Moussa Kerkar, and Ali Yachir. "Energy-centered and QoS-aware services selection for Internet of Things," *IEEE Transactions on Automation Science and Engineering(T-ASE)*, vol. 13, no. 3, pp. 1256–1269, Jul. 2016.

[11] Mohammad Alrifai and Thomas Risse. "Combining global optimization with local selection for efficient QoS-aware service composition." *Proceedings of the 18th international conference on World wide web*. Madrid, Spain, pp. 881–890. Apr. 2009.

[12] Jinghai Rao and Xiaomeng Su. "A survey of automated web service composition methods." *International workshop on semantic web services and web process composition*. Berlin, Heidelberg, pp. 43–54. 2005.

[13] Martin Bauer, Nicola Bui, Jourik De Loof, Carsten Magerkurth, Andreas Nettsträter, Julinda Stefa, and Joachim W Walewski. "IoT Reference Model." In, *Enabling Things to Talk: Designing IoT Solutions with the IoT Architectural Reference Model*. Berlin, Germany: Springer, 2013, pp. 113–162.

[14] Rainer Berbner, Michael Spahn, Nicolas Repp, Oliver Heckmann, and Ralf Steinmetz. "Heuristics for QoS-aware web service composition." *2006 IEEE International Conference on Web Services (ICWS'06)*. Chicago, IL, USA, pp. 72–82. Dec. 2006.

[15] Mohamed Essaid Khanouche, Nawel Atmani, Asma Cherifi, Abdelghani Chibani, Eric T Matson, and Yacine Amirat. "QoS-aware Agent Capabilities Composition in HARMS multi-agent systems." *Advances in Practical Applications of Survivable Agents and Multi-Agent Systems: The PAAMS Collection: 17th International Conference, PAAMS 2019*. Ávila, Spain, pp. 127–138. Jun. 2019.

[16] Mohammad Alrifai, Thomas Risse, and Wolfgang Nejdl. "A hybrid approach for efficient Web service composition with end-to-end QoS constraints," *ACM Transactions on the Web (TWEB)*, vol. 6, no. 2, pp. 1–31, May 2012.

[17] Mohamed Essaid Khanouche, Ferhat Attal, Yacine Amirat, Abdelghani Chibani, and Moussa Kerkar. "Clustering-based and QoS-aware services composition algorithm for ambient intelligence," *Information Sciences*, vol. 482, pp. 419–439, May 2019.

[18] Danilo Ardagna and Barbara Pernici. "Adaptive service composition in flexible processes," *IEEE Transactions on software engineering*, vol. 33, no. 6, pp. 369–384, Jun. 2007.

[19] Salma Hameche, Mohamed Essaid Khanouche, and Abdelkamel Tari. "Mobility and energy efficient services composition algorithm with QoS guarantee for large scale Cyber–Physical–Social Systems," *Expert Systems with Applications*, vol. 249, pp. 123683, Sep. 2024.

[20] Salma Hameche, Mohamed Essaid Khanouche, Abdelghani Chibani, and Abdelkamel Tari. "A Group Teaching Optimization-Based Approach for Energy and QoS-Aware Internet of Things Services Composition," *Journal of Network and Systems Management*, vol. 32, no. 1, pp. 4, Oct. 2024.

[21] Yan Hai, Xin Xu, and Zhizhong Liu. "Dynamic multi-objective service composition based on improved social learning optimization algorithm," *Applied Soft Computing*, vol. 167, pp. 112266, Dec. 2024.

[22] Zhizhong Liu, Quan Z Sheng, Dianhui Chu, Xiaofei Xu, Hedan Zheng, and Kai Feng. "Proactive Recommendation of Composite Services in Multi-Access Edge Computing," *IEEE Transactions on Services Computing*, Mar.-Apr. 2024.

[23] Shuiguang Deng, Longtao Huang, Wei Tan, and Zhaohui Wu. "Top-K Automatic Service Composition: A Parallel Method for Large-Scale Service Sets," *IEEE Transactions on Automation Science and Engineering(T-ASE)*, vol. 11, no. 3, pp. 891–905, Jul. 2014.

[24] Mark H Burstein. "Dynamic invocation of semantic web services that use unfamiliar ontologies," *IEEE Intelligent Systems*, vol. 19, no. 4, pp. 67–73, Jul.-Aug. 2004.

[25] David Martin, Massimo Paolucci, Sheila McIlraith, Mark Burstein, Drew McDermott, Deborah McGuinness, Bijan Parsia, Terry Payne, Marta Sabou, Monika Solanki, et al. "Bringing semantics to web services: The OWL-S approach." *Semantic Web Services and Web Process Composition: First International Workshop, SWSWPC 2004*. San Diego, CA, USA, pp. 26–42. 2005.

[26] Jacek Kopeckỳ, Tomas Vitvar, Carine Bournez, and Joel Farrell. "Sawsdl: Semantic annotations for wsdl and xml schema," *IEEE Internet Computing*, vol. 11, no. 6, pp. 60–67, Dec. 2007.

[27] Chen Wang, Hui Ma, Gang Chen, and Sven Hartmann. "Evolutionary multitasking for semantic web service composition." *2019 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2490–2497. Jun. 2019.

[28] Chen Wang, Hui Ma, Gang Chen, and Sven Hartmann. "Using an estimation of distribution algorithm to achieve multitasking semantic web service composition," *IEEE Transactions on Evolutionary Computation (TEC)*, vol. 27, no. 3, pp. 490–504, Jun. 2023.

[29] Virginie Gabrel, Maude Manouvrier, Kamil Moreau, and Cecile Murat. "QoS-aware automatic syntactic service composition problem: Complexity and resolution," *Future Generation Computer Systems (FGCS)*, vol. 80, pp. 311–321, Mar. 2018.

[30] Siva Kumar Gavvala, Chandrashekar Jatoth, GR Gangadharan, and Rajkumar Buyya. "QoS-aware cloud service composition using eagle strategy," *Future Generation Computer Systems*, vol. 90, pp. 273–290, Aug. 2019.

[31] Mohamed Essaid Khanouche, Nawel Atmani, and Asma Cherifi. "Improved Teaching Learning-Based QoS-Aware Services Composition for Internet of Things," *IEEE Systems Journal*, vol. 14, no. 3, pp. 806–817, Sep. 2020.

[32] Fateh Seghir. "FDMOABC: fuzzy discrete multi-objective artificial bee colony approach for solving the non-deterministic QoS-driven web service composition problem," *Expert Systems with Applications*, vol. 167, pp. 114413, 2021.

[33] Qinglan Peng, Yunni Xia, MengChu Zhou, Xin Luo, Shu Wang, Yuandou Wang, Chunrong Wu, Shanchen Pang, and Mingwei Lin. "Reliability-Aware and Deadline-Constrained Mobile Service Composition Over Opportunistic Networks," *IEEE Transactions on Automation Science and Engineering(T-ASE)*, vol. 18, no. 3, pp. 1012–1025, Jul. 2021.

[34] Feng Li, Lin Zhang, Yongkui Liu, and Yuanjun Laili. "QoS-Aware Service Composition in Cloud Manufacturing: A Gale–Shapley Algorithm-Based Approach," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 7, pp. 2386–2397, Apr. 2018.

[35] Hongbing Wang, Danrong Yang, Qi Yu, and Yong Tao. "Integrating modified cuckoo algorithm and creditability evaluation for QoS-aware service composition," *Knowledge-Based Systems*, vol. 140, pp. 64–81, Jan. 2018.

[36] Chandrashekar Jatoth, GR Gangadharan, and Rajkumar Buyya. "Optimal fitness aware cloud service composition using an adaptive genotypes evolution based genetic algorithm," *Future Generation Computer Systems*, vol. 94, pp. 185–198, May 2019.

[37] Chandrashekar Jatoth, GR Gangadharan, and Ugo Fiore. "Optimal fitness aware cloud service composition using modified invasive weed optimization," *Swarm and evolutionary computation*, vol. 44, pp. 1073–1091, Feb. 2019.

[38] Zhi-Zhong Liu, Dian-Hui Chu, Cheng Song, Xiao Xue, and Bao-Yun Lu. "Social learning optimization (SLO) algorithm paradigm and its application in QoS-aware cloud service composition," *Information Sciences*, vol. 326, pp. 315–333, Jan. 2015.

[39] Xin Zhao, Liwei Shen, Xin Peng, and Wenyun Zhao. "Toward SLA-constrained service composition: An approach based on a fuzzy linguistic preference model and an evolutionary algorithm," *Information Sciences*, vol. 316, pp. 370–396, Sep. 2014.

[40] Yefeng Yang, Bo Yang, Shilong Wang, Tianguo Jin, and Shi Li. "An enhanced multi-objective grey wolf optimizer for service composition in cloud manufacturing," *Applied Soft Computing*, vol. 87, pp. 106003, Feb. 2020.

[41] Mohamadali Yaghoubi and Ali Maroosi. "Simulation and modeling of an improved multi-verse optimization algorithm for QoS-aware web service composition with service level agreements in the cloud environments," *Simulation Modelling Practice and Theory*, vol. 103, pp. 102090, Sep. 2020.

[42] Seyedsalar Sefati and Nima Jafari Navimipour. "A qos-aware service composition mechanism in the internet of things using a hidden-markov-model-based optimization algorithm," *IEEE Internet of Things Journal*, vol. 8, no. 20, pp. 15620–15627, Oct. 2021.

[43] Hong Jin, Shengping Lv, Zhou Yang, and Ying Liu. "Eagle strategy using uniform mutation and modified whale optimization algorithm for QoS-aware cloud service composition," *Applied Soft Computing*, vol. 114, pp. 108053, Jan. 2022.

[44] ChenYang Li, Jun Li, HuiLing Chen, and Ali Asghar Heidari. "Memetic Harris Hawks Optimization: Developments and perspectives on project scheduling and QoS-aware web service composition," *Expert Systems with Applications*, vol. 171, pp. 114529, Jun. 2021.

[45] Samar Haytamy and Fatma Omara. "Enhanced qos-based service composition approach in multi-cloud environment." *2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE)*. Aswan, Egypt, pp. 33–38. Mar. 2020.

[46] Amal Kouicem, Mohamed Essaid Khanouche, and Abdelkamel Tari. "Novel bat algorithm for QoS-aware services composition in large scale internet of things," *Cluster Computing*, vol. 25, pp. 1–15, May 2022.

[47] Rabah Boucetti, Ouassila Hioual, and Sofiane Mounine Hemam. "An approach based on genetic algorithms and neural networks for QoS-aware IoT services composition," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 8, pp. 5619–5632, Sep. 2022.

[48] Jiajun Zhou and Xifan Yao. "Multi-population parallel self-adaptive differential artificial bee colony algorithm with application in large-scale service composition for cloud manufacturing," *Applied Soft Computing*, vol. 56, pp. 379–397, Jul. 2017.

[49] Doaa H Elsayed, Mervat H Gheith, Eman S Nasr, and M Alaa El Din. "Integration of Parallel Genetic Algorithm and Q-learning for QoS-aware Web Service Composition." *2017 12th International Conference on Computer Engineering and Systems (ICCES)*. Cairo, Egypt, pp. 221–226. Dec. 2017.

[50] Jiajun Zhou, Xifan Yao, Yingzi Lin, Felix TS Chan, and Yun Li. "An adaptive multi-population differential artificial bee colony algorithm for many-objective service composition in cloud manufacturing," *Information Sciences*, vol. 456, pp. 50–82, Aug. 2018.

[51] Mohamed Essaid Khanouche, Sihem Mouloudj, and Melissa Hammoum. "Two-steps QoS-aware services composition algorithm for Internet of Things." *Proceedings of the 3rd International Conference on Future Networks and Distributed Systems (ICFNDS)*. Paris, France, pp. 1–6. Jul. 2019.

[52] Idir Aoudia, Laid Kahloul, Saber Benharzallah, and Okba Kazar. "QoS-aware service composition in Fog-IoT computing using multi-population genetic algorithm." *2020 21st International Arab Conference on Information Technology (ACIT)*. Giza, Egypt, pp. 1–9. Nov. 2020.

[53] Godar J Ibrahim, Tarik A Rashid, and Mobayode O Akinsolu. "An energy efficient service composition mechanism using a hybrid meta-heuristic algorithm in a mobile cloud environment," *Journal of Parallel and Distributed Computing*, vol. 143, pp. 77–87, Sep. 2020.

[54] Jiajun Zhou, Liang Gao, Xifan Yao, Chunjiang Zhang, Felix TS Chan, and Yingzi Lin. "An adaptive dual-population evolutionary paradigm with adversarial search: Case study on many-objective service consolidation," *Applied Soft Computing*, vol. 90, pp. 106160, May 2020.

[55] Helan Liang, Bincheng Ding, Yanhua Du, and Fanzhang Li. "Parallel optimization of QoS-aware big service processes with discovery of skyline services," *Future Generation Computer Systems*, vol. 125, pp. 496–514, Dec. 2021.

[56] Pablo Rodriguez-Mier, Manuel Mucientes, and Manuel Lama. "Hybrid optimization algorithm for large-scale QoS-aware service composition," *IEEE Transactions on Services Computing (TSC)*, vol. 10, no. 4, pp. 547–559, Jul.-Aug. 2017.

[57] Jing Li, Yuhong Yan, and Daniel Lemire. "Full solution indexing for top-k web service composition," *IEEE Transactions on Services Computing (TSC)*, vol. 11, no. 3, pp. 521–533, May-Jun. 2018.

[58] Jing Li, Guodong Fan, Ming Zhu, and Yuhong Yan. "Pre-joined semantic indexing graph for qos-aware service composition." *Proceedings of the IEEE International Conference on Web Services (ICWS)*. Milan, Italy, pp. 116–120. Jul. 2019.

[59] Soumi Chattopadhyay and Ansuman Banerjee. "QoS-aware automatic Web service composition with multiple objectives," *ACM Transactions on the Web (TWEB)*, vol. 14, no. 3, pp. 1–38, May 2020.

[60] Alexandre Sawczuk da Silva, Yi Mei, Hui Ma, and Mengjie Zhang. "Fragment-based genetic programming for fully automated multi-objective web service composition." *Proceedings of the Genetic and Evolutionary Computation Conference (GEC)*. Berlin, Germany, pp. 353–360. Jul. 2017.

[61] Soheila Sadeghiram, Hui Ma, and Gang Chen. "Composing distributed data-intensive Web services using a flexible memetic algorithm." *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*. Wellington, New Zealand, pp. 2832–2839. Jun. 2019.

[62] Alexandre Sawczuk da Silva, Yi Mei, Hui Ma, and Mengjie Zhang. "Evolutionary computation for automatic web service composition: an indirect representation approach," *Journal of Heuristics*, vol. 24, pp. 425–456, Apr. 2018.

[63] Chen Wang, Hui Ma, Gang Chen, and Sven Hartmann. "Memetic eda-based approaches to qos-aware fully automated semantic web service composition," *IEEE Transactions on Evolutionary Computation (TEC)*, vol. 26, no. 3, pp. 570–584, Jun. 2022.

[64] Alexandre Sawczuk da Silva, Yi Mei, Hui Ma, and Mengjie Zhang. "A memetic algorithm-based indirect approach to web service composition." *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*. Vancouver, BC, Canada, pp. 3385–3392. Jul. 2016.

[65] Chen Wang, Hui Ma, Gang Chen, Sven Hartmann, and Jürgen Branke. "Robustness estimation and optimisation for semantic web service composition with stochastic service failures," *IEEE Transactions on Emerging Topics in Computational Intelligence (TETCI)*, vol. 6, no. 1, pp. 77–92, Feb. 2022.

[66] Soheila Sadeghiram, Hui Ma, and Gang Chen. "A distance-based genetic algorithm for robust data-intensive web service composition in dynamic bandwidth environment." *2020 IEEE International Conference on Services Computing (SCC)*. Beijing, China, pp. 248–255. Nov. 2020.

[67] Soheila Sadeghiram, Hui Ma, and Gang Chen. "Multi-objective distributed Web service composition—a link-dominance driven evolutionary approach," *Future Generation Computer Systems (FGCS)*, vol. 143, pp. 163–178, Jun. 2023.

[68] Soheila Sadeghiram, Hui Ma, and Gang Chen. "Priority-based selection of individuals in memetic algorithms for distributed data-intensive web service compositions," *IEEE Transactions on Services Computing (TSC)*, vol. 15, no. 5, pp. 2939–2953, Sept.-Oct. 2022.

[69] Alexandre Sawczuk Da Silva, Hui Ma, Yi Mei, and Mengjie Zhang. "A hybrid memetic approach for fully automated multi-objective web service composition." *Proceedings of the IEEE International Conference on Web Services (ICWS)*. San Francisco, CA, USA, pp. 26–33. Sep. 2018.

[70] Zhaoning Wang, Bo Cheng, Wenkai Zhang, and Junliang Chen. "QoS-aware automatic service composition based on service execution timeline with multi-objective optimization." *Proceedings of the IEEE International Conference on Services Computing (SCC)*. Beijing, China, pp. 296–303. Dec. 2020.

[71] Cheyma Ben Njima, Chirine Ghedira Guegan, Youssef Gamha, and Lotfi Ben Romdhane. "Web Service Composition in Mobile Environment: A survey of Techniques," *IEEE Transactions on Services Computing (TSC)*, Mar.-Apr. 2024.

[72] Qiping She, Xiaochao Wei, Guihua Nie, and Donglin Chen. "QoS-aware cloud service composition: A systematic mapping study from the perspective of computational intelligence," *Expert Systems with Applications*, vol. 138, pp. 112804, Dec. 2019.

[73] Melissa Hammoum, Mohamed Essaid Khanouche, Nadjet Khoulalene, and Boualem Benatallah. "Uncertainty QoS-aware services composition: a systematic literature review for services community," *Service Oriented Computing and Applications*, vol. 18, no. 2, pp. 121–143, Jun. 2024.

[74] Firas D Ahmed and Mazlina Abdul Majid. "Towards agent-based petri net decision making modelling for cloud service composition: A literature survey," *Journal of Network and Computer Applications*, vol. 130, pp. 14–38, Mar. 2019.

[75] Vahideh Hayyolalam, Behrouz Pourghebleh, Ali Asghar Pourhaji Kazem, and Ali Ghaffari. "Exploring the state-of-the-art service composition approaches in cloud manufacturing systems to enhance upcoming techniques," *The International Journal of Advanced Manufacturing Technology*, vol. 105, pp. 471–498, Nov. 2019.

[76] Saied Asghari and Nima Jafari Navimipour. "Nature inspired meta-heuristic algorithms for solving the service composition problem in the cloud environments," *International Journal of Communication Systems*, vol. 31, no. 12, pp. e3708, May 2018.

[77] Mohammadreza Razian, Mohammad Fathian, Rami Bahsoon, Adel N Toosi, and Rajkumar Buyya. "Service composition in dynamic environments: A systematic review and future directions," *Journal of Systems and Software*, vol. 188, pp. 111290, Jun. 2022.

[78] Mohammad Masdari, Mehdi Nozad Bonab, and Suat Ozdemir. "QoS-driven metaheuristic service composition schemes: a comprehensive overview," *Artificial Intelligence Review*, vol. 54, pp. 3749–3816, Jun. 2021.

[79] Yang Syu, Shang-Pin Ma, Jong-Yih Kuo, and Yong-Yi FanJiang. "A survey on automated service composition methods and related techniques." *2012 IEEE Ninth International Conference on Services Computing*, pp. 290–297. Aug. 2012.

[80] Parvaneh Asghari, Amir Masoud Rahmani, and Hamid Haj Seyyed Javadi. "Service composition approaches in IoT: A systematic review," *Journal of Network and Computer Applications*, vol. 120, pp. 61–77, Oct. 2018.

[81] Alexandre Sawczuk da Silva, Hui Ma, Yi Mei, and Mengjie Zhang. "A survey of evolutionary computation for web service composition: A technical perspective," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 4, pp. 538–554, Aug. 2020.

[82] Pablo García-Sánchez, Gustavo Romero, Jesús González, Antonio Miguel Mora, Maribel García Arenas, Pedro Ángel Castillo, Carlos Fernandes, and Juan Julián Merelo. "Studying the effect of population size in distributed evolutionary algorithms on heterogeneous clusters," *Applied Soft Computing*, vol. 38, pp. 530–547, Jan. 2016.

[83] Adam P Piotrowski. "Review of differential evolution population size," *Swarm and Evolutionary Computation*, vol. 32, pp. 1–24, Feb. 2017.

[84] Ryoji Tanabe and Alex S Fukunaga. "Improving the search performance of SHADE using linear population size reduction." *2014 IEEE congress on evolutionary computation (CEC)*. Beijing, China, pp. 1658–1665. Jul. 2014.

[85] Mostafa Z Ali, Noor H Awad, Ponnuthurai Nagaratnam Suganthan, and Robert G Reynolds. "An adaptive multipopulation differential evolution with dynamic population reduction," *IEEE transactions on cybernetics (TCYB)*, vol. 47, no. 9, pp. 2768–2779, Sep. 2017.

[86] Laizhong Cui, Genghui Li, Zexuan Zhu, Qiuzhen Lin, Zhenkun Wen, Nan Lu, Ka-Chun Wong, and Jianyong Chen. "A novel artificial bee colony algorithm with an adaptive population size for numerical function optimization," *Information Sciences*, vol. 414, pp. 53–67, Nov. 2017.

[87] Mengnan Tian and Xingbao Gao. "Differential evolution with neighborhood-based adaptive evolution mechanism for numerical optimization," *Information Sciences*, vol. 478, pp. 422–448, Apl. 2019.

[88] Zhenyu Meng, Jeng-Shyang Pan, and Kuo-Kun Tseng. "PaDE: An enhanced Differential Evolution algorithm with novel control parameter adaptation schemes for numerical optimization," *Knowledge-Based Systems*, vol. 168, pp. 80–99, Mar. 2019.

[89] Bo Wei, Xuewen Xia, Fei Yu, Yinglong Zhang, Xing Xu, Hongrun Wu, Ling Gui, and Guoliang He. "Multiple adaptive strategies based particle swarm optimization algorithm," *Swarm and Evolutionary Computation*, vol. 57, pp. 100731, Sep. 2020.

[90] Rohit Salgotra, Urvinder Singh, Sriparna Saha, and Amir H Gandomi. "Self adaptive cuckoo search: analysis and experimentation," *Swarm and Evolutionary Computation*, vol. 60, pp. 100751, Feb. 2021.

[91] Xi Yang and Wenyin Gong. "Opposition-based JAYA with population reduction for parameter estimation of photovoltaic solar cells and modules," *Applied Soft Computing*, vol. 104, pp. 107218, Jun. 2021.

[92] Asma Cherifi, Mohamed Essaid Khanouche, and Zoubeyr Farah. "A comprehensive review on autonomous and plan-based QoS-aware service composition approaches in service community," *Under review by Computing*, Apr. 2025.

[93] Asma Cherifi, Mohamed Essaid Khanouche, Yacine Amirat, and Zoubeyr Farah. "A parallel approach for user-centered QoS-aware services composition in the Internet of Things," *Engineering Applications of Artificial Intelligence*, vol. 123, pp. 106277, Aug. 2023.

[94] Asma Cherifi, Mohamed Essaid Khanouche, and Zoubeyr Farah. "Database concepts-driven failure recovery algorithms for autonomous QoS-aware composition in service community," *Submitted to IEEE Transactions on Industrial Informatics*, Aug. 2025.

[95] Antonio Brogi and Sara Corfini. "Ontology-and behavior-aware discovery of Web service compositions," *International Journal of Cooperative Information Systems*, vol. 17, no. 03, pp. 319–347, 2008.

[96] Amin Mesmoudi, Michaël Mrissa, and Mohand-Said Hacid. "Combining configuration and query rewriting for Web service composition." *2011 IEEE International Conference on Web Services*. Washington, DC, USA, pp. 113–120. Sep. 2011.

[97] D. Maier. The Theory of Relational Databases. Computer software engineering series. Computer Science Press, 1983. ISBN: 9780914894421. URL: https://books.google.dz/books?id=INdQAAAAMAAJ.

[98] Jeffrey D Ullman. Principles of database systems. Galgotia publications, 1983.

[99] Edmund K Burke and Yuri Bykov. "The late acceptance hill-climbing heuristic," *European Journal of Operational Research*, vol. 258, no. 1, pp. 70–78, Apr. 2017.

[100] Phillip J Chase. "Algorithm 382: Combinations of m out of n objects [g6]," *Communications of the ACM*, vol. 13, no. 6, pp. 368, 1970.

[101] Angel Lagares Lemos, Florian Daniel, and Boualem Benatallah. "Web service composition: a survey of techniques and tools," *ACM Computing Surveys (CSUR)*, vol. 48, no. 3, pp. 1–41, Dec. 2015.

[102] Dominic D Martinelli. "Generative machine learning for de novo drug discovery: A systematic review," *Computers in Biology and Medicine*, vol. 145, pp. 105403, Jun. 2022.

[103] Aboul Ella Hassanien, Eiman Tamah Al-Shammari, and Neveen I Ghali. "Computational intelligence techniques in bioinformatics," *Computational biology and chemistry*, vol. 47, pp. 37–47, Dec. 2013.

[104] Kerstin Dächert, Jochen Gorski, and Kathrin Klamroth. "An augmented weighted Tchebycheff method with adaptively chosen parameters for discrete bicriteria optimization problems," *Computers & Operations Research*, vol. 39, no. 12, pp. 2929–2943, Dec. 2012.

[105] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[106] Vivek Sharma, Ashish Kumar Tripathi, and Himanshu Mittal. "Technological revolutions in smart farming: Current trends, challenges & future directions," *Computers and Electronics in Agriculture*, vol. 201, pp. 107217, Oct. 2022.

[107] Babak Poursartip, Arash Fathi, and John L Tassoulas. "Large-scale simulation of seismic wave motion: A review," *Soil Dynamics and Earthquake Engineering*, vol. 129, pp. 105909, Feb. 2020.

[108] Derrick N Joanes and Christine A Gill. "Comparing measures of sample skewness and kurtosis," *Journal of the Royal Statistical Society: Series D (The Statistician)*, vol. 47, no. 1, pp. 183–189, 1998.

[109] David E Goldberg. "Cenetic algorithms in search," *Optimization, Machine Learning*, 1989.

[110] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, Philip S Yu, et al. "Top 10 algorithms in data mining," *Knowledge and information systems*, vol. 14, pp. 1–37, 2008.

[111] Anil K Jain. "Data clustering: 50 years beyond K-means," *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, Jun. 2010.

[112] Soumi Chattopadhyay and Ansuman Banerjee. "QoS constrained large scale web service composition using abstraction refinement," *IEEE Transactions on Services Computing (TSC)*, vol. 13, no. 3, pp. 529–544, May-Jun. 2020.

[113] Soumi Chattopadhyay, Ansuman Banerjee, and Nilanjan Banerjee. "A fast and scalable mechanism for web service composition," *ACM Transactions on the Web (TWEB)*, vol. 11, no. 4, pp. 1–36, 2017.

[114] Liangzhao Zeng, Boualem Benatallah, Marlon Dumas, Jayant Kalagnanam, and Quan Z Sheng. "Quality driven web services composition." *Proceedings of the 12th international conference on World Wide Web*, pp. 411–421. May 2003.

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

[116] Ajay Bansal, M Brian Blake, Srividya Kona, Steffen Bleul, Thomas Weise, and Michael C Jaeger. "WSC-08: continuing the web services challenge." *2008 10th IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services*, pp. 351–354. Jul. 2008.

[117] Srividya Kona, Ajay Bansal, M Brian Blake, Steffen Bleul, and Thomas Weise. "WSC-2009: a quality of service-oriented web services challenge." *2009 IEEE conference on commerce and enterprise computing*, pp. 487–490. Jul. 2009.

[118] Ulrich Küster, Birgitta König-Ries, and Andreas Krug. "Opossum-an online portal to collect and share sws descriptions." *2008 IEEE International Conference on Semantic Computing*, pp. 480–481. Aug. 2008.

[119] Zibin Zheng, Yilei Zhang, and Michael R Lyu. "Investigating QoS of real-world web services," *IEEE Transactions on Services Computing (TSC)*, vol. 7, no. 1, pp. 32–39, Jan.-Mar. 2014.

[120] Wei Jiang, Dongwon Lee, and Songlin Hu. "Large-scale longitudinal analysis of soap-based and restful web services." *2012 ieee 19th international conference on web services*, pp. 218–225. Jun. 2012.

[121] "WS-Challenge. (2010) Testsetgenerator2009. [Online]. Available: https://code.google.com/p/wsc-pku-tcs/downloads/list."

[122] "S. Kona et al. 2005. The web services challenge. In Proceedings of the ICEBE. Retrieved from http://www.comp.hkbu.edu.hk/simctr/wschallenge/."

[123] Levent Yilmaz, Simon JE Taylor, Richard Fujimoto, and Frederica Darema. "Panel: The future of research in modeling & simulation." *Proceedings of the Winter Simulation Conference 2014*, pp. 2797–2811. Dec. 2014.

[124] Mohamed Essaid Khanouche, Ferhat Attal, Yacine Amirat, Abdelghani Chibani, and Moussa Kerkar. "Clustering-based and QoS-aware services composition algorithm for ambient intelligence," *Information Sciences*, vol. 482, pp. 419–439, May 2019.

[125] Rainer Storn and Kenneth Price. "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, Dec. 1997.

[126] Jingqiao Zhang and Arthur C Sanderson. "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on evolutionary computation*, vol. 13, no. 5, pp. 945–958, May 2009.

[127] Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, Mar. 2011.

[128] Adrian Satja Kurdija, Marin Silic, Goran Delac, and Klemo Vladimir. "Fast multi-criteria service selection for multi-user composite applications," *IEEE Transactions on Services Computing (TSC)*, vol. 15, no. 1, pp. 174–187, Jan.-Feb 2022.

[129] Massimo Paolucci, Takahiro Kawamura, Terry R Payne, and Katia Sycara. "Semantic matching of web services capabilities." *The Semantic Web—ISWC 2002: First International Semantic Web Conference Sardinia, Italy, June 9–12, 2002 Proceedings 1*, pp. 333–347. Jan. 2002.

[130] KC Shet, U Dinesh Acharya, et al. "A new similarity measure for taxonomy based on edge counting," *arXiv preprint arXiv:1211.4709*, Nov. 2012.

[131] Philip A Bernstein. "Synthesizing third normal form relations from functional dependencies," *ACM Transactions on Database Systems (TODS)*, vol. 1, no. 4, pp. 277–298, Dec. 1976.

[132] James Clifford and David S Warren. "Formal semantics for time in databases," *ACM Transactions on Database Systems (TODS)*, vol. 8, no. 2, pp. 214–254, Jun. 1983.

[133] Catriel Beeri, Ronald Fagin, and John H Howard. "A complete axiomatization for functional and multivalued dependencies in database relations." *Proceedings of the 1977 ACM SIGMOD international conference on Management of data*, pp. 47–61. Agu. 1977.

[134] Victor Vianu. "Dynamic functional dependencies and database aging," *Journal of the ACM (JACM)*, vol. 34, no. 1, pp. 28–59, Jan. 1987.

[135] Marco A Casanova, Ronald Fagin, and Christos H Papadimitriou. "Inclusion dependencies and their interaction with functional dependencies." *Proceedings of the 1st ACM SIGACT-SIGMOD symposium on Principles of database systems*, pp. 171–176. Mar. 1982.

[136] Thorsten Papenbrock and Felix Naumann. "A hybrid approach to functional dependency discovery." *Proceedings of the 2016 International Conference on Management of Data*, pp. 821–833. Jun. 2016.

[137] JS Appleby, DV Blake, and EA Newman. "Techniques for producing school timetables on a computer and their application to other scheduling problems," *The Computer Journal*, vol. 3, no. 4, pp. 237–245, Jan. 1961.

[138] Tingyang Wei, Shibin Wang, Jinghui Zhong, Dong Liu, and Jun Zhang. "A review on evolutionary multitask optimization: Trends and challenges," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 5, pp. 941–960, Oct. 2022.

[139] Jonas Fritzsch, Justus Bogner, Stefan Wagner, and Alfred Zimmermann. "Microservices migration in industry: intentions, strategies, and challenges." *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 481–490. Sep.-Oct 2019.

[140] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. "Microservices: yesterday, today, and tomorrow," *Present and ulterior software engineering*, pp. 195–216, Sep. 2017.

[141] KR1442 Chowdhary and KR Chowdhary. "Natural language processing," *Fundamentals of artificial intelligence*, pp. 603–649, Apr. 2020.

[142] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. "A comprehensive overview of large language models," *ACM Transactions on Intelligent Systems and Technology*, May 2025.

[143] Qinglin Qi, Fei Tao, Tianliang Hu, Nabil Anwer, Ang Liu, Yongli Wei, Lihui Wang, and Andrew YC Nee. "Enabling technologies and tools for digital twin," *Journal of Manufacturing Systems*, vol. 58, pp. 3–21, Jan. 2021.

[144] Rui Zhang, Rui Xue, and Ling Liu. "Security and privacy on blockchain," *ACM Computing Surveys (CSUR)*, vol. 52, no. 3, pp. 1–34, Jul. 2019.

[145] David Heckerman. "A tutorial on learning with Bayesian networks," *Learning in graphical models*, pp. 301–354, 1998.

[146] Siddhartha Chib. "Markov chain Monte Carlo methods: computation and inference," *Handbook of econometrics*, vol. 5, pp. 3569–3649, 2001.

[147] Anna Vettoruzzo, Mohamed-Rafik Bouguelia, Joaquin Vanschoren, Thorsteinn Rögnvaldsson, and KC Santosh. "Advances and challenges in meta-learning: A technical review," *IEEE transactions on pattern analysis and machine intelligence*, vol. 46, no. 7, pp. 4763–4779, Jan. 2024.

Abstract

The rapid growth of the Internet of Things (IoT) has led to the proliferation of services, making Quality of Service (QoS)-aware service composition a critical challenge. This thesis addresses this issue through three complementary contributions. The first contribution presents a systematic literature review of QoS-aware service composition approaches, introducing a two-layered taxonomy that distinguishes between plan-based and autonomous approaches. This review identifies key limitations in the state-of-the-art, such as the lack of semantic matching consideration, the assumption of a prior existence of an abstract composition plan, limited scalability, and the use of fixed population sizes. These findings highlight the need for more efficient and adaptive approaches. To overcome these issues, the second contribution proposes the parallel differential evolution-based approach with population size reduction for QoS-aware services composition (PDE-QSC). By evolving two parallel sub-populations with distinct strategies and adaptively reducing population size, the PDE-QSC approach improves the composition quality and computation time compared to five baseline approaches. However, this approach still relies on the existence of an abstract plan and does not account for the semantic matching aspect. The third contribution addresses these remaining limitations by introducing two database concepts-driven approaches for autonomous QoS-aware semantic service composition (HCFDSSC and ESFDSSC). The proposed approaches leverage functional dependency theory to ensure semantic feasibility, reduce the search space, achieve fault tolerance in the case of service failures, and generate high-quality compositions. This thesis advances QoS-aware service composition in IoT by linking plan-based optimization and autonomous semantic approaches, opening new perspectives for scalable, adaptive, and resilient service systems.

Keywords: Quality of Service (QoS), Service selection, Quality of Semantic Matching (QoSM), Autonomous service composition, Multi-population Differential Evolution, Population size reduction, Functional dependencies, Semantic services composition.

Résumé

La croissance rapide de l'Internet des objets (IoT) a entraîné une prolifération des services, faisant de la composition de services tenant compte de la qualité de service (QoS) un défi majeur. Cette thèse aborde cette question à travers trois contributions complémentaires. La première contribution présente une revue des approches de composition de services tenant compte de la QoS, en introduisant une taxonomie à deux niveaux qui distingue les approches basées sur un plan et les approches autonomes. Cette revue identifie les principales limites de l'état de l'art, telles que l'absence de prise en compte de la correspondance sémantique, l'hypothèse de l'existence préalable d'un plan de composition abstrait, la scalabilité limitée et l'utilisation de populations à tailles fixes. Ces constats soulignent la nécessité d'approches plus efficaces et adaptatives. Pour pallier ces problèmes, la deuxième contribution propose une approche parallèle basée sur l'évolution différentielle avec réduction de la taille de la population pour la composition de services tenant compte de la QoS (PDE-QSC). En faisant évoluer deux sous-populations parallèles avec des stratégies distinctes et en réduisant de manière adaptative la taille de la population, l'approche PDE-QSC améliore la qualité de la composition et le temps de calcul par rapport à cinq approches de référence. Cependant, l'approche PDE-QSC repose toujours sur l'existence d'un plan abstrait et ne tient pas compte de la correspondance sémantique de services. La troisième contribution remédie à ces limitations en introduisant deux approches basées sur des concepts de base de données pour la composition sémantique autonome de services avec prise en compte de la QoS (HCFDSSC et ESFDSSC). Les approches proposées s'appuient sur la théorie des dépendances fonctionnelles pour garantir la faisabilité sémantique, réduire l'espace de recherche, assurer la tolérance aux défaillances de services et générer des compositions de haute qualité. Cette thèse fait progresser la composition de services tenant compte de la QoS dans l'IoT en reliant l'optimisation basée sur des plans et les approches sémantiques autonomes, ouvrant ainsi de nouvelles perspectives pour des systèmes de services évolutifs et adaptatifs.

Mots Clée : Qualité de service (QoS), Sélection de services, Qualité de la correspondance sémantique (QoSM), Composition autonome de services, Évolution différentielle multi-population, Réduction de la taille de la population, Dépendances fonctionnelles, Composition sémantique de services.

ملخص

أدى النمو السريع لإنترنت الأشياء (IoT) إلى انتشار الخدمات، مما جعل تكوين الخدمات التي تراعي جودة الخدمة (QoS) تحديًا بالغ الأهمية. تتناول هذه الأطروحة هذه المسألة من خلال ثلاث مساهمات متكاملة. تقدم المساهمة الأولى مراجعة منهجية للأدبيات المتعلقة بنهج تكوين الخدمات التي تراعي جودة الخدمة، وتقدم تصنيفًا من طبقتين يميز بين النهج القائم على التخطيط والنهج المستقل. تحدد هذه المراجعة القيود الرئيسية في أحدث ما توصلت إليه التكنولوجيا، مثل عدم مراعاة المطابقة الدلالية، وافتراض وجود خطة تكوين مجردة مسبقًا، وقابلية التوسع المحدودة، واستخدام أحجام سكانية ثابتة. تسلط هذه النتائج الضوء على الحاجة إلى نُهج أكثر كفاءة وتكيفًا. للتغلب على هذه المشكلات، تقترح المساهمة الثانية نهجًا موازيًا قائمًا على التطور التفاضلي مع تقليل حجم السكان لتكوين الخدمات التي تراعي جودة الخدمة .(PDE-QSC) من خلال تطوير مجموعتين فرعيتين متوازيتين باستراتيجيات متميزة وتقليل حجم المجموعة بشكل تكيفي، يحسن نهج PDE-QSC جودة التكوين ووقت الحساب مقارنة بخمسة نُهج أساسية. ومع ذلك، لا يزال هذا النهج يعتمد على وجود خطة مجردة ولا يأخذ في الاعتبار جانب المطابقة الدلالية. تتناول المساهمة الثالثة هذه القيود المتبقية من خلال تقديم نهجين قائمين على مفاهيم قاعدة البيانات لتكوين خدمات دلالية مستقلة تراعي جودة الخدمة HCFDSSC) و .(ESFDSSC تستفيد النهج المقترحة من نظرية التبعية الوظيفية لضمان الجدوى الدلالية، وتقليل مساحة البحث، وتحقيق التسامح مع الأخطاء في حالة فشل الخدمة، و إنشاء تركيبات عالية الجودة. تقدم هذه الأطروحة تركيب الخدمة المراعية لجودة الخدمة في إنترنت الأشياء من خلال ربط التحسين القائم على الخطة والنهج الدلالية المستقلة، مما يفتح آفاقًا جديدة لأنظمة الخدمة القابلة للتطوير والتكيف والمرونة.

الكلمات المفتاحية: جودة الخدمة (QoS) اختيار الخدمة، جودة المطابقة الدلالية (QoSM) تكوين الخدمة المستقلة، التطور التفاضلي متعدد السكان، تقليل حجم السكان، التبعيات الوظيفية، تكوين الخدمات الدلالية.

.