الجمهورية الجزائرية الديمقراطية الشعبية

Democratic and Popular Algerian Republic

Ministry of Higher Education and Scientific Research

FERHAT ABBAS UNIVERSITY - SETIF1

FACULTY OF SCIENCES

THESIS

Presented for the Department of Computer Science

For the graduation of

PhD

Domain: Mathematics and Computer Science

Field: Computer Science **Option: Software Engineering and human-machine interfaces**

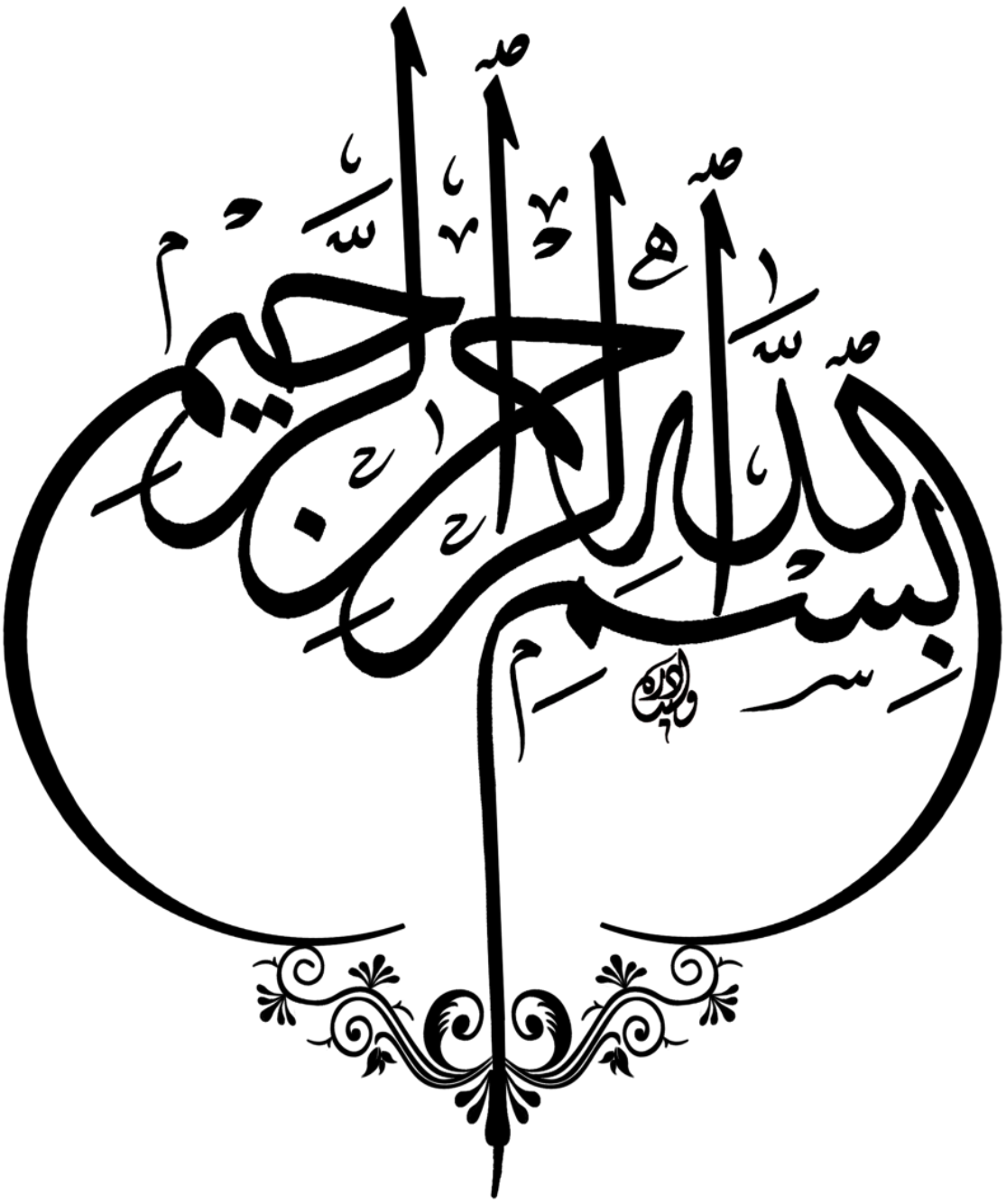
By: Ghizlane KHABABA

THEME

Nondeterministic algorithms for solving the dynamic QoS-aware service composition under ambiguous QoS parameters

Thesis defended in 09/07/2025 in front of the jury members:

LAKHFIF Abdelaziz	**MCA**	**Faculty of Sciences, UFAS 1**	**President**
BESSOU Sadik	**MCA**	**Faculty of Sciences, UFAS 1.**	**Supervisor**
SEGHIR Fateh	**MCA**	**Faculty of Technology, UFAS 1**	**Co-supervisor**
AKHROUF Samir	**Professor**	**University Mohamed Boudiaf MSILA**	**Examiner**
BEGHOURA Med Lamine	**MCA**	**University of Bordj Bou Arreridj**	**Examiner**
BOUBATRA Abdelhak	**Professor**	**Retired of University of Bordj Bou Arreridj**	**Invited**
TOUAHRIA Mohamed	**Professor**	**Retired of UFAS 1**	**Invited**


بسم الله الرحمن الرحيم

Dedication

In the name of Almighty Allah,
to my beloved parents,
whose unwavering love, sacrifices, and faith in me
have made this journey—and this work—possible.

Acknowledgments

All praise is due to Allah, the Most Beneficent, the Most Merciful, who granted me health, strength, and courage to complete this work despite all obstacles. As the Prophet Muhammad (peace be upon him) said : “He who is thankless to people is thankless to Allah.”

I would like to express my deepest gratitude to my supervisors : first to Dr. Bessou Sadik, for his invaluable guidance, insightful comments, and unwavering support throughout this research ; and to Dr. Fateh Seghir, whose constructive feedback, patience, and dedication helped me refine my ideas and correct key errors. I am profoundly indebted to both for their encouragement and for the time they invested in mentoring me.

My sincere thanks go to my internship mentors, whose expertise greatly enriched my PhD journey : Dr. Nor Hazlyna Harun, Head of the Data Science Research Lab at Universiti Utara Malaysia, and Prof. Saad Harous, Head of the Department of Computer Science at the University of Sharjah.

I also wish to thank the members of the examination jury for their careful reading and thoughtful evaluation of this thesis :

— Dr. Abdelaziz Lakhfif

— Prof. Akhrouf Samir

— Dr. Baghoura Mohamed Lamine

And I am particularly honored to have invited members whose expertise enriched this work :

— Prof. Touahria Mohamed

— Prof. Boubetra Abdelhak

I owe a debt of love and gratitude to my family—my mother, my father, my brother, and my sister—for their unwavering support during the rough patches of this PhD journey and their undivided happiness in my accomplishments.

To my friends—Sisters Farizah and Amera Hanna—and special thanks to Sameera, Manel, and Safia : your companionship and encouragement have meant the world to me.

And last, but by no means least, to my bestie, Asma : thank you for being my constant source of strength, laughter, and inspiration.

Finally, I am forever grateful to everyone who has walked alongside me on this path ; this work would not have been possible without you all.

Abstract

This thesis addresses three pivotal challenges in web services environment : QoS-aware service composition under uncertainty, QoS prediction in dynamic environments, and a comprehensive review of collaborative filtering techniques for QoS prediction. The first contribution introduces an extended artificial bee colony algorithm with local search (EABC) to solve the interval-constrained QoS-aware service composition (IQSC) problem. Here, QoS uncertainty is modeled using an interval-number representation and the skyline operator is applied to eliminate redundant services, with experimental results demonstrating superior performance compared to skyline-based PSO, an efficient discrete gbest-guided artificial bee colony, and Harris Hawks optimization with an elite evolutionary strategy. The second contribution, denoted as QSCFIoT, tackles QoS ambiguity in fuzzy IoT environments by representing QoS parameters with generalized trapezoidal fuzzy numbers. This approach integrates a fuzzy skyline-based module with an improved discrete flower pollination algorithm, and its efficacy is confirmed through experiments on both real and synthetic datasets, outperforming EFPA, PSO, and ITL-QCA in terms of composition quality, computational time, and stability. The third contribution is a systematic literature review (SLR) that rigorously examines QoS prediction methods for web services, focusing on Collaborative Filtering (CF) techniques in static and dynamic settings. Following PRISMA guidelines, 512 studies were initially identified and 146 were thoroughly analyzed, revealing that while traditional CF methods perform well in static environments, they face significant challenges in dynamic contexts due to data sparsity and variability. This review highlights the advancements in hybrid and context-aware models and underscores the need for adaptive, real-time prediction approaches to better meet user demands. Collectively, these contributions provide a robust framework for enhancing both the composition and prediction of QoS in web services, advancing their reliability and adaptability in complex, real-world scenarios.

Keywords : QoS-aware service composition ; Quality of Service (QoS) ; Interval number ; Generalized trapezoidal fuzzy number ; single objective optimization ; combinatorial optimization ; metaheuristics ; QoS uncertainty ; Dynamic Environments ; Web Service Composition ; Collaborative Filtering ; Fuzzy Logic ; IoT ; Adaptive Algorithms ; Swarm Intelligence ; Systematic Literature Review.

Résumé

Cette thèse aborde trois défis essentiels dans le domaine des services web : la composition de services tenant compte de la qualité de service (QoS) en contexte d'incertitude, la prédiction de la QoS dans des environnements dynamiques, et une revue exhaustive des techniques de filtrage collaboratif pour la prédiction de la QoS. La première contribution présente une version étendue de l'algorithme de colonie d'abeilles artificielles avec recherche locale (EABC) destinée à résoudre le problème de composition de services QoS-consciente à contraintes d'intervalle (IQSC). Dans cette approche, l'incertitude de la QoS est modélisée à l'aide d'une représentation par nombre intervalle, et l'opérateur skyline est utilisé pour éliminer les services redondants, les résultats expérimentaux démontrant une performance supérieure par rapport à la PSO basée sur le skyline, à une colonie d'abeilles artificielles discrète guidée par gbest efficace, ainsi qu'à l'optimisation par Harris Hawks combinée à une stratégie évolutive élite. La deuxième contribution, dénommée QSCFIoT, traite de l'ambiguïté de la QoS dans des environnements IoT flous en représentant les paramètres de la QoS par des nombres flous trapézoïdaux généralisés. Cette approche intègre un module basé sur le skyline flou avec un algorithme de pollinisation de fleurs discret amélioré, et son efficacité est confirmée par des expériences menées sur des jeux de données réels et synthétiques, surpassant l'EFPA, la PSO et l'ITL-QCA en termes de qualité de composition, de temps de calcul et de stabilité. La troisième contribution est une revue systématique de la littérature (SLR) qui examine rigoureusement les méthodes de prédiction de la QoS pour les services web, en se concentrant sur les techniques de filtrage collaboratif dans des contextes statiques et dynamiques. Conformément aux directives PRISMA, 512 études ont été initialement identifiées et 146 ont été analysées en profondeur, révélant que, bien que les méthodes de filtrage collaboratif traditionnelles fonctionnent bien dans des environnements statiques, elles rencontrent des défis significatifs dans des contextes dynamiques en raison de la rareté et de la variabilité des données. Cette revue met en lumière les avancées des modèles hybrides et sensibles au contexte et souligne la nécessité d'approches de prédiction adaptatives et en temps réel pour mieux répondre aux exigences des utilisateurs. Collectivement, ces contributions offrent un cadre robuste pour améliorer à la fois la composition et la prédiction de la QoS dans les services web, renforçant ainsi leur fiabilité et leur adaptabilité dans des scénarios complexes et réels.

Mots clés : Composition des services Web ; Qualité de service ; l'incertitude QoS ; Nombre intervalle ; des nombres flous trapézoïdaux ; Optimisation mono-objectif ; Optimisation combinatoire ; l'Internet des Objets ; méta-heuristiques ; Environnements dynamiques ; Filtrage collaboratif ; Logique floue ; L'IoT ; Algorithmes adaptatifs ; Intelligence en essaim ; Revue systématique de la littérature

ملخص

تتناول هذه الرسالة ثلاث تحديات محورية في خدمات الويب: تركيب الخدمات مع مراعاة جودة الخدمة (QoS)في ظل عدم اليقين، والتنبؤ بجودة الخدمة في البيئات الديناميكية، ومراجعة شاملة لتقنيات الترشيح التعاوني لتنبؤ جودة الخدمة. تُقدّم المساهمة الأولى خوارزمية مستعمرة النحل الاصطناعية الموسعة مع البحث المحلي (EABC) لحل مشكلة تركيب الخدمات ذات قيود الفترات (IQSC) مع مراعاة جودة الخدمة. في هذه المقاربة، يتم نمذجة عدم اليقين في جودة الخدمة باستخدام تمثيل الأعداد الفاصلة، ويُطبّق عامل الـ"سكاي لاين" لإزالة الخدمات الزائدة، حيث تُظهر النتائج التجريبية أداءً متفوّقاً مقارنةً بخوارزمية PSO المعتمدة على الـ"سكاي لاين"، ومستعمرة النحل الاصطناعية الموجهة بكفاءة بواسطةgbest ، وخوارزمية تحسين هاريس هوكس مع استراتيجية تطورية نخبويّة. تُعالِج المساهمة الثانية، والتي يُشار إليها باسم QSCFIoT، غموض جودة الخدمة في بيئات إنترنت الأشياء الغامضة عن طريق تمثيل معايير جودة الخدمة باستخدام أعداد غامضة شبه منحرفية معممة. تدمج هذه المقاربة وحدة مبنية على الـ"سكاي لاين" الغامض مع خوارزمية تحسين تلقيح الزهور المقطعية المحسّنة، وقد تأكدت فعاليتها من خلال تجارب على مجموعات بيانات حقيقية وتركيبية، حيث تفوقت على خوارزميات EFPA و PSOو ITL-QCAمن حيث جودة التركيب، والوقت الحسابي، والاستقرار. أما المساهمة الثالثة فهي مراجعة منهجية للأدبيات (SLR) تفحص بدقة أساليب التنبؤ بجودة الخدمة لخدمات الويب، مع التركيز على تقنيات الترشيح التعاوني في البيئات الثابتة والديناميكية. وباتباع إرشاداتPRISMA ، تم تحديد 512 دراسة في البداية وتم تحليل 146 دراسة بشكل مفصل، مما كشف أن الطرق التقليدية للترشيح التعاوني تؤدي أداءً جيداً في البيئات الثابتة، لكنها تواجه تحديات كبيرة في السياقات الديناميكية بسبب قلة وتفاوت البيانات. تبرز هذه المراجعة التطورات في النماذج الهجينة والمعتمدة على السياق وتؤكد الحاجة إلى مقاربات تنبؤية تكيفية وفي الوقت الفعلي لتلبية متطلبات المستخدمين بشكل أفضل. مجتمعة، توفّر هذه المساهمات إطاراً قوياً لتعزيز كل من تركيب الخدمات والتنبؤ بجودة الخدمة في خدمات الويب، مما يُحسّن من موثوقيتها وقدرتها على التكيف في سيناريوهات معقدة وواقعية.

الكلمات المفتاحية:

تركيب الخدمات مع مراعاة جودة الخدمة؛ جودة الخدمة(QoS) ؛ العدد الفاصل؛ العدد الغامض شبه المنحرف المعمم؛ التحسين لهدف واحد؛ التحسين التركيبي؛ الميتاوريستيك؛ عدم يقين جودة الخدمة؛ البيئات الديناميكية؛ تركيب خدمات الويب؛ الترشيح التعاوني؛ المنطق الغامض؛ إنترنت الأشياء(IoT) ؛ الخوارزميات التكيفية؛ ذكاء السرب؛ المراجعة المنهجية للأدبيات.

Table of Contents

List of Figures

List of Tables

Chapter 1

General introduction

1 Overview on Context and Problem Introduction

Over the last few decades, Information Technology (IT) has undergone a remarkable evolution, shaping and revolutionizing various aspects of our technological landscape. One notable and adaptable transformation within IT is the emergence and advancement of Service-Oriented Architecture (SOA) Papazoglou et al. [2007]. SOA represents a paradigm shift in software design and deployment, focusing on the creation and utilization of services as fundamental building blocks.

This architectural approach heavily relies on Service-Oriented Computing (SOC), which serves as the underlying framework for managing and utilizing these 'services.' Services, within the context of SOA, are conceptualized as modular, independent, and loosely coupled components that can be discovered, invoked, and composed to facilitate the implementation of complex functionalities, whether in the form of hardware or software Benatallah and Motahari Nezhad [2008].

In the realm of IT literature, the concept of services encapsulates a versatile approach wherein these modular components, often considered as the fundamental units of computation, are brought together to fulfill specific tasks or functionalities. These services possess the flexibility to be combined, reused, and orchestrated in various ways to build larger and more intricate systems or applications. Their loosely coupled nature allows them to operate independently, interact with each other through standardized protocols, and adapt to changing demands and requirements. This integration and interoperability between services are pivotal in the development of sophisticated and adaptable business applications Benatallah and Motahari Nezhad [2008].

In essence, the core principle behind SOA is the notion of encapsulating business functionalities as modular services, promoting reusability, flexibility, and scalability in IT systems. This paradigm shift has significantly influenced the development and deployment of modern software architectures by fostering an environment where diverse services can be orchestrated and combined to create agile, adaptable, and complex systems that cater to evolving business needs and technological advancements.

Within the realm of modern technological advancements, another significant paradigm that leans on the foundations of SOA is the Internet of Things (IoT). The IoT represents a vast and interconnected network comprising smart objects embedded with sensors, actuators, and computational capabilities, enabling them to communicate, collaborate, and interact seamlessly through the Internet Moreno et al. [2015]. These smart objects, or 'things,' encompass a diverse range of devices such as sensors, wearables, appliances, vehicles, and industrial equipment. They are empowered with the ability to gather data, process information, and initiate actions, ultimately contributing to an ecosystem where physical objects merge with digital connectivity to offer innovative functionalities and services.

The core functionalities and capabilities of these IoT-enabled devices are encapsulated as 'IoT services.' Similar to the concept of web services (WSs) within the domain of SOA, these IoT services represent modular functionalities or tasks that can be advertised, discovered, and composed to facilitate the creation and implementation of IoT applications Viriyasitavat et al. [2019]. Much like web services, IoT services are designed to be discoverable and accessible over network protocols, allowing for their integration into larger systems or applications. They serve as the foundational building blocks upon which diverse IoT applications are constructed, offering functionalities that range from data sensing and collection to analysis, decision-making, and control. These IoT services can be orchestrated and combined in various configurations to address specific user requirements, thus enabling the creation of innovative and tailored IoT applications catering to diverse domains such as smart homes, healthcare, transportation, agriculture, and industrial automation. Their interoperability and adaptability form the backbone of the IoT ecosystem, fostering an environment where the seamless integration of diverse devices and services contributes to the advancement of technology and enhances user experiences.

The applications, whether they belong to the realm of business applications or fall within the scope of the Internet of Things (IoT), often take the form of composite services. These composite services represent a conglomeration or assembly of individual services, each contributing specific functionalities or capabilities that collectively address a user's needs or demands. The selection and assembly of these composite services are guided by the overarching principle of Quality of Service (QoS). QoS encapsulates a set of metrics or attributes that define the performance, reliability, efficiency, and other qualitative aspects of services. These QoS metrics act as the yardstick or criteria against which the composite services are evaluated and selected to meet the global QoS constraints established for the targeted composite service. In essence, the user's requirements, expectations, or preferences for the composite service are translated into specific QoS criteria. These criteria, which could include parameters such as response time, reliability, security, cost-effectiveness, or other relevant factors, serve as constraints that the composite service must adhere to in order to fulfill the user's expectations effectively. For instance, in the context of a business application, the composite service might comprise various software components or modules, each contributing specific functionalities such as data processing, storage, or user interface. The selection and arrangement of these components are governed by their individual QoS attributes, ensuring that the composite service meets the desired level of performance and reliability as required by the user. Similarly, in IoT applications, where various smart devices and sensors collaborate to provide functionalities, the

composition of these devices into a coherent service also considers their QoS attributes. For example, an IoT application aimed at smart home automation might involve devices like sensors, actuators, and controllers, each contributing functionalities like temperature sensing, lighting control, or security monitoring. The selection and orchestration of these devices into the composite service are driven by their QoS attributes, ensuring the overall reliability, responsiveness, and efficiency of the smart home system.

The advent and widespread adoption of Service-Oriented Computing (SOC) have led to the proliferation of candidate services available on the web. These candidate services, although offering similar functionalities, exhibit varying Quality of Service (QoS) parameters such as response time, price, availability, reliability, and others. This diversity in QoS attributes among candidate services creates a scenario where selecting the most optimal services to construct an efficient and effective composite service becomes a complex and challenging task Zeng et al. [2004], Alrifai et al. [2012]. The process of selecting the most suitable candidate services aims at constructing an optimal composite service that meets the global end-to-end QoS constraints. Global end-to-end QoS constraints represent a set of predefined criteria or requirements established by users or applications, encompassing various QoS parameters crucial for the overall performance and satisfaction of the composite service. However, due to the diverse nature of candidate services and their varying QoS parameters, the selection process becomes intricate and non-trivial. Determining the best combination of services that collectively fulfill the global end-to-end QoS constraints while optimizing performance, cost, reliability, and other pertinent factors is a challenging optimization problem. This latter is formally recognized in the domain of Service-Oriented Architecture (SOA) as the QoS-aware web service composition problem (QSC). QSC is identified as an NP-hard optimization problem, implying that finding an optimal solution within a reasonable time frame becomes increasingly complex and resource-intensive as the number of candidate services and QoS parameters grows Zeng et al. [2004], Alrifai et al. [2012]. Addressing the QSC problem involves developing efficient algorithms, heuristics, or meta-heuristic approaches capable of intelligently selecting and composing candidate services while adhering to the global end-to-end QoS constraints. Researchers and practitioners in the field employ various methodologies, optimization techniques, and intelligent algorithms to tackle this complex problem and provide solutions that cater to the evolving demands and requirements of service-based systems and applications.

The primary objective of QoS-aware service composition is to identify and assemble the most optimal combination of individual atomic services, forming what is referred to as a composite service. This composite service is tailored and orchestrated to execute a specific business process or fulfill a particular functional requirement. The ultimate goal is to meet the global constraints and requirements defined by users or applications, ensuring that the composite service aligns with the desired Quality of Service (QoS) attributes Jatoth et al. [2015], Hamzei and Navimipour [2018], Asghari et al. [2018]. Despite the extensive research efforts in this field, a predominant focus within the literature has been on QoS-aware service composition approaches that consider QoS attributes as fixed and precise values. However, an important aspect often overlooked in these approaches is the inherent uncertainty

and imprecision associated with QoS attributes Jatoth et al. [2015], Hamzei and Navimipour [2018], Asghari et al. [2018]. In reality, open and dynamic environments, such as those encountered in modern web services or IoT ecosystems, face numerous challenges. One significant challenge revolves around the uncertainty and volatility of QoS attribute values associated with atomic services. These values can dynamically change over time due to various unforeseen factors, including network congestion, system overloads, variations in service invocations, or other environmental dynamics. For instance, the QoS attributes such as response time, availability, or reliability of a web or IoT service might vary unpredictably with different service invocations, leading to fluctuations in their performance characteristics. The uncertainty in these QoS attributes poses a challenge in accurately predicting or guaranteeing the behavior of composite services, thereby affecting their reliability and performance.

In addition to service composition, the prediction of web service QoS in both static and dynamic environments plays a critical role in ensuring optimal service selection and composition. In static environments, QoS prediction methods rely on historical data to estimate service performance under fixed conditions, while in dynamic environments, the prediction becomes more complex due to fluctuating network conditions, system loads, and other temporal factors Shao et al. [2007], Zhang et al. [2011b], Ye et al. [2021]. Addressing these challenges requires adaptive prediction approaches that can accommodate real-time data and evolving service behaviors.

The focus of this thesis lies in addressing the QoS-aware service composition problem within the context of uncertain QoS attributes associated with elementary web or IoT services, and in developing robust methodologies for **web service QoS prediction in static and dynamic environments**. By acknowledging and incorporating the uncertainty and imprecision of QoS attributes into the service composition process, and by advancing QoS prediction techniques, this research aims to develop novel methodologies or frameworks that can handle the dynamic nature of QoS attributes while satisfying the QoS constraints imposed by users or applications.

The subsequent sections of this chapter are structured as follows :

Section 2 : This section delves into an in-depth exploration of the primary challenges encountered within the scope of our research. It comprehensively examines the hurdles, complexities, or limitations that the research aims to address or overcome. By thoroughly examining these challenges, the section sets the stage for understanding the critical issues that form the basis of the research inquiry.

Section 3 : In this section, a compelling and illustrative motivating example is presented. The purpose of this example is to provide readers with a tangible scenario or use case that vividly demonstrates the relevance and objectives of this thesis. Through the depicted example, readers gain insight into the practical implications and real-world applications driving the focus and objectives of the research.

Section 4 : Here, the primary contributions and significant advancements introduced by this thesis are outlined and discussed in detail. This section elucidates the novel methodologies, innovative approaches, or key findings that constitute the core contributions of the research. It provides an overview of the unique aspects and noteworthy outcomes resulting from the conducted research efforts.

Section 5 : Finally, this section offers a succinct yet comprehensive overview outlining the structure and sequence of the thesis chapters. It provides a road-map or preview of the subsequent chapters, highlighting the thematic content and focus areas covered in each section. This brief description helps readers anticipate the flow and organization of the thesis, enabling them to navigate through the subsequent chapters with clarity.

2 Challenges of QoS-aware service composition

Service-Oriented Computing (SOC) emphasizes the integration of discrete service components to create composite services capable of executing abstract composite services (ACS). These ACSs, serving as higher-level representations of functionalities or tasks, necessitate the invocation of elementary services to fulfill the abstract tasks. Within the realm of Service-Oriented Architecture (SOA), the fundamental objective of QoS-aware service composition is to adeptly select a subset of elementary services, each corresponding to an abstract task, in a manner that the resultant service composition meets both the functional requirements and the defined QoS constraints. In typical scenarios, multiple services might qualify as candidates for fulfilling a specific abstract task. To discern and differentiate among these candidate services, the selection process goes beyond merely addressing the functional requirements. Instead, it places significant emphasis on non-functional attributes, commonly referred to as Quality of Service (QoS) attributes. These QoS attributes encompass various parameters such as response time, reliability, availability, cost, and other performance-related metrics associated with services. However, achieving the optimal selection of services that collectively fulfill all the imposed constraints represents a daunting challenge, particularly in scenarios involving large-scale problems. In such scenarios, the sheer magnitude of services, tasks, and constraints complicates the selection process, making it arduous and resource-intensive Zeng et al. [2004], Alrifai et al. [2012].

The pursuit of identifying the best service composition has garnered substantial attention and effort from researchers in the field. Despite the active research and numerous advancements in service selection methodologies, several challenges persist, remaining unresolved in current literature. These unresolved challenges underscore the complexities inherent in service selection processes and can be summarized as follows :

1. **Scalability and Optimality** : The scalability challenge stands as a pivotal concern within the domain of QoS-aware service composition. Essentially, the QSC process endeavors to identify the most suitable composite service from an exhaustive array of possibilities. This quest for the optimal composite service necessitates comparing and evaluating all potential combinations of services (i.e., composite services), ultimately selecting the most optimal one among them. However, when confronted with an extensive search space comprising an incredibly vast number of possible combinations, the issue of scalability promptly arises. The sheer magnitude of potential combinations to consider, particularly while accommodating

user-defined constraints, poses a formidable scalability challenge Zeng et al. [2004], Alrifai et al. [2012].

Moreover, ensuring the optimality of the selected composite service further compounds the challenge. Achieving an optimal solution that adheres to user-defined constraints is imperative for the QSC problem. Nonetheless, the task of selecting the best composite service from a multitude of feasible solutions becomes an intricate and time-consuming endeavor. In practical terms, exhaustively evaluating and comparing all feasible solutions to determine the optimal one inevitably intertwines with the scalability issue. The quest for optimality while considering large volumes of potential solutions accentuates the inherent scalability challenge in this context.

Furthermore, striking a balance between providing an optimal solution and delivering timely responses to user requests amplifies the complexity. The need to obtain an optimal solution within an acceptable time-frame to cater to user requirements brings forth the interplay between scalability and achieving solution quality or optimality. The simultaneous operation of scalability and optimality becomes a critical concern, requiring innovative methodologies and efficient algorithms that can navigate and resolve these intertwined challenges effectively in QSC scenarios.

2. **Scalability and Linearization** : In recent studies addressing the QSC problem Jatoth et al. [2015], Hamzei and Navimipour [2018], Asghari et al. [2018], exact service selection algorithms have been prominent. These algorithms typically model the QSC problem using integer or mixed-integer linear programming frameworks Alrifai et al. [2012], Ardagna and Pernici [2007]. To solve these formulated QSC models, solvers such as LpSolve [1] and CPLEX [2] are utilized. When the objective functions and constraints of the QSC problem are linearized, these solvers can efficiently derive optimal solutions.

   However, the efficacy of exact service selection algorithms is inversely proportional to the scalability of the QSC problem. As the complexity and scalability of the QSC increase, the efficiency of these exact algorithms decreases significantly. This phenomenon highlights an inherent trade-off between scalability and the performance of exact algorithms, making them relatively unsuitable for larger, more complex QSC instances.

   Consequently, researchers in the field have shifted their focus towards evolutionary and bio-inspired algorithms, such as Genetic Algorithm (GA) Canfora et al. [2005], Particle Swarm Optimization (PSO) Liao et al. [2014], and Artificial Bee Colony (ABC) Huo et al. [2015], x. wang et al. [2019]. These algorithms have garnered attention due to their distinct advantages :

   (a) **Assurance of Optimal/Near-optimal Solutions** : Evolutionary and bio-inspired algorithms demonstrate an ability to provide optimal or near-optimal solutions within reasonable computational timeframes, even in scenarios involving significant scalability.

1. http ://lpsolve.sourceforge.net/5.5/
2. https ://www.ibm.com/analytics/cplex-optimizer

(b) **Flexibility in Objective Functions** : Unlike exact algorithms reliant on linearization, these evolutionary and bio-inspired algorithms operate without constraints imposed by linearized objective functions. This flexibility allows for a more robust exploration of solution spaces, especially in scenarios where linearization might be challenging or impractical.

3. **Ambiguity within dynamic environments** : Traditionally, QoS-aware service composition (QSC) studies have typically operated under the assumption that Quality of Service (QoS) values are unambiguous and constant. However, in real-world Service Oriented Architecture (SOA) environments, numerous unforeseen factors like changes in network topologies, economic policies, or varying network conditions contribute to the uncertainty and variability of these QoS values Razian et al. [2020].

   In dynamic environments, QoS attributes of candidate services often exhibit uncertainty, making them inherently uncertain and unreliable for exact quantification Razian et al. [2020]. For instance, the response time of a web service might fluctuate between different transactions or invocations due to varying network connectivity, leading to varying QoS response times for composite services built from these fluctuating services. Similarly, pricing of candidate services may vary during peak hours, influencing the overall cost of a composite service.

   Consequently, the deterministic nature of QoS attributes breaks down in dynamic settings, rendering QoS values unreliable and subject to constant change. This uncertainty needs to be considered and addressed while solving the QoS-aware service composition problem. Recent advancements in this area have proposed approaches that model uncertain QoS parameters using different methodologies such as :

   (a) **Interval Numbers** : Modeling QoS uncertainty by representing QoS attributes as intervals Jian et al. [2016], Seghir et al. [2019].

   (b) **Probabilistic Variables** : Considering the uncertain nature of QoS by representing attributes as probabilistic variables Zheng et al. [2016].

   (c) **Fuzzy Numbers** : Addressing QoS uncertainty through the use of fuzzy numbers to capture imprecise QoS attributes Xu et al. [2018].

3 Motivational example

The process of composing services, as outlined by AlSedrani and Touir [2016], involves several sequential stages as illustrated in Figure 1.1. These stages are essential for creating a composite service that meets the user's requirements :

— **Composition Planning** : This initial stage involves planning and specifying the requested service, breaking it down into an abstract set of tasks. It defines the overall structure and requirements of the desired composite service.

— **Service Discovery** : In the service discovery phase, there's a search for available services that match both the functional and non-functional requirements, including Quality of Service (QoS) attributes, for each task identified in the composition plan. This step involves identifying potential candidate services that can fulfill the defined tasks.

— **Service Selection** : Once multiple services matching the requirements from the previous phase are discovered, the service selection stage comes into play. This phase involves selecting the most suitable service for each task within the composition. The selection aims to ensure that the user's requirements, including functional and QoS aspects, are adequately met.

— **Service Execution** : The final phase, service execution, involves invoking and executing individual tasks identified in the composition plan using the selected services. This phase integrates these tasks to produce the final composite service that fulfills the user's intended purpose.

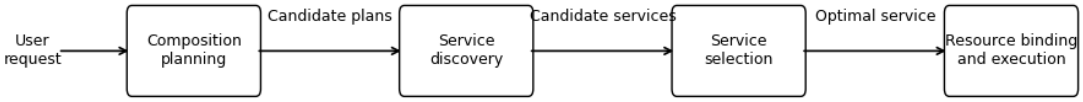


FIGURE 1.1 Service Composition Processes
AlSedrani and Touir [2016]

In order to understand the QoS-aware service composition problem, an illustrative example will help to fulfill the task. Let's consider, for instance, an abstract composite service (ACS) of four (04) abstract tasks, as shown in Figure 1.2 , where each task $T_i (i \in \{1,2,3,4\})$ has a set of functionality similar to concrete services $CS_i^j (j \in \{1 \ldots 5\})$, and for each concrete service CS_i^j , two QoS attributes are considered in this example : the response time and the price (These attributes need to get minimal values as it will be explained in further chapters with details). The QoS values of these concrete services are detailed in Table 1.1.

This ACS example takes into consideration only two global end-to-end QoS constraints :

1. The global response time composite service must not exceed 50 ms.
2. The global price of the composite service must not exceed 15 DZD.

To find the best composite service of four concrete services that implements the ACS example of Figure 1.1 by giving each task T_i its appropriate selected concrete service CS_i^j , where these selected services must satisfy the above two global constraints.

In this example, the best composite service for ACS that fulfills the QoS user's constraints is $CCS = (CS_1^2, CS_2^3, CS_3^2, CS_4^2)$. It is selected among $5_4 = 625$ solutions by applying the dominance notion of Huo et al. [2015] to define the CCS solution. We shall explain, each set of concrete services is associated to each task T_i, we select the concrete service that dominates all the other concrete

TABLE 1.1 QoS values of candidate services for the abstract task

Tasks	Concrete services	QoS attributes	
		Response time (ms)	Price (DZD)
T_1	CS_1^1	5	7
	CS_1^2	8	10
	CS_1^3	10	8
	CS_1^4	8	6
	CS_1^5	13	14
$T2$	CS_2^1	7	6
	CS_2^2	39	14
	CS_2^3	7	3
	CS_2^4	10	11
	CS_2^5	42	6
T_3	CS_3^1	10	9
	CS_3^2	10	5
	CS_3^3	12	7
	CS_3^4	12	8
	CS_3^5	25	3
T_4	CS_4^1	20	9
	CS_4^2	18	5
	CS_4^3	35	7
	CS_4^4	40	13
	CS_4^5	49	15

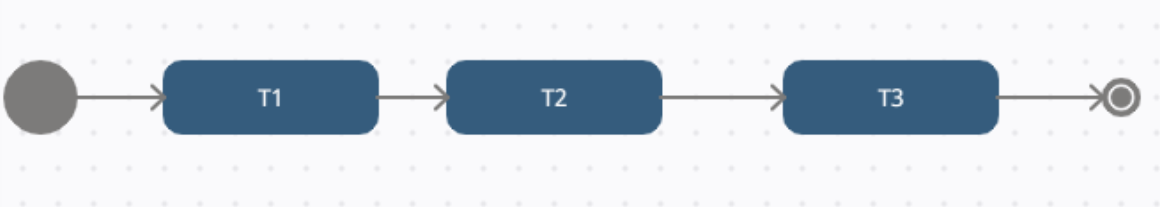


FIGURE 1.2 State chart of the proposed ACS example

services in this set of candidate services (e.g. CS_1^1 is selected to implement the task T_1 as it offers the best QoS values compared to the QoS values of the concrete services of the same class).

The previous example of Abstract Composite Service (ACS) illustrates QoS values as fixed. However, real-world QoS attributes of services are often imprecise and uncertain due to various factors. Therefore, representing these QoS attributes as fixed values doesn't align with reality. It's crucial to account for the uncertainty associated with these QoS attributes to ensure the selection of the most appropriate composite service.

The QSC problem is notably complex, given its NP-hard complexity and the dynamic nature of the environments where services are utilized. This inherent complexity makes solving this problem challenging. Some prior studies Alrifai et al. [2010], Wang et al. [2013], Ying and Jiande [2020] have employed the Skyline operator Borzsony et al. [2001] to streamline the QoS-aware service composition process. This operator reduces the search space by eliminating WSs that cannot be part of the final solutions, as they are dominated by other functionally equivalent WSs. However, these existing approaches mostly consider static QoS values and fail to handle the imprecision and uncertainty associated with QoS parameters.

To address these limitations, our work proposes two approaches. Firstly, we aim to tackle the scalability issue associated with solving the QSC problem, ensuring optimal solutions even in a large search space. Secondly, we focus on managing the uncertainty linked to QoS attributes. Our motivation lies in presenting an efficient approach to solve the QoS uncertainty-aware service composition problem within a reduced search space.

In the subsequent section, we'll outline and present the distinct contributions of our work, emphasizing how our proposed approaches aim to overcome the challenges posed by scalability, optimality, and QoS attribute uncertainty in service composition.

4 Research major contributions

The central focus of this thesis revolves around the QoS-aware service composition problem, which can be comprehensively described as follows :

Abstract Composite Service Structure : The problem entails the presence of an abstract composite service (ACS) that intricately defines the interconnections among its elementary tasks. Additionally, it encapsulates the overarching global Quality of Service (QoS) constraints mandated by

users. Each atomic task within this abstract composite service is associated with a set of functionally equivalent candidate services. These candidates offer the necessary functionality for each specific task but exhibit diverse QoS attributes.

Web Service Selection Objective : The primary goal of web service selection is to meticulously choose the most suitable services. These selections are instrumental in constructing the most optimal composite service capable of effectively implementing the ACS while concurrently adhering to the specified user requirements and constraints.

Web Service QoS Prediction : The primary purpose of the prediction module is to estimate the performance characteristics of candidate services. In static environments, QoS prediction leverages historical data to forecast service performance under stable conditions, whereas in dynamic environments, advanced adaptive techniques are employed to capture time-varying QoS parameters affected by fluctuating network loads, user demands, and other temporal factors. This prediction process plays a crucial role in guiding the selection of services, ensuring that the assembled composite service consistently meets the global QoS constraints and adapts effectively to changing conditions.

Our goal in this work is to propose new service selection approaches in uncertain environments to deal with the issues previously presented in Section 2. The contributions of this thesis can be seen from three different angles :

1. **Solving Scalability and Optimality Issues** : To mitigate issues related to scalability and optimality, two distinct yet effective approaches have been proposed :
   (a) **Utilization of Skyline Operator for Pruning** : Both approaches incorporate the use of the skyline operator, a concept introduced by Borzsony et al. Borzsony et al. [2001], aimed at efficiently eliminating redundant and dominated web services from their respective sets of functionally equivalent options. Subsequently, both the Extended Artificial Bee Colony (EABC) and the Improved Discrete Flower Pollination Algorithm (IDFPA) leverage this pruned dataset to solve the Interval QoS-aware Web Service Composition Problem (IQSC) and the Fuzzy-IoT Service Composition Problem (QSCFIoT), respectively, in a reduced search space.
   (b) **Approach 1 : Extended Artificial Bee Colony (EABC) for IQSC** : The first approach introduces an Extended Artificial Bee Colony (EABC) algorithm specifically tailored to tackle the Interval QoS-aware Web Service Composition Problem (IQSC). EABC ensures rapid identification of the best optimal composite service while maintaining a high standard of solution quality. Its strength lies in embedding a local search capability that significantly enhances the exploration process. Comparative analysis against existing algorithms like skyline-based PSO, discrete gbest-guided artificial bee colony, and Harris Hawks optimization with an elite evolutionary strategy demonstrates EABC's superiority. It excels in managing end-to-end constraints and efficiently identifying optimal solutions within a reasonable computation timeframe.

(c) **Approach 2 : Improved Discrete Flower Pollination Algorithm (IDFPA) for QSC-FIoT** : The second approach introduces an Improved Discrete Flower Pollination Algorithm (IDFPA) designed to swiftly search for near-optimal Composite Services (CSs) within the Fuzzy-IoT Service Composition Problem. IDFPA employs novel discrete global and local pollination processes to update the positions of solutions. Additionally, it integrates a discard abandoned solutions mechanism, which effectively repositions stagnated CSs that haven't updated their positions after a predetermined number of iterations during the near-optimal CSs' search process. Experimental results illustrate IDFPA's superiority over EFPA, PSO, and ITL-QCA in terms of efficiency, showcasing its capability to outperform existing algorithms when dealing with near-optimal CSs in the Fuzzy-IoT Service Composition Problem.

2. **Addressing Uncertainty in Dynamic Environments** : To tackle the challenge of uncertainty inherent in dynamic environments, both proposed approaches offer solutions for solving the QoS-aware Service Composition Problem (QSC) under uncertain QoS properties :

   (a) **Approach 1 : IQSC with Interval-based QoS Parameters** : The first approach introduces an Interval Constrained Single Objective Optimization Model aimed at addressing the QoS Uncertainty-Aware Web Service Composition Problem (IQSC). In this model, the QoS parameters are represented as interval numbers. The IQSC problem deals with uncertain QoS attributes associated with web services, allowing for a more realistic representation of QoS parameters in an uncertain environment.

   (b) **Approach 2 : QSCFIoT with GTrFN-based QoS Parameters** : The second approach focuses on a Fuzzy Constrained Optimization Model for the IoT-services Composition Problem, denoted as QSCFIoT. QSCFIoT operates within the realm of IoT services and employs Generalized Trapezoidal Fuzzy Numbers (GTrFN) to represent QoS parameters. By leveraging GTrFN-based QoS attributes, this model accounts for uncertainties in QoS attributes within the IoT services domain.

3. **Systematic Literature Review on Web Service QoS Prediction** : In addition to the above contributions, a comprehensive systematic literature review (SLR) has been conducted to examine the state-of-the-art in web service QoS prediction. This SLR focuses on collaborative filtering techniques applied in both static and dynamic environments. Following PRISMA guidelines, 512 studies were initially identified, and 146 were thoroughly analyzed. The review reveals that while traditional collaborative filtering methods perform adequately in static settings, they face significant challenges in dynamic environments due to data sparsity and variability. The SLR highlights advancements in hybrid and context-aware models and underscores the need for adaptive, real-time prediction approaches that can better accommodate fluctuating QoS attributes. This insight into QoS prediction further informs and complements the proposed service selection strategies by providing accurate, context-aware performance estimates essential for optimal composite service construction.

5 Thesis Organization

The thesis in hand has three main parts where the first part including Chapter 2 is devoted to give general information about Web Services and Internet of Things ; Optimization and the uncertain numbers are introduced in Chapter 3. Whereas, the second part including Chapter 4 and Chapter 5 are derived from research results that were conducted during the course of the doctorate period. Furthermore, the last chapter 7 is devoted to conclude the thesis and highlight further perspectives Thus, the thesis is organized as follows :

1. First Part (Chapters 1,2 and 3) :
   - Chapter 1 - **General Introduction** : This chapter provides an overview of the study's focus, identifies challenges, and elucidates the motivation behind the research. Additionally, it outlines the organization of the subsequent chapters.
   - Chapter 2 - **From Web Services to Internet of Things** : This chapter delves into the fundamentals of web services and Internet of Things (IoT). It covers basic definitions, architectures, and standards pertaining to both technologies.
   - Chapter 3 - **Preliminaries on Optimization and Uncertain Numbers** : This chapter introduces essential concepts in optimization, elucidates various types of optimization problems, and specifically focuses on uncertain numbers such as Interval Numbers and Fuzzy Numbers (GTrFN). It provides detailed explanations of arithmetic operations on these numbers and their ranking, as they play a crucial role in formulating problems and preparing datasets in subsequent chapters (4 and 5).
2. Second Part (Chapters 4, 5 and 6) :
   - Chapter 4 - **Extended Artificial Bee Colony (ABC) for Interval-based QoS Properties** : This chapter introduces an Extended Artificial Bee Colony algorithm enhanced with local search to solve Skyline-based web service composition problems under Interval QoS properties. The proposed approach demonstrates superior performance compared to existing references, as evaluated on a new interval-valued version of the public QWS dataset Khababa et al. [2022].

     "G. Khababa, F. Seghir, and S. Bessou. An extended artificial bee colony with local search for solving the skyline-based web services composition under interval qos properties. *Journal of Intelligent & Fuzzy Systems*, 42(4) :3855–3870, 2022."
   - Chapter 5 - **Improved Discrete Flower Pollination Algorithm for Fuzzy QoS-aware IoT Services Composition** : This chapter presents an Improved Discrete Flower Pollination Algorithm designed to address the challenges of Fuzzy QoS-aware IoT services composition, utilizing the Skyline Operator Seghir and Khababa [2023]

     "F. Seghir and G. Khababa. An improved discrete flower pollination algorithm for fuzzy qos-aware iot services composition based on skyline operator. *The Journal of Supercomputing*, 79(10) : 10645–10676, 2023"

— Chapter 6 - **Collaborative Filtering Techniques for Predicting Web Service QoS Values in Static and Dynamic Environments : A Systematic and Thorough Analysis** : This chapter presents a systematic literature review of state-of-the-art prediction methods applied in the web service environment. Furthermore, it analyzes these methods from the perspective of whether they were applied in static or dynamic environments. The sole purpose of this chapter is to be a roadmap for further implementations Khababa et al. [2025].

"G. Khababa, S. Bessou, F. Seghir, N. H. Harun, A. S. Almazyad, P. Jangir, and A. W. Mohamed. Collaborative filtering techniques for predicting web service qos values in static and dynamic environments : A systematic and thorough analysis. *IEEE Access*, 2025."

3. **Conclusion** (Chapter 7) :

— Chapter 7 - Conclusion : This chapter summarizes and concludes the findings and contributions presented in the previous chapters. It also outlines potential future perspectives that warrant further exploration as potential contributions.

The thesis is logically organized, starting with foundational knowledge and gradually progressing to innovative research findings and conclusions, providing a comprehensive understanding of the topics covered.

Chapter 2

From Web Services to Internet of Things : Architectures and Technologies

Introduction

In recent years, technological advancements have revolutionized task execution through the Internet. Among the latest innovations, the Internet of Things (IoT) has emerged, amplifying the use of Web services and establishing standards that have proliferated across various domains. These technologies have significantly impacted diverse sectors, from streamlining business operations in hospitality (such as hotel and restaurant management) to enhancing traffic management systems (enabling real-time vehicle tracking and automatic monitoring of environmental conditions such as temperature fluctuations).

This chapter aims to provide an in-depth understanding of Web services and the IoT by elucidating the fundamental definitions, concepts, and standards that underpin these technological frameworks. Navigate through the intricacies of these technologies, shedding light on their significance and role in transforming various industries and daily lives.

1 Web Services : Standards and Technologies

In this section, we dive into the theoretical underpinnings and design principles of the Web services technology. Here, we explore the intricate models, detailed specifications, and accessibility of this technology as a robust tool that allows disparate systems to collaborate seamlessly, achieving desired tasks with efficiency and effectiveness.

Web services technology stands as a pivotal mechanism for facilitating interoperability among diverse systems, allowing them to communicate and work together harmoniously. Its design principles encompass various protocols, standards, and methodologies that ensure the seamless integration of functionalities across different platforms and domains.

At its core, Web Services are built on standards like XML (eXtensible Markup Language), SOAP (Simple Object Access Protocol), and WSDL (Web Services Description Language), providing a framework for communication and data exchange between applications over the Internet. These services encapsulate discrete functions, allowing them to be accessed and utilized remotely by other software systems Newcomer [2002].

The discussion in this section aims to elucidate the fundamental aspects of Web service technology, focusing on its conceptual framework, standardization protocols, and practical applications. By exploring these theoretical foundations and design elements, we gain a comprehensive understanding of how Web services function as an enabling tool for achieving seamless integration and collaborative operations among diverse systems, fostering interoperability and streamlined functionalities across different platforms and applications.

1.1 A Brief History on Web Services : from Distributed Computing to Service Oriented Architecture

Since its inception in the 1990s, the World Wide Web (WWW) has revolutionized the accessibility of information for users worldwide. Despite the initial technological constraints associated with the early days of the WWW that didn't adequately support online product promotion, the focus shifted toward developing Business-to-Business (B2B) infrastructures. These infrastructures aimed to assist organizations in optimizing their processes and enhancing the efficiency of business transactions conducted over the Internet.

To foster greater integration, many B2B solutions leveraged XML as the primary language for representing data. XML's versatility and extensibility made it a preferred choice for facilitating data interchange and communication between different systems.

The evolution of B2B strategies led to the emergence of the SOA concept. SOA was conceived to overcome the limitations inherent in B2B approaches, especially in terms of the flexibility and dynamic adaptability of Information Technology (IT) systems. It introduced a methodology focused on the design, development, deployment, and management of discrete components of computer logic or services.

Using XML as its primary language, SOA aimed to structure, promote loose coupling, and standardize the exchange of business functionalities among various software applications. By employing XML for data representation, SOA facilitated the seamless interaction and interoperability of diverse systems, enabling the creation of agile and adaptable IT infrastructures.

Figure 2.1 illustrates the evolutionary trajectory from the initial limitations of WWW technologies to the development of B2B infrastructures and the subsequent evolution into SOA, highlighting the pivotal role of XML in facilitating data structuring and interoperability.

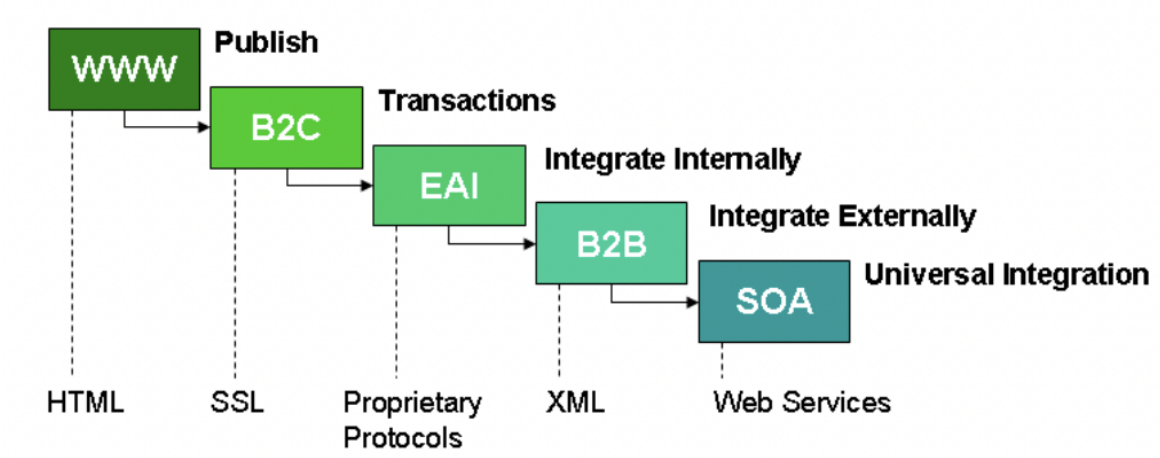


FIGURE 2.1 The evolution of business usage on the WWW
Pennington [2007]

1.1.1 Distributed Computing and Web Services

Distributed computing traced its roots back to the 1980s when sockets emerged as the pioneering communication technology. At this early stage, Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) served as foundational transport protocols, facilitating low-level messaging over Internet Protocol (IP) networks. This process involved encoding messages, transmitting them to recipients, and subsequently decoding them. However, this approach incurred significant time overheads for both senders and recipients in the processing of messages, hindering efficiency.

Addressing these challenges, Remote Procedure Call (RPC) Birrell and Nelson [1984] emerged in the late 1980s, primarily introduced by Sun Microsystems. RPC proved to be a viable solution, especially in the development of two-tier client/server architectures. It facilitated communication between different applications by enabling a client application to execute procedures or functions on a remote server as if they were local.

As the demand escalated for more complex N-tier applications, the industry witnessed the advent of two significant technologies : Common Object Request Broker Architecture (CORBA) and Distributed Common Object Model (DCOM)Brown and Kindel [1996]. CORBA aimed to enable communication between distributed objects without considering the operating systems on which they operated. Meanwhile, DCOM, developed by Microsoft, facilitated communication between applications running on distributed computers under reliable conditions.

Subsequently, Java Remote Method Invocation (RMI) Downing and Java [1998] emerged, providing a framework for creating and executing distributed Java programs. It streamlined object method calls between different Java Virtual Machines (JVMs) across a network Raghavan and Waghmare [2002], Pennington [2007].

Despite the successes of CORBA, DCOM, and Java RMI, these technologies also exhibited limitations. They tended to create tightly coupled distributed systems; they were often platform-

specific (for instance, COM/DCOM was limited to Windows environments), and they relied on complex proprietary protocols, message formats, and data representations. These factors led to challenges in interoperability and system scalability, particularly within heterogeneous environments Pennington [2007], Booth et al. [2004].

In response to these limitations, Web services emerged as a more standardized and interoperable solution for developing distributed systems on the web. Web services offer a platform-independent, language-neutral approach to facilitate communication and interaction between different applications over the internet. They rely on standardized protocols such as HTTP, XML, SOAP, and WSDL, fostering loose coupling and providing a more flexible and scalable architecture for distributed systems.

1.1.2 Service-Oriented Architecture

Recognizing the limitations inherent in traditional distributed objects, the evolution of Service-Oriented Architecture (SOA) emerged in the early 2000s, offering a more effective solution.

Definition 1 *Pennington [2007], Booth et al. [2004] SOA can be defined as a technology that aims at simplifying the development of modular services that can subsequently be integrated to construct distributed applications. The fundamental goal of SOA is to facilitate the creation of flexible and adaptable IT infrastructures by breaking down applications into smaller, more manageable services that can be accessed independently.*

In essence, SOA promotes a modular and loosely-coupled design philosophy, where each service encapsulates a specific business functionality or task. These services communicate and interact with each other using standard protocols, enabling them to be flexibly combined and reused to support various applications and business processes.

1.2 Service Oriented Architecture and Web Services

Web Services play a pivotal role in resolving interoperability issues among applications and services by providing robust standards to build effective distributed systems. Within the SOA, web services are associated with three fundamental actions : discovery, request, and response.

The SOA architecture typically involves three main actors : the requester, provider, and registry. As depicted in Figure 2.2, the process of service discovery and interaction involves several sequential steps Pennington [2007], Booth et al. [2004], which is described in the following steps :

1. **Step 1.** This initial phase establishes a connection between two participants — the service provider and the requester. The service provider publishes crucial details like the WSD and semantics (Sem) to a registry upon the discovery of the service by the requester.
2. **Step 2.** Subsequently, there's an exchange of data during the communication phase. This step occurs after establishing a mutual understanding between the semantics and the description provided by the service.

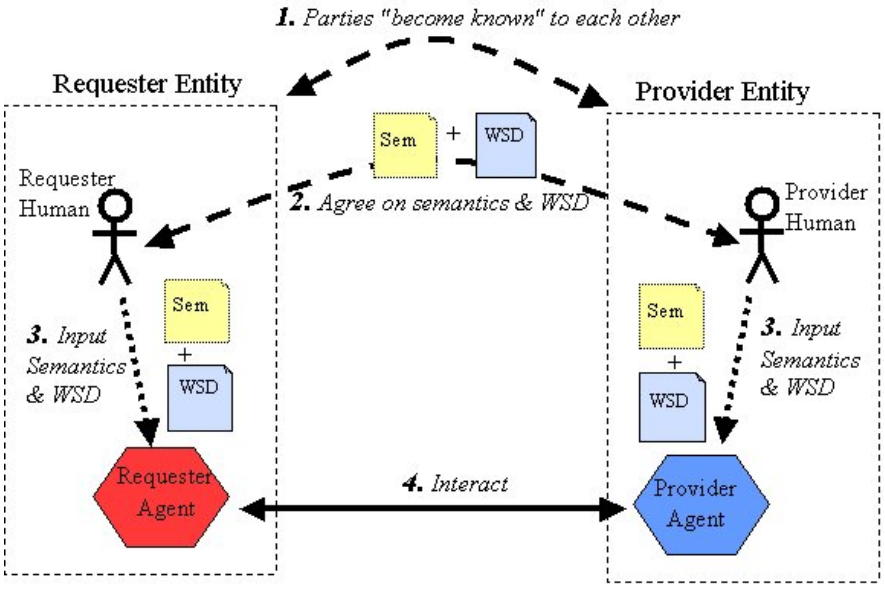


FIGURE 2.2 Process of discovery
Booth et al. [2004]

3. **Step 3.** The WSD and Sem are acknowledged and loaded into both the provider and the requester, facilitating their interaction to execute the necessary operation.

In the context of Web Services within a SOA, a service provider has the capability to create multiple distinct Web services. Within this framework, each service is required to consist of at least one operation.

Operations, also referred to as endpoints, play a pivotal role within a service. They are specifically responsible for processing and managing information. Each operation encapsulates a specific functionality or task that the service provides.

These operations act as entry points or interfaces through which external systems or clients can interact with the service. They define the methods or functions that external entities can call or utilize to request specific functionalities or data from the service.

An operation typically defines the required inputs (parameters or data), processes these inputs according to the service logic, and produces outputs (response or result) that are sent back to the requester. Operations encapsulate the core functionalities and capabilities of a service, serving as the functional units with which external systems interact.

These endpoints or operations, as the building blocks of a Web service, are crucial elements that enable services to perform specific tasks or provide particular functionalities in response to requests from various clients or systems Pennington [2007], Booth et al. [2004].

1.3 SOA and Web Service Standards

SOA heavily relies on standardized protocols to deploy interoperable and compatible services effectively. Currently, the preferred approach for developing SOA-based products is through Web service standards. Several key standards associated with Web services play a pivotal role in the development and implementation of SOA-based solutions. These standards are fundamental in ensuring interoperability, communication, and seamless interaction between various services and systems within a SOA environment. Figure 2.3 illustrates some of the essential standards closely associated with Web services, which are crucial for building robust SOA-based solutions.

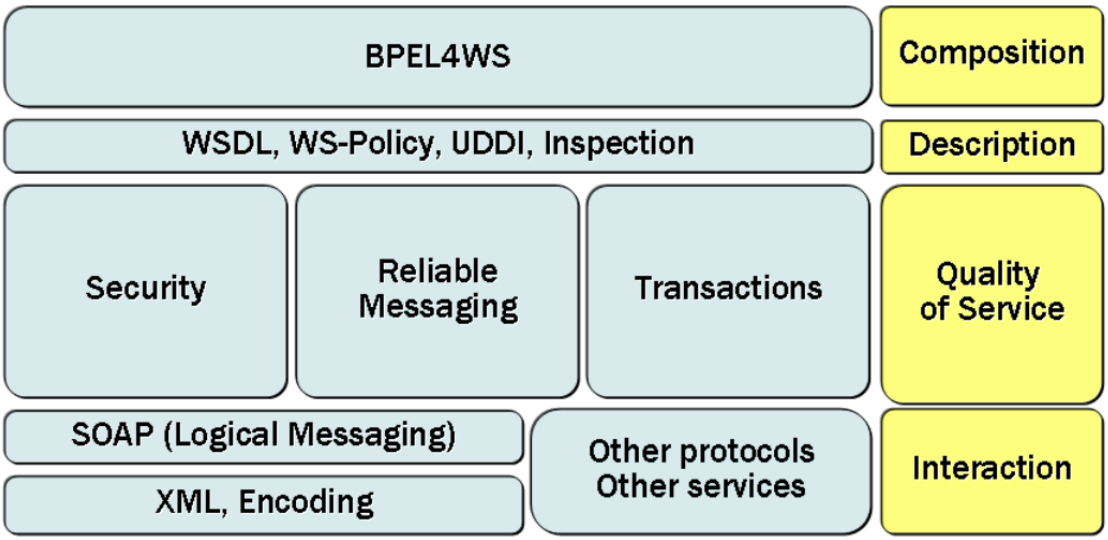


FIGURE 2.3 List of Standards associated to Web Services
Cardoso et al. [2004]

To further understand the concept of Web services and their role in SOA, the following definitions provide clarity on the core elements and characteristics of Web services.

Definition 2 *Curbera et al. [2001] Web services are modular, self-describing, self-contained applications that are accessible over the Internet. They represent the most popular realization of the SOA.*

Definition 3 *Bray et al. [2004] A Web service is a software component that can be invoked over the Web using an XML message following the SOAP standard. Each Web service provides one or more operations to perform specific actions on behalf of the invoking client. These operations, along with the formats of input and output messages, are described using WSDL.*

Definition 4 *Masdari et al. [2021] Web Services are self-contained and reusable software components which can be accessed and invoked remotely based on the specified protocols and standards such*

FIGURE 2.4 The relationship between XML/SOAP/WSDL/UDDI
Cardoso et al. [2004]

as Universal Description, Discovery and Integration (UDDI), Web Services Description Language (WSDL), Web Services Inspection Language (WSIL), SOAP, and Web Services Interoperability (WS-I). Subsequently, they can be applied to create more complicated composite services for users according to the specified Quality of Service.

Definition 5 *Booth et al. [2004] A Web service is a software system designed to support interoperable machine-to-machine interaction over the network. It contains an interface which is described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a prescribed way by*

1.4 Basic Web Service Standards

According to IBM [1], XML, SOAP, WSDL, and UDDI are the fundamental standards to deploy SOA infrastructures based on Web services where XML is the standard for data representation ; SOAP specifies the transport layer to send messages between consumers and providers ; WSDL describes Web services ; and UDDI is used to register and search for Web services, as shown in Figure 2.4.

1.4.1 XML

XML serves as the foundation for data interchange on the Web. It stands as a standard language for semi-structured data, providing a flexible means of coding and displaying information. Through XML, data are described using metadata, allowing the creation of a structured tree of nested sets through open and closed tags, each capable of including multiple attribute-value pairs. This structure

1. https ://www.ibm.com/docs/en/dmrt/9.5 ?topic=overview-web-services-standards checked on December 2023)

facilitates flexible and readable data representation, often using definitions such as DTD (Document Type Definition) or XSD (XML Schema Definition) to describe the data structure.

1.4.2 SOAP

SOAP establishes the structure and format for XML-based message exchange within decentralized, distributed environments. It operates as a communication protocol, fostering seamless interaction among applications created using diverse programming languages, operating systems, and platforms.

Major technology companies, including Sun, Microsoft, and IBM, have integrated SOAP implementations into their systems. SOAP 1.2 [2] stands as the latest version, currently undergoing standardization processes by the World Wide Web Consortium (W3C). As the standard evolves, minor changes are anticipated to further refine and enhance its functionality and compatibility across various environments.

1.4.3 WSDL

WSDL serves as the standard language for describing the syntactical details of Web services. It offers a model and an XML format that precisely defines the interface of a Web service. Specifically, WSDL, developed by the World Wide Web Consortium (W3C), plays a critical role in separating the description of abstract functionality provided by a service from the concrete implementation details. It defines the interface between the Web service and its requesters.

The interface, referred to as a **port type** in WSDL 1.x and an **interface** in version 2.0, provides comprehensive signatures for all operations. These include the operation name, inputs, outputs, and potential faults. Additionally, it furnishes information about the service itself.

The latest version of the WSDL standard is WSDL 1.1, available at the link [3]. However, WSDL 2.0 has emerged as a candidate recommendation. WSDL 1.1 incorporates XML Schema Definition (XSD), a specification enabling the creation of complex types [4]. The utilization of XSD in WSDL 1.1 facilitates the definition of sophisticated data types used within the service.

WSDL acts as a crucial intermediary, offering a structured and standardized means to describe the interface details of Web services, thereby enabling seamless interaction between services and requesters.

1.4.4 UDDI

To register, advertise, and discover Web services, the UDDI specification Alston et al. [2002] is the needed standard in this case. UDDI standard defines a SOAP-based Web service for locating WSDL descriptions of Web services to the mitigation of the web by discovering the suitable services among thousands by following the upcoming process :

2. http ://www.w3.org/TR/soap (checked online on March 2025)
3. http ://www.w3.org/TR/wsdl (checked online on March 2025)
4. http ://www.w3.org/XML/Schema (checked online on March 2025)

1. The standard defines the content and the type of access provided by service registries.
2. These registries provide the advertisement of the services that can be invoked by a client.
3. UDDI can store descriptions about internal Web services across an organization and public Web services located on the Internet. Pennington [2007], Booth et al. [2004]

1.5 Other Web Services Standards and Protocols

In addition to the four aforementioned basic standards, these standards play crucial roles in extending the capabilities and functionalities of web services :

— **WS-Policy and WS-PolicyAttachment** : WS-Policy provides a framework to define a service's requirements and capabilities. It organizes policies in XML, allowing service providers to publish these policies alongside service descriptions. WS-PolicyAttachment, on the other hand, defines methods for attaching these policies to WSDL files, making them available in service directories like UDDI. These policies can be referenced directly within WSDL or linked as separate XML files via URI, providing flexibility and hierarchy in policy application. Both WS-Policy and WS-PolicyAttachment are currently under W3C standardization.

— **Web Service Security** : Security is crucial in web services to prevent unauthorized access and safeguard privacy. Web services, unlike traditional client-server models, are distributed and thus more vulnerable to privacy issues due to the increased sharing of information across networks. Security threats include attacks like Man-In-The-Middle, unauthorized access, and SQL injection. Encryption, proper authentication, and secure coding practices are essential to mitigate these risks. WS-Security specifies frameworks for securing messages, including encryption and integrity mechanisms. Additionally, protocols like WS-Trust, WS-SecureConversation, and WS-Privacy offer solutions for establishing trust and securing sensitive data exchanges.

— **WS-Transaction and Composite Applications** : In distributed environments, transaction management is essential for ensuring reliability across multiple participants. WS-Transaction specifications like WS-Coordination, WS-AtomicTransaction, and WS-BusinessActivity help define transactional contexts and properties for short or long-running transactions. Composite applications benefit from managed environments that provide features like event logging, fault tolerance, and security, which are essential for ensuring reliability.

— **Messaging Reliability** : WS-ReliableMessaging ensures message delivery even in unreliable environments, guaranteeing messages are received exactly once and in the correct order. WS-Eventing and WS-Notification are specifications for handling asynchronous messaging and event notifications, enabling services to communicate efficiently in dynamic environments.

Web services security, transaction management, and messaging reliability are fundamental to their effective use in distributed applications. Policies like WS-Policy and WS-PolicyAttachment, along with security frameworks like WS-Security, address the critical needs for trust, encryption, and privacy in service-oriented architectures.

1.6 Developing Web Services

Creating a Web Service is similar to developing other software, but there are some differences as it can begin by creating a WSDL specification, or by creating, for example, a Java interface or abstract class. Since tools such as Axis or Radiant can convert one form to the other, it is a matter of preference where to start. By following fundamental software engineering techniques, Web services can be created as follows : Pennington [2007], Booth et al. [2004]

1. **Create an UML Class Diagram** : The initial step involves establishing a Unified Modeling Language (UML) Class Diagram. This diagram defines the classes that will be essential for the web service. UML provides a comprehensive representation to model these classes, aiding in the conceptualization of the service's structure.
2. **Generate Java Code** : The UML Class Diagram is then transformed into a Java class skeleton. Adhering to Java bean programming conventions, this step includes implementing "getters "and "setters "for each member variable, outlining the basic structure of the service in Java code.
3. **Integrating Web Services Annotations** : Java 6 includes the necessary annotations that simplify the process, allowing the compiler to recognize the program code as a Web service. A partial list of available annotations is provided below :
   - javax.jws.WebService
   - javax.jws.WebMethod
   - javax.jws.WebParam
   - javax.jws.WebResult
   - javax.jws.HandlerChain
   - javax.jws.soap.SOAPBinding
4. **Generate WSDL** : The annotations from the previous step indicate to the Annotation Processing Tool or the Java compiler that a WSDL file should be generated at compile time. This description can be used in two ways : First, it serves as an advertisement when published on the Web. The information from the WSDL file is registered in UDDI repositories, allowing queries to discover the required service. Second, it provides all the necessary information to invoke the service remotely.
5. **Implement Methods** : It is needed to create an implementation class that extends the abstract class. The difference resides within the need for the developer to write a proper code from the beginning. Any created class must have **getters** and **setters** for all member variables and they are used during invocation by the SOAP engine to serialize and deserialize the data that is in the SOAP messages into Java objects and back to SOAP.
6. **Deploy Service** : Deployment involves hosting the service on a web application server along with a SOAP engine, like Tomcat and Axis2 respectively. This step puts the service code onto the server, making it accessible to clients for interaction.

7. **Test Service** : A simple Java program can be enough to test a service. However,it requires from time to time a more complex client code which needs End Point Reference that is a URL to the service, a call setting the target, and setting the transport information.
8. **Publish Service** : to publish a service, UDDI registries are required after choosing one. After deployment and testing, the service is open and ready to accept requests, however, before getting published, it is unlikely for the service would be known. Tools that can simplify this process are Radiant and Lumina , both from the METEOR-S tool suite.

2 Internet of Things (IoT)

The Internet of Things (IoT) represents a revolutionary paradigm that bridges the gap between the physical and digital realms, reshaping the landscape of information technology. This convergence has brought forth a multitude of possibilities, redefining the way we interact with the world around us. The idea of having the ability to monitor and control things in the physical world electronically has inspired a wave of innovation, which can be briefly outlined as follows :

1. **Interconnected Physical-Digital Fusion** : IoT's core essence lies in interweaving physical objects, devices, and systems with digital technologies. This integration enables remote interaction, monitoring, and management of physical entities through electronic means.
2. **Inspiration for Innovation** : The notion of electronically monitoring and controlling physical entities has sparked an era of profound innovation. IoT's potential to transform various aspects of our lives has been a driving force behind revolutionary advancements.
3. **Revolutionizing Asset Management** : IoT has redefined how businesses manage their physical assets. Through real-time tracking, predictive maintenance, and data analytics, companies optimize asset utilization and operational efficiency.
4. **Personal Health and Fitness Monitoring** : IoT-enabled devices allow users to monitor their health and fitness in real time. Wearables, health trackers, and connected devices enable individuals to track, analyze, and proactively manage their well-being.
5. **Smart City Development** : IoT's influence extends to urban environments, transforming cities into smart, efficient hubs. It facilitates better waste management, traffic control, energy consumption optimization, and enhanced public safety measures.
6. **Enhancing Vehicle Efficiency** : In the automotive sector, IoT plays a pivotal role in improving vehicle efficiency. IoT devices enable tracking, performance monitoring, and data-driven insights that optimize fuel consumption and reduce environmental impact.

This section introduces the definitions, concepts and standards related to Internet of Things and their technologies.

2.1 Definitions of IoT

A multitude of IT companies have put forward various definitions associated with the IoT, emphasizing its relationship with technology and communication facilitated by Internet connectivity, infrastructure, or the interconnection of disparate entities. Some notable definitions among these include :

— **IEEE** : (Institute of Electrical and Electronics Engineers) “The IoT is an intelligent network which connects all things to the Internet for the purpose of exchanging information and communicating through the information sensing devices in accordance with agreed protocols. It achieves the goal of intelligent identifying, locating, tracking, monitoring, and managing things.”Stankovic [2014]

— **ITU** : (International Telecommunication Union) “Internet of things (IoT) is a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving inter-operable information and communication technologies.” [5]

— **IETF** : (Internet Engineering Task Force) “A world-wide network of interconnected objects uniquely addressable based on standard communication protocols.”Lee et al. [2012]

— **CCSA** : (China Communications Standards Association) “a network, which can collect information from the physical world objects through various deployed devices with capability of perception, computation, execution and communication, and support communications between human and things or between things by transmitting, classifying and processing information.” [6]

— **CASAGRAS** : (Coordination And Support Action for Global RFID-related Activities and Standardisation) “a global network infrastructure, linking physical and virtual objects through the exploitation of data capture and communication capabilities.”Smith et al. [2009]

— **IBM** :(The International Business Machines Corporation) “The Internet of Things is the concept of connecting any device (so long as it has an on/off switch) to the Internet and to other connected devices. The IoT is a giant network of connected things and people – all of which collect and share data about the way they are used and about the environment around them. ” [7]

— **Oracle** : “The Internet of Things (IoT) describes the network of physical objects “things” that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet. These devices range from ordinary household objects to sophisticated industrial tools. ” [8]

5. https ://www.itu.int/en/ITU-T/Workshops-and-Seminars/iot/201402/Documents/S1P1_Marco_CARUGI.pdf (Checked on March 2025)
6. http ://www.ccsa.org.cn/english/list_std.php ?tbname=ydb_doc&keyword=&page_currentPage=4 (Checked on March 2025)
7. https ://www.ibm.com/blogs/internet-of-things/what-is-the-iot/ (Checked on line in March 2025)
8. https ://www.oracle.com/internet-of-things/what-is-iot/ (Checked on line in March 2025)

As a summary, the Internet of Things can be simply identified as a network which is connected to the Internet for the purpose of bringing together all the objects (intelligent or transformed into intelligent) that can be interconnected and communicate with other systems which are also connected to the Internet.

2.2 From a Traditional Thing to a Smart Thing

The concept of a "thing" in the context of IoT is versatile, encompassing a wide range of objects and environmental settings. These "things" can take various forms, from a person with a medical implant monitoring their organs (such as a heart monitor or hearing aid) to a vehicle equipped with sensors that alert the driver about surrounding obstacles. Each of these entities, whether natural or artificially created, possesses specific properties that define its role within the IoT ecosystem :

— **The identifier** : Every IoT-enabled entity is assigned a unique identifier. This identifier often manifests as a specific Internet Protocol (IP) address, typically drawn from the IPv6 protocol, ensuring that each entity has its distinct digital identity.

— **Sensors** : Smart objects within the IoT framework are equipped with one or multiple sensors that enable them to gather diverse types of information from their surroundings. These sensors play a crucial role in data collection, processing, and subsequent transmission.

— **Intelligence** : To make informed decisions based on the collected data, smart entities integrate inference engines and possess computational capabilities. This intelligence allows them to process data and execute actions autonomously.

— **Internet connection** : Connectivity forms the backbone of IoT, enabling seamless communication between smart objects and facilitating interaction with the broader environment or other interconnected devices.

— **Security** : Given their connectivity to the internet, ensuring secure communication and safeguarding shared data from potential cyber threats, attacks, and viruses is paramount. Employing robust network security protocols and data encryption methods is crucial.

— **Energy saving** : Smart objects need to operate efficiently with minimal energy consumption. This requirement is essential for their sustained functionality, especially when operating remotely or away from a direct power source.

— **The minimum cost** : The proliferation of IoT devices necessitates cost-effectiveness to ensure scalability. Minimizing costs associated with manufacturing, deployment, and maintenance becomes a crucial consideration.

— **Quality Standards** : Maintaining high-quality standards is imperative for IoT devices to perform reliably in various environments. Ensuring that environmental factors do not adversely affect information processing is essential to guarantee trustworthy results and decisions.

2.3 Enabling Technologies

The convergence of various enabling technologies has led to the realization of the Internet of Things (IoT), reshaping how the physical and digital realms interact. This section delves into some of the pivotal technologies that contribute significantly to the IoT landscape :

— **RFID** (Radio Frequency Identification) : RFID systems comprise readers and tags. RFID tags, affixed to objects, are equipped with unique identifiers and typically consist of a microchip attached to an antenna. These tags interact with RFID readers, which are powerful devices equipped with sufficient memory and computational resources. RFID systems facilitate the identification and tracking of objects in the physical world using radio waves. The reader queries and communicates with the tags, retrieving information stored in them, enabling efficient object tracking and identification. RFID technology finds widespread use in supply chain management, inventory tracking, and asset monitoring due to its capability for seamless identification.Mishra [2018]

— **Sensor networks** : In conjunction with RFID systems, sensor networks play a crucial role in collecting and relaying real-time information about various physical parameters related to objects. These networks enhance the tracking capabilities of IoT by monitoring parameters such as location, temperature, movement, and other environmental variables. Sensor networks consist of multiple sensing nodes interconnected wirelessly, communicating in a multi-hop manner. Typically, these nodes collect data and transmit it to specific nodes called sinks or base stations, which process and aggregate the collected information. This collaborative setup forms a bridge between the physical and digital worlds, providing comprehensive insights into the status and behavior of physical objects or environments. Atzori et al. [2010]

— **WISP** [9] (wireless identification and sensing platforms) : aWISPs represent a specific type of RFID tag that stands out due to its unique functionality. Unlike traditional RFID tags, WISPs possess the ability to collect sensory data beyond identification purposes. These platforms are powered and read by standard RFID readers, harnessing energy from the reader's querying signal. WISPs have found utility in diverse applications, measuring various environmental parameters such as light, temperature, acceleration, strain, and liquid levels. By amalgamating identification and sensing capabilities, WISPs contribute significantly to IoT applications where data collection and environmental monitoring are essential. Atzori et al. [2010]

These technologies, in their synergy, form the backbone of IoT implementations, providing the infrastructure and mechanisms necessary to connect and monitor physical objects, transforming them into intelligent entities capable of interacting with the digital world.

9. http ://seattle.intel-research.net/wisp/ (checked online on March 2025)

2.4 The Architecture of an IoT System

The architecture for IoT projects lacks a universally agreed-upon standard ; however, a prevalent approach involves a layered structure that outlines the path of data flow, enabling the connection between physical devices and the virtual world. This architecture typically consists of several layers, where each layer communicates with others to facilitate the integration of physical devices with digital systems. One commonly adopted architecture is the four-layer model, as seen in Figure. 2.5, illustrated as follows :

— **IoT device layer** : Positioned as the primary layer, this tier encompasses interconnected smart devices, which serve as the foundational elements equipped with sensors and actuators. These devices can establish connections through wired or wireless means, engaging in bidirectional communication facilitated by a gateway or Data Acquisition System (DAS). In this layer, each device operates as either a sensor or an actuator :

1. **Sensor** : A device designed to measure physical quantities in the environment, converting this data into interpretable digital signals. Sensors gather information about various physical parameters and can identify other smart objects within their surroundings.
2. **Actuator** : The component responsible for controlling or manipulating a mechanism or system. Actuators transform the digital data generated by smart devices into physical actions, influencing the environment based on received instructions.

— **IoT gateway layer** : This layer assumes the responsibility of transmitting data over networks like Wi-Fi or wired local networks. Gateways ensure data security by employing encryption tools, thereby preventing data leaks into the IoT platform. This security measure mitigates the risks associated with potential malicious external attacks on IoT devices.

— **Data processing layer** : Tasked with managing the storage, analysis, and processing of vast amounts of data, this layer employs robust data analytics engines and machine learning mechanisms. To achieve effective functionality, various technologies such as databases, cloud computing, and big data frameworks are utilized.

— **Application layer** : Serving as the top layer, this tier delivers application-specific services to users through terminals like smartphones and computers. The application layer simplifies interactions for non-expert users, presenting information in an easily understandable manner through user-friendly interfaces.

This four-layer architecture provides a structural framework for IoT systems, enabling the seamless integration of physical devices with digital systems while facilitating data flow and interaction between the physical and virtual worlds.

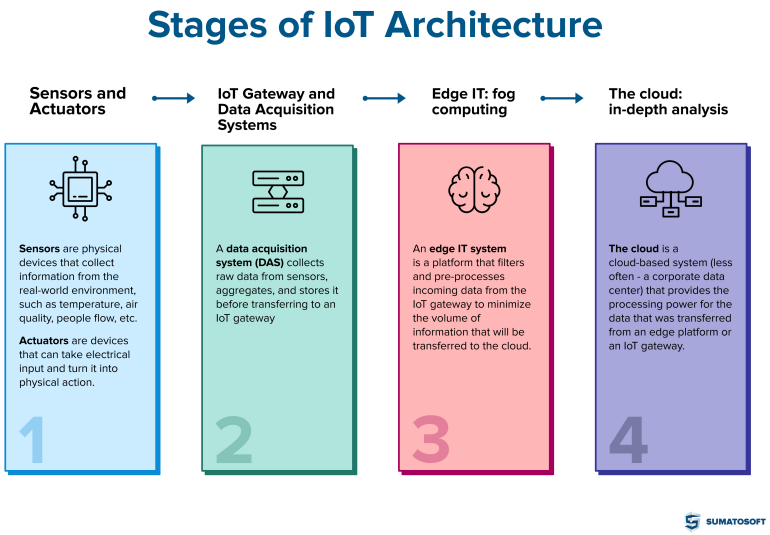


FIGURE 2.5 IoT Architecture (https ://sumatosoft.com/blog/what-is-iot-architecture-4-stages-of-iot-architecture, checked on March 2025)

2.5 Middleware

The middleware is a software layer or a set of sub-layers that plays a crucial role in bridging the gap between the technology layers and the application levels in the context of IoT. As it facilitates the development of new services, its significance has been increasingly recognized.

In recent years, middleware architectures for IoT have often embraced the principles of SOA. This approach allows for the decomposition of intricate systems into applications consisting of simpler and well-defined components. A significant advantage of SOA is its flexibility in terms of technology, enabling software and hardware reuse without mandating specific implementation technologies.

Most studies emphasize the benefits of the SOA approach in middleware solutions for IoT. Although a universally accepted layered architecture is absent, these solutions tackle similar challenges of abstracting device functionalities and communication capabilities. They aim to provide a standardized set of services and a platform for **service composition**, as depicted in Figure 2.6. Atzori et al. [2010], De Deugd et al. [2006], Pasley [2005], Spiess et al. [2009]

The middlware relies on the following layers :

1. **Applications Layer** : Positioned at the top of the architecture, this layer delivers the system's functionalities to end-users. While not strictly considered part of the middleware, it leverages all the capabilities of the underlying layers. By utilizing standard web service protocols and service composition technologies, this layer ensures seamless integration between distributed systems and applications. Atzori et al. [2010]
2. **Service composition Layer** : This layer facilitates the composition of individual services provided by networked objects to build specific applications. Here, there's no direct representation

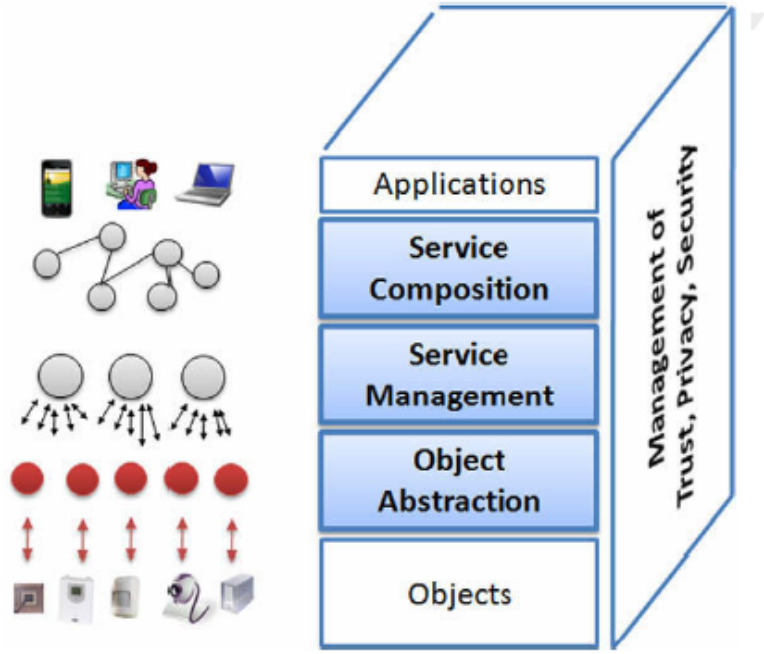


FIGURE 2.6 SOA-based architecture for the IoT middleware. Atzori et al. [2010]

of devices ; instead, services are the focal point. Services are dynamically executed at run-time to construct composite services. Workflow languages define business processes interacting with external entities via Web Service operations using WSDL.Atzori et al. [2010].

3. **Service management Layer** : This layer furnishes fundamental functions essential for managing objects in the IoT scenario. It includes object dynamic discovery, status monitoring, service configuration, and Quality of Service (QoS) management. Some middleware proposals extend these functionalities to support remote deployment of new services during runtime to meet application requirements. A service repository correlates service catalogs with respective objects in the network. Atzori et al. [2010]
4. **Object abstraction Layer** : The nature of devices and objects in the realm of IoT is incredibly diverse. Each of these objects possesses unique functionalities and interacts via distinct languages or protocols. The Object Abstraction Layer plays a pivotal role in unifying and simplifying the way various devices are accessed and managed within the IoT ecosystem. The primary objectives and functionalities of this layer include : Standardizing Communication by establishing a common interface, allowing applications and services to interact with various devices seamlessly. Unified Access and Control to access and control different devices, irrespective of their underlying technologies or manufacturers. Interoperability and Integration as it ensures that different devices, regardless of their types or functionalities, can work together harmoniously within the IoT ecosystem.Atzori et al. [2010]

5. **Trust, privacy and security management** : embedded RFID tags in the personal devices can unknowingly be triggered to reply with their ID and other information. This possibility calls for a surveillance mechanism that would invade large parts of our lives. The middleware must then include functions related to the management of the trust, privacy and security of all the exchanged data. The related functions may be either built on one specific layer of the previous ones or distributed through the entire stack, from the object abstraction to the service composition, in a manner that does not affect system performance or introduce excessive overheads. Atzori et al. [2010]

2.6 The advantages of Internet of Things

In order to operate efficiently, most organizations in the industrial community are using IoT where IoT technology has become an integral part, providing numerous benefits that significantly impact business value and operational efficiency. These advantages include :

1. **Bridging Digital and Physical Realm**s : IoT strengthens the connection between the digital and physical worlds, enabling seamless interaction between devices, sensors, and systems. This integration fosters a more interconnected and efficient ecosystem.
2. **Cost Reduction and Resource Optimization** : By leveraging IoT devices and data analytics, organizations can optimize resource usage, reduce operational costs, and identify areas of waste or inefficiency in processes, thereby enhancing overall cost-effectiveness.
3. **Enhanced Assistance for Professional and Personal Activities** : IoT technologies offer valuable support in both professional and personal spheres, providing assistance in tasks, decision-making processes, and operational workflows, thereby improving productivity and efficiency.
4. **Precise System Control with Minimal Manual Intervention** : Through IoT-enabled systems, users gain precise control over devices and processes, often without the need for manual intervention. This automation ensures accurate and efficient operations.
5. **High-Speed Data Collection and Safety Sensation** : Sensors deployed in IoT systems gather and process vast amounts of data rapidly. This real-time data collection provides a sense of security, triggering alerts or warnings in case of potential dangers or anomalies, ensuring prompt responses.
6. **Enhanced System Availability and Safety** : IoT systems contribute to increased system availability and safety by proactively identifying and mitigating potential risks or hazards, thereby preventing down-times and ensuring uninterrupted operations.
7. **Time and Effort Savings via Remote Control** : Remote management capabilities facilitated by IoT technologies allow for the control and monitoring of devices and systems from remote locations, saving time and effort that would otherwise be required for manual operations.

8. **Improved Device Performance with Reduced Human Intervention** : IoT-driven automation minimizes the need for human intervention in system operations, thereby optimizing device performance and reducing the probability of errors.
9. **Enhanced Convenience and Comfort in Daily Life** : IoT applications in daily life ensure convenience and comfort by simplifying routine tasks, improving accessibility, and offering personalized experiences tailored to individual needs.
10. **Creation of New Opportunities and Services** : The rise of IoT technology creates opportunities for the development of new services, job roles, and technological advancements, fostering innovation and growth in various industries.

2.7 The disadvantages of Internet of Things

Similarly to any existing technology, though the internet of things has brought many benefits in several areas, it exposed humanity to many types of threats that can cause specific types of damage to businesses and consumers like :

— **Human Dependence on Technology** : The increased reliance on IoT devices can lead to human inactivity, causing physical health issues due to reduced physical activity and psychological problems related to overdependence on technology.

— **Rise in Unemployment** : With automation and smart machinery taking over various tasks, there is a concern about increased unemployment rates, especially in developed societies, where technology substitutes human labor.

— **Privacy and Security Concerns** : The vast interconnectedness of IoT devices creates a potential vulnerability for cyberattacks. Hacker attacks can compromise the confidentiality of personal data transmitted and stored within IoT systems, leading to privacy breaches and identity theft.

— **Trust and Data Privacy** : Users' trust in companies manufacturing IoT devices may decline due to concerns about data privacy. Companies might misuse or exploit user data collected by IoT devices, leading to breaches of privacy and ethical concerns about data usage.

— **System Vulnerabilities and Malfunctions** : Glitches, software bugs, or vulnerabilities in IoT systems could lead to system failures or corruption. If a bug or error occurs in the system controlling connected devices, it can cause disruptions, malfunctions, or even potential safety hazards.

2.8 The challenges of Internet of Things

The Internet of Things revolution is a recent technology developed in a step-by-step process. There still problems to be solved and flaws to be eliminated according to Li et al. [2015] such as :

— **Technical Challenges** : Designing one SOA for IoT is challenging which leads service-based things to suffer in terms of their performances. Furthermore, the lack of a common service description language makes the services incompatible. Moreover, IoT is a complex heterogeneous network, which includes the connections among various types of networks through various communication technologies. Thus, the devices and methodologies to address things are still a challenge.

— **Standardization** : aims to easily integrate the new service providers and users as it can improve the interoperability and allow products or services to compete better at a higher level. The issue within standardization includes interoperability, radio access level issues, semantic interoperability, and security and privacy issues.

— **Security and privacy protection** : a reliable security protection mechanism for IoT is still in demand for data confidentiality, privacy, and trust.

— **Innovation in IoT environment** : IoT is a complex network that might be managed with great skills, and the cross-domain systems supporting innovation are still lacking.

— **Development Strategies** : Getting new opportunities for investments, short term IoT projects and integrating IoT in all IT infrastructures are the main strategies to spread the IoT technologies

In conclusion, the Internet of Things, while promising, faces significant challenges related to technical complexity, standardization, security, innovation, and development strategies. Addressing these issues will be critical for the sustained growth and success of IoT technologies in various industries.

2.9 The application fields of Internet of Things

The potential applications of the IoT span various facets of daily life for different user profiles including individuals, businesses, and society at large, impacting several domains :

— **Smart city** : IoT-enabled technologies facilitate smart traffic management, efficient lighting systems, and integrated smart home solutions, contributing to enhanced urban living experiences. Arasteh et al. [2016]

— **Intelligent environment** : Utilizing IoT sensors for predicting and detecting natural disasters like earthquakes and promptly identifying and responding to fire outbreaks. Alphonsa and Ravi [2016], Sharma et al. [2020]

— **Security and emergency management** : Implementing IoT-based solutions for monitoring radiation levels, detecting attacks and explosions, and responding to emergencies to ensure public safety. Ahmad et al. [2021], Ahmed and Jaaz [2014]

— **Logistics** : Real-time asset tracking with IoT sensors to monitor and manage location, temperature, humidity, and vibrations during transportation.Tran-Dang et al. [2022]

— **Industrial control** : Utilizing IoT for predictive maintenance, enabling timely intervention to prevent equipment breakdowns and conducting remote troubleshooting.Shahzad et al. [2017]

— **Healthcare** : IoT devices for monitoring vital signs and health metrics, aiding in remote patient monitoring and facilitating personalized healthcare.Valsalan et al. [2020], Khan et al. [2022]

— **Smart agriculture** : Applying IoT for precise environmental control in agriculture by regulating light, humidity, and temperature for optimized crop growth.Gondchawar et al. [2016]

3 Conclusion

This chapter has been focused on delineating and citing the architectures and technologies integral to both Web Services and the IoT, emphasizing their role in shaping a higher quality of life compared to the current standards.

In the upcoming chapter titled "Preliminaries on Optimization and Uncertain Numbers (Interval and Fuzzy Numbers)," our aim is to delve into the fundamental objectives and various types of optimization methods. Additionally, we will explore the key definitions and concepts related to uncertain numbers, specifically focusing on two significant types : fuzzy numbers and interval numbers.

Chapter 3

Preliminaries on optimization and uncertain numbers (Interval and fuzzy numbers)

Introduction

The upcoming chapter serves as a foundational platform to comprehend optimization problems, interval numbers, and generalized trapezoidal fuzzy numbers. These preliminaries constitute an essential stepping stone for a comprehensive understanding of the subsequent chapters, particularly Chapter 4 and Chapter 5.

By delving into these fundamental concepts in the preliminary chapter, readers will gain a solid grasp of the theoretical underpinnings essential for comprehending the subsequent chapters. The knowledge acquired regarding optimization problems, interval numbers, and generalized trapezoidal fuzzy numbers will serve as a crucial foundation for the in-depth discussions and applications illustrated in further chapters.

1 Background on optimization

1.1 Optimization problems

Optimizing refers to the process of improving or refining a problem or system in order to achieve the best possible outcome. It involves searching for the optimal solution from a range of available choices. This optimal solution represents the most efficient state or configuration of the problem under consideration.

In mathematical terms, optimization is typically framed as a maximization or minimization problem. It seeks to either maximize or minimize a particular objective function or set of criteria, depending on the nature of the problem and the variables involved.

In the realm of Computer Science, optimization techniques are employed to enhance the performance of computer systems or algorithms. This enhancement is based on specific predefined criteria or metrics that measure the system's efficiency, speed, resource utilization, or other relevant factors. The goal is to optimize the system's behavior or output, ensuring it operates at its best possible level according to the established benchmarks or objectives.

Non-scalable problems, typically simpler or less complex ones, might find their optimal solutions without the need for computers. These problems could be solved through traditional analytical or manual methods without requiring computational resources. Conversely, complex problems, especially those involving numerous variables or intricate constraints, usually demand computational power to derive optimal solutions. Computers are essential for handling the complexity and computational load of these intricate problems.Amiri et al. [2017], Ruder [2016]

Many optimization problems are nonlinear, meaning the relationship between the variables and the objective function isn't a simple linear one. These problems often operate under multifaceted constraints where various objectives might conflict with each other. Such complexities make finding an optimal solution challenging Amiri et al. [2017], Ruder [2016].

Due to the complexities involved, obtaining an optimal solution for nonlinear and multi-objective optimization problems can be considered a difficult task. Conflicting objectives, intricate constraints, and nonlinear relationships between variables pose significant challenges. Achieving an optimal or near-optimal solution for these problems is computationally demanding and often requires sophisticated optimization techniques and algorithms.Amiri et al. [2017], Ruder [2016]

The realm of optimization is supported by a diverse array of algorithms, each aimed at attaining optimal or near-optimal solutions for various problems. These algorithms can be broadly classified into different categories, with meta-heuristic algorithms emerging as effective tools for addressing complex optimization challenges Amiri et al. [2017], Ruder [2016].

Figure 3.1 illustrates the classification of optimization algorithms into two primary classes :

1. **Exact optimization methods** refer to algorithms and techniques used to solve optimization problems precisely, aiming to find the globally optimal solution within a given problem space. These methods guarantee finding the best possible solution based on specific criteria, such as maximizing or minimizing an objective function while satisfying defined constraints.

   These approaches explore various strategies to mathematically model and systematically search for the most favorable solution without approximation or error. They are particularly advantageous for problems where an optimal solution must be determined with certainty, regardless of the computational resources required. Key characteristics of exact optimization methods include :

— **Iterative methods** : These optimization techniques involve a process of continuously refining a solution by iteratively moving towards an optimal. They start with an initial solution and iteratively modify it to improve its quality based on certain criteria or constraints. Through successive iterations, these methods adjust the solution until a stopping criterion is met, typically when no further improvements can be made. Gradient descent, Newton's method, and linear programming solvers like the Simplex algorithm are examples of iterative methods used in optimization.

— **Enumerative methods** : These methods involve an exhaustive search through the entire solution space to identify the optimal solution. Unlike iterative methods that refine a given solution, enumerative methods systematically explore all possible solutions within the solution space to identify the best one. While they guarantee finding the optimal solution in theory, enumerative methods can be computationally expensive, especially in high-dimensional or complex problem spaces, as they require evaluating a large number of potential solutions. Examples include brute-force search, exhaustive enumeration, and certain types of branch-and-bound algorithms used in optimization problems.

2. **Approximate methods** : are techniques employed to find solutions that are close to the optimal solution of an optimization problem. Unlike exact optimization methods that guarantee the precise best solution, approximate methods aim to deliver solutions that are near-optimal within a reasonable computational time frame. They can be categorized into two main parts :

   — **Search algorithms** : refer to the capability of optimization algorithms to explore and navigate through the solution space effectively in search of an optimal or near-optimal solution for a given problem. The search ability of an algorithm is crucial, as it determines how well and efficiently the algorithm can traverse the solution space to find the best possible solution. They can be simply classified into :

   (a) **Local search algorithms** : These methods are geared towards investigating solutions within a limited or specific range of the solution space. They intensively explore a local neighborhood around a given solution, aiming to improve it incrementally. However, due to their localized nature, they often converge to local optima, achieving the best solution within the immediate vicinity but failing to explore beyond it. As a result, these algorithms might miss out on the best possible solution available globally.

   (b) **Global search algorithms** : These algorithms address the limitations of local search techniques. They adopt strategies, often utilizing meta-heuristic approaches, to break away from local optima and navigate the broader solution space. By exploring multiple regions simultaneously and employing more comprehensive search strategies, global search algorithms aim to locate the global optimum, which represents the best solution across the entire solution space. These methods are

generally more effective in discovering the optimal solution in complex, multi-dimensional problem spaces compared to local search algorithms.

— **Meta-heuristics** : Many meta-heuristic algorithms exist in literature such as Genetic Algorithms (GA) ,Particle Swarm Optimization (PSO), Bee Colony Algorithm (BCA), and Firefly Algorithm (FA Amiri et al. [2017], Eusuff et al. [2006]. These algorithms can be classified into :

(a) **Trajectory-based algorithms** : These types of optimization algorithms employ a single agent or solution trajectory to navigate towards the optimal solution. At each iteration, the algorithm progresses along a specific path, making adjustments or changes to refine the solution towards optimization. It's akin to a single path-finding approach, where the algorithm steadily advances with each step, gradually converging towards an optimal or near-optimal solution.

(b) **Population-based algorithms** : Unlike trajectory-based methods, population-based optimization algorithms involve multiple agents or solutions exploring various potential paths simultaneously. Each agent or solution represents a possible solution to the problem. These methods operate by maintaining a population of solutions and iteratively evolving or updating these solutions based on certain rules or algorithms. This simultaneous exploration of multiple paths allows for a broader exploration of the solution space, increasing the likelihood of finding a globally optimal solution rather than converging to a local optimum.

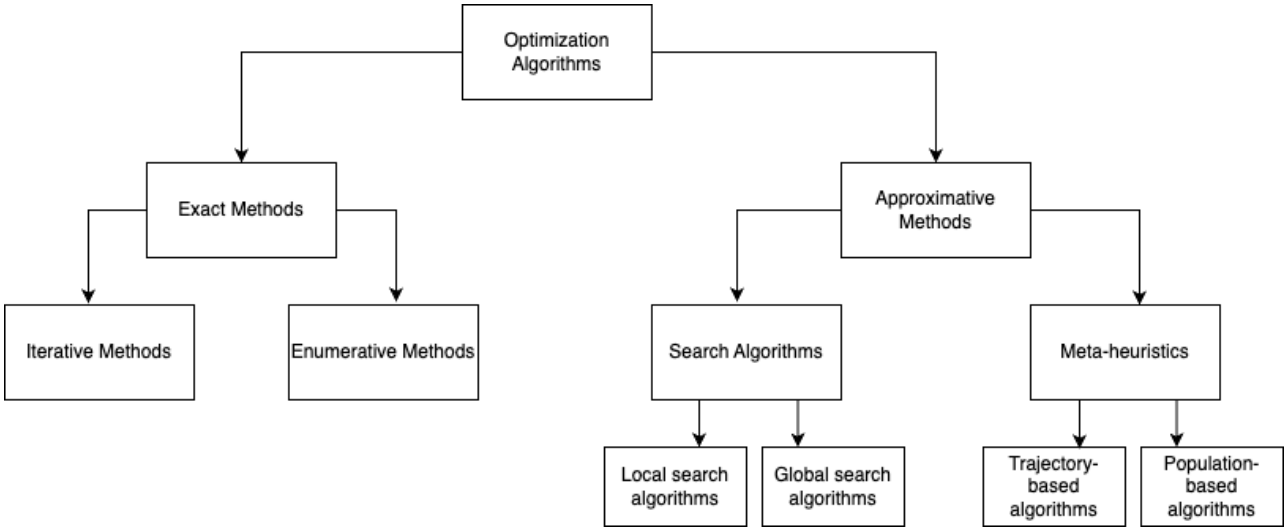


FIGURE 3.1 Classification of optimization techniques

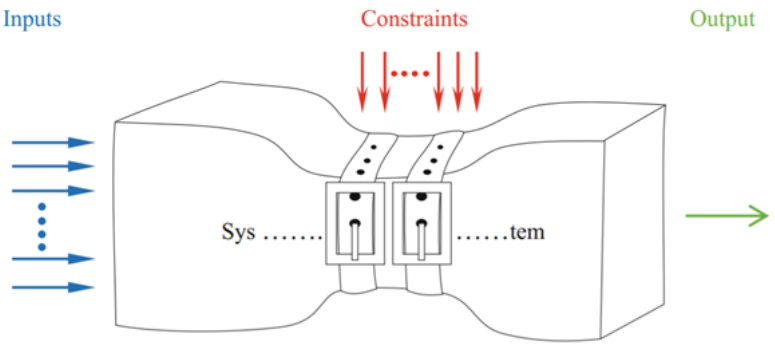


FIGURE 3.2 Three main components of an optimization system with single objective : inputs, constraints (operating conditions), and output Mirjalili [2019].

1.2 Single/Multi-Objective Optimization

1.2.1 Single objective optimization

In any single-objective problem, several critical components contribute to defining and solving the optimization challenge. These components include inputs seeking optimal values, constraints that must be adhered to, and the system's resulting outputs, as depicted in Figure 3.2. These elements collectively shape the boundaries of a system, guiding the search for a suitable solution by identifying valid sets of input values. Any solution that violates a constraint is considered invalid, regardless of its objective value Mirjalili [2019].

When faced with multiple criteria, merging them into a single-objective optimization problem becomes possible by formulating a unified cost function. This single-objective function aggregates the various criteria by assigning weights to each and computing a weighted sum of their normalized costs. Each metric contribution to the final function is scaled and combined, allowing for a comprehensive assessment of the trade-offs between different objectives.

By combining multiple metrics into a single-objective function, decision-makers can streamline the optimization process, creating a unified criterion that captures the essence of various considerations. This approach simplifies the evaluation and facilitates the search for an optimal solution that balances multiple objectives effectively.

A minimized single-objective optimization problem can be formulated as follows :

$$\begin{cases} \min : f(x_1, x_2, \ldots, x_n) \\ \text{s.t} : g_i(x_1, x_2, \ldots, x_n) \geq 0, i = 1, 2, \ldots, m \\ g_i(x_1, x_2, \ldots, x_n) = 0, i = 1, 2, \ldots, p \\ lb_i \leq x_i \leq ub_i, i = 1, 2, \ldots, n \end{cases} \tag{3.1}$$

Where :

— n is the number of variables,

— m is the number of inequality constraints,

— p is the number of equality constraints,

— lb_i is the lower bound of the i-th variable,

— ub_i is the upper bound of the i-th variable.

1.2.2 Multi-objective optimization

Several fields, including engineering, economics, and logistics, encounter scenarios where optimal decisions must navigate trade-offs between two or more conflicting objectives, all needing simultaneous optimization. Multi-Objective Optimization (MOO), credited to ***Vilfredo Pareto***, has proven instrumental in handling such complexities. In MOO, an objective function vector exists, where each vector represents a function of the solution vector. Contrary to seeking a singular best solution, MOO aims to generate multiple solutions that offer trade-offs among conflicting objectives.

Mathematically, the formulation of a MOO problem involves defining an equation that encapsulates multiple objective functions. The objective function vector represents various criteria or goals that need to be simultaneously optimized Gunantara [2018]. The problem can be represented as :

$$\begin{cases} \min / \max : f_1(x), f_1(x), \ldots, f_n(x) \\ \text{s.t} : x \in U \end{cases} \tag{3.2}$$

Where :

— x is a solution,

— n is the number of objective function,

— U is feasible set,

— $f_n(x)$ is n-th objective function,

— $\min / \max$ is combined object operations.

A set of optimal Pareto solutions exists because it's not feasible to have one solution that optimizes every goal simultaneously. In this case, all Pareto solutions are considered equally good.

MOO is also known as **multi-objective programming**, **vector optimization**, **multi-criteria optimization**, **multi-attribute optimization**, or **Pareto optimization**.

1.3 Continuous and discrete optimization problems

Models in optimization problems vary based on the type of variables they incorporate. The distinction lies between discrete optimization problems, where variables are confined to discrete sets—often subsets of integers, and continuous optimization problems, where variables can take on any real value within a specific range impacting the problem-solving approach due to mathematical and computational differences.

In particular, optimization problems dealing with binary (restricted to values 0 and 1) or integer variables face distinct challenges compared to continuous problems. The key challenge arises from the nature of these variables and their effect on the mathematical functions used in optimization, especially regarding derivatives.

1.3.1 Continuous optimization problem

In **continuous optimization problems**, as outlined by Rubinstein [1999], Rao et al. [2012], variables possess the freedom to assume real number values. This stands in contrast to **discrete optimization**, where variables can be binary (limited to values of 0 and 1), integer, or selected from sets with a finite number of elements.

Within continuous optimization, a pivotal differentiation exists between problems with no constraints on variables (**unconstrained optimization problems**) and those with explicit limitations on variables (**constrained optimization problems**). Unconstrained optimization problems are common in various practical applications and often surface when constraints are transformed into penalty terms within the objective function.

On the other hand, **constrained optimization problems** emerge from scenarios where specific limitations govern the variables. Various specialized areas within constrained optimization have their own sets of algorithms tailored to address these constraints effectively.

1.3.2 Discrete optimization problems

Discrete optimization problems restrict some or all variables within a model to discrete sets, contrasting with continuous optimization where variables have a continuous range. Discrete optimization branches into two key categories :

— **Integer Programming**, where the discrete set comprises a subset of integers.

— **Combinatorial Optimization**, where the discrete set is composed of combinatorial structures, like assignments, combinations, routes, schedules, or sequences. This subject has been extensively explored in literature, particularly in works such as Lawler [1972], Bertsimas and Sim [2003], Lawler and Bell [1966].

It's notable that optimization problems involving binary or integer variables face challenges with discontinuous derivatives, rendering these problems more intricate compared to continuous problems.

2 Background on Generalized Trapezoidal Fuzzy Numbers (GTrFNs)

In this section, we delve into the concept of the generalized trapezoidal fuzzy number (GTrFN) and its fundamental properties. A GTrFN is a type of fuzzy number characterized by a membership

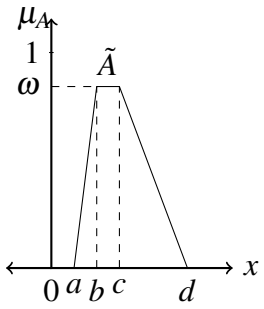


FIGURE 3.3 Graphical representation of a generalized trapezoidal fuzzy number

function that represents uncertainty within a defined range. It extends the concept of a standard trapezoidal fuzzy number to provide a more versatile and flexible representation of uncertainty.

We explore the basic arithmetic operations applicable to sets of GTrFNs, which include operations like addition, subtraction, multiplication, and division. These operations allow for computations involving fuzzy numbers, enabling the manipulation and analysis of uncertain data in various applications.

Furthermore, we introduce two essential fuzzy operators : the fuzzy minimum operator and the fuzzy maximum operator. These operators are crucial in the context of fuzzy optimization problems. The fuzzy minimum operator helps rank GTrFN values, particularly in scenarios involving fuzzy minimization problems, where the objective is to find the minimum value under uncertainty. Conversely, the fuzzy maximum operator aids in ranking GTrFN values for fuzzy maximization problems, where the goal is to find the maximum value amidst uncertainty.

Understanding these fundamental operations and operators is pivotal in handling uncertainty within optimization problems involving generalized trapezoidal fuzzy numbers, enabling decision-making in scenarios where imprecision and ambiguity are present.

Definition 6 *Chen and Chen [2003] A GTrFN $\tilde{A} = (a,b,c,d;\omega)$ is a fuzzy subset of the real line $\mathbb{R}$ that is defined by the membership function μ_A as*

$$\mu_A(x) = \begin{cases} \frac{x-a}{b-a}, & a < x < b \\ \omega, & b \leq x \leq c \\ \frac{d-x}{d-c}, & c < x < d \\ 0, & otherwise \end{cases}, \tag{3.3}$$

where $a,b,c,d,w \in \mathbb{R}$ with $a \leq b \leq c \leq d$ and $0 < \omega \leq 1$.

For a graphical representation, $\tilde{A}$ can be shown as given in Figure 3.3, which represents the real values of $\mu_A(x) \in [0,\omega]$ for each $x \in \mathbb{R}$.

Definition 7 *Given a GTrFN $\tilde{A} = (a,b,c,d;\omega)$, if $b = c$ then $\tilde{A}$ is called a generalized triangular fuzzy Number (GTFN), and if $a = b$ and $c = d$ then $\tilde{A}$ is called a generalized interval fuzzy number (GIFN), and if $a = b = c = d$ then $\tilde{A}$ is called a generalized crisp fuzzy number (GCFN).*

Definition 8 *Given a GTrFN $\tilde{A} = (a,b,c,d;\omega)$, If $\omega = 1$, then a GTrFN $\tilde{A}$ is called a normal trapezoidal fuzzy number (TrFN) $\tilde{A} = (a,b,c,d)$, and a GTFN $\tilde{A}$ is called a normal triangular fuzzy number (TFN) $\tilde{A} = (a,b,b,d)$, and a GIFN $\tilde{A}$ is called a normal interval fuzzy number (IFN) $\tilde{A} = (a,a,d,d)$, and a GCFN $\tilde{A}$ is called a normal crisp fuzzy number (CFN) $\tilde{A} = (a,a,a,a)$.*

Figure 3.4 shows the graphical representations of TrFN, TFN, IFN and CFN where the GTrFN is more general and covers them as particular cases.

Definition 9 *Chen and Chen [2003] The fuzzy arithmetic operations between two GTrFNs $\tilde{A} = (a_1,a_2,a_3,a_4;\omega_A)$ and $\tilde{B} = (b_1,b_2,b_3,b_4;\omega_B)$ are as follows :*

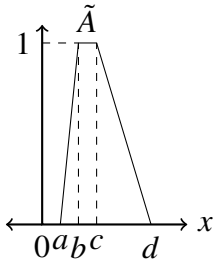
- ***GTrFNs Addition*** $\oplus$ ***:***
  $\tilde{A} \oplus \tilde{B} = (a1,a2,a3,a4;\omega_A) \oplus (b1,b2,b3,b4;\omega_B) = (a1+b1, a2+b2, a3+b3, a4+b4; min(\omega_A,\omega_B))$
- ***GTrFNs Subtraction*** $\ominus$ ***:***
  $\tilde{A} \ominus \tilde{B} = (a1,a2,a3,a4;\omega_A) \ominus (b1,b2,b3,b4;\omega_B) = (a1-b4, a2-b3, a3-b2, a4-b1; min(\omega_A,\omega_B))$
- ***GTrFNs Multiplication*** $\otimes$ ***:***
  $\tilde{A} \otimes \tilde{B} = (a1,a2,a3,a4;\omega_A) \otimes (b1,b2,b3,b4;\omega_B) = (a,b,c,d; min(\omega_A,\omega_B))$, *where :* $a = min(a1*b1, a1*b4, a4*b1, a4*b4)$, $b = min(a2*b2, a2*b3, a3*b2, a3*b3)$, $c = max(a2*b2, a2*b3, a3*b2, a3*b3)$ *and* $d = max(a1*b1, a1*b4, a4*b1, a4*b4)$.
- ***GTrFNs Scalar Multiplication :*** *for any real number* $\lambda \in \mathbb{R}$,
  $$\lambda \otimes \tilde{A} = \lambda \otimes (a1,a2,a3,a4;\omega_A) = \begin{cases} (\lambda*a1, \lambda*a2, \lambda*a3, \lambda*a4; \omega_A) \text{ if } \lambda >= 0 \\ (\lambda*a4, \lambda*a3, \lambda*a2, \lambda*a1; \omega_A) \text{ if } \lambda < 0 \end{cases}$$
- ***GTrFNs Division*** $\oslash$ ***:***
  - *For the GTrFN $\tilde{A} = (a1,a2,a3,a4,\omega_A)$, if $a1,a2,a3$ and $a4$ are all nonzero positive-real numbers or all nonzero negative-real numbers, then, the inverse of $\tilde{A}$ is defined as follows :* $1 \oslash \tilde{A} = (\frac{1}{a4}, \frac{1}{a3}, \frac{1}{a2}, \frac{1}{a1}, \omega_A)$.
  - *For the GTrFNs $\tilde{A} = (a1,a2,a3,a4,\omega_A)$ and $\tilde{B} = (b_1,b_2,b_3,b_4;\omega_B)$, if $a1, b1, a2, b2, a3, b3$, $a4$, and $b4$ are all nonzero positive-real numbers or all nonzero negative-real numbers, then, the division of $\tilde{A}$ and $\tilde{B}$ is defined as follows :*
    $\tilde{A} \oslash \tilde{B} = (\frac{a1}{b4}, \frac{a2}{b3}, \frac{a3}{b2}, \frac{a4}{b1}; min(\omega_A,\omega_B))$.

Given a GTrFN $\tilde{A} = (a_1,a_2,a_3,a_4;\omega_A)$, the Mid-Support ($\mathscr{MS}$), Mid-Kernel ($\mathscr{MK}$), Support-Width ($\mathscr{SW}$) and Kernel-Width ($\mathscr{KW}$) of $\tilde{A}$, are defined respectively, as follows :

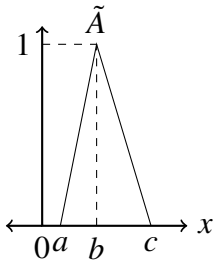
$$\mathscr{MS}(\tilde{A}) = \frac{(a4+a3+a2+a1)}{4} \tag{3.4}$$

$$\mathscr{MK}(\tilde{A}) = \frac{(a3+a2)}{2} \tag{3.5}$$

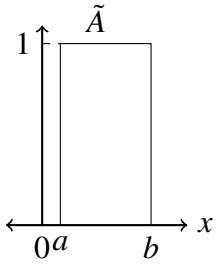
$$\mathscr{SW}(\tilde{A}) = \frac{(a4-a1)}{2} \tag{3.6}$$



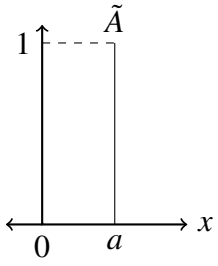
(a) TrFN



(b) TFN



(c) IFN



(d) CFN

FIGURE 3.4 Graphical representations of TrFN, TFN, IFN and CFN.

$$\mathscr{KW}(\tilde{A}) = \frac{(a3-a2)}{2} \tag{3.7}$$

Based on the prescribed equations, two new definitions of fuzzy order relations for ranking GTrFNs values of minimization (maximization) optimization problems are concluded as in the following :

Definition 10 *For two GTrFNs $\tilde{A} = (a_1, a_2, a_3, a_4; \omega_A)$ and $\tilde{B} = (b_1, b_2, b_3, b_4; \omega_B)$, the fuzzy minimum order relation for a fuzzy minimization optimization problem, denoted as $\prec_{min}$, is defined as follows :*

1. *If $\mathscr{MS}(\tilde{A}) < \mathscr{MS}(\tilde{B})$ then $\tilde{A}$ is smaller than $\tilde{B}$ (represented as $\tilde{A} \prec_{min} \tilde{B}$)*
2. *If $\mathscr{MS}(\tilde{A}) = \mathscr{MS}(\tilde{B})$ then :*
   (a) *If $\mathscr{MK}(\tilde{A}) < \mathscr{MK}(\tilde{B})$ then $\tilde{A} \prec_{min} \tilde{B}$*
3. *If $\mathscr{MS}(\tilde{A}) = \mathscr{MS}(\tilde{B})$ and $\mathscr{MK}(\tilde{A}) = \mathscr{MK}(\tilde{B})$ then :*
   (a) *If $\mathscr{SW}(\tilde{A}) < \mathscr{SW}(\tilde{B})$ then $\tilde{A} \prec_{min} \tilde{B}$*
4. *If $\mathscr{MS}(\tilde{A}) = \mathscr{MS}(\tilde{B})$, $\mathscr{MK}(\tilde{A}) = \mathscr{MK}(\tilde{B})$ and $\mathscr{SW}(\tilde{A}) = \mathscr{SW}(\tilde{B})$ then :*
   (a) *If $\mathscr{KW}(\tilde{A}) < \mathscr{KW}(\tilde{B})$ then $\tilde{A} \prec_{min} \tilde{B}$*
5. *If $\mathscr{MS}(\tilde{A}) = \mathscr{MS}(\tilde{B})$, $\mathscr{MK}(\tilde{A}) = \mathscr{MK}(\tilde{B})$, $\mathscr{SW}(\tilde{A}) = \mathscr{SW}(\tilde{B})$ and $\mathscr{KW}(\tilde{A}) = \mathscr{KW}(\tilde{B})$ then :*
   (a) *If $\omega_A < \omega_B$ then $\tilde{A} \prec_{min} \tilde{B}$*

Definition 11 *For two GTrFNs $\tilde{A} = (a_1, a_2, a_3, a_4; \omega_A)$ and $\tilde{B} = (b_1, b_2, b_3, b_4; \omega_B)$, the fuzzy maximum order relation for a fuzzy maximization optimization problem, denoted as $\succ_{max}$, is defined as follows :*

1. *If $\mathscr{MS}(\tilde{A}) > \mathscr{MS}(\tilde{B})$ then $\tilde{A}$ is greater than $\tilde{B}$ (represented as $\tilde{A} \succ_{max} \tilde{B}$)*
2. *If $\mathscr{MS}(\tilde{A}) = \mathscr{MS}(\tilde{B})$ then :*
   (a) *If $\mathscr{MK}(\tilde{A}) > \mathscr{MK}(\tilde{B})$ then $\tilde{A} \succ_{max} \tilde{B}$*
3. *If $\mathscr{MS}(\tilde{A}) = \mathscr{MS}(\tilde{B})$ and $\mathscr{MK}(\tilde{A}) = \mathscr{MK}(\tilde{B})$ then :*
   (a) *If $\mathscr{SW}(\tilde{A}) < \mathscr{SW}(\tilde{B})$ then $\tilde{A} \succ_{max} \tilde{B}$*
4. *If $\mathscr{MS}(\tilde{A}) = \mathscr{MS}(\tilde{B})$, $\mathscr{MK}(\tilde{A}) = \mathscr{MK}(\tilde{B})$ and $\mathscr{SW}(\tilde{A}) = \mathscr{SW}(\tilde{B})$ then :*
   (a) *If $\mathscr{KW}(\tilde{A}) < \mathscr{KW}(\tilde{B})$ then $\tilde{A} \succ_{max} \tilde{B}$*
5. *If $\mathscr{MS}(\tilde{A}) = \mathscr{MS}(\tilde{B})$, $\mathscr{MK}(\tilde{A}) = \mathscr{MK}(\tilde{B})$, $\mathscr{SW}(\tilde{A}) = \mathscr{SW}(\tilde{B})$ and $\mathscr{KW}(\tilde{A}) = \mathscr{KW}(\tilde{B})$ then :*
   (a) *If $\omega_A > \omega_B$ then $\tilde{A} \succ_{max} \tilde{B}$*

Adding to the two aforesaid definitions, we have provided the following observations :

$(O1)$ For the fuzzy minimization (maximization) optimization problems, If $\mathscr{MS}(\tilde{A}) = \mathscr{MS}(\tilde{B})$, $\mathscr{MK}(\tilde{A}) = \mathscr{MK}(\tilde{B})$, $\mathscr{SW}(\tilde{A}) = \mathscr{SW}(\tilde{B})$, $\mathscr{KW}(\tilde{A}) = \mathscr{KW}(\tilde{B})$ and $\omega_A = \omega_B$ then $\tilde{A}$ and $\tilde{B}$ have the same rank (represented as $\tilde{A} \sim \tilde{B}$).

$(O2)$ For the fuzzy minimization optimization problems, the fuzzy smaller or equal operator, denoted as $\preccurlyeq_{min}$, is defined as

— $\tilde{A} \preccurlyeq_{min} \tilde{B} \Leftrightarrow (\tilde{A} \prec_{min} \tilde{B})$ or $(\tilde{A} \sim \tilde{B})$

$(O3)$ For the fuzzy maximization optimization problems, the fuzzy greater or equal operator, denoted as $\succcurlyeq_{max}$, is defined as

— $\tilde{A} \succcurlyeq_{max} \tilde{B} \Leftrightarrow (\tilde{A} \succ_{max} \tilde{B})$ or $(\tilde{A} \sim \tilde{B})$

Moreover, based on the above ranking definitions, given any three GTrFN $\tilde{A}$, $\tilde{B}$ and $\tilde{C}$, the following proprieties are introduced :

$(\mathscr{P}1)$ $\tilde{A} \preccurlyeq_{min} (\succcurlyeq_{max})\tilde{A}$, which is called reflexivity.

$(\mathscr{P}2)$ If $\tilde{A} \preccurlyeq_{min} (\succcurlyeq_{max})\tilde{B}$ and $\tilde{B} \preccurlyeq_{min} (\succcurlyeq_{max})\tilde{A}$ then $\tilde{A} \sim \tilde{B}$, which is called anti-symmetry.

$(\mathscr{P}3)$ If $\tilde{A} \preccurlyeq_{min} (\succcurlyeq_{max})\tilde{B}$ and $\tilde{B} \preccurlyeq_{min} (\succcurlyeq_{max})\tilde{C}$ then $\tilde{A} \preccurlyeq_{min} (\succcurlyeq_{max})\tilde{C}$, which is called transitivity.

For the proofs of these proprieties, readers can refer to appendix B. As result, $\preccurlyeq_{min}$ and $\succcurlyeq_{max}$ are partial order relations among GTrFNs.

3 Generalities on Interval-numbers

In this section, we delve into the concept of interval numbers and their fundamental characteristics. Interval numbers are mathematical constructs that represent a range of real numbers within defined bounds. They are commonly depicted as intervals between lower and upper bounds, signifying uncertainty or imprecision in the represented values.

We explore the basic arithmetic operations applicable to sets of interval numbers, including addition, subtraction, multiplication, and division. These operations allow for computations involving intervals, enabling the manipulation and analysis of uncertain or imprecise data in various applications.

Furthermore, we introduce two essential interval operators : the interval minimum operator and the interval maximum operator. These operators are instrumental in the context of interval optimization problems. The interval minimum operator facilitates the ranking of interval values in scenarios involving interval minimization problems, where the objective is to determine the minimum value within the interval bounds. Conversely, the interval maximum operator aids in ranking interval values for interval maximization problems, where the goal is to identify the maximum value within the interval bounds.

Definition 12 *(interval number) [Bhunia and Samanta, 2014] Let's have* $(a^l, a^u) \in \mathbb{R}^2$ *two real numbers with* $a^l \leq a^u$. *The set* $A = \{x : x \in \mathbb{R} \text{ and } a^l \leq x \leq a^u\}$ *is named an interval number, which is represented as* $A = [a^l, a^u]$, *where* a^l *and* a^u *are considered as its lower (left) and upper (right) limits, respectively. The interval number A can be also expressed as* $A =< a^c, a^w >$, *where the real numbers* $a^c = (a^l + a^u)/2$ *and* $a^w = (a^u - a^l)/2$ *are its mid-point (center) and mid-width (radius), respectively. If* $a^l = a^u$, *then* $A = [a^l, a^l] = [a^u, a^u] =< a^l, 0 >=< a^u, 0 >$ *is a real number, which has zero mid-width.*

Arithmetic operations on interval numbers We have two interval numbers $A = [a^l, a^u]$ and $B = [b^l, b^u]$, the arithmetic operations, known as addition, subtraction, multiplication and division are applied on A and B as follows [Mahato and Bhunia, 2006] :

— **Interval-numbers addition $\oplus$:**
$A \oplus B = [a^l, a^u] \oplus [b^l, b^u] = [a^l + b^l, a^u + b^u]$

— **Interval-numbers subtraction $\ominus$:**
$A \ominus B = [a^l, a^u] \ominus [b^l, b^u] = [a^l - b^u, a^u - b^l]$

— **Interval-numbers multiplication $\otimes$:**
$A \otimes B = [a^l, a^u] \otimes [b^l, b^u] = [m^l, m^u]$, where $m^l = min(a^l b^l, a^l b^u, a^u b^l, a^u b^u)$ and $m^u = max(a^l b^l, a^l b^u, a^u b^l, a^u b^u)$

— **Interval-numbers scalar multiplication :**
Given any real number $\lambda \in \mathbb{R}$, $\lambda \otimes A = \lambda \otimes [a^l, a^u] = \begin{cases} [\lambda * a^l, \lambda * a^u] \text{ if } \lambda >= 0 \\ [\lambda * a^u, \lambda * a^l] \text{ if } \lambda < 0 \end{cases}$

— **Interval-numbers division $\oslash$:**
$A \oslash B = [a^l, a^u] \oslash [b^l, b^u] = [a^l, a^u] \otimes \left[\frac{1}{b^u}, \frac{1}{b^l}\right]$, provided $0 \notin [b^l, b^u]$

Ranking interval numbers Let $A = \left[a^l, a^u\right]$ and $B = \left[b^l, b^u\right]$ be any two interval numbers that can be classified as three possible overlapping types : **Type I** (non-overlapping interval numbers) as depicted in Figure 3.5a, **Type II** (partially overlapping interval numbers) as seen in Figure 3.5b, or **Type III** (completely overlapping interval numbers) as shown in Figure 3.5c [Karmakar and Bhunia, 2012]. These types are used for ordering and ranking interval numbers of the maximization (minimization) optimization problems along with objective functions modeled using interval numbers as it will be described in the definitions that follows :

Definition 13 *(smaller order relation $<_{\min}$) [Seghir et al., 2019] For interval minimization optimization problems, the order relation $<_{\min}$ that is applied to rank two given interval numbers $A = \left[a^l, a^u\right]$ and $B = \left[b^l, b^u\right]$ is shaped as follows :*

1. *For interval numbers of Type I and Type II, $A <_{\min} B$ if $(a^l < b^l)$ and $(a^u < b^u)$;*
2. *For interval numbers of Type III, $A <_{\min} B$ if :*
   (a) $(a^l - b^l) < (b^u - a^u)$;
   (b) or $\left((a^l - b^l) = (b^u - a^u)\right)$ and $(a^u < b^u)$

Definition 14 *(greater order relation $>_{\max}$) [Seghir et al., 2019] For interval maximization optimization problems, the order relation $>_{\max}$ that is used to rank any two interval numbers $A = \left[a^l, a^u\right]$ and $B = \left[b^l, b^u\right]$ is defined as follows :*

1. *For interval numbers of Type I and Type II, $A >_{\max} B$ if $(a^l > b^l)$ and $(a^u > b^u)$;*
2. *For interval numbers of Type III, $A >_{\max} B$ if :*
   (a) $(a^l - b^l) > (b^u - a^u)$;

(b) or $\big((a^l - b^l) = (b^u - a^u)\big)$ *and* $(a^u < b^u)$

Furthermore, if the two order relations $<_{\min}$ and $>_{\max}$ are not taken as two closed interval numbers A and B, which implies by deduction that $a^l = b^l$ and $a^u = b^u$, and, thus, A and B are considered equivalent interval numbers or have the same rank, denoted as $A \sim B$. Subsequently, we define the greater or equal ($\geq_{\max}$) and the smaller or equal ($\leq_{\min}$) order relations that can be used upon any two interval numbers A and B for the interval maximization (minimization) optimization problems, respectively, as given in the following two definitions :

Definition 15 *(smaller or equal order relation* $\leq_{\min}$*) For interval minimization optimization problems, the order relation* $\leq_{\min}$ *that is used to rank any two interval numbers* $A = \left[a^l, a^u\right]$ *and* $B = \left[b^l, b^u\right]$*, is defined as follows :* $A \leq_{\min} B \Leftrightarrow (A <_{\min})$ *or* $(A \sim B)$

Definition 16 *(greater or equal order relation* $\geq_{\max}$*) For interval maximization optimization problems, the order relation* $\geq_{\max}$ *that is used to rank any two interval numbers* $A = \left[a^l, a^u\right]$ *and* $B = \left[b^l, b^u\right]$*, is defined as follows :* $A \geq_{\max} B \Leftrightarrow (A >_{\max})$ *or* $(A \sim B)$

Theorem 1 *Given any three interval numbers A, B and C, the following proprieties are provided :*

$(\mathscr{P}1)$ $A \leq_{\min} (\geq_{\max})A$*, which is called reflexivity.*

$(\mathscr{P}2)$ *If* $A \leq_{\min} (\geq_{\max})B$ *and* $B \leq_{\min} (\geq_{\max})A$ *then* $A \sim B$*, which is called anti-symmetry.*

$(\mathscr{P}3)$ *If* $A \leq_{\min} (\geq_{\max})B$ *and* $B \leq_{\min} (\geq_{\max})C$ *then* $A \leq_{\min} (\geq_{\max})C$*, which is called transitivity.*

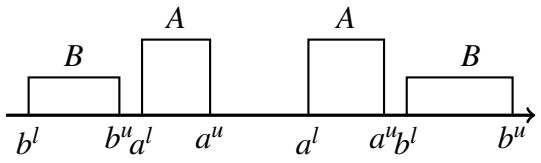
For the proof of the Theorem 1's proprieties, you can refer to Appendix A. Consequently, $\leq_{\min}$ and $\geq_{\max}$ are partial order relations for the sets of interval numbers.

4 Conclusion

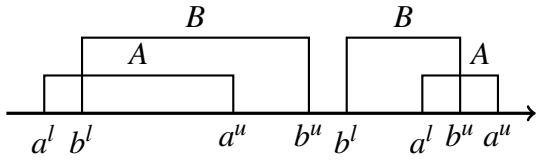
As we conclude this chapter, we will have acquired comprehensive insights into various types of optimization problems and a profound understanding of uncertain numbers, encompassing fuzzy and interval numbers. This acquired knowledge forms a robust foundation that will prove to be immensely valuable in comprehending and engaging with the subsequent chapters' content.

Our exploration into the diverse spectrum of optimization problems will provide us with a nuanced understanding of the different problem types, such as single-objective optimization, multi-objective optimization, continuous optimization problems, and discrete optimization problems. Understanding these distinct problem types equips us with the necessary knowledge to approach varied scenarios in decision-making, problem-solving, and mathematical modeling, where optimization techniques play a crucial role.

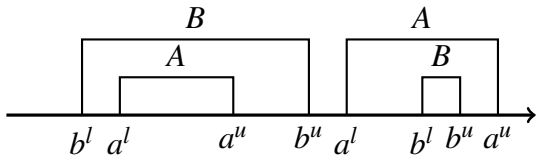
Furthermore, our immersion into uncertain numbers, specifically fuzzy and interval numbers, will empower us to grapple with imprecision, variability, and uncertainty in data representation



(a) Type I : non-overlapping interval numbers



(b) Type II : partially overlapping interval numbers



(c) Type III : completely overlapping interval numbers

FIGURE 3.5 Types of interval-numbers overlapping

and analysis. We will have gained insights into fuzzy arithmetic operations, generalized trapezoidal fuzzy numbers, and interval arithmetic operations. These mathematical concepts will be pivotal in handling and interpreting ambiguous or imprecise information in real-world scenarios, especially in optimization problems where uncertainty prevails.

This accumulated wealth of knowledge and understanding serves as a robust toolkit, enabling us to navigate through complex optimization problems and uncertainty-laden scenarios effectively. Such insights will play a pivotal role in dissecting and comprehending the intricacies of **the composition problem**, enhancing our ability to address real-world challenges and make informed decisions based on uncertain or imprecise data.

Chapter 4

An extended artificial bee colony with local search for solving the Skyline-based web services composition under interval QoS properties

Introduction

In the domain of SOA, virtually any hardware or software resource can be perceived as a web service accessible to end-users. However, as the number of web services offering similar functionalities grows, their distinctions emerge in nonfunctional attributes such as Quality of Service (QoS) parameters like response time, cost, availability, and more. This proliferation of services with functionally equivalent capabilities but varying QoS metrics poses a significant challenge : selecting the most suitable web services among functionally similar ones to compose the optimal Composite Service (CS) that meets both local and global QoS requirements of end users. This dilemma is known as the QoS-aware web service composition (QSC), an intricate problem that has attracted the attention of both industry practitioners and academic researchers Zeng et al. [2004], Alrifai et al. [2012].

QSC represents an NP-hard optimization problem, implying its complexity and the computational challenges associated with solving it. To address this, diverse optimization techniques have been developed, ranging from precise and exact methods to more heuristic and metaheuristic approaches within the domain of web service composition Jatoth et al. [2015].

The core issue underlying QSC arises from the need to identify the most optimal combination of web services, considering their similar functionalities but varied QoS attributes. This task involves determining an optimal selection that fulfills users' QoS expectations while considering both local and global requirements.

This multifaceted problem has garnered significant attention from researchers who are looking to devise efficient algorithms and methodologies to tackle the complexities of the composition of web services. Various approaches have been explored, from exact optimization methods that rigorously handle the QSC problem to heuristic and metaheuristic approaches that offer more pragmatic solutions.

The review conducted in Jatoth et al. [2015] highlights that the majority of the prevailing algorithms for exact web service selection in QSC have been formulated using integer / mixed integer linear programming models Alrifai et al. [2012], Ardagna and Pernici [2007]. These models are typically solved using powerful solvers such as LpSolve [1] and CPLEX [2], which are capable of deriving optimal solutions. However, to utilize these solvers effectively, the QSC problem requires the linearization of its objective functions and constraints. Unfortunately, as the scale of QSC expands, the efficiency of these exact Web service selection algorithms tends to diminish.

In response to this efficiency challenge, evolutionary and bio-inspired algorithms such as the Genetic Algorithm (GA) Canfora et al. [2005], Particle Swarm Optimization (PSO) Liao et al. [2014], Artificial Bee Colony (ABC) Huo et al. [2015], x. wang et al. [2019], and Harris Hawks Optimization (HHO) Li et al. [2021b], among others, have gained attention from QSC researchers. These algorithms have demonstrated the ability to produce near-optimal solutions in reasonable computational times without necessitating the linearization of objective functions or user-defined QoS constraints in the QSC problem.

However, previous studies focusing on QSC have predominantly assumed QoS values as unambiguous, neglecting real-world complexities and inherent uncertainties prevalent in SOA, such as network topology alterations and economic policy fluctuations Razian et al. [2020]. As a result, QoS attributes associated with web services are inherently uncertain and ambiguous in nature.

Recent approaches have emerged to tackle QoS uncertainty in web service composition by representing QoS parameters as interval numbers Jian et al. [2016], Seghir et al. [2019], probabilistic variables Zheng et al. [2016], or fuzzy numbers Xu et al. [2018], Seghir [2021]. These approaches consider all available web services as potential candidates for constructing the final CS. However, certain web services may be dominated by functionally equivalent alternatives, making them unsuitable for inclusion in the final solution. To address this issue, some studies Alrifai et al. [2010], Wang et al. [2013], Ying and Jiande [2020] have employed the Skyline operator Borzsony et al. [2001] to prune web services that cannot contribute to the final solutions, thereby reducing the search space in QSC. It's crucial to note that in these Skyline-based QSC studies, QoS parameters were assumed to possess precise and exact values.

The motivation behind this study lies in devising an efficient approach to tackle QoS uncertainty in web service composition, particularly within a reduced search space. The primary contributions of this chapter can be summarized as follows :

1. http ://lpsolve.sourceforge.net/5.5/
2. https ://www.ibm.com/analytics/cplex-optimizer

1. First, an interval constrained single-objective optimization model termed IQSC for QoS uncertainty-aware web service composition, where QoS parameters are represented as interval numbers, is presented ;
2. Then, a proposal of an approach comprising two components is conducted : **(1) Utilization of the Skyline operator** to drastically reduce the search space in IQSC, enabling quick discovery of high-quality CS solutions. **(2) Introduction of an extended version of the basic ABC algorithm**, named EABC, augmented with an effective local search method to solve IQSC efficiently within a reduced search space.
3. Finally, the demonstration of the efficiency and performance of the proposed approach was illustrated through comparative experiments against existing methods Wang et al. [2013], Huo et al. [2015], Li et al. [2021b] on a modified interval-based QWS dataset Al-Masri and Mahmoud [2008].

The subsequent sections of this chapter are structured as follows : Section 1 delves into a comprehensive review of relevant works in the domain, categorizing them based on the nature and precision of QoS values. Section 2 presents the mathematical formulation of QoS uncertainty-aware web service composition under interval QoS properties (IQSC). Section 3 details the proposed approach, encompassing the Skyline operator and the EABC algorithm, designed to solve the formulated IQSC efficiently. Section 4 discusses the outcomes of comparative experiments, showcasing the efficiency of the proposed approach. Lastly, Section 6 concludes and offers insights into future research directions.

1 Related works and literature review

In the existing body of literature, the QSC problem has been classified into three primary categories within web service selection methodologies : **(1)** Exact **(2)** Heuristic and **(2)** Meta-heuristic optimization approaches, specifically those rooted in evolutionary and bio-inspired algorithms, as discussed by Jatoth et al. [2015]. These categories serve as frameworks for devising strategies to select optimal WSs based on QoS criteria. To enhance the efficacy of QSC methodologies, researchers have explored the integration of the Skyline operator, a concept introduced by Borzsony et al. [2001]. This operator is leveraged by certain studies to streamline the process of web service selection, consequently reducing the time involved. Noteworthy works in this regard include the contributions of Alrifai et al. [2010], Wang et al. [2013], Ying and Jiande [2020]. Furthermore, acknowledging the unpredictable nature of SOA environments due to factors such as network topological changes and economic policies, as highlighted by Razian et al. [2020], several QSC studies have extended their focus to address QoS parameters characterized by ambiguous values. This is evident in research efforts by Jian et al. [2016], Zheng et al. [2016], Xu et al. [2018], where QoS parameters are considered with uncertain values.

This section provides an overview of state-of-the-art studies in the field, classifying them based on their approaches to addressing the QSC problem. The initial segment delves into a review of

solution approaches dedicated to resolving the QSC problem when confronted with precise and exact QoS values. Subsequently, attention is directed towards recently proposed solution methods designed to tackle the QSC problem in scenarios characterized by uncertain QoS parameters, and a brief exposition of these approaches is presented.

1.1 Solution approaches based on precise QoS values

Numerous research endeavors have focused on effectively addressing the QSC problem by treating the QoS values as precise and exact quantities, as highlighted in studies such as those by Jatoth et al. [2015] and She et al. [2019].

These studies contribute to a nuanced understanding of solution approaches designed for tackling the QSC problem under the assumption of precise QoS values. These approaches can be broadly classified into three main categories : **(1) Exact (non-heuristic) approaches, (2) Heuristic methods, and (3) Meta-heuristic and bio-inspired approaches**.

1.1.1 Exact approaches

In these methodologies, the QSC problem has been conceptualized as a mixed-integer linear programming (MILP) model, ensuring a reliable solution through the utilization of mixed-integer linear programming solvers like CPLEX Ardagna and Pernici [2007] and LpSolve Alrifai et al. [2012].

In the work presented by Zeng et al. [2004], the QSC problem has been cast as an Integer Linear Programming (ILP) optimization model. Two distinct approaches have been integrated into addressing this model : **(1)Local Optimization Approach** which was characterized by its computational efficiency, ensuring a swift computation time. However, it comes with the drawback of not guaranteeing a feasible optimal solution when adhering to global QoS user-constraints. **(2) Global Optimization Method** : In contrast to its predecessor, it addresses the limitations of the local optimization approach as It aims to find an optimal feasible composite service that satisfies global QoS user-constraints. Despite overcoming the issues of its predecessor, this method entails a notable consumption of processing time in the pursuit of identifying the optimal solution.

In the study conducted by Ardagna and Pernici [2007], the QSC problem was cast as a mixed-integer linear programming (MILP) optimization model, and the researchers employed the CPLEX solver for its resolution. Additionally, due to the initial absence of a feasible solution for the solved QSC problem, the researchers explored negotiation techniques to secure the necessary feasible solutions.

In the work presented by Alrifai et al. [2012], a hybrid approach was introduced, amalgamating both local and global optimization methods to address the QSC problem. Initially, the global optimization method was executed to delineate the optimal decomposition of global QoS user-constraints into local quality levels. This was achieved by solving a relatively smaller Mixed Integer Linear Programming (MILP) model compared to those in previous works such as Zeng et al. [2004], Ardagna

and Pernici [2007]. Subsequently, a distributed local optimization method utilized the derived local thresholds to meticulously select the most suitable atomic WSs from their respective service providers, ultimately constructing the best feasible composite service.

In the work conducted by Ghobaei-Arani and Souri [2019], the QSC problem within geographically distributed environments was systematically addressed by formulating it as an Integer Linear Programming (ILP) model. In response to this modeling framework, the researchers introduced a novel methodology known as the linear programming web service composition (LP-WSC) approach. This approach was meticulously designed to provide a systematic and structured solution for the intricacies posed by the QSC problem within the context of geographically distributed environments.

Despite the effectiveness of exact methods in providing high-quality solutions for addressing the QSC problem, there are inherent challenges associated with the functional linearization of the totality of optimized objectives and QoS user-constraints within LIP or MILP optimization models. This necessitates a thorough linearization process to ensure the compatibility and feasibility of the solutions obtained through these methods. Moreover, when employing solvers such as CPLEX and LpSolve, a noteworthy limitation arises in terms of scalability. These solvers may exhibit suboptimal performance when confronted with constrained QSC problems of considerably large scales. The ability to efficiently handle and resolve QSC problems characterized by extensive scales remains a significant concern, urging researchers and practitioners to explore alternative methodologies or enhancements to existing solvers to address scalability challenges effectively.

In the pursuit of enhancing the efficiency QSC optimization approaches, a notable strategy known as the Skyline operator Borzsony et al. [2001] has been employed to strategically narrow down the search space of the QSC problem. This application of the Skyline operator plays a pivotal role in reducing the computational time cost associated with these approaches.

Pioneering this initiative, the authors in Alrifai et al. [2010] took a significant step by integrating the Skyline operator into the QSC problem. In their approach, a novel service dominance criterion, based on QoS attributes of WSs, was introduced among the set of WSs. This criterion facilitated the pruning of dominated web services, thereby effectively reducing the overall search space. The QSC problem, within the context of this study, was formulated as an Integer Linear Programming (ILP) model. Leveraging the Skyline operator and the service dominance criterion, the authors demonstrated its ability to solve the problem more efficiently. Notably, the utilization of the existing LpSolve solver in the reduced search space further contributed to the optimization of the computational process. This innovative approach stands as an illustrative example of how the integration of the Skyline operator can lead to improved efficiency in solving the QSC problem by strategically managing the exploration of potential service compositions.

In the work presented by Ying and Jiande [2020], the authors employed the mathematical programming language (AMPL) Gay [2015] to articulate the QSC problem as a nonlinear integer programming model. The solution to this formulated model was achieved using the established

Bonmin solver [3]. To enhance the efficiency of the QSC optimization process, the study incorporated the utilization of the Skyline operator, strategically applied to curtail the expansive search space associated with the QSC problem. This integration of the Skyline operator serves as a valuable mechanism for refining and streamlining the exploration of potential service compositions within the nonlinear integer programming framework.

1.1.2 Heuristic approaches

Within the realm of heuristic approaches, examples of widely employed methods include the hill-climbing algorithm Klein et al. [2011] and the A^* algorithm Neiat et al. [2014]. These heuristic strategies have been applied effectively to tackle the QSC problem when dealing with precise QoS values.

In the study presented by Luo et al. [2011], the authors introduced a heuristic-enhanced cross-entropy algorithm (HCE) designed for addressing the QSC problem. This heuristic approach draws inspiration from the hill-climbing algorithm, which was previously developed in Klein et al. [2011].

In the study conducted by Neiat et al. [2014], the A^* algorithm was implemented to address the spatio-temporal composition of sensor cloud services. In this context, two novel QoS attributes including freshness and certainty were introduced to enhance the compositional considerations. Furthermore, an additional heuristic algorithm aimed at the efficient selection and composition of trustworthy services was put forth in a separate work by Li et al. [2014].

Heuristic approaches hold a distinct advantage over exact algorithms, particularly in their ability to swiftly identify good near-optimal or optimal solutions within a reasonable time-frame. This efficiency is particularly valuable in addressing complex optimization problems where exact algorithms might incur significant computational costs. However, a notable drawback warrants attention : heuristic approaches are crafted based on "experience" and are often tailored for specific optimization problems. This specialization renders them susceptible to converging to a local optimum, potentially limiting their applicability in exploring the broader solution space.

Recognizing this limitation, researchers have turned their attention to metaheuristic and bio-inspired algorithms Jatoth et al. [2015], She et al. [2019] as a promising avenue to mitigate the challenges associated with heuristic approaches. Metaheuristic algorithms, inspired by natural processes or optimization techniques, offer a more versatile and adaptive approach to optimization problems. By harnessing concepts from evolutionary processes, swarm intelligence, and other bio-inspired mechanisms, these algorithms can explore diverse regions of the solution space, potentially avoiding the pitfalls of local optima encountered by traditional heuristic methods. This shift towards metaheuristic and bio-inspired algorithms reflects a strategic response to enhance the robustness and global exploration capabilities of optimization techniques, ensuring more effective solutions to complex problems such as the QSC problem.

3. https ://github.com/coin-or/Bonmin

1.1.3 Metaheuristic and bio-inspired approaches

Some of the frequently used Metaheuristic and bio-inspired algorithms in addressing the QSC problem, are listed as follows : genetic algorithm (GA) Wu et al. [2014a], particle swarm optimization (PSO) Wang et al. [2013], artificial bee colony (ABC) Huo et al. [2015], ant colony optimization (ACO) Wu and Zhu [2013], teaching learning based optimization (TLBO) Deng et al. [2014a], Khanouche et al. [2020b], and so on.

In the quest for solutions to scalable optimization problems, meta-heuristic methods emerge as efficient alternatives, capable of achieving near-optimal solutions with significantly reduced computation time when compared to the Linear Integer Programming (ILP)/ Mixed Integer Linear Programming (MILP) approaches. While ILP/MILP approaches may yield optimal solutions, they often demand excessive processing time.

An additional advantage of metaheuristic methods lies in their flexibility to accommodate non-linear function formulations for optimized objectives and constraint equations within the optimization problems. This adaptability contributes to their versatility in handling a broad spectrum of optimization challenges.

Metaheuristic approaches represent a higher-level of heuristic algorithms, intentionally designed to autonomously address general and complex optimization problems. Consequently, over the past few decades, a noteworthy trend has emerged wherein evolutionary and bio-inspired optimization algorithms have been repurposed and tailored to effectively tackle the specific intricacies presented by the QSC problem. This shift underscores the recognition of meta-heuristic methods as valuable tools for addressing scalability, computational efficiency, and complexity challenges in the domain of optimization.

The pioneers in leveraging genetic algorithms for addressing the QSC problem were the authors of Canfora et al. [2005]. To guide the evolution process of the proposed genetic algorithm (GA) towards satisfying constraints, the authors experimented with two distinct fitness functions, each incorporating penalties of either static or dynamic nature. The comparative analysis of the results revealed that the near-optimal solutions achieved by the static and dynamic fitness functions did not exhibit notably distinguishable differences.

In recent times, several optimization approaches based on genetic algorithms (GAs) have been employed to address the challenges posed by the QSC problem. Notably, in works such as Wu et al. [2014a], Jin et al. [2017], authors have delved into the consideration of inter-dependencies and business correlations among elementary WSs when formulating QSC problems, particularly in the context of cloud manufacturing. These formulated problems have then been effectively solved through the application of genetic algorithms.

In the study outlined in Wu and Zhu [2013], the incorporation of transactional properties among atomic WSs led to the formulation of the QSC problem as a directed acyclic graph. Subsequently, this formulated problem was addressed through the application of an ACO algorithm.

The application of the PSO algorithm for addressing the QSC problem in manufacturing grid systems is exemplified in Tao et al. [2008]. In an effort to enhance the computational efficiency of the PSO-based approach introduced in Wang et al. [2013], the researchers incorporated the skyline operator. This addition served to mitigate the computational burden by reducing the search space of the solved QSC problem, effectively pruning redundant candidate WSs that were dominated by others.

The authors in Huo et al. [2015] introduced a discrete variant of the ABC algorithm tailored for addressing the QSC problem within cloud environments.

In pursuit of effective near-optimal solutions for addressing the QSC problem, researchers have introduced several recent optimization methods based on the ABC algorithm, as evidenced by works such as x. wang et al. [2019], Dahan et al. [2017].

In addressing the QSC problem within mobile and IoT environments, the authors in Deng et al. [2014a], Khanouche et al. [2020b] applied the TLBO algorithm.

In reference Fathollahi-Fard et al. [2020], the authors introduced a novel nature-inspired metaheuristic algorithm known as the red deer algorithm (RDA), drawing inspiration from the behavior of Scottish red deer. The efficacy and efficiency of the developed RDA were showcased through its application to various real-world engineering optimization problems akin to QSC. These issues included the single-machine scheduling problem, traveling salesman problem, fixed-charge transportation problem, and vehicle routing problem.

Several multi-objective optimization approaches have been introduced to address the QSC problem. Unlike the conventional metaheuristic algorithms mentioned earlier, which typically yield a single compromise solution, multi-objective optimization approaches offer a distinct advantage. These approaches guarantee the generation of multiple non-dominated near-optimal or optimal solutions, commonly known as Pareto optimal solutions. Consequently, users have the flexibility to choose a solution that aligns with their specific preferences or requirements from the set of non-dominated solutions. For an in-depth exploration of the various multi-objective optimization approaches proposed for the QSC problem, interested readers are encouraged to delve into comprehensive literature reviews such as those presented in Cremene et al. [2016], Ramírez et al. [2017]. These reviews offer a detailed analysis and comparison of the most prominent multi-objective optimization methodologies devised to tackle the challenges inherent in QSC, providing valuable insights for researchers and practitioners in the field.

It is worth noting that the previously mentioned approaches primarily operate under the assumption that QoS values are certain and precise. However, in practical applications of the QSC problem, QoS values often exhibit inexactness or ambiguity due to diverse factors such as network topology changes, variations in workstation load, system congestion, and economic policies, as discussed in Razian et al. [2020]. These uncertainties inherent in QoS values emphasize the need for a more realistic modeling approach. In response to this practical consideration, some researchers have chosen to formulate the QSC problem as a non-deterministic optimization model. This involves representing QoS values using interval numbers Jian et al. [2016], Seghir et al. [2019], fuzzy numbers Xu et al. [2018], Seghir [2021], or probabilistic models Zheng et al. [2016]. These alternative formulations account for the inherent

uncertainty in QoS values, enabling a more accurate representation of the real-world complexities associated with dynamic and unpredictable service-oriented environments.

1.2 Solution approaches based on ambiguous QoS values

Addressing the inherent ambiguity and vagueness in QoS parameters' values necessitates the utilization of three (03) representation models capable of expressing and representing this uncertainty : **(1)** Random variables, defined by probability distribution functions, **(2)** Interval-numbers, can bound the ambiguity within their lowers (lefts) and uppers (rights) limits on the real line $\mathbb{R}$, or **(3)** Fuzzy numbers that can be expressed using membership functions.

In the following, a concise overview of contemporary approaches that employ probabilistic, interval-number-based, and fuzzy-number-based methodologies for addressing the QSC problem, specifically in the context of ambiguous QoS parameters, will be provided.

1.2.1 Probabilistic approaches

Recently, some works have been devoted to address the QSC problem under uncertain QoS parameters, where the vagueness and ambiguity of the QoS values are expressed with random variables Chattopadhyay and Banerjee [2016], Kim et al. [2016] that are defined through probability distribution functions Zheng et al. [2016].

In the work detailed in Chattopadhyay and Banerjee [2016], the authors introduced a stochastic Integer Linear Programming (ILP) model for the QSC problem. This model served as the foundation for the development of a probabilistic heuristic algorithm, offering potential advantages, particularly in terms of scalability, for solving the QSC problem.

In the study outlined in Kim et al. [2016], a robust approach for web service selection in the presence of outliers was devised to address a probabilistic QoS model. This model was formulated based on historical executions of requested web services. The proposed method focuses on selecting a candidate service for each task within a composite service. This selection is determined by considering the service's ranking score and its satisfaction of a local threshold, derived from the decomposition of global constraints.

The probabilistic approaches mentioned earlier necessitate prior knowledge of the probability distribution functions associated with ambiguous QoS attributes Zheng et al. [2016]. However, accurately defining the specific type of QoS probability distribution poses a challenge for researchers, as it requires reliable historical QoS values, which may not always be readily available or easily defined with precision.

1.2.2 Interval-numbers-based approaches

In the study presented in Jian et al. [2016], the authors employed interval numbers to encapsulate the uncertainty associated with QoS attributes. They introduced a novel interval-based model to address

the uncertain QSC problem. The authors introduced a unique interval-based fuzzy ranking method to rank the interval numbers. This method was subsequently integrated into the PROMETHEE decision-making approach for resolving uncertain QSC problems with smaller dimensions. Additionally, for larger-scale QSC problems, the authors developed a genetic algorithm incorporating the extended PROMETHEE method (GAP).

In the study outlined in Niu et al. [2019], the authors conceptualized the uncertain QSC problem as an Interval Number-based Multi-Objective Optimization Problem with Global Constraints (IMOPs). To address this formulation, they devised a non-deterministic multi-objective optimization-based decomposition algorithm (NDmoea/d) for its resolution.

In the work presented in Seghir et al. [2019], the authors approached the uncertain QSC problem by formulating it as an Interval Number-based Multi-Objective Optimization Problem with Global Constraints (IMOPs). To address this model, they introduced the interval-based multi-objective artificial bee colony (IM_ABC) algorithm. This algorithm incorporates a novel interval-based feasibility technique designed to handle interval constraints effectively. Additionally, it integrates a new interval-distance metric, an extension of the original crowding distance of NSGA-II. This extension ensures the IM_ABC's non-dominated solutions maintain a diverse set, thereby enhancing the algorithm's control over solution diversity.

While interval approaches have proven effective in addressing interval-based models for uncertain QSC problems, they face limitations in accurately describing ambiguous Quality of Service (QoS) values. Consider a dynamic web service where the response time attribute varies in the interval $[1,4]$ seconds, but it is more likely to fall within the interval $[2,3]$ seconds. In such cases, the interval-number model may inadequately capture the uncertainty of this QoS attribute. Consequently, the utilization of fuzzy numbers becomes crucial to precisely describe and account for the ambiguity associated with such QoS values.

1.2.3 Fuzzy-numbers-based approaches

In the study outlined in Zhang et al. [2017], the authors addressed the ambiguity associated with four fundamental QoS properties : price, execution time, reliability, and availability. To represent this ambiguity, they employed triangular fuzzy numbers (TFNs). These TFNs were then utilized to formulate the QSC problem within dynamic manufacturing environments as a fuzzy optimization problem. The authors proposed a solution approach leveraging an extended flower pollination algorithm (EFPA) to effectively solve the formulated fuzzy optimization problem.

In the work presented in Zhang et al. [2019c], the authors employed the Triangular Fuzzy Number (TFN) model to articulate the impreciseness inherent in the uncertain QoS attributes relevant to both vertical and horizontal QSC formulations within fuzzy manufacturing environments. Additionally, they devised an enhanced flower pollination algorithm to effectively address and solve the TFN-based optimization model associated with the QSC problem.

In the study presented in Xu et al. [2018], the authors employed both Triangular Fuzzy Number (TFN) and crisp-valued models to represent uncertainty in QoS properties and non-ambiguous ones, respectively. Leveraging these models, the authors introduced a fuzzy optimization model for the QSC problem. To solve this model, they developed a triangular fuzzy genetic algorithm (TGA).

In the study outlined in Zhang et al. [2018], the authors utilized the Intuitionistic Fuzzy Entropy Weight (IFEW) method to accurately determine preference weights for QoS properties. They applied an extended Biogeography-Based Optimization (BBO) algorithm to address the challenges of manufacturing service supply chain optimization. Within this algorithm, the Triangular Fuzzy Number (TFN) model was employed to effectively represent the uncertainty associated with the considered QoS attributes.

In the study outlined in Feng and Kong [2015], the authors introduced a Fuzzy Multi-Objective Genetic Algorithm (FMOGA) as a solution to the QSC problem. In this approach, both the QoS attributes and the users' preference weights were formulated based on the TFN model.

In the study presented in Seghir and Khababa [2021b], the authors introduced a novel fuzzy optimization approach tailored for the QSC problem under uncertain Qos parameters. These laters were represented using Generalized Trapezoidal Fuzzy Numbers (GTrFN). Furthermore, the QSC problem was formulated as a fuzzy optimization (FQSC) problem, modeling QoS properties with GTrFNs. To address the FQSC model, the authors developed a new Fuzzy Teaching Learning-Based Optimization (FTLBO) algorithm. This later incorporates a local search method and an elitism search operator to enhance exploitation ability and expedite the convergence of FTLBO in discovering optimal or suboptimal composite service solutions.

Table 4.1 provides a summary of the studies mentioned, categorizing them based on the representation model for QoS parameters, the mathematical formulation of the QSC problem, and the respective solution approaches. Unlike most referenced papers, our representation of QoS uncertainty adopts the interval number model. Consequently, we formulate the QSC problem as a single-objective optimization with interval constraints. This formulation is addressed using an extended artificial bee colony algorithm. Additionally, an interval-valued version of the existing crisp Skyline operator is applied to narrow the search space of the Interval QSC (IQSC), enhancing the efficiency of the proposed algorithm.

TABLE 4.1 Literature review of th QoS-aware service composition problem for static and dynamic environments

Study	Year	QoS representation model	QSC mathematical formulation	Proposed QSC solution approach
Zeng et al. [2004]	2004	Real numbers	ILP	ILP-based approach using the CPLEX solver
Ardagna and Pernici [2007]	2007	Real numbers	MILP	ILP-based approach using the CPLEX solver
Alrifai et al. [2012]	2012	Real numbers	MILP	ILP-based approach using the LpSolve solver
Wang et al. [2013]	2013	Real numbers	Single objective optimization	PSO with Skyline operator
Canfora et al. [2005]	2005	Real numbers	Constrained single objective optimization	GA-based approach
Xu et al. [2018]	2018	Fuzzy numbers	Fuzzy single objective optimization	GA-based approach
Seghir et al. [2019]	2019	Interval numbers	Interval constrained multiple objectives optimization	Interval multi-objectives artificial bee colony (IM_ABC)
Huo et al. [2015]	2015	Real numbers	Single objective optimization	discrete gbest-guided artificial bee colony (DGABC)

x. wang et al. [2019]	2019	Real numbers	Single objective optimization	Discrete ABC-based approaches
Jian et al. [2016]	2016	Interval numbers	Interval single objective optimization	GA with PROMETHEE method
Seghir and Khababa [2021b]	2021	Fuzzy numbers	Fuzzy constrained single objectives optimization	FTLBO with local search method and elitism operator
Khababa et al. [2022]	2022	Interval numbers	Interval constrained single objective optimization	EABC with Skyline operator
Wu et al. [2014a]	2014	Real numbers	Constrained single objective optimization	Genetic algorithm
Ghobaei-Arani and Souri [2019]	2019	Real numbers	Single objective optimization	ILP-based approach
Klein et al. [2011]	2011	Real numbers	Single objective optimization	LP-based approach with Hill Climbing algorithm
Zhang et al. [2017]	2017	Triangular fuzzy numbers	Triangular constrained single objective optimization	EFPA
Zhang et al. [2019c]	2019	Triangular fuzzy numbers	Triangular constrained single objective optimization	Enhanced EFPA

Zhang et al. [2018]	2018	Triangular fuzzy numbers	Triangular constrained single objective optimization	EBBO
Niu et al. [2019]	2019	Interval numbers	Interval constrained multi-objective optimization	(NDmoea/d) algorithm
Wu and Zhu [2013]	2013	Real numbers	Single objective optimization	GA
Ji et al. [2017]	2017	Real numbers	Single objective optimization	Improved GA
Neiat et al. [2014]	2014	Real numbers	Single objective optimization	A* Algorithm with spatio-temporal composition
Li et al. [2014]	2014	Real numbers	Single objective optimization	A* Algorithm with trustworthiness
Dahan et al. [2017]	2017	Real numbers	Single objective optimization	Discrete ABC-based approaches
Deng et al. [2014a]	2014	Real numbers	Single objective optimization	TLBO
Tao et al. [2008]	2008	Real numbers	Single objective optimization	PSO in manufacturing grid system
Cremene et al. [2016]	2016	Real numbers	Multi-objective optimization	EA
Ramírez et al. [2017]	2017	Real numbers	Multi-objective optimization	EA

Feng and Kong [2015]	2015	Triangular Fuzzy numbers	Fuzzy constrained multi-objective optimization	GA
Jian et al. [2016]	2016	Interval numbers	Interval constrained single objective optimization	GAP
Chattopadhyay and Banerjee [2016]	2016	Probabilistic distribution	Probabilistic constrained single objective optimization	stochastic ILP
Kim et al. [2016]	2016	Probabilistic distribution	Probabilistic constrained single objective optimization	outlier-robust approach
Khanouche et al. [2020b]	2020	Real numbers	Single objective optimization	Improved TLBO
Fathollahi-Fard et al. [2020]	2020	Real numbers	Single objective optimization	RDA

2 Interval model of the QoS uncertainty-aware web service composition problem

To understand the issue in hand, we shall start by introducing the general concepts which will provide the needed information for the rest of this study :

— Consider a set of atomic web services, with each web service, denoted as *ws*, characterized by two distinct types of properties : functional and non-functional. The functional parameters, encompassing input and output attributes, serve to represent the supported functionality of a given *ws*. In contrast, the non-functional properties, including QoS attributes such as response time, availability, reputation, price, and more, reflect the quality parameters associated with a *ws*.

— Each QoS attribute, represented by q_t where $t = 1,2,\ldots,r$ and r is the count of the considered QoS parameters, is viewed as either a positive or a negative parameter. For positive q_t values, a higher value signifies better performance, while for negative q_t values, a lower value indicates inferior performance. In essence, a web service with larger positive q_t values is considered better, whereas a web service with smaller negative q_t values is also deemed worse.

— The set of positive q_t values is denoted as QoS^+, while the negative values are represented by QoS^-. For instance, attributes such as reputation and availability are categorized in the QoS^+ list, whereas parameters like price and response time fall under the QoS^- list.

— A class of web services, denoted as $S = ws_1, ws_2, \ldots, ws_m$, represents a collection of m atomic web services sharing similar functionalities while exhibiting distinct values in their respective q_t attributes.

— As illustrated in Fig. 4.1, providing a graphical representation of the QSC problem, consider an abstract composite service denoted as $ACS = S_1, S_2, \ldots, S_n$. This abstract composite service represents the n required web service classes to fulfill a user request.

— A set of k user's global Quality of Service (QoS) requirements, indicated as Cst_{q_t} with $t = 1,2,\ldots,k$ and $k \leq r$.

A concrete composite service, denoted as *CS* and representing the response to a user request, is constructed by choosing a unique web service ws_i^j from each web service class $S_i = ws_i^1, ws_i^2, \ldots, ws_i^{m_i}$, where $j \leq m_i$. The global QoS values of *CS*, determined by aggregating the QoS values of its atomic web services *ws*s, must align with the specified Cst_{q_t} requirements (i,e. a feasible concrete composition is characterized by its adherence to the global QoS constraints of the end-user. This composition is constructed by choosing a distinct *ws* and aggregating QoS values for both benefit and cost aspects in the intended feasible solution must align with the global QoS user-constraints).

In this study, as depicted in Fig 4.2, we have considered four fundamental connection structures : sequential, parallel, branch, and loop to compose the atomic web services of composite services (*CS*s). Additionally, owing to uncontrollable factors in SOAs, such as changes in network architectures and

economic policies, the QoS values associated with elementary web services are inherently uncertain Razian et al. [2020]. Recognizing the efficacy of the interval number as a versatile and straightforward representation model for expressing uncertain QoS values Jian et al. [2016], Seghir et al. [2019], and considering the four most commonly used QoS parameters in solving the QSC problem Alrifai et al. [2012], including two positive QoS attributes—*availability* (q_1) and *throughput* (q_2), and two negative ones—*response time* (q_3) and *price* (q_4), we formulate the interval constrained single-objective optimization model for QSC, denoted as IQSC, as given in the following subsection.

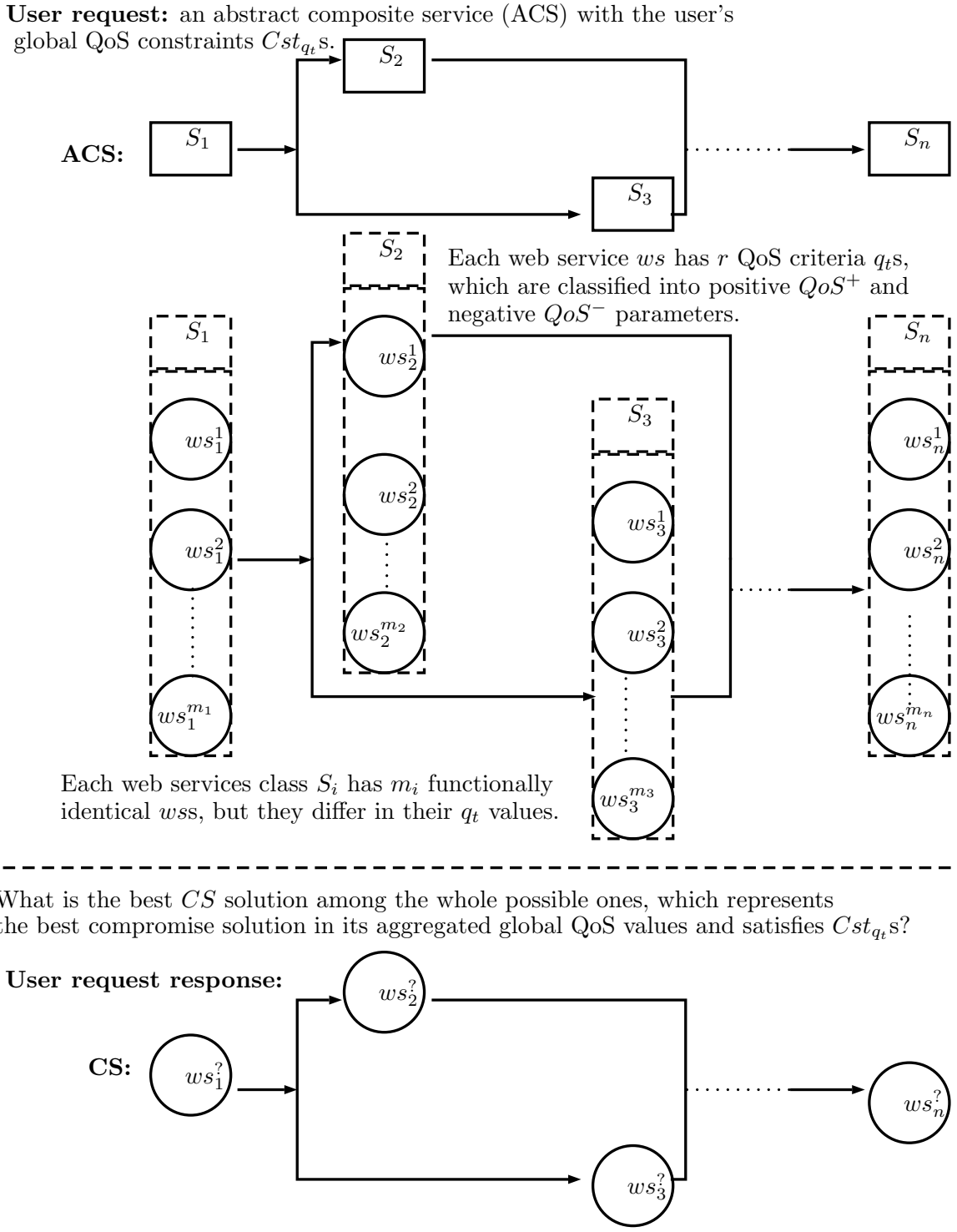
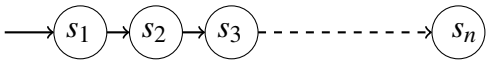
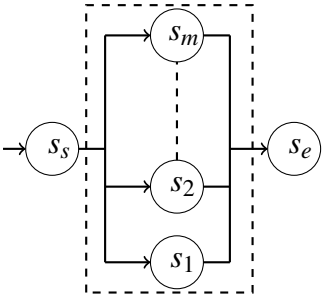


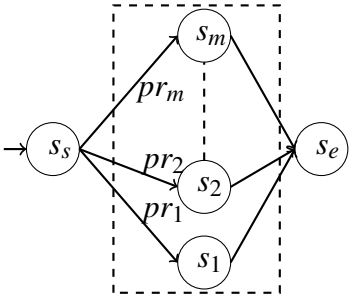
FIGURE 4.1 Graphic depiction of the QoS-aware web service composition problem



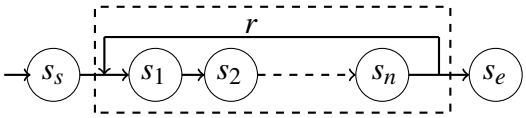
(a) Sequential composition of n web services



(b) Parallel composition of m web services



(c) Conditional composition of m web services with their pr_is probabilities, $\forall i = 1,2\ldots,m : pr_i \in [0,1]$ and $\sum_{i=1}^{m} pr_i = 1$



(d) Loop composition of n web services with r calls

FIGURE 4.2 The common control structures for composing elementary web services

TABLE 4.2 Interval QoS aggregation formulas for evaluating the global QoS interval-valued of concrete composite services *CS*s

QoS	n sequential web services ws_i	m parallel web services ws_i	branch[a] of m web services ws_is with their pr_is probabilities	call a web service ws with p times
q_1	$\prod_{i=1}^{n}\left[ws_{i,q_1}^{l}, ws_{i,q_1}^{u}\right]$	$\prod_{i=1}^{m}\left[ws_{i,q_1}^{l}, ws_{i,q_1}^{u}\right]$	$\mathrm{Min}_{i=1}^{m}\left\{\left[ws_{i,q_1}^{l}, ws_{i,q_1}^{u}\right]\right\}$	$\prod_{i=p}^{n}\left[ws_{i,q_1}^{l}, ws_{i,q_1}^{u}\right]$
q_2	$\mathrm{Min}_{i=1}^{n}\left\{\left[ws_{i,q_2}^{l}, ws_{i,q_2}^{u}\right]\right\}$	$\mathrm{Min}_{i=1}^{m}\left\{\left[ws_{i,q_2}^{l}, ws_{i,q_2}^{u}\right]\right\}$	$\mathrm{Min}_{i=1}^{m}\left\{\left[ws_{i,q_2}^{l}, ws_{i,q_2}^{u}\right]\right\}$	$\left[ws_{i,q_2}^{l}, ws_{i,q_2}^{u}\right]$
q_3	$\sum_{i=1}^{n}\left[ws_{i,q_3}^{l}, ws_{i,q_3}^{u}\right]$	$\mathrm{Max}_{i=1}^{m}\left\{\left[ws_{i,q_3}^{l}, ws_{i,q_3}^{u}\right]\right\}$	$\mathrm{Max}_{i=1}^{m}\left\{\left[ws_{i,q_3}^{l}, ws_{i,q_3}^{u}\right]\right\}$	$p*\left[ws_{i,q_3}^{l}, ws_{i,q_3}^{u}\right]$
q_4	$\sum_{i=1}^{n}\left[ws_{i,q_4}^{l}, ws_{i,q_4}^{u}\right]$	$\sum_{i=1}^{m}\left[ws_{i,q_4}^{l}, ws_{i,q_4}^{u}\right]$	$\mathrm{Max}_{i=1}^{m}\left\{\left[ws_{i,q_4}^{l}, ws_{i,q_4}^{u}\right]\right\}$	$p*\left[ws_{i,q_4}^{l}, ws_{i,q_4}^{u}\right]$

[a]In the branch connection structure, according to the pr_is of ws_is, only one web service among them is executed.
q_1 :Availability, q_2 :Throughput, q_3 :Response time and q_4 :Price

2.1 Problem formulation

Identify the best *CS* solution among all the possible ones [4], which maximizes

$$\text{Maximize } IU(CS) = \sum_{t=1}^{r} w_{q_t} \otimes \overline{CS_{q_t}} \tag{4.1}$$

Subject to k global QoS constraints

$$\forall t = 1 \ldots k \begin{cases} CS_{q_t} \leq_{\min} Cst_{q_t}, \text{ if } q_t \in QoS^{-} \\ CS_{q_t} \geq_{\max} Cst_{q_t}, \text{ if } q_t \in QoS^{+} \end{cases} \tag{4.2}$$

Where :

— CS_{q_t} is the global QoS interval-valued of the *CS* solution in the q_t attribute that can be evaluated, as seen in Table 4.2, using the interval arithmetic operations given in Definitions 3 and 14 as seen in the previous chapter, the four basic connection structures as depicted in Fig. 4.2

— the interval-valued $ws_{i,q_t} = \left[ws^{l}_{i,q_t}, ws^{u}_{i,q_t}\right]$ of each atomic web service ws_i of *CS* in the q_t attribute.

— $IU(CS)$ is the interval utility function of *CS* that maps the CS_{q_t}s values into a single interval-valued.

This function adopts the well-known *Simple Additive Weighting* (SAW) method Rao [2007] through scaling the interval values CS_{q_t}s into their normalized interval ones $\overline{CS_{q_t}}$s.

Then afterwards, the normalized interval values are weighted and summed-up using the weights w_{q_t}s with $w_{q_t} \in [0,1]$ and $\sum_{t=1}^{r} w_{q_t} = 1$, which represent the importance and priority of each q_t by the user.

The normalized interval-valued $\overline{CS_{q_t}}$ of its related original one $CS_{q_t} = \left[CS^{l}_{q_t}, CS^{u}_{q_t}\right]$ is calculated according to the type of the used q_t criterion, i.e., positive or negative parameter, as given in the following interval positive and negative normalization Equations 4.3 and 4.4, receptively.

— Interval positive normalization

$$\forall q_t \in QoS^{+}, \overline{CS_{q_t}} = \begin{cases} \left[\frac{CS^{l}_{q_t} - min^{l}_{q_t}}{max^{u}_{q_t} - min^{l}_{q_t}}, \frac{CS^{u}_{q_t} - min^{l}_{q_t}}{max^{u}_{q_t} - min^{l}_{q_t}}\right] & \text{if } max^{u}_{q_t} \neq min^{l}_{q_t} \\ [1,1] & \text{if } max^{u}_{q_t} = min^{l}_{q_t} \end{cases} \tag{4.3}$$

— Interval negative normalization

$$\forall q_t \in QoS^{-}, \overline{CS_{q_t}} = \begin{cases} \left[\frac{max^{u}_{q_t} - CS^{u}_{q_t}}{max^{u}_{q_t} - min^{l}_{q_t}}, \frac{max^{u}_{q_t} - CS^{l}_{q_t}}{max^{u}_{q_t} - min^{l}_{q_t}}\right] & \text{if } max^{u}_{q_t} \neq min^{l}_{q_t} \\ [1,1] & \text{if } max^{u}_{q_t} = min^{l}_{q_t} \end{cases} \tag{4.4}$$

4. For an *ACS* of n web services classes, where each one has m candidate web services, then m^n different *CS*s can be obtained

Where the real limit values $min^{l}_{q_t}$ and $max^{u}_{q_t}$ for an $ACS = (S_1, S_2, \ldots, S_n)$ with $\forall i \in \{1,2\ldots,n\}$, $S_i = \{ws^1_i, ws^2_i, \ldots, ws^{mi}_i\}$ and $\forall j \in \{1,2\ldots,m_i\}$, $ws^j_{i,q_t} = \left[ws^{j,l}_{i,q_t}, ws^{j,u}_{i,q_t}\right]$ are evaluated as follows

$$max^{u}_{q_t} = Agg_{q_t}{}^{n}_{i=1}\left(\max^{m_i}_{j=1}\left\{ws^{j,u}_{i,q_t}\right\}\right) \tag{4.5}$$

$$min^{l}_{q_t} = Agg_{q_t}{}^{n}_{i=1}\left(\min^{m_i}_{j=1}\left\{ws^{j,l}_{i,q_t}\right\}\right) \tag{4.6}$$

Where :

$Agg_{q_t}{}^{n}_{i=1}$ denotes the related crisp QoS aggregation formula, i.e., $\sum$, $\prod$, min and max, for the q_t attribute, as defined in Table 4.2, that has been employed to aggregate the n obtained real values $\max\left\{ws^{j,u}_{i,q_t}\right\}/\min\left\{ws^{j,l}_{i,q_t}\right\}$ from each web services class S_i with $j = 1,2,\ldots,m_i$.

3 Proposed approach

To address the IQSC problem, a proposed approach incorporates two components. The first component utilizes the Skyline operator Borzsony et al. [2001] to diminish the search space of IQSC by eliminating dominated candidate web services. The second component involves swiftly obtaining near-optimal solutions for IQSC by implementing an extended version of the basic Artificial Bee Colony (ABC) algorithm, enhanced with an effective local search method, named EABC.

3.1 Skyline service

The primary objective is to discover a near-optimal solution for the IQSC problem, aiming to maximize the interval utility value defined in Equation 4.1 while adhering to the user's overall QoS constraints outlined in Equation 4.2.

The near-optimal solution comprises a set of atomic web services (*wss*), with each selected from its respective web services class. A challenge arises when not all *wss* in each web services class are potential candidates for constructing the final near-optimal solutions of IQSC. Some previous studies Alrifai et al. [2010], Wang et al. [2013], Ying and Jiande [2020] have utilized the Skyline operator Borzsony et al. [2001] to narrow down the search space of QSC by eliminating *wss* that are not eligible for inclusion in the construction of final solutions, as they are dominated by some of their functionally equivalent *wss* partners. However, a limitation of the existing skyline-based studies is that they consider QoS parameters with precise and exact values. To overcome this limitation, we extend the crisp definitions of *Service Dominance* and *Skyline Service* from Alrifai et al. [2010], Wang et al. [2013], Ying and Jiande [2020] to incorporate the representation of QoS properties using the interval-number model, as explained below.

Definition 17 *(Service Dominance) Let's consider a web services class S and two web services* ws_1, $ws_2 \in S$, *where each one has a set of QoS parameters* q_t*s. We say* ws_1 *dominates* ws_2, *denoted as*

*$ws_1 \prec ws_2$, if and only if : **(1)** $\forall q_t \in QoS^+ : ws_{1,q_t} \geq_{\max} ws_{2,q_t}$, and **(2)** $\forall q_t \in QoS^- : ws_{1,q_t} \leq_{\min} ws_{2,q_t}$, and **(3)** $\exists q_t \in QoS^+, ws_{1,q_t} >_{\max} ws_{2,q_t}$ or $\exists q_t \in QoS^- , ws_{1,q_t} <_{\min} ws_{2,q_t}$.*

Definition 18 *(Skyline Service) For a given web services class $S = \{ws_1, ws_2, \ldots, ws_m\}$ of m functionally equivalent wss. The skyline service of S, denoted by SkS, contains the candidate wss in S that cannot be dominated by any other ws of S. i.e, $SkS = \{ws_i \in S \mid \nexists ws_j \in S : ws_j \prec ws_i\}$.*

To define the skyline service (*SkS*) for each web service class (*S*), the skyline computation process performs pairwise comparisons between the interval values of ws_{q_t} for the compared web services (*wss*) within *S*. This computational process can be computationally expensive, particularly if *S* encompasses a large number of functionally equivalent *wss*. However, for the IQSC interval-number based problem, the skyline service calculation is independent of any online user request Alrifai et al. [2010]. Therefore, the skyline computation can be performed offline using any of the existing efficient skyline algorithms Borzsony et al. [2001]. In our study, we calculate each skyline service for IQSC by adapting the well-known non-dominated method from Deb et al. [2002], employing the service dominance operator $\prec$ as defined in Definition 17. The pseudo code for the interval non-dominated procedure, evaluating each skyline service *SkS* for its associated web services class *S*, is presented in Algorithm 1.

Algorithm 1 Interval non-dominated procedure for evaluating skyline service.

Input : The web service class S
Output : The calculated skyline service SkS of S

```
1:  SkS = ∅
2:  for each web service ws ∈ S do
3:      N_ws ← 0                          ▷ N_ws : is the number of web services in S dominating ws
4:      for each web service ws' ∈ S that differs to ws do
5:          if ws' ≺ ws then
6:              N_ws ← N_ws + 1
7:          end if
8:      end for
9:      if N_ws = 0 then
10:         SkS ← SkS ∪ {ws}
11:     end if
12: end for
```

3.2 Artificial bee colony algorithm

To comprehend the extended version of the ABC algorithm, it is essential to first introduce a brief overview of the canonical ABC :

The ABC algorithm, a swarm intelligence research method introduced by Karaboga and Basturk [2007], draws inspiration from the intelligent behavior of bee swarms in their search for optimal solutions. In this algorithm, the evolution within the bee swarm is guided and targeted.

In the basic ABC algorithm, each food source represents a feasible solution to the problem being solved, and its nectar amount signifies the quality (fitness value) of that solution. The ABC algorithm employs three (03) types of bees to achieve global optimization :

1. Employed Bees : Each food source is assigned a single employed bee. This bee explores its assigned food source to exploit new neighboring food sources and employs a greedy selection technique to choose the best. The information obtained is then disseminated to onlookers within the hive.
2. **Onlooker Bees** : Within their hive, the onlooker bees await the information from the employed bees to select an exploitable food source compared to the one they originally exploited. When they identify better sources, they notify the relevant bees employed to update their locations.
3. **Scout Bees** : If a food source remains unchanged in position after being exploited by employed and onlooker bees for a specific duration, it is considered abandoned. The bee used associated with this source transforms into a scout, tasked with searching for a new food source by exploring a different search area.

The synergy of the three types of bees in the aforementioned algorithm leads to a gradual convergence in the search for the optimal feasible solution. The fundamental strategy of the ABC algorithm is summarized in the following steps :

— **1st Step.** The process begins with the random generation of a population consisting of SN food sources or solutions. Subsequently, the fitness value, representing the nectar amount, is evaluated for each solution within the generated population.

— **2nd Step. Employed Bee Phase** : During the employed bee phase, each bee associated with a particular food source generates a new solution by exploring the neighborhood of its assigned food source. Following the generation of the solution, the fitness value is evaluated. A greedy selection strategy is then applied to decide whether to update the position. This update is executed only if the nectar amount of the newly generated solution surpasses that of the existing one. The process ensures that the algorithm progresses towards more promising solutions.

— **3rd Step. Onlooker Bee Phase** : In the onlooker bee phase, each onlooker bee selects a food source from the population based on the probability value associated with that particular source. Subsequently, the onlooker bee explores the neighborhood of the chosen food source to create a new solution. The nectar amount of the newly formed solution is then evaluated, and a greedy selection mechanism is employed between the old and new solutions to determine the position interchange. This phase ensures a dynamic exploration of the search space by allowing onlooker bees to adapt their choices based on the probability and neighboring solutions.

— **4th Step. Scout Bee Phase** : In the scout bee phase, the algorithm introduces scout bees, which play a crucial role in discovering new food sources. If a food source remains unchanged in

position after being exploited by employed and onlooker bees for a certain number of iterations, it indicates abandonment. The employed bee associated with this abandoned source transforms into a scout bee. Subsequently, the scout bee is dispatched to explore new regions in the search space, aiming to identify potential food sources that can contribute to improving the overall solution quality. This phase enhances the algorithm's exploration capability and ensures adaptability to changes in the optimization landscape.

— **5th Step. Memorization of Best Solution** : The algorithm maintains a record of the best solution identified throughout its execution. This best solution, referred to as the food source, represents the optimal outcome discovered up to the current iteration. The memorization process allows the algorithm to track the most promising solution encountered during its search. This information is crucial for reporting the final result once the algorithm concludes its exploration. The algorithm continually updates this memorized solution whenever a superior food source is found, ensuring that the best-known solution is always retained for further evaluation.

— **6th Step. Verification of Stopping Criterion** : The algorithm regularly assesses whether the predefined stopping criterion has been met. If the specified criterion is not fulfilled, the algorithm returns to the **2nd Step**, resuming the iterative process. On the contrary, if the stopping criterion is satisfied, indicating that a satisfactory solution has been reached or a predetermined number of iterations has been completed, the algorithm concludes. At this point, the algorithm showcases the best-known food source discovered throughout its execution. This final display provides insights into the optimal solution obtained by the algorithm, completing the execution cycle.

3.3 Extended Artificial Bee Colony : EABC

The ABC algorithm has found widespread application in addressing the QSC problem across various service-based environments, such as service-oriented applications x. wang et al. [2019] and cloud computing Huo et al. [2015]. Similar to the fundamental ABC algorithm, the EABC method aims to iteratively and successively employ three types of bees (Employees, Onlookers, and Scouts) within each explored food source area src_g. This iterative process seeks to discover a better alternative, denoted as src_{g+1}, with $g \in 0,1,2,\ldots,MITR$, where $MITR$ represents the maximum iteration number for the EABC algorithm. To initiate the algorithm, an initial set of food sources Z, represented by solutions $CS_1, CS_2, \ldots, CS_Z$ with their initial positions $src_0 = CS_1^0, CS_2^0, \ldots, CS_Z^0$ is randomly generated. The dimensional values (web services) of each food source position CS_z^0, with $z = 1,2,\ldots,Z$, are defined according to the encoding schema provided in 3.3.1. The nectar amount, representing the fitness value, is evaluated using the interval utility function, denoted as $IU(CS_z^0)$, as defined in Equation 4.1. Additionally, for each generated food source CS_z, an integer variable $trial_z$ is assigned, initialized to zero.

Moreover, to augment the local search capability of the EABC algorithm, a **local search method** is applied to the best food source position within each newly discovered food source area src_{g+1},

denoted as CS_{best}^{g+1}. The EABC algorithm is illustrated by a flowchart in Fig 4.3. Subsequently, in the following subsections, the encoding schema of *CS*s, the generation of the initial food source area src_0, and the operations of the three types of bees involved in searching for favorable food source areas—namely, *Employees' work*, *Onlookers' work*, and *Scouts' work* are detailed. Additionally, the integrated local search method and the termination criterion of the proposed EABC algorithm are thoroughly described.

3.3.1 Encoding schema of food sources positions and generation of the initial food sources area

Each food source position CS_z^g in the gth food source area src_g is represented by an n-dimensional array of integers, where $z = 1,2,\ldots,Z$, and Z is the population size of EABC. The gth food source position CS_z^g for the solution CS_z, denoted as $CS_z^g = (CS_{z,1}^g, CS_{z,2}^g, \ldots, CS_{z,n}^g)$, consists of n integer elements indicating the selected candidate *ws*s from their skyline services SkS_i, where $i = 1,2,\ldots,n$. For the initial food source area src_0, each integer element of each food source position $CS_z^0 = (CS_{z,1}^0, CS_{z,2}^0, \ldots, CS_{z,n}^0)$ was randomly generated using the following procedure :

$$\begin{gathered} \forall : i = 1,2,\ldots,n \\ CS_{z,i}^0 = 1 + \lceil rand(0,1) * (m_i - 1) \rceil \end{gathered} \tag{4.7}$$

Where $CS_{z,i}^0$ is an integer number representing the ith selected *ws* from the ith skyline service SkS_i that has m_i functionally equivalent *ws*s, $rand(0,1)$ is considered as a real number, which was randomly generated from the range $[0,1]$, and $\lceil\ \rceil$ represents the rounding up integer function.

3.3.2 Employees work

The employed bees explore each current food source area src_g to find a new and better source src_g^{Emp}. Similar to the conventional updating positions of food sources in ABC Karaboga and Basturk [2007], only one dimension (a unique web service) of each food source position $CS_z^g = (CS_{z,1}^g, CS_{z,2}^g, \ldots, CS_{z,j}^g, \ldots, CS_{z,n}^g)$ with n atomic web services is considered to update the CS_z^g position of CS_z.

Hence, the new food source position $CS_z^{Emp} = (CS_{z,1}^g, CS_{z,2}^g, \ldots, CS_{z,j}^{Emp}, \ldots, CS_{z,n}^g)$ has the same atomic web services as its old one CS_z^g, except for the jth web service (i.e., the $CS_{z,j}^{Emp}$ value), which is defined as follows :

$$CS_{z,j}^{Emp} = \lceil CS_{z,j}^g + \phi_j * (CS_{z,j}^g - CS_{l,j}^g) \rceil \tag{4.8}$$

Where :

For each explored CS_z^g, j is a randomly-selected dimension, i.e., the jth skyline service SkS_j, from the range $[1,n]$, CS_l^g with $l \neq z$ represents a randomly-selected food source position from src_g,

ϕ_j is a real value randomly generated within the range $[-1,1]$ for every selected skyline service SkS_j and $\lceil\ \rceil$ is the rounding up integer function.

If the $CS_{z,j}^{Emp}$ value is out of the range $[1,m_j]$ with m_j is the total number of the functionally equivalent *wss* of SkS_j, then the $CS_{z,j}^{Emp}$ value will be updated by the nearest one of the two bounds $\{1,m_j\}$.

After the definition of the new food sources positions CS_z^{Emp}s with $z = 1,2,\ldots,Z$, their interval utility values, i.e., $IU(CS_z^{Emp})$, are calculated using Equation 4.1.

In order to update each old food source position CS_z^g of CS_z by its new defined one CS_z^{Emp}, the greedy selection mechanism of ABC Karaboga and Basturk [2007] is adapted by using the following Deb's solutions selection rules Deb [2000].

R 1. If CS_z^{Emp} and CS_z^g are feasible food sources positions, i.e., the user's global QoS constraints as given in Equation 4.2 are satisfied, then :

- **R 1.1.** If $IU(CS_z^{Emp}) >_{\max} IU(CS_z^g)$ then CS_z^{Emp} is the new food source position of CS_z and its $trial_z$ is reset to zero ;
- **R 1.2.** Otherwise, the old food source position CS_z^g of CS_z is maintained and its $trial_z$ is incremented by one.

R 2. If CS_z^{Emp} is a feasible food source position and CS_z^g is an infeasible one, then CS_z^{Emp} is the new food source position of CS_z and its $trial_z$ is reset to zero.

R 3. If CS_z^{Emp} is an infeasible food source position and CS_z^g is a feasible one, then the old food source position CS_z^g of CS_z is maintained and its $trial_z$ is incremented by one.

R 4. If CS_z^{Emp} and CS_z^g are infeasible food sources positions then :

- **R 4.1.** If the normalized global constraint violation of CS_z by considering its new food source position CS_z^{Emp}, as will be defined in Equation 4.11, is larger than its one by considering its old food source position CS_z^g, then CS_z^{Emp} is the new food source position of CS_z and its $trial_z$ is reset to zero ;
- **R 4.2.** Otherwise, the old food source position CS_z^g of CS_z is maintained and its $trial_z$ is incremented by one.

Normalized global constraint violation of an infeasible food source position : Given any infeasible food source position, denoted by *ICS*, the constraint violation amounts of its violated user's overall QoS requirements, denoted by $ICS_{q_k}^{cst}$s, are evaluated as follows.

$$\begin{aligned} \forall q_k \in QoS^+ : \text{ if } Cst_{q_k} >_{\max} ICS_{q_k} \text{ then} \\ ICS_{q_k}^{cst} = Cst_{q_k} \ominus ICS_{q_k} \end{aligned} \tag{4.9}$$

$$\begin{aligned} \forall q_k \in QoS^- : \text{ if } Cst_{q_k} <_{\min} ICS_{q_k} \text{ then} \\ ICS_{q_k}^{cst} = ICS_{q_k} \ominus Cst_{q_k} \end{aligned} \tag{4.10}$$

By adapting the simple additive weighting -SAW- method Rao [2007] to support the interval numbers' calculations, the $ICS_{q_k}^{cst}$s interval-values are aggregated into a single interval value ICS_{cst} through using the following equation.

$$ICS_{cst} = \frac{1}{V} \otimes \sum_{t=1}^{k} \overline{ICS_{q_t}^{cst}} \tag{4.11}$$

Where :

V is the total number of the violated overall QoS constraints by ICS,

$\overline{ICS_{q_k}^{cst}}$ is the normalized interval-valued of its associated original one $ICS_{q_k}^{cst} = \left[ICS_{q_k}^{l,cst}, ICS_{q_k}^{u,cst}\right]$ for the kth violated user's global QoS constraint $Cst_{q_k} = \left[Cst_{q_k}^{l}, Cst_{q_k}^{u}\right]$.

Since the lower $ICS_{q_k}^{cst}$s are, the lower its global constraint violation amount is (i.e., in other words, the higher its aggregated normalized interval-valued ICS_{cst} is). Therefore, the lower and upper limit values of the interval-valued $\overline{ICS_{q_k}^{cst}} = \left[\overline{ICS_{q_k}^{l,cst}}, \overline{ICS_{q_k}^{u,cst}}\right]$ are calculated by using the interval negative normalization process as given in Equations 4.12 and 4.13, respectively.

$$\overline{ICS_{q_k}^{l,cst}} = \begin{cases} \frac{max_{q_k}^{u} - Cst_{q_k}^{l} - ICS_{q_k}^{u,cst}}{\left(max_{q_k}^{u} - Cst_{q_k}^{l}\right) - \left(min_{q_k}^{l} - Cst_{q_k}^{u}\right)} & \text{if } \left(max_{q_k}^{u} - Cst_{q_k}^{l}\right) \neq \left(min_{q_k}^{l} - Cst_{q_k}^{u}\right) \\ 1 & \text{if } \left(max_{q_k}^{u} - Cst_{q_k}^{l}\right) = \left(min_{q_k}^{l} - Cst_{q_k}^{u}\right) \end{cases} \tag{4.12}$$

$$\overline{ICS_{q_k}^{u,cst}} = \begin{cases} \frac{max_{q_k}^{u} - Cst_{q_k}^{l} - ICS_{q_k}^{l,cst}}{\left(max_{q_k}^{u} - Cst_{q_k}^{l}\right) - \left(min_{q_k}^{l} - Cst_{q_k}^{u}\right)} & \text{if } \left(max_{q_k}^{u} - Cst_{q_k}^{l}\right) \neq \left(min_{q_k}^{l} - Cst_{q_k}^{u}\right) \\ 1 & \text{if } \left(max_{q_k}^{u} - Cst_{q_k}^{l}\right) = \left(min_{q_k}^{l} - Cst_{q_k}^{u}\right) \end{cases} \tag{4.13}$$

Where $max_{q_k}^{u}$ and $min_{q_k}^{l}$ have been previously defined in Equations 6.11 and 6.12, respectively.

3.3.3 Onlookers work

The onlooker bees select Z food source positions from those discovered by the employed bees, i.e., src_g^{Emp}, for further exploration. In the basic ABC algorithm, the selection criterion for the Z food source positions is based on the roulette wheel selection Karaboga and Basturk [2007]. However, this method faces the problem of local optima stagnation Xiang and An [2013]. Therefore, in this study, the binary tournament selection method Miller et al. [1995] is employed to determine the newly selected food source positions, denoted by src_g^{Sel}. The same solution selection rules as defined by Deb in the above subsection are used to determine src_g^{Sel} by repeating this selection procedure Z times for each pair of randomly selected food source positions from src_g^{Emp}.

src_g^{Sel} is re-explored to find new food sources area src_g^{Onl} by employing the same updating formula of food sources positions, as given in Equation 4.8, where the aforesaid Deb's solutions selection rules, which are itemized in the *Employees work* part, are used to update the explored food sources positions src_g^{Sel} by its new discovered ones src_g^{Onl}.

3.3.4 Scouts work

Similar to the scouts work of the basic ABC algorithm, the scout bees of EABC are used to update one randomly-selected food source position from the discovered ones by the onlooker bees, i.e., src_g^{Onl}, that have not updated theirs positions after $limit$ iterations, where the $limit$ parameter of EABC indicates the criterion to identify an abandoned food source position. This latter is updated using Equation 4.7 and its associated $trial$ variable is reinitialized to zero. As a result, by performing the above works of Employee, Onlooker and Scout bees, a new food sources area src_{g+1} is discovered to replace the old one src_g.

3.3.5 Local search method

To enhance the quality of the final best CS solution (CS_{best}). Therefore, for each new discovered food sources area src_{g+1}, a local search method is employed for its best food source position, represented by $CS_{best}^{g+1} = (CS_{best,1}^{g+1}, CS_{best,2}^{g+1}, \ldots, CS_{best,n}^{g+1})$, where n indicates the number of its skylines services and $CS_{best,i}^{g+1}$ with $i \leq n$ represents the subscript value of the i^{th} selected web service from its associated skyline services SkS_i. The following steps describe the incorporated local search method to the EABC algorithm.

Step 1. j is a randomly selected integer number from the set $\{1,2,\ldots,n\}$ that represents the selected skyline service SkS_j form the n existing ones.

Step 2. L are the different generated new food source positions $CS_{best,l}^{g+1}$ with $l \in \{1,2,\ldots,L\}$ from CS_{best}^{g+1}, where each $CS_{best,l}^{g+1}$ is created with the same atomic web services of CS_{best}^{g+1} except for its j^{th} web service, which has been randomly selected from its related SkS_j.

Step 3. The interval utility values of the generated $CS_{best,l}^{g+1}$s are evaluated, as given in Equation 4.1, and determine the best one among the new generated $CS_{best,l}^{g+1}$s and CS_{best}^{g+1} to update the food source position of CS_{best} by the determined one.

3.3.6 Ending criterion of EABC

The afore-explained tasks of the Employee, Onlooker, and Scout bees, along with the integrated local search method, are iteratively performed for each newly discovered food source area until the stopping criterion of the EABC algorithm is met, i.e., the maximum iteration number $MITR$ is reached. The best feasible food source position among those in the last discovered food source area, with the highest interval utility value, represents the final near-optimal solution to the IQSC problem using EABC.

4 Experiments and tools

The compared algorithms are implemented with Matlab R2016b, and performed on the same personnel computer, which runs Windows 7 and has an Intel(R) Core(TM) i5-4570, CPU 3.20 GHz and 4 Go of memory as a hardware configuration.

In order to assess the effectiveness of the proposed approach,two comparison metrics have been used.

— **(a) Running time** : This metric denotes the computation time required by each executed algorithm to identify its near-optimal solutions for solving the IQSC problem.

— **(b) Optimality** : This metric serves as an indicator of the solution quality, assessing the obtained solutions based on their interval utility values as defined in Equation 4.1. This metric provides insights into how well the solutions meet the criteria specified in the optimization process.

4.1 Interval version of the public QWS dataset

In the comparison experiments, as the public QWS dataset Al-Masri and Mahmoud [2008] was published with 2507 atomic web services, where each web service has 09 non-ambiguous QoS values for 09 QoS properties including (1) Response Time, (2) Availability, (3) Throughput, (4) Successability, (5) Reliability, (6) Compliance, (7) Best Practices, (8) Latency and (9) Documentation. Therefore, an interval version of QWS, denoted by IQWS, is provided to be used in the upcoming experiments, where the QoS interval values of IQWS are generated via multiplying the precise QoS values of each considered QoS parameter of QWS by the random interval number $[r_1, r_2]$ with $r_1 = 0.9 + 0.1 * rand(0,1)$, $r_2 = 1.0 + 0.1 * rand(0,1)$, and $rand(0,1)$ is a uniformly distributed random real number in the range $[0,1]$. Since the QWS dataset does not contain the web service price parameter, and as mentioned previously, only the availability (q_1), throughput (q_2), response time (q_3) and service price (q_4) attributes are considered to generate the IQWS dataset. Hence, the interval values of the IQSW's web services in the price attribute have been randomly generated by $r3 \otimes [r_1, r_2]$ with $r3$ is a random generated real number from the range $[2,5]$\$. Moreover, the importance and priorities of the four considered QoS attributes were set to the same value, i.e, $\forall t \in \{1,2,3,4\}, w_{q_t} = \frac{1}{4}$, and each considered user's global QoS requirement Cst_{q_t} for the q_t attribute was set as given in the following Equation.

$$
\begin{array}{l}
Cst_{q_t} = \\
\left[\max\left(r_1 * SU_{q_t}, min^l_{q_t}\right), \min\left(r_2 * SU_{q_t}, max^u_{q_t}\right)\right] \\
\text{With } SU_{q_t} = \\
\begin{cases}
\mu_{q_t} * (max^u_{q_t} - min^l_{q_t}) + min^l_{q_t} & \text{if } q_t \in QoS^+ \\
max^u_{q_t} - \mu_{q_t} * (max^u_{q_t} - min^l_{q_t}) & \text{if } q_t \in QoS^-
\end{cases}
\end{array}
\tag{4.14}
$$

Where :

$\mu_{q_t} \in [0,1]$ is a severity factor used to adjust the considered user's global QoS constraint Cst_{q_t}

$max^u_{q_t}$ and $min^l_{q_t}$ are calculated as given by Equations 6.11 and 6.12, respectively.

Here, only two global QoS constraints Cst_{q_3} and Cst_{q_4} of the response time and price attributes are considered in solving IQSC by setting their severity factors μ_{q_3} and μ_{q_4} to 0.3 and 0.2, respectively, whereas, μ_{q_1} and μ_{q_2} of the availability and throughput properties are set to the zero values.

4.2 Parameters setting of the compared algorithms

To investigate the performance and the efficiency of the proposed EABC, a series of comparison to the one obtained using the PSO algorithm with skyline operator Wang et al. [2013], the proposed Discrete Gbest-guided Artificial Bee Colony (DGABC) approach in Huo et al. [2015] and the improved Harris Hawks Optimization (HHO) algorithm by a developed Elite Evolutionary Strategy (EES) in Li et al. [2021b] that has been named by its authors EESHHO, have been conducted. The compared approaches to the proposed one (i.e., PSO, DGABC and EESHHO) were proposed to solve the QSC problem with non-ambiguous QoS parameters. Hence, these approaches have been adopted to support interval utility calculations of solutions as defined in Equation 4.1. To simplify things, the interval extended versions of the PSO-based approach Wang et al. [2013], the DGABC Huo et al. [2015] and the EESHHO Li et al. [2021b] algorithms are named IPSO, IDGABC and IEESHHO, respectively.

For the parameters setting of the compared algorithms, EABC, IPSO, IDGABC and IEESHHO share the same population size ($Z = 40$) and the same stopping criterion, which is the number of solutions evaluations that was set to 50000. However, for their appropriate parameters, the inertia weight (w) and the two accelerating coefficients (c_1 and c_2) of IPSO were set to $w = 0.8$ and $c_1 = c_2 = 2.0$, as they were recommended in their related reference Wang et al. [2013].

The *limit* parameter for both EABC and IDGABC, which indicates the criterion to identify an abandoned food source position, was set to 80. Whereas, the L parameter of EABC that represents the number of neighbor food source positions of the best solution in the EABC's local search method was set to 3. Moreover, the control parameters E and sp of IEESHHO, which are used to switch between the exploration and exploitation phases, and controls the proportion of the best parental genes in EES, respectively, were set by Equations 4.15 and 4.16 as designed by the authors of EESHHO Li et al. [2021b].

$$sp = rand(-1,1) \times \left(1 - \frac{t}{T}\right) \tag{4.15}$$

$$E = 2E_0 \times \left(1 - \frac{t}{T}\right) \tag{4.16}$$

Where :

t and T are respectively the current and the maximum number of the IEESHHO's iterations cycles

$rand(-1,1)$ and E_0 are random generated numbers between -1 and 1 with E_0 is updated by each agent (solution) for each population evolution.

The afore-listed algorithms are performed to solve five abstract composite services ACS^m_ns, with $(n,m) \in \{(5,501),(10,250),(15,167),(20,125)\ ,(25,100)\}$, where each ACS^m_n consists of solving IQSC with n web services classes per m functionally equivalent ones that have been randomly selected

from the 2507 WSs of IQWS. For each ACS_n^m, the compared EABC, IPSO, IDGABC and IEESHHO are carried out 30 independent times to define the best, worst and average optimality values, and average running times of their obtained near-optimal solutions.

4.3 Comparison results discussion

For each solved ACS_n^m, the best, worst and average interval utility values of the obtained near-optimal solutions by the compared algorithms have been listed in Tables 4.3, 4.4 and 4.5, respectively. As it is shown from the results of solving the five ACS_n^ms written down in these Tables that the proposed EABC in this study reaches very higher interval optimality values compared to the ones of its opponent approaches.

Furthermore, as seen in Table 4.6, listing the interval variances (IVs), as calculated by Equation 4.17 of the compared algorithms in solving each ACS_n^m over the 30 independent running times where EABC and IDGABC have obtained two optimal IVs in solving the five ACS_n^ms, IPSO has obtained one optimal IVs in solving the five ACS_n^ms. Whereas, the IEESHHO algorithm has not obtained any optimal IV for the five solved ACS_n^ms. Moreover, the average IVs for the five solved ACS_n^ms of the EABC algorithm is superior to the ones of other compared algorithms. Therefore, EABC has better stability than IDGABC, IPSO and IEESHHO.

$$IV = \frac{1}{30} \otimes \sum_{i=1}^{30} \left(IU_{best}^{i} \ominus \overline{IU}\right) \otimes \left(IU_{best}^{i} \ominus \overline{IU}\right) \tag{4.17}$$

Where :

IU_{best}^{i} is the i^{th} interval utility value of the i^{th} obtained near-optimal solution by a compared algorithm. $\overline{IU} = \sum_{i=1}^{30} IU_{best}^{i}$ is the average interval-valued of the all obtained near-optimal solutions.

In addition, as we can see from Fig. 4.4, the one that plots the average running times of the compared approaches to get their near-optimal solutions in solving the listed ACS_n^ms in Table 4.6 over the 30 independent executions when the number of web services classes n was set with small values, i.e., $n \leq 5$, the average running times of IPSO are slightly better than the ones of EABC. However, when n was set with medium or large values, $n \geq 10$, the proposed EABC algorithm obtained near-optimal solutions with less average running times compared to the ones of IPSO. Besides, for all the solved ACS_n^ms, EABC is faster than IEESHHO and slightly better than IDGABC in terms of computation time.

From this discussion, EABC outperforms the compared IPSO, IDGABC and IEESHHO algorithms in terms of final CS solutions optimality as-well-as efficiency, especially in solving users' requests that need the composition of an important number of elementary web services.

TABLE 4.3 Comparison results of the best interval utility values of the obtained near-optimal solutions by the compared approaches in solving each ACS_n^m of IQSC with n web services classes per m functionally equivalent web services

ACS_n^m	*EABC*	*IPSO*	*IDGABC*	*IEESHHO*
ACS_5^{501}	**[0.7075,0.8234]**	[0.6849,0.7894]	[0.6825,0.8080]	[0.7156,0.8037]
ACS_{10}^{250}	**[0.6240,0.7140]**	[0.5690,0.7051]	[0.5843,0.7102]	[0.5980,0.7006]
ACS_{15}^{167}	**[0.5603,0.7116]**	[0.5079,0.6015]	[0.5482,0.6935]	[0.5440,0.6412]
ACS_{20}^{125}	**[0.5305,0.6839]**	[0.4753,0.5477]	[0.5126,0.6454]	[0.4898,0.5664]
ACS_{25}^{100}	**[0.5141,0.6239]**	[0.4813,0.5208]	[0.5231,0.5624]	[0.5107,0.5396]

The best interval utility values are in boldface.

TABLE 4.4 Comparison results of the worst interval utility values of the obtained near-optimal solutions by the compared approaches in solving each ACS_n^m of IQSC with n web services classes per m functionally equivalent web services

ACS_n^m	*EABC*	*IPSO*	*IDGABC*	*IEESHHO*
ACS_5^{501}	**[0.6936,0.8024]**	[0.6337,0.7443]	[0.6623,0.7615]	[0.6474,0.7584]
ACS_{10}^{250}	**[0.5826,0.7303]**	[0.5296,0.6487]	[0.5705,0.6793]	[0.5377,0.6227]
ACS_{15}^{167}	**[0.5407,0.7106]**	[0.4616,0.5295]	[0.5413,0.6421]	[0.4825,0.5924]
ACS_{20}^{125}	**[0.5273,0.6529]**	[0.4605,0.5002]	[0.4929,0.6003]	[0.4573,0.5059]
ACS_{25}^{100}	**[0.5122,0.5839]**	[0.4430,0.4631]	[0.5016,0.5247]	[0.4579,0.4840]

The best interval utility values are in boldface.

4.4 Effectiveness of the local search method

In this subsection, the feasibility of the employed local search method is investigated, the aforementioned ACS_n^ms of Table 4.6 have been solved by the EABC algorithm without the local search method (i.e., the basic ABC algorithm) over 30 independent executions where its obtained results are compared to the ones of the developed EABC. As shown in Table 4.7 that illustrates the obtained results by EABC and ABC algorithms, it is obvious that the best, worst and average interval utility values of the obtained near-optimal solutions by EABC, as listed previously in Tables 4.3, 4.4 and 4.5, are better than the ones of the basic ABC algorithm. Moreover, for comparing the convergence of EABC to that of the basic ABC algorithm, Fig. 4.5 plots the average mid-points [5] of the interval utility values of the obtained near-optimal solutions of both EABC and ABC in solving the ACS_{20}^{125} instance over 30 independent running times.

In this experiment, the stopping criterion of each compared algorithm is set to 15 seconds. It is clear from this figure that EABC gives a better convergence rate compared to the basic ABC algorithm. Therefore, by integrating the proposed local search method into EABC, its convergence is improved in order to get near-optimal solutions with high quality.

5. A mid-point (i.e., center) of an interval number $A = [a^l, a^u]$, which represents its performance, is calculated as $(a^l + a^u)/2$ Bhunia and Samanta [2014].

TABLE 4.5 Comparison results of the average interval utility values of the obtained near-optimal solutions by the compared approaches in solving each ACS_n^m of IQSC with n web services classes per m functionally equivalent web services

$\mathbf{ACS}_n^m$	***EABC***	***IPSO***	***IDGABC***	***IEESHHO***
ACS_5^{501}	**[0.7039,0.8164]**	[0.6742,0.7711]	[0.6742,0.7885]	[0.6769,0.7829]
ACS_{10}^{250}	**[0.5924,0.7366]**	[0.5626,0.6691]	[0.5727,0.7022]	[0.5646,0.6631]
ACS_{15}^{167}	**[0.5545,0.7102]**	[0.4949,0.5718]	[0.5422,0.6714]	[0.5133,0.6015]
ACS_{20}^{125}	**[0.5279,0.6722]**	[0.4566,0.5287]	[0.5028,0.6177]	[0.4636,0.5442]
ACS_{25}^{100}	**[0.5245,0.5947]**	[0.4640,0.4940]	[0.5091,0.5447]	[0.4893,0.5212]

The best interval utility values are in boldface.

TABLE 4.6 Comparison results of the interval variances of the obtained near-optimal solutions by the compared approaches in solving each ACS_n^m of IQSC with n web services classes per m functionally equivalent web services

$\mathbf{ACS}_n^m$	***EABC***	***IPSO***	***IDGABC***	***IEESHHO***
ACS_5^{501}	[-0.0130,0.0149]	[-0.0096,0.0116]	**[-0.0135,0.0153]**	[-0.0115,0.0136]
ACS_{10}^{250}	**[-0.0216,0.0221]**	[-0.0117,0.0136]	[-0.0174,0.0190]	[-0.0100,0.0120]
ACS_{15}^{167}	**[-0.0251,0.0259]**	[-0.0060,0.0079]	[-0.0173,0.0189]	[-0.0080,0.0100]
ACS_{20}^{125}	[-0.0215,0.0227]	**[-0.0055,0.0066]**	[-0.0137,0.0151]	[-0.0067,0.0085]
ACS_{25}^{100}	[-0.0052,0.0060]	[-0.0008,0.0017]	**[-0.0013,0.0019]**	[-0.0009,0,0019]
Average	**[-0.0173 0.0183]**	[-0.0067 0.0083]	[-0.0126 0.0140]	[-0.0074 0.0092]

The best interval variances are in boldface.

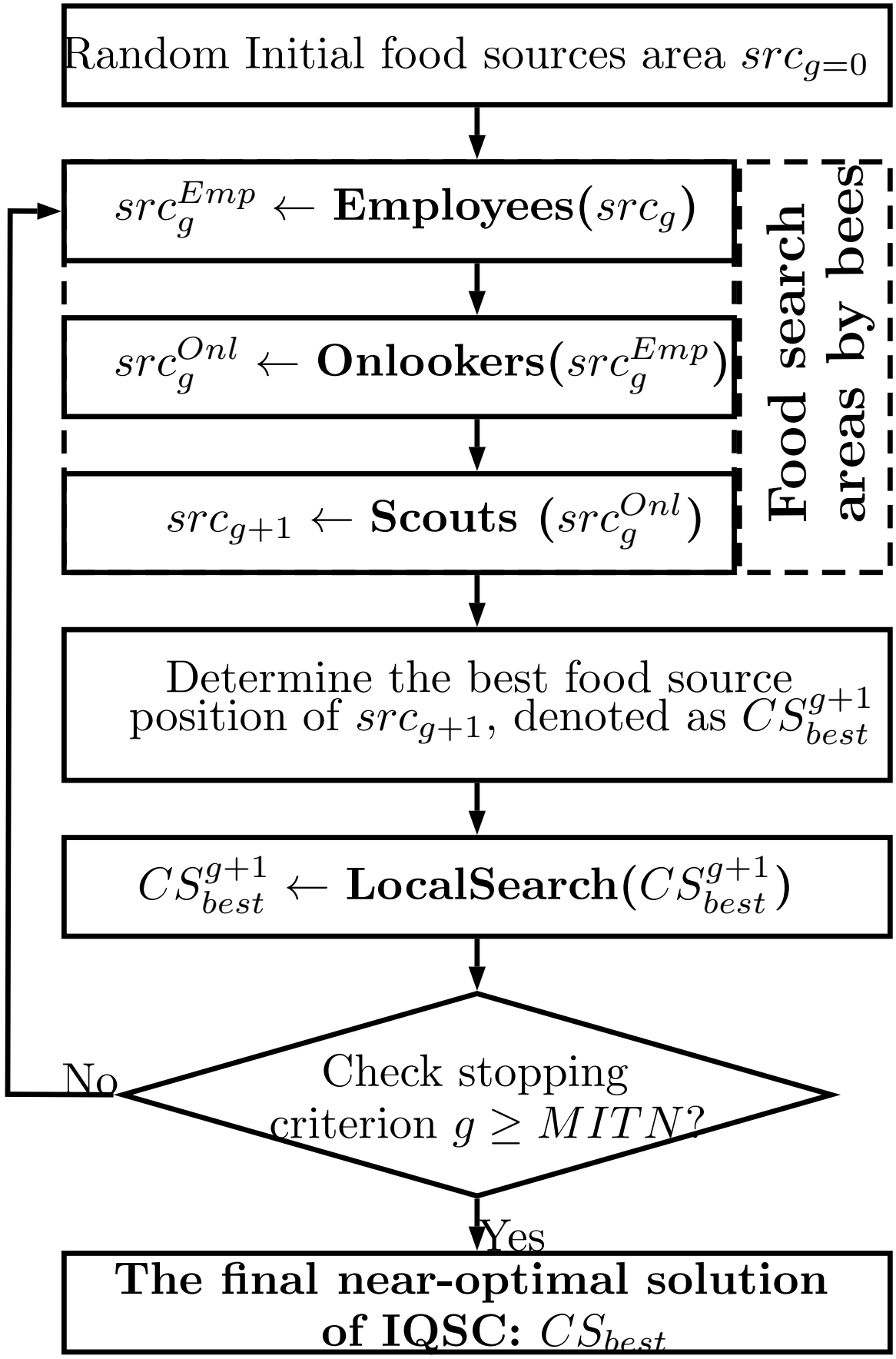


FIGURE 4.3 The flowchart of EABC.

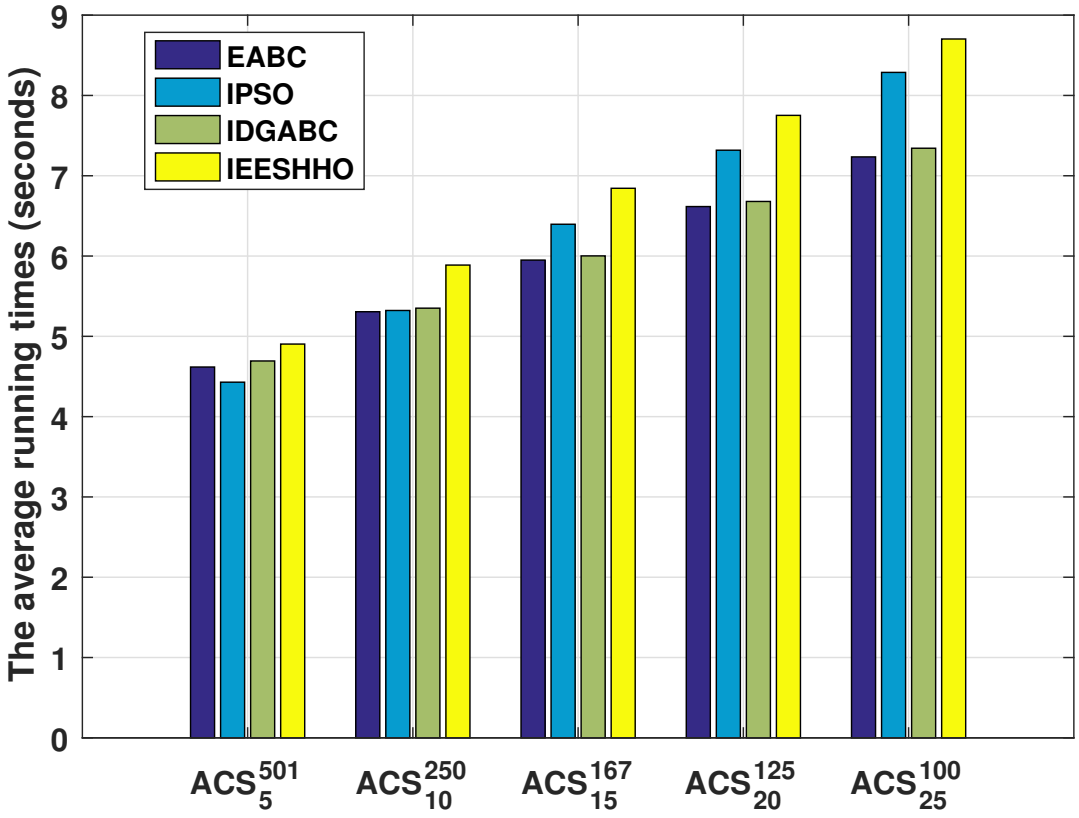


FIGURE 4.4 The average running times of the compared algorithms in solving the ACS_n^ms of n web services classes per m functionally equivalent web services (for colors, see online version).

TABLE 4.7 Comparison results of the best, worst and average interval utility values of the obtained near-optimal *CS*s solutions by the EABC and the basic ABC algorithms in solving each ACS_n^m of IQSC with n web services classes per m functionally equivalent web services

ACS_n^m	EABC			Basic ABC		
	Best	***Worst***	***Average***	***Best***	***Worst***	***Average***
ACS_5^{501}	**[0.7075,0.8234]**	**[0.6936,0.8024]**	**[0.7039,0.8164]**	**[0.7075,0.8234]**	[0.7043,0.7777]	[0.7000,0.8049]
ACS_{10}^{250}	**[0.6240,0.7140]**	**[0.5826,0.7303]**	**[0.5924,0.7366]**	[0.6109,0.7219]	[0.5788,0.7265]	[0.5885,0.7317]
ACS_{15}^{167}	**[0.5603,0.7116]**	**[0.5407,0.7106]**	**[0.5545,0.7102]**	[0.5518,0.7137]	[0.5340,0.7069]	[0.5512,0.7028]
ACS_{20}^{125}	**[0.5305,0.6839]**	**[0.5273,0.6529]**	**[0.5279,0.6722]**	[0.5294,0.6662]	[0.4998,0.6539]	[0.5143,0.6583]
ACS_{25}^{100}	**[0.5141,0.6239]**	**[0.5122,0.5839]**	**[0.5245,0.5947]**	[0.5183,0.5956]	[0.51389,0.5556]	[0.5181,0.5757]

The best interval utility values are in boldface.

5 Time complexity of the EABC algorithm

Given n is the number of services classes, m is the total number of the equivalent web services per each services class, r is the total number of the QoS parameters, and K is the total number of the users' global QoS constraints. Moreover, by considering the population size Z of food sources and the maximum iteration number $MITR$ of the proposed extended artificial bee colony EABC algorithm, so, its time complexity, which is decided by one loop of three food search works by bees : **(1) Employees work**, **(2) Onlookers work**, and the **(3) Scoots work**, is analyzed as follows :
The Employees **(1)** and the Onlookers **(2)** works need $2*Z$ calculation times to define the new positions of Z food sources, $2*Z*n*r$ computation times to calculate the global QoS interval-values of the new defined food sources' positions, and $2*Z$ solution comparisons to update the old food sources' potions by the new ones through using the greedy selection mechanism of ABC. Therefore, the complexity of **(1)** and **(2)** is $\mathcal{O}(2*Z+2*Z*n*r+2*Z) = \mathcal{O}(Z*n*r)$. The complexity of the Scoots **(3)** work is decided by one update of an arbitrary selected abandoned food source from the abounded ones. Hence, the complexity of **(3)** is $\mathcal{O}(n*r)$ indicating the computation time to evaluate the global QoS interval-valued of the selected abandoned food source. As a conclusion, by considering the $MITR$ iteration cycles for the proposed EABC; so, its time complexity is $\mathcal{O}(MIN*Z*n*r)$.

6 Conclusion

In this chapter, we proposed a novel approach that combines two key components, namely the skyline operator and an extended artificial bee colony with a local search method (EABC), to tackle the QoS-aware web service composition problem under uncertain QoS parameters dubbed as IQSC which is demonstrated to be a part of the NP-hard class. We represented the ambiguity of QoS values using interval numbers and formulated the problem as an interval constrained single-objective optimization model (IQSC).

The introduced approach leverages both the skyline operator and EABC to address the formulated IQSC. The skyline operator plays a crucial role in reducing the search space of the model by eliminating redundant and dominated web services from sets of functionally equivalent ones. Meanwhile, the EABC efficiently explores the reduced search space to identify near-optimal composite services of IQSC, incorporating a local search method for enhanced performance.

To assess the effectiveness of EABC, we conducted experiments using an interval-extended version of the public QWS dataset. The results demonstrates that our proposed approach outperforms existing methods, including skyline-based PSO, an efficient discrete gbest-guided artificial bee colony, and a recently introduced Harris Hawks optimization with an elite evolutionary strategy, in terms of both Optimality and Running Time.

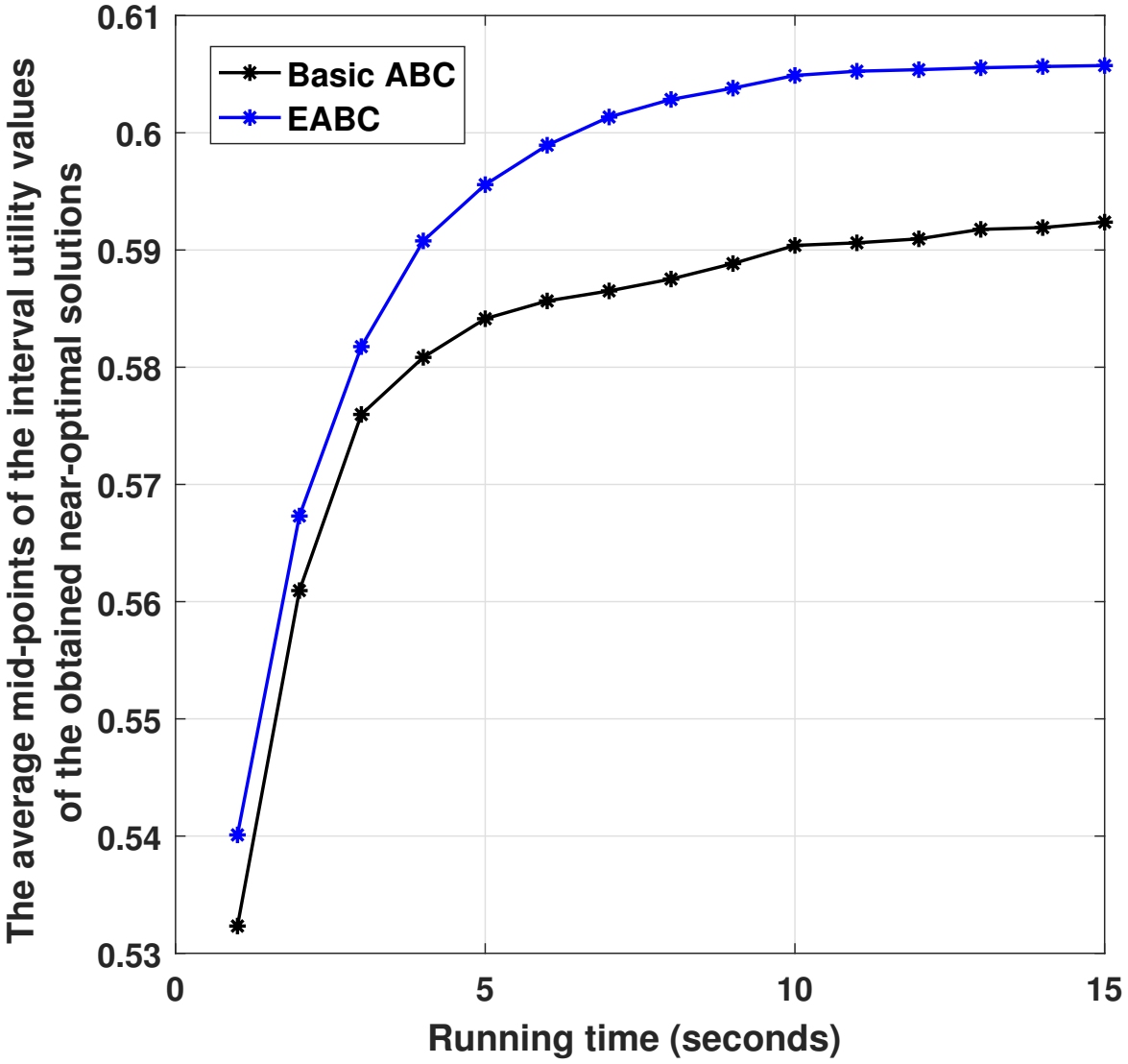


FIGURE 4.5 The convergence of the EABC and the basic ABC algorithms to find-out their near optimal solutions in solving ACS_{20}^{125} over 30 independent running times (for colors, see online version).

Chapter 5

An Improved Discrete Flower Pollination Algorithm for Fuzzy QoS-aware IoT Services Composition Based on Skyline Operator

Introduction

The Internet of Things (IoT) encompasses a global network of smart objects that communicate and collaborate to deliver IoT-based applications for end-users Mashal et al. [2015]. Typically, these applications leverage the functionalities provided by combining atomic IoT services from cooperating smart devices Viriyasitavat et al. [2019]. However, as the number of atomic IoT services with similar functionalities grows, the challenge arises of selecting and composing these services for optimal performance, taking into account their varying non-functional parameters, known as Quality of Service (QoS) properties (e.g., cost, reputation, execution and response times, reliability, and availability) Hamzei and Navimipour [2018].

Developing IoT service selection and composition engines that automatically choose the most suitable atomic IoT services for a given complex user request (IoT-based application) and provide an optimal Composite of IoT-Services (CS) in terms of QoS values has become challenging. The composition of this CS involves selecting several atomic IoT services, each chosen from its associated set of functionally-equivalent IoT services. Furthermore, the global QoS values of the composed CS, calculated by aggregating the QoS values of its atomic IoT services, must meet the global QoS constraints specified by the end-user. This challenge, recognized in the literature as a formidable NP-hard constrained optimization problem, is referred to as the QoS-aware service composition/selection (QSC) problem Abosaif and Hamza [2020].

Approaches addressing the QoS-aware service composition (QSC) problem can be categorized into several types, including those utilizing integer linear programming (ILP), meta-heuristic optimization algorithms (bio-inspired or evolutionary), and Skyline-based (Pareto-based) QSC methods Jatoth et al. [2015], She et al. [2019]. ILP solvers, such as LpSolve [1] or CPLEX [2], have been applied in studies like Ardagna and Pernici [2007], Alrifai et al. [2012] to find optimal Composite of IoT-Services (CS) solutions in terms of Quality of Service (QoS). However, the ILP/MILP formulations in these studies require linear equations for optimized QoS-based objectives and global QoS constraints, posing limitations on non-linear QoS models. Additionally, due to the NP-hard nature of the QSC problem, ILP solvers become inefficient, especially for problems with large search spaces.

To overcome these limitations, researchers have turned to bio-inspired and evolutionary optimization algorithms to obtain near-optimal solutions within reasonable processing times. For instance, Ding et al. [2015] introduced a QoS-aware service selection approach using the evolutionary Genetic Algorithm (GA) to solve the QSC problem under transactional properties of web services and CSs. Bio-inspired intelligent optimization algorithms such as particle swarm optimization (PSO) Naseri and Jafari Navimipour [2019], artificial bee colony (ABC) Xu et al. [2016a], and whale optimization algorithm (WOA) Jin et al. [2022], among others, have been employed in various QSC approaches.

Skyline-based QSC approaches Alrifai et al. [2010], Wang et al. [2013], Ying and Jiande [2020], Guo et al. [2017] leverage the skyline operator Borzsony et al. [2001], based on Pareto dominance, to reduce the search space. This technique filters out poor-quality candidate services, improving computational efficiency in QSC approaches.

Recent review papers Abosaif and Hamza [2020], Masdari and Khezri [2021], Razian et al. [2022] emphasize a significant limitation in existing approaches addressing the QoS-aware service composition (QSC) problem within IoT environments. The prevalent trend in these approaches involves the consideration of Quality of Service (QoS) parameters with deterministic values Khanouche et al. [2020a], Sefati and Navimipour [2021], Chai et al. [2021], Asghari et al. [2020]. However, this approach is deemed impractical due to the dynamic nature of QoS values in IoT services. These values are subject to constant changes influenced by variations in the topological structure of IoT networks, the mobility of IoT devices, system congestion, and economic policies Razian et al. [2020]. Consequently, the primary motivation behind this study is to address the QoS-aware IoT service composition problem while taking into account the inherent uncertainty associated with QoS parameters.

Addressing the Quality of Service-aware service composition (QSC) problem under uncertain QoS properties requires the consideration of various uncertainty representation models for QoS parameters Razian et al. [2022], Masdari and Khezri [2021]. These models encompass (1) Interval numbers Jian et al. [2016], Khababa et al. [2022], (2) Fuzzy numbers Xu et al. [2018], Zhang et al. [2019b], Seghir and Khababa [2021a], (3) Probabilistic models Zheng et al. [2016], (4) Machine learning-based methods, and (5) Service recommendation approaches. Researchers have leveraged these models to articulate the ambiguity inherent in QoS values. In this study, we specifically adopt the

1. http ://lpsolve.sourceforge.net/5.5/
2. https ://www.ibm.com/analytics/cplex-optimizer

generalized trapezoidal fuzzy number (GTrFN) due to its perceived versatility compared to interval numbers and its simplicity in contrast to probabilistic functions, machine learning-based methods, and service recommendation approaches. The primary contributions of our research are outlined below :

1. First, we propose a fuzzy constrained optimization model for the IoT-service composition problem, considering Generalized Trapezoidal Fuzzy Number (GTrFN)-based QoS parameters. This model, denoted as QSCFIoT, addresses the uncertainty in QoS values.
2. Second, to solve the QSCFIoT problem, we introduce an approach consisting of two modules. The first module employs a fuzzy extension of the deterministic Skyline operator Borzsony et al. [2001] to reduce the search space of QSCFIoT. The second module utilizes an improved discrete flower pollination algorithm (IDFPA) along with an effective best solution improvement method. This two-module approach allows for the efficient search for near-optimal Composite of IoT-Services (CS) in the reduced search space. Recognizing QSCFIoT as a combinatorial optimization problem, we incorporate two innovative processes within IDFPA : discrete global and local pollination processes. These processes update the positions of solutions, contributing to the algorithm's exploration capabilities. IDFPA employs a "discard abandoned solutions mechanism" to enhance exploration during the search. This mechanism regenerates new positions for stagnated CSs that have not updated their positions after a predetermined number of iterations.
3. Finally, the performance and efficiency of the proposed approach are validated in solving QSCFIoT with different scales over using fuzzy versions of the QWS Al-Masri and Mahmoud [2008] and a synthetic datasets, where IDFPA is compared to other bio-inspired-based QSC algorithms including extended flower pollination algorithm (EFPA) Zhang et al. [2019b], particle swarm optimization (PSO) algorithm Wang et al. [2013] and a recently proposed improved teaching learning-based QoS-aware services composition algorithm (ITL-QCA) Khanouche et al. [2020a].

The remaining sections of this paper are organized as follows :

— Section 1 : Fuzzy Constrained Optimization Model for QSCFIoT : This section offers a comprehensive explanation of the fuzzy constrained optimization model tailored to tackle the QoS-aware service composition problem in the context of the Internet of Things (QSCFIoT).

— Section 2 : Basic Flower Pollination Algorithm (FPA) : Here, we introduce the basic flower pollination algorithm (FPA) as the foundational optimization algorithm upon which our proposed solution approach builds.

— Section 3 : Proposed Solution Approach : In this section, we present our comprehensive solution approach, which involves the integration of a fuzzy Skyline-based module and an improved discrete flower pollination algorithm (IDFPA) for effectively tackling the QSCFIoT problem.

— Section 4 : Experimental Results : The performance and efficiency of the proposed approach are thoroughly validated through a series of comparison experiments, addressing QSCFIoT with varying scales.

— Section 5 : Conclusion and Future Directions : The paper concludes in this section, summarizing the research findings and providing insights into potential directions for future work.

1 Fuzzy constrained optimization model of QSCFIoT

This section is dedicated to formulating the fuzzy constrained optimization model for QSCFIoT. To facilitate the mathematical representation of QSCFIoT, the following definitions are presented before delving into the detailed formulation of the problem.

Definition 19 *(IoT-service) refers to a software entity offered by an IoT device, which serves as the core component of IoT technology. In this technological paradigm, smart objects (IoT devices) can provide their published services, encompassing functions and operations, to end-users via the internet.*

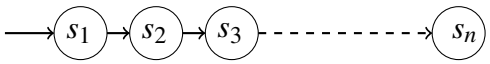
Definition 20 *(IoT-service properties) an IoT service denoted as S through a 2-tuple notation, expressed as* $S = (F, NF)$*. Here,* F *represents the functional properties, encompassing input and output attributes that define the functions and operations provided by the IoT service. On the other hand,* NF *denotes the non-functional properties, commonly referred to as Quality of Service (QoS) attributes. These QoS attributes include factors such as price, response time, availability, and throughput, providing a comprehensive description of the service quality.*

The QoS attributes are further categorized into two classes : QoS^+ *and* QoS^-*, representing the benefit and cost criteria, respectively. Higher values for* QoS^+ *attributes (e.g., availability and throughput) indicate superior performance, while lower values for* QoS^- *attributes (e.g., response time and price) suggest enhanced service quality. Conversely, lower (higher) values for* QoS^+ *(*QoS^-*) are associated with poorer service quality. This formal representation establishes a clear framework for understanding and characterizing IoT services based on their functional and non-functional attributes.*

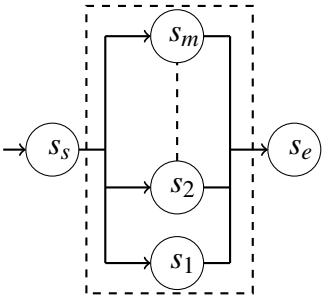
Definition 21 *(IoT-services class) An IoT-services class, denoted as* $C = S_1, S_2, \ldots, S_m$*, refers to a collection of m IoT services sharing similar functionalities. In other words, these services within the class have identical functional properties, signifying common functions and operations. However, despite their shared functionalities, each IoT service within the class exhibits noticeable differences in their Quality of Service (QoS) values. These distinctions in QoS values contribute to the unique characteristics and performance attributes of individual services within the class, reflecting the diverse quality aspects that each service can provide to end-users.*

Definition 22 *An abstract composite of IoT-services, represented by* $ACS = C_1, C_2, \ldots, C_n$*, serves to outline the required IoT-service classes, each contributing to the fulfillment of a complex user request. The specific instance of this abstract composite, denoted as* $CS = (S^1 i1, S^2 i2, \ldots, S^n in)$*, is termed a concrete composite of IoT-services. In this concrete composite, each* $S^j ij$*, where* $j \in 1, 2, \ldots, n$ *and* $i_j \in 1, 2, \ldots, m_j$*, refers to an IoT-service selected from the* j^{th} *IoT-service class* $C_j = S^j 1, S^j 2, \ldots, S^j mj$

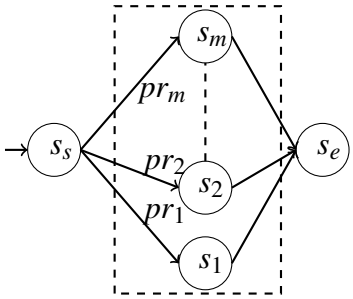
comprising m_j functionally equivalent IoT-services. The assembly of these n selected atomic IoT services follows various composition patterns, such as sequential, parallel, branch, and loop structures, as illustrated in Fig. 5.1. These composition patterns, including sequential, parallel, branch, and loop structures, are commonly employed in constructing concrete composites (CSs) Seghir and Khababa [2021a].



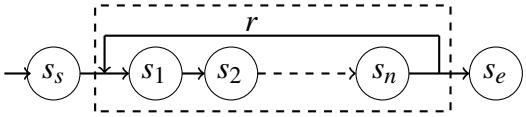
(a) Sequential composition of *n Ss*



(b) Parallel composition of *m Ss*



(c) Branch composition of *m Ss* with their execution probabilities pr_is



(d) Loop composition of *n Ss* with *p* calls

FIGURE 5.1 The frequently used composition structures to construct *CSs* Seghir and Khababa [2021a]

1.1 Mathematical formulation of QSCFIoT

The uncertainty in the QoS values of IoT services arises from dynamic factors such as changes in the topological structure of IoT networks, mobility of IoT devices, congestion in IoT systems, and economic policies Razian et al. [2020]. To capture this uncertainty, the Generalized Trapezoidal Fuzzy Number (GTrFN) model is employed due to its effectiveness in representing uncertain parameters Chen and Chen [2003].

In this study, we focus on four widely used QoS attributes in QSC problem-solving Alrifai et al. [2012] : two beneficial QoS attributes (availability (q_{avail}) and throughput (q_{thrpt})) and two cost QoS parameters (response time (q_{time}) and price (q_{price})). The fuzzy constrained optimization model for QSCFIoT is formulated as follows :

TABLE 5.1 Fuzzy QoS aggregation functions for evaluating global fuzzy QoS values of a *CS* using different composition structures

QoS parameter	***n* sequential IoT services**	***m* parallel IoT services**	**branch of *m* IoT-services with their pr_is probabilities**[a]	**Loop : call an IoT-service *S* with *p* times**
q_{price}	$\tilde{\sum}_{i=1}^{n} \tilde{S}_{i,q_{price}}$	$\tilde{\sum}_{i=1}^{m} \tilde{S}_{i,q_{price}}$	$\tilde{\text{Max}}_{i=1}^{m}\{\tilde{S}_{i,q_{price}}\}$	$p \otimes \tilde{S}_{i,q_{price}}$
q_{time}	$\tilde{\sum}_{i=1}^{n} \tilde{S}_{i,q_{time}}$	$\tilde{\text{Max}}_{i=1}^{m}\{\tilde{S}_{i,q_{time}}\}$	$\tilde{\text{Max}}_{i=1}^{m}\{\tilde{S}_{i,q_{time}}\}$	$p \otimes \tilde{S}_{i,q_{time}}$
q_{avail}	$\tilde{\prod}_{i=1}^{n} \tilde{S}_{i,q_{avail}}$	$\tilde{\prod}_{i=1}^{m} \tilde{S}_{i,q_{avail}}$	$\tilde{\text{Min}}_{i=1}^{m}\{\tilde{S}_{i,q_{avail}}\}$	$\tilde{\prod}_{i=1}^{p} \tilde{S}_{i,q_{avail}}$
q_{thrpt}	$\tilde{\text{Min}}_{i=1}^{n}\{\tilde{S}_{i,q_{thrpt}}\}$	$\tilde{\text{Min}}_{i=1}^{n}\{\tilde{S}_{i,q_{thrpt}}\}$	$\tilde{\text{Min}}_{i=1}^{m}\{\tilde{S}_{i,q_{thrpt}}\}$	$\tilde{S}_{i,q_{thrpt}}$

[a] Depending on the executions probabilities pr_is of S_is, only one IoT-service among the S_is ones is selected to be executed for a *CS*.

Identify the best *CS* among all the possible ones [3] that :

$$\tilde{\text{Max}}\ \tilde{U}(CS) = \Big((w_{q_{time}} \otimes \overline{\tilde{CS}_{q_{time}}}) \oplus (w_{q_{price}} \otimes \overline{\tilde{CS}_{q_{price}}}) \\ \oplus (w_{q_{avail}} \otimes \overline{\tilde{CS}_{q_{avail}}}) \oplus (w_{q_{thrpt}} \otimes \overline{\tilde{CS}_{q_{thrpt}}})\Big) \tag{5.1}$$

Subject to the following global QoS constraints of end-users :

— Fuzzy benefit QoS constraints

$$\tilde{CS}_{q_{avail}} \geq_{max} \tilde{Cst}_{q_{avail}}, \tilde{CS}_{q_{thrpt}} \geq_{max} \tilde{Cst}_{q_{thrpt}} \tag{5.2}$$

— Fuzzy cost QoS constraints

$$\tilde{CS}_{q_{time}} \leq_{min} \tilde{Cst}_{q_{time}}, \tilde{CS}_{q_{price}} \leq_{min} \tilde{Cst}_{q_{price}} \tag{5.3}$$

where :

— $\tilde{CS}_{q_b}$ and $\tilde{CS}_{q_c}$, with $q_b \in \{q_{avail}, q_{thrpt}\}$ and $q_c \in \{q_{time}, q_{price}\}$, are the global fuzzy benefit and cost QoS values of *CS*, respectively.

— $\tilde{Cst}_{q_b}$ and $\tilde{Cst}_{q_c}$ are the fuzzy benefit and cost QoS requirements of users, respectively.

— As illustrated in Table 5.1, depending on **(1)** The used fuzzy QoS aggregation function, i.e., $\tilde{\sum}$,$\tilde{\prod}$,$\tilde{\text{Min}}$ or $\tilde{\text{Max}}$, **(2)** The type of the composition structure, i.e., sequential, parallel, branch or loop, and **(3)** The considered benefit or cost QoS parameter ;

— the $\tilde{CS}_{q_b}$ and $\tilde{CS}_{q_c}$ are evaluated by aggregating the fuzzy benefit $\tilde{S}_{q_b} = (S^1_{q_b}, S^2_{q_b}, S^3_{q_b}, S^4_{q_b}; S^w_{q_b})$ and cost $\tilde{S}_{q_c} = (S^1_{q_c}, S^2_{q_c}, S^3_{q_c}, S^4_{q_c}; S^w_{q_c})$ QoS values of the assembled IoT-services *S*s of *CS*.

— $\tilde{U}(CS)$ is the fuzzy utility function of *CS* that melts its fuzzy benefit $\tilde{CS}_{q_b}$ and cost $\tilde{CS}_{q_c}$ values into a single fuzzy value.

The afore-used function adapts the *Simple Additive Weighting* (SAW) method Rao [2007] by scaling the fuzzy values $\tilde{CS}_{q_b}$ and $\tilde{CS}_{q_c}$ into their normalized fuzzy ones $\overline{\tilde{CS}_{q_b}}$ and $\overline{\tilde{CS}_{q_c}}$, respectively.

Subsequently, the normalized fuzzy values are subjected to weighting and summation using the benefit weight (w_{q_b}) and the cost weight (w_{q_c}), where $w_{q_c} \in [0,1]$, $w_{q_b} \in [0,1]$, and $w_{q_{time}} + w_{q_{price}} + w_{q_{thrpt}} + w_{q_{avail}} = 1$. This approach emphasizes the significance and priority assigned to each QoS parameter by the user. The normalized fuzzy values $\overline{\tilde{CS}_{q_b}}$ and $\overline{\tilde{CS}_{q_c}}$ of their related original ones $\tilde{CS}_{q_b} = (CS^1_{q_b}, CS^2_{q_b}, CS^3_{q_b}, CS^4_{q_b}; CS^w_{q_b})$ and $\tilde{CS}_{q_c} = (CS^1_{q_c}, CS^2_{q_c}, CS^3_{q_c}, CS^4_{q_c}; CS^w_{q_c})$ are calculated as given in the following fuzzy benefit and cost QoS parameter normalization Equations 5.4 and 5.5, respectively.

3. For instance, if an *ACS* has *n* IoT-services classes, where each one has *m* functionally similar IoT-services, then m^n different *CS*s can be instantiated

— Fuzzy benefit QoS parameter normalization

$$\overline{\tilde{CS}_{q_b}} = \begin{cases} \left(\frac{CS^1_{q_b} - min^1_{q_b}}{max^4_{q_b} - min^1_{q_b}}, \frac{CS^2_{q_b} - min^1_{q_b}}{max^4_{q_b} - min^1_{q_b}}, \frac{CS^3_{q_b} - min^1_{q_b}}{max^4_{q_b} - min^1_{q_b}}, \right. \\ \left. \frac{CS^4_{q_b} - min^1_{q_b}}{max^4_{q_b} - min^1_{q_b}}; CS^w_{q_b} \right) \text{ if } max^4_{q_b} \neq min^1_{q_b} \\ \left(1,1,1,1; CS^w_{q_b}\right) \text{ if } max^4_{q_b} = min^1_{q_b} \end{cases} \tag{5.4}$$

— Fuzzy cost QoS parameter normalization

$$\overline{\tilde{CS}_{q_c}} = \begin{cases} \left(\frac{max^4_{q_c} - CS^4_{q_c}}{max^4_{q_c} - min^1_{q_c}}, \frac{max^4_{q_c} - CS^3_{q_c}}{max^4_{q_c} - min^1_{q_c}}, \frac{max^4_{q_c} - CS^2_{q_c}}{max^4_{q_c} - min^1_{q_c}}, \right. \\ \left. \frac{max^4_{q_c} - CS^1_{q_c}}{max^4_{q_c} - min^1_{q_c}}; CS^w_{q_c} \right) \text{ if } max^4_{q_c} \neq min^1_{q_c} \\ \left(1,1,1,1; CS^w_{q_c}\right) \text{ if } max^4_{q_c} = min^1_{q_c} \end{cases} \tag{5.5}$$

Where :

$min^1_{q_t}$ and $max^4_{q_t}$ are the crisp QoS limit values of $ACS = (C_1, C_2, \ldots, C_n)$ for each considered QoS parameter q_t, with $\forall j \in \{1,2\ldots,n\}$, $C_j = \{S^j_1, S^j_2, \ldots, S^j_{m_j}\}$, and $\forall i \in \{1,2\ldots,m_j\}$, $\tilde{S}^j_{i,q_t} = (S^{j,1}_{i,q_t}, S^{j,2}_{i,q_t}, S^{j,3}_{i,q_t}, S^{j,4}_{i,q_t}; S^{j,w}_{i,q_t})$. $min^1_{q_t}$ and $max^4_{q_t}$ are evaluated as follows :

$$min^1_{q_t} = Agg_{q_t}{}^n_{j=1}\left(\text{Min}^{m_j}_{i=1}\{S^{j,1}_{i,q_t}\}\right) \tag{5.6}$$

$$max^4_{q_t} = Agg_{q_t}{}^n_{j=1}\left(\text{Max}^{m_j}_{i=1}\{S^{j,4}_{i,q_t}\}\right) \tag{5.7}$$

where :

— $Agg_{q_t}{}^n_{j=1}$ is the associated crisp QoS aggregation function of the q_t parameter as described in Table 5.1, (i.e., $\sum$, $\prod$,Min and Max), it was applied to aggregate the n obtained real values $\text{Max}\{S^{j,4}_{i,q_t}\}/\text{Min}\{S^{j,1}_{i,q_t}\}$ from each IoT-services class C_j of m_j functionally equivalent IoT-services.

An illustrative example will show how the fuzzy utility value of a CS is calculated. Let's consider, for example, a sequential composition structure of 3 IoT-services classes $C_{i=1,2,3}$ per 3 functionally equivalent IoT-services $S^{j=1,2,3}_i$, where each S^j_i has two cost QoS criterion : the response time q_{time} (ms) and the price q_{price} ($) attributes, which are assigned by the $w_{q_{time}} = 0.5$ and the $w_{q_{price}} = 0.5$ cost-weights, respectively.

The fuzzy QoS values of these nine atomic IoT-services in the considered q_{time} and q_{price} parameters are shown in Tables 5.2 and 5.3, respectively. Using the formulas of Equations 5.6 and 5.7, it is possible to evaluate the $min^1_{q_{time}}$, $max^4_{q_{time}}$, $min^1_{q_{price}}$ and $max^4_{q_{price}}$ as follows : $min^1_{q_{time}} =$

$\sum_{q_{time}\,j=1}^{3}\left(\text{Min}_{i=1}^{3}\{S_{i,q_{time}}^{j,1}\}\right)=1+1+1=3$, $max_{q_{time}}^{4}=\sum_{q_{time}\,j=1}^{3}\left(\text{Max}_{i=1}^{3}\{S_{i,q_{time}}^{j,4}\}\right)=6+5+6=17$, $min_{q_{price}}^{1}=\sum_{q_{price}\,j=1}^{3}\left(\text{Min}_{i=1}^{3}\{S_{i,q_{price}}^{j,1}\}\right)=1+1+1=3$ and

$max_{q_{price}}^{4}=\sum_{q_{price}\,j=1}^{3}\left(\text{Max}_{i=1}^{3}\{S_{i,q_{price}}^{j,4}\}\right)=5+5+5=15$

In the illustrated example, let's give $CS'=(S_1^1,S_3^2,S_1^3)$ as an instance of an implemented composite of IoT-services, its $\tilde{CS}'_{q_{time}}$ and $\tilde{CS}'_{q_{price}}$ can be calculated as demonstrated in Table 5.1 using the fuzzy aggregation function $\tilde{\sum}$ for a sequential composition structure of three atomic IoT-services as follows :

$\tilde{CS}'_{q_{time}}=(1,2,3,4;0.9)\oplus(1,2,2,3;0.9)\oplus(3,4,5,6;0.9)=(1+1+3,2+2+4,3+2+5,4+3+6;min(0.9,0.9,0.9))=(5,8,10,13;0.9)$ and

$\tilde{CS}'_{q_{price}}=(2,3,3,5;0.8)\oplus(3,4,5,5;0.9)\oplus(1,2,2,3;0.9)=(2+3+1,3+4+2,3+5+2,5+5+3;min(0.8,0.9,0.9))=(6,9,10,13;0.8)$.

Using Equation 5.4, the normalized fuzzy values $\overline{\tilde{CS}'_{q_{time}}}$ and $\overline{\tilde{CS}'_{q_{price}}}$ of $\tilde{CS}_{q_{time}}$ and $\tilde{CS}_{q_{price}}$, respectively are calculated as given in the following :

$$\begin{cases}\overline{\tilde{CS}'_{q_{time}}}=\left(\frac{5-3}{17-3},\frac{8-3}{17-3},\frac{10-3}{17-3},\frac{13-3}{17-3};0.9\right)=(0.14,0.36,0.5,0.71;0.9)\\ \overline{\tilde{CS}'_{q_{price}}}=\left(\frac{6-3}{15-3},\frac{9-3}{15-3},\frac{10-3}{15-3},\frac{13-3}{15-3};0.8\right)=(0.25,0.5,0.58,0.83;0.8)\end{cases}$$

Finally, using the above $\overline{\tilde{CS}'_{q_{time}}}$ and $\overline{\tilde{CS}'_{q_{price}}}$ fuzzy values, the fuzzy utility value of CS' can be calculated by Equation 5.1 as $\tilde{U}(CS')=\left(w_{q_{time}}\otimes\overline{\tilde{CS}'_{q_{time}}}\right)\oplus\left(w_{q_{price}}\otimes\overline{\tilde{CS}'_{q_{price}}}\right)=\left(0.5\otimes(0.14,0.36,0.5,0.71;0.9)\right)\oplus\left(0.5\otimes(0.25,0.5,0.58,0.83;0.8)\right)=(0.19,0.43,0.54,0.77;0.8)$.

TABLE 5.2 An example of nine atomic IoT-services assessed on the q_{time} parameter

C_1	C_2	C_3
$\tilde{S}_{1,q_{time}}^{1}=(1,2,3,4;0.9)$	$\tilde{S}_{1,q_{time}}^{2}=(2,2,3,4;0.9)$	$\tilde{S}_{1,q_{time}}^{3}=(3,4,5,6;0.9)$
$\tilde{S}_{2,q_{time}}^{1}=(2,3,4,5;0.8)$	$\tilde{S}_{2,q_{time}}^{2}=(1,3,4,5;0.8)$	$\tilde{S}_{2,q_{time}}^{3}=(1,2,4,5;0.8)$
$\tilde{S}_{3,q_{time}}^{1}=(3,4,5,6;0.9)$	$\tilde{S}_{3,q_{time}}^{2}=(1,2,2,3;0.9)$	$\tilde{S}_{3,q_{time}}^{3}=(2,3,3,5;0.8)$

TABLE 5.3 An example of nine atomic IoT-services assessed on the q_{price} parameter

C_1	C_2	C_3
$\tilde{S}_{1,q_{price}}^{1}=(2,3,3,5;0.8)$	$\tilde{S}_{1,q_{price}}^{2}=(1,2,2,3;0.9)$	$\tilde{S}_{1,q_{price}}^{3}=(1,2,2,3;0.9)$
$\tilde{S}_{2,q_{price}}^{1}=(1,2,3,4;0.8)$	$\tilde{S}_{2,q_{price}}^{2}=(2,3,3,4;0.8)$	$\tilde{S}_{2,q_{price}}^{3}=(2,3,3,4;0.8)$
$\tilde{S}_{3,q_{price}}^{1}=(2,2,3,4;0.9)$	$\tilde{S}_{3,q_{price}}^{2}=(3,4,5,5;0.9)$	$\tilde{S}_{3,q_{price}}^{3}=(3,4,4,5;0.9)$

2 Basic Flower Pollination Algorithm

The Flower Pollination Algorithm (FPA) is considered as a nature-inspired intelligent optimization algorithm which simulates the flower pollination behavior to solve global optimization problems Yang [2012]. It has been proven as an effective algorithm to handle the QSC problem under precise Wu and

Tan [2021] or ambiguous Zhang et al. [2019b] QoS parameters. FPA searches for the optimal solutions of an optimization problem using the abiotic and self-pollination, and the biotic and cross-pollination mechanisms, where the switch probability parameter $P \in [0,1]$ is used by FPA to alter between its global (the biotic and cross-pollination) and local (the abiotic and self-pollination) search processes of optimal solutions. In the flower pollination, the abiotic and self-pollination processes can intervene more than the biotic and cross-pollination. Therefore, in order to perform the local search more than the global one in the pollination processes of FPA, Yang [2012] suggested that the switch probability P to be set with a value of 0.8.

2.1 Global pollination

In the biotic and cross-pollination process, pollinators such as insects and birds who can move and fly over long distances, are employed to transport flower pollen gametes.

Mathematically, the global pollination is given by Equation 5.8, where each pollen/solution x_i updates its old position x_i^t by a new one x_i^{t+1} using the Lévy flight behavior pollinators, which is mathematically formulated by the Lévy flight step size $L(\lambda)$ as illustrated in Equation 5.9.

$$x_i^{t+1} = x_i^t + \gamma L(\lambda)(x_i^t - g^*) \tag{5.8}$$

Where :

— $i = 1,2,\ldots,z$ and z is the number of pollens (population size of FPA),

— g^* is the position of the current best pollen found among the whole z solutions at the t^{th} iteration,

— $\gamma \in [0,1]$ is a scaling factor, which is used to control the step size of pollinators.

According to Yang [2012], γ was suggested to be set with a value of 0.1.

$$L \sim \frac{\lambda\Gamma(\lambda)\sin(\pi\lambda/2)}{\pi} * \frac{1}{S^{1+\lambda}}, (S >> S_0 > 0) \tag{5.9}$$

Where :

— $\Gamma(\lambda)$ is the standard gamma function,

— S is the step size,

— S_0 is the smallest step size

— λ is the distribution factor.

Referring to Yang [2012], S_0 and λ were recommended to be set to 0.1 and 1.5, respectively.

2.2 Local pollination

The mathematical formulation of the abiotic and self-pollination process to update the old pollen position x_i^t of a given solution x_i by a new one x_i^{t+1} is defined by Equation 5.10

$$x_i^{t+1} = x_i^t + U(x_p^t - x_k^t) \tag{5.10}$$

Where :

— x_p^t and x_k^t represent the positions of two different pollens that have been randomly selected from the same plant, i.e., the current population of FPA,

— U is a random generated number drawn from a uniform distribution in [0,1].

For each iteration cycle $t = 1, 2, \ldots maxItr$, where $maxItr$ is the maximum iteration number of FPA, for both the local and global pollinations, x_i^t is replaced by x_i^{t+1}, if the latter is better than x_i^t in terms of optimality value to solve the global optimization problem ; otherwise x_i^{t+1} is discarded and x_i^t is maintained for the $(t+1)^{th}$ iteration. The pseudo-code of the basic FPA can be referred to in Algorithm 2

3 Proposed solution approach for QSCFIoT

The proposed solution approach for QSCFIoT comprises two main modules. The first module utilizes a fuzzy extended version of the Skyline operator to narrow down the search space of QSCFIoT. In this module, only the non-dominated IoT services from each IoT services class are selected as candidate services to construct the Composite of IoT Services (*CS*) for end-user requests.

The second module involves an improved discrete Flower Pollination Algorithm (FPA) with an effective best solution improvement method, termed IDFPA. This module is responsible for efficiently searching for the final *CS* solutions of QSCFIoT, leveraging the reduced search space obtained from the first module.

3.1 Skyline IoT-Services

Upon careful consideration, the objective of this study is to acquire a near-optimal Composite of IoT Services (*CS*) solution. This solution aims to maximize the fuzzy utility function, as articulated in Equation 5.1, while simultaneously adhering to the fuzzy benefit and cost overall Quality of Service (QoS) constraints outlined in Equations 5.2 and 5.3, respectively.

A Composite of IoT Services (*CS*) solution is formulated by assembling a set of IoT services, with each service selected from its respective class. However, not all IoT service partners within each class are considered potential candidates for constructing *CS*s. Therefore, certain studies Alrifai et al. [2010], Wang et al. [2013], Ying and Jiande [2020] have employed the Skyline operator Borzsony et al. [2001] to streamline the QoS-aware service composition (QSC) process. The Skyline operator is

Algorithm 2 Basic Flower Pollination Algorithm.

Initialize a population of z pollens $\{x_1, x_2, \ldots, x_z\}$
Evaluate each pollen (solution) of the initialized population using the optimization function of the solved problem
Identify the best pollen g^* in the initialized population
for $t = 1, 2, \ldots maxItr$ **do**
    for $i = 1, 2, \ldots, z$ **do**
        if $rand \leq P$ **then**
            Calculate the Lévy flight step size $L(\lambda)$ using $L \sim \frac{\lambda\Gamma(\lambda)\sin(\pi\lambda/2)}{\pi} * \frac{1}{S^{1+\lambda}}, (S >> S_0 > 0)$
            Perform the global pollination by $x_i^{t+1} = x_i^t + \gamma L(\lambda)(x_i^t - g^*)$
        else
            Perform the local pollination by $x_i^{t+1} = x_i^t + U(x_j^t - x_k^t)$
        end if
        Evaluate x_i using its new position x_i^{t+1}
        if x_i^{t+1} is better than x_i^t **then**
            update x_i^t with the new position x_i^{t+1}
        else
            Discard x_i^{t+1} and maintain the old position x_i^t of x_i for the $(t+1)^{th}$ iteration
        end if
    end for
    Find the current best pollen g^* in the current population
end for
    return the best solution g^*

utilized to eliminate candidate services that may not contribute significantly to *CS* construction, as they are dominated by some of their functionally equivalent counterparts.

A challenge surfaced in the aforementioned Skyline-based studies is the consideration of QoS parameters with crisp values. This approach is not realistic for dynamic IoT environments, where addressing the QoS uncertainty becomes crucial in solving the QoS-aware Service Composition for IoT (QSCFIoT) problem, particularly due to challenges posed by IoT architecture Razian et al. [2020]. To address this limitation, we propose fuzzy extended versions of the crisp *Service Dominance Operator* and *Skyline Service* Alrifai et al. [2010], Wang et al. [2013], Ying and Jiande [2020]. These extended versions incorporate the Generalized Trapezoidal Fuzzy Number (GTrFN) model for representing QoS properties, as defined in the following two expressions :

Definition 23 *(Fuzzy Service Dominance Operator :* $\tilde{\prec}$*) Let's consider an IoT-services class C and two IoT-services* S_1*,* S_2 *that belong to C, where each one has a set of QoS parameters* q_t*s. We say* S_1 *fuzzy dominates* S_2*, represented by* $S_1 \tilde{\prec} S_2$*, if and only if :* ***(1)*** $\forall q_t \in QoS^+ : \tilde{S}_{1,q_t} \geq_{\max} \tilde{S}_{2,q_t}$*,* ***(2)*** $\forall q_t \in QoS^- : \tilde{S}_{1,q_t} \leq_{\min} \tilde{S}_{2,q_t}$*, and* ***(3)*** $\exists q_t \in QoS^+, \tilde{S}_{1,q_t} >_{\max} \tilde{S}_{2,q_t}$ *or* $\exists q_t \in QoS^-, \tilde{S}_{1,q_t} <_{\min} \tilde{S}_{2,q_t}$*.*

Definition 24 *(Skyline IoT-Services) Considering a class* $C = \{S_1, S_2, \ldots S_m\}$ *of m functionally equivalent IoT-services. The Skyline of this latter, represented by SkC, gathers the candidates Ss from C that cannot be fuzzy dominated by any other S from C. i.e,* $SkS = \{S_i \in C \mid \nexists S_j \in C : S_j \tilde{\prec} S_i\}$*.*

Each IoT services class C is associated with a definition of its Skyline IoT services SkC, which undergoes the Skyline calculation process. This process involves conducting pair-wise comparisons between the fuzzy values $\tilde{S}qt$ of the compared IoT services S within class C. However, the computation time for the Skyline calculation process can be prohibitively expensive, particularly when class C encompasses a large number of functionally equivalent Ss. Furthermore, it's worth noting that the Skyline services calculation is conducted independently of any online user request for web services composition Alrifai et al. [2010]. As a result, this computation process is typically performed offline using existing efficient Skyline algorithms Borzsony et al. [2001].

For this study, the computation of each Skyline IoT services set SkC from its corresponding IoT services class C follows the well-known non-dominated method introduced by Deb et al. [2002]. This approach employs the fuzzy service dominance operator $\tilde{\prec}$, as defined in Definition 23, to establish dominance relationships among the fuzzy service values. The adapted algorithm is outlined in Algorithm 3.

3.2 Improved Discrete Flower Pollination Algorithm (IDFPA)

The pseudo-code for the proposed Improved Discrete Flower Pollination Algorithm (IDFPA) is outlined in Algorithm 4, encompassing six main procedures : (1) Population Initialization, (2) Population Evaluation, (3) Discrete Global Pollination, (4) Discrete Local Pollination, (5) Discard Abandoned Solutions Positions, and (6) Best Solution Improvement. While the first two procedures

Algorithm 3 Fuzzy non-dominated method to calculate the *SkC* from its associated IoT-services class *C*.

```
1: SkC = ∅ ▷ SkC : is the set of the non-dominated IoT-services (i.e., Skyline IoT-services)
   to be calculated from its associated IoT-services class C
2: for each IoT-service S ∈ C do
3:     N_S ← 0                          ▷ N_S : is the number of IoT-services that fuzzy dominate S
4:     for each IoT-service S' ∈ C and different to S do
5:         if S' ≾̃ S then               ▷ if S' fuzzy dominates S using Definition 23
6:             N_S ← N_S + 1
7:         end if
8:     end for
9:     if N_S = 0 then
10:        SkC ← SkC ∪ {S}
11:    end if
12: end for
        return SkC .
```

are elaborated within the algorithm, the subsequent subsections provide detailed descriptions of the remaining four procedures.

3.2.1 Discrete global and local pollination procedures

In the standard Flower Pollination Algorithm (FPA), solution positions are updated with continuous values. However, the QoS-aware Service Composition for the Internet of Things (QSCFIoT) poses a unique challenge as it is a discrete global optimization problem. In this context, each Composite of IoT-Services (*CS*) solution position, consisting of n atomic IoT-services, is encoded as an n-dimensional vector of integer values, represented as $CS = (CS_1, CS_2, \ldots, CS_n)$. Here, each value CS_j, with $j \in 1, 2, \ldots, n$, signifies the index number of an atomic IoT-service selected from its associated Skyline IoT-services class $SkC_j = S_j^1, S_j^2, \ldots, S_j^{m_j}$ containing m_j functionally equivalent IoT-services.

Given the discrete nature of the QSCFIoT problem, the original continuous pollination processes of FPA are adapted to discrete global and local pollinations. These adaptations, presented in the following subsections, facilitate the exploration of the discrete search space to effectively address the QSCFIoT problem.

Discrete global pollination : Deriving inspiration from the original continuous global pollination process in FPA (refer to Equation 6.13), the discrete global pollination in the proposed Improved Discrete Flower Pollination Algorithm (IDFPA) is executed as outlined below :

For a given current population pop_t of z solutions at the t^{th} iteration, each CS_i solution, with $i \in \{1, 2, \ldots, z\}$, updates its old position $CS_i^t = (CS_{i,1}^t, CS_{i,2}^t, \ldots, CS_{i,n}^t)$ by a new one

$CS_i^{t+1} = (CS_{i,1}^{t+1}, CS_{i,2}^{t+1}, \ldots, CS_{i,n}^{t+1})$ using so far the best-solution position $CS_{best}^t = (CS_{best,1}^t, CS_{best,2}^t, \ldots, CS_{best,n}^t)$ of pop_t, as described by the following steps :

Algorithm 4 Improved Discrete Flower Pollination Algorithm.

Population initialization : initialize a population of z solutions $\{CS_1, CS_2, \ldots, CS_z\}$, where each initial dimensional value $CS^0_{i,j}$ of each CS_i's initial position, denoted by $CS^0_i = (CS^0_{i,1}, CS^0_{i,2}, \ldots, CS^0_{i,n})$ with $i \in \{1,2,\ldots,z\}$, $j \in \{1,2,\ldots,n\}$ and n is the number of the Skyline IoT-services classes SkCs, is initialized with the index number $d \in \{1,2,\ldots m_j\}$ of a randomly selected IoT-service S^d_j from its $SkC_j = \{S^1_j, S^2_j, \ldots, S^{m_j}_j\}$ of m_j functionally equivalent IoT-services.

Population evaluation : for each CS_i, if its initial position CS^0_i is a feasible one, evaluate its fuzzy utility value $\tilde{U}(CS^0_i)$ by Equation 4.1. Otherwise, evaluate its global aggregated normalized QoS constraints violations $\tilde{CS}^0_{i,cst}$ by Equation 5.13.

Identify the initial best-solution position CS^0_{best} among the z generated ones of the initial population.

for $t = 0,1,2,\ldots maxItr$ **do**

  for $i = 1,2,\ldots,z$ **do**

    if $rand \leq P$ **then**

      Discrete global pollination : calculate the new solution position CS^{t+1}_i of CS_i using its current one CS^t_i and the so far best-solution position CS^t_{best}. (see Subsection 3.2.1).

    else

      Discrete local pollination : calculate the new position CS^{t+1}_i of CS_i using its current one CS^t_i and two different neighborhood solutions CS^t_p and CS^t_k selected randomly from the current population. (see Subsection 3.2.1).

    end if

    If CS^{t+1}_i is a feasible solution position, evaluate its fuzzy utility value $\tilde{U}(CS^{t+1}_i)$ using Equation 4.1. Otherwise, evaluate its global aggregated normalized QoS constraints violations $\tilde{CS}^{t+1}_{i,cst}$ using Equation 5.13.

    Decide if the current position CS^t_i of CS_i will be updated by its new calculated one CS^{t+1}_i (see Subsection 3.2.2).

  end for

  Discard abandoned solutions positions : update the position of each CS_i solution that does not change its current position after *Limit* iterations (see Subsection 3.2.3).

  Find the current best-solution position $CS^t_{curBest}$ of the current population.

  If $CS^t_{curBest}$ is better than CS^t_{best} update this latter by $CS^t_{curBest}$.

  Best solution improvement : improve CS^t_{best} by a new neighborhood one (see Subsection 3.2.4).

end for

return the best solution position of CS_{best}.

— Initialize the CS_i^{t+1} position with the same n-dimensional values of the old position CS_i^t, i.e., $\forall j = 1,2\ldots,n : CS_{i,j}^{t+1} \leftarrow CS_{i,j}^t$.

— Select randomly a dimension $j \in [1,n]$.

— If the j^{th} dimension value of CS_i^{t+1} is different than the one of CS_{best}^t, i.e., $CS_{i,j}^{t+1} \neq CS_{best,j}^t$, then update the $CS_{i,j}^{t+1}$ value by the $CS_{best,j}^t$ one. Otherwise, update the j^{th} dimension value of CS_i^{t+1} by a different integer value selected randomly from the range $[1,m_j]$, which represents the indexes values of the m_j IoT-services that belong to the j^{th} Skyline IoT-services class SkC_j.

Discrete local pollination Similar to the local pollination process of the basic FPA (as seen in Equation 5.10), in our proposed IDFPA, each CS_i solution updates its old position CS_i^t according to the positions CS_p^t and CS_k^t of two different neighborhood solutions, with $p \neq k \neq i$, selected randomly from the current population pop_t at the t^{th} iteration. The following steps describe the introduced discrete local pollination to update the old position $CS_i^t = (CS_{i,1}^t, CS_{i,2}^t, \ldots, CS_{i,n}^t)$ by a new one $CS_i^{t+1} = (CS_{i,1}^{t+1}, CS_{i,n}^{t+1}, \ldots, CS_{i,n}^{t+1})$ using the two different solutions positions CS_p^t and CS_k^t :

— Identify the best-solution position, denoted as $CS_b^t = (CS_{1,b}^t, CS_{2,b}^t, \ldots, CS_{n,b}^t)$, between the two selected ones CS_p^t and CS_k^t.

— Initialize the CS_i^{t+1} position with the same n-dimensional values of the old position CS_i^t, i.e., $\forall j = 1,2\ldots,n : CS_{i,j}^{t+1} \leftarrow CS_{i,j}^t$.

— Select randomly a dimension $j \in [1,n]$.

— If the j^{th} dimension value of CS_i^{t+1} is different than the one of CS_b^t, i.e., $CS_{i,j}^t \neq CS_{b,j}^t$, then update the $CS_{i,j}^{t+1}$ value by the $CS_{b,j}^t$ one. Otherwise, update the j^{th} dimension value of CS_i^{t+1} by a different integer value selected randomly from the range $[1,m_j]$, which represents the indexes values of the m_j IoT-services that belong to the j^{th} Skyline IoT-services class SkC_j.

3.2.2 Updating solutions positions decision

Following the appliance of the discrete local (global) pollination processes by IDFPA, and similar to the solutions positions update of the basic FPA, here in our proposed IDFPA, updating each old solution position CS_i^t by its new calculated one CS_i^{t+1}, at a given iteration cycle t, is decided according to the following adapted Deb's selection criteria Deb [2000].

— CS_i^t is replaced by CS_i^{t+1} if the latter is a feasible solution position and CS_i^t is an infeasible one, i.e., CS_i^{t+1} satisfies the global QoS requirements as given by the fuzzy benefit and cost QoS constraints of Equations 5.2 and 5.3, respectively ; whereas, at least one global QoS constraint of Equations 5.2 and 5.3 is not satisfied by CS_i^t.

— If CS_i^t is a feasible solution position and CS_i^{t+1} is an infeasible one, then CS_i^{t+1} is discarded and CS_i^t is maintained for the $(t+1)^{th}$ iteration.

— When CS_i^t and CS_i^{t+1} are both feasible solutions positions. If the fuzzy utility value of CS_i^t calculated by Equation 5.1 is higher than the one of CS_i^{t+1}, then the latter is discarded and CS_i^t is maintained for the $(t+1)^{th}$ iteration. Otherwise, CS_i^t is replaced by CS_i^{t+1}.

— If both CS_i^t and CS_i^{t+1}, are infeasible solutions positions, then the one with the higher aggregated normalized global QoS constraints violations is maintained for the $(t+1)^{th}$ iteration, i.e., if $\tilde{CS}_{i,cst}^{t+1} >_{\max} \tilde{CS}_{i,cst}^{t}$ where $\tilde{CS}_{i,cst}^{t+1}$ and $\tilde{CS}_{i,cst}^{t}$ are calculated as will be presented in Equation 5.13, then CS_i^t is replaced by CS_i^{t+1}. Otherwise, CS_i^{t+1} is rejected and CS_i^t is maintained.

Given any infeasible solution position ICS, the fuzzy benefit and cost constraint violations quantities of its violated fuzzy benefit $\tilde{Cst}_{q_b}$ and cost $\tilde{Cst}_{q_c}$ global QoS constraints, denoted respectively by $I\tilde{C}S_{q_b}^{cst}$ and $I\tilde{C}S_{q_c}^{cst}$, are evaluated as follows.

$$\forall q_b \in QoS^+ : \text{ if } \tilde{Cst}_{q_b} >_{\max} I\tilde{C}S_{q_b} \text{ then } I\tilde{C}S_{q_b}^{cst} = \tilde{Cst}_{q_b} \ominus I\tilde{C}S_{q_b} \tag{5.11}$$

$$\forall q_c \in QoS^- : \text{ if } \tilde{Cst}_{q_c} <_{\min} I\tilde{C}S_{q_c} \text{ then } I\tilde{C}S_{q_c}^{cst} = I\tilde{C}S_{q_c} \ominus \tilde{Cst}_{q_c} \tag{5.12}$$

To calculate the global aggregated normalized QoS constraints violations of ICS, denoted by $I\tilde{C}S_{cst}$, the SAW method Rao [2007] has been adapted and employed to aggregate the fuzzy values $I\tilde{C}S_{q_b}^{cst}$ and $I\tilde{C}S_{q_c}^{cst}$ into a single fuzzy value $I\tilde{C}S_{cst}$, as given by the following Equation.

$$I\tilde{C}S_{cst} = \frac{1}{V} \otimes \left(\tilde{\sum}_{q_b \in QoS^+} \overline{I\tilde{C}S_{q_b}^{cst}} \oplus \tilde{\sum}_{q_c \in QoS^-} \overline{I\tilde{C}S_{q_c}^{cst}} \right) \tag{5.13}$$

Where :

— V is the number of the violated global QoS constraints by ICS,

— $\overline{I\tilde{C}S_{q_b}^{cst}}$ and $\overline{I\tilde{C}S_{q_c}^{cst}}$ are the normalized fuzzy values of their associated original ones $I\tilde{C}S_{q_b}^{cst}$ and $I\tilde{C}S_{q_c}^{cst}$, respectively.

— $\overline{I\tilde{C}S_{q_b}^{cst}}$ and $\overline{I\tilde{C}S_{q_c}^{cst}}$ have been calculated following the same formula of the fuzzy cost QoS parameter normalization given by Equation 5.5, since the lower fuzzy values $I\tilde{C}S_{q_b}^{cst}$ and $I\tilde{C}S_{q_c}^{cst}$ of ICS are, the higher its aggregated normalized QoS constraints violations $I\tilde{C}S_{cst}$ is.

3.2.3 Discard abandoned solutions positions

To enhance the exploration search of IDFPA through escaping from local optimums, for each CS_i solution of a population pop_t at the t^{th} iteration having z solutions, with $i \in \{1,2,\ldots,z\}$, and which does not update its current position $CS_i^t = (CS_{i,1}^t, CS_{i,2}^t, \ldots, CS_{i,n}^t)$ after a predetermined number of iterations $Limit$, then CS_i^t is replaced by a new solution position $CS_i'^t = (CS_{i,1}'^t, CS_{i,2}'^t, \ldots, CS_{i,n}'^t)$, where each new dimension value $CS_{i,j}'^t$ of the CS_i solution is set to the index number $d \in \{1,2,\ldots,m_j\}$ of an IoT-service S_j^d that has been randomly selected from its associated Skyline IoT-services class

$SkC_j = \{S_j^1, S_j^2, \ldots, S_j^{m_j}\}$ of m_j functionally equivalent IoT-services. Accordingly, if the new position of the CS_i solution (i.e.,$CS_i'^t$) that replaces its stagnated old position (i.e., CS_i^t) is a feasible one, then the fuzzy utility value of $CS_i'^t$ is evaluated using Equation 5.1. Otherwise, the global aggregated normalized QoS constraints violations of this new $CS_i'^t$ position is calculated by Equation 5.13.

3.2.4 Best solution improvement

To accelerate the convergence rate of IDFPA in discovering its near-optimal solution, a neighborhood local search method is executed at each iteration t by IDFPA to enhance the best-solution position CS_{best}^t of the current population pop_t. The following steps describe this process :

— Generate L neighborhood solutions positions that have the same IoT-services as the ones of CS_{best}^t.

— For each generated neighborhood solution position, select randomly one of its IoT-services and update it by a different one from its Skyline IoT-services class.

— Evaluate the L updated neighborhood solutions positions and identify the best one, denoted as G_{best}^t, among them.

— If G_{best}^t is better than CS_{best}^t, then CS_{best}^t is replaced by G_{best}^t. Otherwise, the CS_{best}^t position is maintained for the $(t+1)^{th}$ iteration.

3.3 Computational complexity of the proposed approach

In order to solve the QSCFIoT problem, the proposed approach includes two modules (1) a fuzzy skyline-based module and (2) an improved discrete flower pollination algorithm (IDFPA). As the Skyline-services calculation is independent from any online user request of web services composition Alrifai et al. [2010]. Therefore, the utilization of this computation process is carried on offline by adapting the first step of the famous non-dominated sorting method Deb et al. [2002] which has in its worst case a time complexity of $\mathcal{O}(r*n*m^2)$, where r is the number of the considered QoS parameters, n is the number the used IoT-services to implement a CS solution using n IoT-services classes, and m is the number of the functionally equivalent IoT-services per each used IoT-services class. For the IDFPA illustrated in Algorithm 4, its time complexity is analyzed as follows : the time complexity of the *Population initialization* and the *Population evaluation* procedures, are respectively calculated as, $Comp_{InitPop} = \mathcal{O}(n*z)$ and $Comp_{EvalPop} = \mathcal{O}(n*r*z)$, where z is the population size (i.e., number of the performed CS solutions by IDFPA). Furthermore, in the *Discrete global/local pollination* procedure, n and $n*r$ computation times are needed to calculate the new position and evaluate the fuzzy utility value of each CS solution, respectively. Hence, the time complexity of the *Discrete global/local pollination* procedure applied to z solutions is $Comp_{GlobalPol/LocalPol} = \mathcal{O}((n*r+n)*z) = \mathcal{O}(n*r*z)$. In addition, the time complexity of the *Discard abandoned solutions positions* procedure for updating and evaluating a new position of an abandoned solution in each iteration cycle of IDFPA is $Comp_{Aband} = \mathcal{O}(n+n*r) = \mathcal{O}(n*r)$. Moreover, the time complexity of the *Best solution*

improvement procedure which needs a number of $n*r*l$ calculations to select the best solution among the l evaluated neighborhood ones per each IDFPA iteration is $Comp_{Impro} = \mathcal{O}(n*r*l)$. As a result, the overall time complexity of IDFPA under $maxItr$ iterations is $Comp_{InitPop} + Comp_{EvalPop} + maxItr*(Comp_{GlobalPol/LocalPol} + Comp_{Aband} + Comp_{Impro}) = \mathcal{O}\big(n*z+n*r*z+maxItr*(n*r*z+n*r+n*r*l)\big) = \mathcal{O}\big(maxItr*n*r*(z+l)\big)$.

4 Experimental results

In this section, the obtained performance results of the proposed IDFPA in solving QSCFIoT with different scales are compared to ones that were obtained by some other nature-inspired optimization algorithms, including EFPA Zhang et al. [2019b], PSO Wang et al. [2013] and ITL-QCA Khanouche et al. [2020a]. For coherent results compared to the originals ones, all the compared algorithms with their parameters setting as presented in Table 5.4 have been implemented with MATLAB R2016b under Windows 7 and performed on the same personal computer having an Intel(R) Core(TM) i5-4570 (3.20GHZ) and 4 GB RAM. The three used performance metrics to compare IDFPA, EFPA, PSO and ITL-QCA are described as follows :

— **Composition time** : measures the time taken by a specific algorithm to identify the optimal Composite of IoT-Services (CS) when solving an instance of the QoS-aware IoT Service Composition (QSCFIoT) problem with an experimental dataset. This metric is crucial for evaluating the efficiency and speed of an algorithm in addressing the composition problem. In experimental scenarios, the composition time serves as a key indicator, helping researchers gauge the algorithm's responsiveness and suitability for handling complex user requests and large datasets.

— **Composition optimality** : assesses the quality of the best feasible Composite of IoT-Services (CS_{best}) found by a given algorithm in terms of its fuzzy utility value for a solved instance of the QoS-aware IoT Service Composition (QSCFIoT) problem using an experimental dataset. The fuzzy utility value represents the overall goodness or desirability of the composed IoT-services, considering both benefit and cost criteria. It is a comprehensive measure that accounts for the trade-off between various QoS parameters, such as availability, throughput, response time, and price. The higher the fuzzy utility value of CS_{best}, the better the quality of the composed solution.

— **Composition stability** : quantifies the fuzzy variance ($v\tilde{a}r$) for a set of n fuzzy utility values of CS_{best} instances. These instances are obtained by a compared algorithm when solving an experimental dataset instance of the QoS-aware IoT Service Composition (QSCFIoT) problem for n independent running times. The fuzzy variance, as expressed in Equation 5.14, provides a measure of the variability or dispersion in the fuzzy utility values of the best-composed solutions across multiple runs. A lower fuzzy variance indicates greater stability, suggesting that the algorithm consistently produces solutions with similar fuzzy utility values under

different experimental conditions.

$$\tilde{var} = \frac{1}{n-1} \otimes \tilde{\sum}_{i=1}^{n} \left(\tilde{CS}_{i,best} \ominus \tilde{avg}\right) \otimes \left(\tilde{CS}_{i,best} \ominus \tilde{avg}\right) \tag{5.14}$$

, where $\tilde{CS}_{i,best} = \tilde{U}(CS_{i,best})$ is the i^{th} fuzzy utility value evaluated by Equation 5.1 of the i^{th} obtained $CS_{i,best}$ and $\tilde{avg} = \frac{1}{n} \otimes \tilde{\sum}_{i=1}^{n} \tilde{CS}_{i,best}$ is the average fuzzy value of the n calculated $\tilde{CS}_{i,best}$s.

TABLE 5.4 Parameters setting of the compared algorithms

Algorithm	Parameters setting
PSO Wang et al. [2013]	The accelerations factors $C1$ and $C2$ are set with a value of 2.0 and the inertia weight w is set to 0.8
EFPA Zhang et al. [2019b]	The step size of the adaptive switch probability δ is set to 0.2
ITL-QCA Khanouche et al. [2020a]	The learning probability P_c is set to 0.8 and the number of iterations to rearrange learners P_m is set to 80
Proposed IDFPA	The switch probability P is set to 0.8, the number of iterations for the abandoned solutions *Limit* is set to 80 and the number of neighborhood solutions positions L is set to 3
For all the compared algorithms, the population size is set to 40 and the maximum number of solutions evaluations is set to 50000	

4.1 Experimental datasets design

Following the experimental dataset design outlined in reference Khanouche et al. [2020a], we have generated various instances of the QoS-aware Service Composition for IoT (QSCFIoT) problem using two datasets : (1) QWS dataset and (2) Random dataset. The QWS dataset, sourced from Guelph University by Eyhab Al-Masri, comprises 2507 real web services with consideration for 09 QoS parameters Al-Masri and Mahmoud [2008]. Conversely, the random dataset is synthetically created to simulate a large-sized dataset of 250,000 IoT services, as the QWS dataset is relatively small. In line with Section 1, where the QSCFIoT problem is detailed, only four QoS parameters—availability (q_{avail}), throughput (q_{thrpt}), response time (q_{time}), and price (q_{price})—with identical QoS weights and priorities are considered to generate the fuzzy QoS values for each IoT service in both the QWS and Random datasets, as presented below :

— Fuzzy QoS values generation of the QWS dataset
For each web service $S_{i=\{1,2,...,2507\}} \in$ QWS :

$$\tilde{S}_{i,q_{avail}} = S_{i,q_{avail}} \otimes \tilde{R}, \tilde{S}_{i,q_{thrpt}} = S_{i,q_{thrpt}} \otimes \tilde{R},$$
$$\tilde{S}_{i,q_{time}} = S_{i,q_{time}} \otimes \tilde{R} \text{ and } \tilde{S}_{i,q_{price}} = rnd_{q_{price}} \otimes \tilde{R}$$

— Fuzzy QoS values generation of the Random dataset
For each simulated IoT-service $S_{i=\{1,2,...,250000\}}$:

$$\tilde{S}_{i,q_{avail}} = rnd_{q_{avail}} \otimes \tilde{R}, \tilde{S}_{i,q_{thrpt}} = rnd_{q_{thrpt}} \otimes \tilde{R},$$
$$\tilde{S}_{i,q_{time}} = rnd_{q_{time}} \otimes \tilde{R} \text{ and } \tilde{S}_{i,q_{price}} = rnd_{q_{price}} \otimes \tilde{R}$$

Where :

$S_{i,q_{avail}}$, $S_{i,q_{thrpt}}$ and $S_{i,q_{time}}$ are the crisp QoS values of the i^{th} web service from QWS in the q_{avail}, q_{thrpt} and q_{time} attributes, respectively. Since the QWS does not have the price parameter, hence each fuzzy QoS value $\tilde{S}_{i,q_{price}}$ of the QWS/Random datasets is generated using a random generated price value $rnd_{q_{price}}$ from the range $[2,5]$\$. Moreover, $rnd_{q_{avail}}$, $rnd_{q_{thrpt}}$ and $rnd_{q_{time}}$ are three random crisp QoS values that are drawn from the ranges $[0.70,0.90]$, $[10,90]$ and $[7,4]$, respectively, and have been used to generate the other fuzzy QoS values of the Random dataset (i.e., $\tilde{S}_{i,q_{avail}}$, $\tilde{S}_{i,q_{thrpt}}$ and $\tilde{S}_{i,q_{time}}$). $\tilde{R} = (0.8, r1, r2, 1.1; w)$ is a random generated GTrFN by the three real numbers $r1, r2$ and w that are randomly drawn from the interval $[0.9,1]$ with $r1 \leq r2$.

The experimental datasets instances of solving each abstract composite IoT-services ACS_n^m with sequential composition structure of n IoT-services classes per m functionally equivalent IoT-services using the QWS and the Random datasets are denoted by QWS_n^m and RND_n^m, respectively.

4.2 Performance comparisons

To assess the performance of IDFPA, a comprehensive statistical experiment is conducted, involving variations in the number of IoT-services classes (n) and the number of functionally equivalent IoT-services (m) within each class. The experiment comprises the solution of five small-sized QWS_n^m instances and five large-sized RND_n^m instances, with each algorithm undergoing 50 independent runs for each QWS_n^m/RND_n^m instance.

Given the stochastic nature of the algorithms under comparison, multiple runs are essential to capture the variability in their performance. By conducting 50 independent runs for each instance, the obtained results are subjected to a robust statistical analysis. The performance of the compared algorithms is then evaluated using the three aforementioned metrics for performance comparison. This rigorous experimental design ensures a comprehensive and reliable assessment of IDFPA across different scenarios, providing insights into its effectiveness under varying conditions.

4.2.1 Composition optimality and composition stability comparisons

The fuzzy average value, fuzzy best value, and fuzzy worst value of the fuzzy utility values for the obtained feasible CS_{best}s by the compared algorithms in solving each instance of the five QWS_n^ms and the five RND_n^ms per 50 independent running times are reported in Table 5.5 and Table 5.7, respectively. Additionally, the fuzzy variances of the obtained CS_{best}s per each QWS_n^m/RND_n^m by the compared algorithms are listed in Table 5.6 and Table 5.8, respectively.

Analyzing the results for the QWS_n^m instances in Table 5.5, IDFPA exhibits superior composition optimality compared to the other algorithms. Specifically, IDFPA achieves higher fuzzy best utility values in 3 out of 5 cases, higher fuzzy worst utility values in all 5 cases, and higher fuzzy average utility values in 4 out of 5 cases. Turning to the RND_n^m instances in Table 5.7, all fuzzy best/worst/average utility values of the obtained feasible CS_{best}s by IDFPA surpass those of the compared approaches.

Regarding composition stability, the fuzzy variances of IDFPA in solving the QWS_n^m instances (as shown in Table 5.6) outperform those of the compared algorithms. Similarly, in Table 5.8, which lists the composition stability results for the five RND_n^m instances, ITL-QCA exhibits the best fuzzy variances in 3 out of 5 cases, IDFPA achieves the best fuzzy variances in 2 out of 5 cases, and EFPA and PSO do not achieve the best fuzzy variances in any of the 5 cases.

Furthermore, the convergence curves associated with the average Mid-Supports of the fuzzy utility values of the obtained feasible CS_{best}s by the compared algorithms in solving the QWS_{25}^{100} and the RND_{30}^{8334} instances over 50 runs are depicted in Fig. 5.2.

For a fair comparison, each compared algorithm is executed to solve each of the two instances per run for a fixed time limit of 15 seconds. The convergence curves in Fig. 5.2 vividly demonstrate that IDFPA achieves a superior convergence rate compared to EFPA, PSO, and ITL-QCA algorithms. The notable improvement in convergence rate for IDFPA can be attributed to its integration of the "Best solution improvement" procedure, which retains improved *CS* solutions, and its utilization of the "Discard abandoned solutions positions" procedure, enhancing its exploration search for more desirable solutions.

It can be concluded from this comparison discussion that IDFPA have better composition optimality and convergence rate as well as higher stability than the compared algorithms for solving both the real and the synthetic fuzzy datasets instances.

TABLE 5.5 Composition optimality comparisons for five QWS_n^ms instance

QWS_n^m	Algorithm	Fuzzy best value	Fuzzy worst value	Fuzzy average value
QWS_5^{501}	IDFPA	**(0.5826,0.6952,0.7283,0.8777 ;0.9016)**	**(0.5834,0.6953,0.7152,0.8494 ;0.9116)**	(0.5833,0.6953,0.7167,0.8528 ;0.9016)
	EFPA	**(0.5826,0.6952,0.7283,0.8777 ;0.9016)**	(0.5862,0.6883,0.7104,0.8416 ;0.9116)	(0.5832,0.6938,0.7162,0.8539 ;0.9013)
	PSO	(0.5815,0.6960,0.7103,0.8501 ;0.9116)	(0.5490,0.6302,0.6541,0.7780 ;0.9031)	(0.5760,0.6671,0.6921,0.8216 ;0.9001)
	ITL-QCA	**(0.5826,0.6952,0.7283 ;0.8777 ;0.9016)**	(0.5978,0.6833,0.7064,0.8478 ;0.9034)	**(0.5839,0.6916,0.7202,0.8688 ;0.9016)**
QWS_{10}^{250}	IDFPA	**(0.5201,0.5835,0.6076,0.8053 ;0.9041)**	**(0.4887,0.5453,0.5804,0.7625 ;0.9044)**	**(0.5189,0.5820,0.6065,0.8036 ;0.9041)**
	EFPA	**(0.5201,0.5835,0.6076,0.8053 ;0.9041)**	(0.4888,0.5450,0.5772,0.7626 ;0.9044)	(0.5143,0.5779,0.6030,0.7963 ;0.9018)
	PSO	(0.4904,0.5572,0.5809,0.7575,0.9041)	(0.4500,0.5098,0.5340,0.6948 ;0.9008)	(0.4782,0.5370,0.5604,0.7168 ;0.9001)
	ITL-QCA	(0.5214,0.5772,0.6020,0.7729 ;0.9013)	(0.5061,0.5649,0.5824,0.7150 ;0.9019)	(0.5164,0.5730,0.5944,0.7370 ;0.9013)
QWS_{15}^{167}	IDFPA	(0.4982,0.5433,0.5645,0.7541 ;0.9001)	**(0.4811,0.5268,0.5418,0.7506 ;0.9063)**	**(0.4922,0.5367,0.5545,0.7586 ;0.9001)**
	EFPA	**(0.4990,0.5510,0.5670,0.7485 ;0.9001)**	(0.4781,0.5249,0.5409,0.7528 ;0.9063)	(0.4912,0.5366,0.5545,0.7542 ;0.9001)
	PSO	(0.4726,0.5149,0.5323,0.7058 ;0.9001)	(0.4356,0.4716,0.4865,0.6509 ;0.9001)	(0.4598,0.5001,0.5157,0.6554 ;0.9000)
	ITL-QCA	(0.5098,0.5534,0.5668,0.6826 ;0.9033)	(0.4868,0.5320,0.5430,0.6621 ;0.9001)	(0.4886,0.5330,0.5485,0.6923 ;0.9001)
QWS_{20}^{125}	IDFPA	**(0.4874,0.5277,0.5433,0.7244 ;0.9010)**	**(0.4870,0.5231,0.5394,0.7140 ;0.9010)**	**(0.4874,0.5258,0.5418,0.7200 ;0.9010)**
	EFPA	**(0.4874,0.5277,0.5433,0.7244 ;0.9010)**	(0.4408,0.4699,0.4849,0.6935 ;0.9014)	(0.4795,0.5153,0.5309,0.7114 ;0.9010)
	PSO	(0.4443,0.4788,0.4962,0.6769 ;0.9006)	(0.4381,0.4725,0.4831,0.5709 ;0.9006)	(0.4467,0.4784,0.4916,0.6093 ;0.9006)
	ITL-QCA	(0.5086,0.5479,0.5602,0.6299 ;0.9034)	(0.4616,0.4978,0.5087,0.6303 ;0.9042)	(0.4821,0.5197,0.5304,0.6223 ;0.9002)
QWS_{25}^{100}	IDFPA	(0.4787,0.5118,0.5239,0.6947 ;0.9006)	**(0.4717,0.5033,0.5135,0.6913 ;0.9006)**	**(0.4758,0.5082,0.5195,0.6933 ;0.9006)**
	EFPA	**(0.4811,0.5158,0.5286,0.6846 ;0.9015)**	(0.4487,0.4765,0.4883,0.6599 ;0.9006)	(0.4693,0.5009,0.5119,0.6837 ;0.9004)
	PSO	(0.4677,0.4989,0.5051,0.5646 ;0.9003)	(0.4333,0.4565,0.4617,0.4907 ;0.9003)	(0.4396,0.4650,0.4714,0.5307 ;0.9001)
	ITL-QCA	(0.4795,0.5142,0.5251,0.6049 ;0.9015)	(0.4614,0.4984,0.5051,0.5678 ;0.9017)	(0.4726,0.5049,0.5138,0.5814 ;0.9003)

For each QWS_n^m, the better fuzzy value among the ones of the compared algorithms is in bold.

TABLE 5.6 Composition stability comparisons for five QWS_n^ms instances

QWS_n^m	**Algorithm**	**Fuzzy variance**
QWS_5^{501}	IDFPA	**(-0.0741,-4.6962e^{-04},5.4591e^{-04},0.0758;0.9016)**
	EFPA	(-0.0748,-4.9972e^{-04},6.3488e^{-04},0.0768;0.9013)
	PSO	(-0.0615,-5.5436e^{-04},0.0011,0.0658;0.9001)
	ITL-QCA	(-0.0828,-8.0784e^{-04},0.0012,0.0855;0.9016)
QWS_{10}^{250}	IDFPA	**(-0.0827,-5.7861e^{-04},7.9728e^{-04},0.0844;0.9041)**
	EFPA	(-0.0810,-5.2021e^{-04},0.0012,0.0859,0.9018)
	PSO	(-0.0581,-3.6168e^{-04},0.0014,0.0632;0.9001)
	ITL-QCA	(-0.0498,-4.1254e^{-04},8.0159e^{-04},0.0532;0.9013)
QWS_{15}^{167}	IDFPA	**(-0,0724,-2.9728e^{-04},5.0072e^{-04},0.0743;0.9001)**
	EFPA	(-0.0706,-2.8371e^{-04},5.5548e^{-04},0.0733;0.9001)
	PSO	(-0.0391,-1.6165e^{-04},6.2049e^{-04},0.0428;0.9000)
	ITL-QCA	(-0.0424,-1.9645e^{-04},4.9341e^{-04},0.0450;0.9001)
QWS_{20}^{125}	IDFPA	**(-0.0552,-2.5551e^{-04},3.3119e^{-04},0.0564;0.9010)**
	EFPA	(-0.0548,-1.8979e^{-05},0.0010,0.0592;0.9010)
	PSO	(-0.0271,-1.3855e^{-04},3.6030e^{-04},0.0315;0.9006)
	ITL-QCA	(-0.0201,-5.0292e^{-05},3.9723e^{-04},0.0221;0.9002)
QWS_{25}^{100}	IDFPA	**(-0.0483,-1.0784e^{-04},2.5789e^{-04},0.0494;0.9006)**
	EFPA	(-0.0469,-4.8811e^{-05},3.9553e^{-04},0.0497;0.9004)
	PSO	(-0.0085,2.9974e^{-05},2.8760e^{-04},0.0104;0.9001)
	ITL-QCA	(-0.0121,-5.0810e^{-05},2.0072e^{-04},0.0133;0.9003)

For each QWS_n^m, the better fuzzy value among the ones of the compared algorithms is in bold.

TABLE 5.7 Composition optimality comparisons for five RND_n^ms instances

RND_n^m	Algorithm	Fuzzy best value	Fuzzy worst value	Fuzzy average value
RND_{10}^{25000}	IDFPA	**(0.5464,0.6830,0.7144,0.9360 ;0.9034)**	**(0.5454,0.6740,0.7110,0.9409 ;0.9034)**	**(0.5455,0.6763,0.7129,0.9399 ;0.9023)**
	EFPA	(0.5458,0.6698,0.7171,0.9447 ;0.9023)	(0.5286,0.6637,0.6973,0.9175 ;0.9010)	(0.5435,0.6708,0.7089,0.9354 ;0.9001)
	PSO	(0.5224,0.6450,0.6841,0.9006 ;0.9057)	(0.5028,0.6058,0.6363,0.8378 ;0.9024)	(0.5111,0.6199,0.6578,0.8728 ;0.9001)
	ITL-QCA	(0.5417,0.6573,0.7053,0.9424 ;0.9039)	(0.5255,0.6467,0.6903,0.9082 ;0.9039)	(0.5351,0.6545,0.6928,0.9163 ;0.9001)
RND_{20}^{12500}	IDFPA	**(0.5383,0.6242,0.6627,0.9194 ;0.9013)**	**(0.5095,0.5989,0.6304,0.9030 ;0.9013)**	**(0.5263,0.6117,0.6416,0.9164 ;0.9013)**
	EFPA	(0.5215,0.6156,0.6431,0.9086 ;0.9030)	(0.4871,0.5696,0.5975,0.8650 ;0.9013)	(0.5133,0.5983,0.6271,0.8893 ;0.9001)
	PSO	(0.4668,0.5518,0.5763,0.7508 ;0.9003)	(0.4353,0.5165,0.5418,0.7451 ;0.9002)	(0.4543,0.5381,0.5634,0.7298 ;0.9001)
	ITL-QCA	(0.5242,0.6099,0.6319,0.8592 ;0.9011)	(0.4895,0.5769,0.6007,0.8243 ;0.9005)	(0.5073,0.5951,0.6207,0.8313 ;0.9001)
RND_{30}^{8334}	IDFPA	**(0.5178,0.6030,0.6209,0.8998 ;0.9002)**	**(0.4849,0.5574,0.5838,0.8571 ;0.9002)**	**(0.5033,0.5801,0.6050,0.8812 ;0.9002)**
	EFPA	(0.5009,0.5867,0.6105,0.8732 ;0.9002)	(0.4550,0.5299,0.5510,0.8082 ;0.9013)	(0.4946,0.5743,0.5968,0.8435 ;0.9000)
	PSO	(0.4599,0.5406,0.5600,0.6809 ;0.9007)	(0.4162,0.4965,0.5115,0.5990 ;0.9001)	(0.4392,0.5193,0.5397,0.6717 ;0.9000)
	ITL-QCA	(0.4952,0.5712,0.6067,0.7971,0.9013)	(0.4852,0.5630,0.5843,0.7358 ;0.9006)	(0.4923,0.5743,0.5941,0.7581 ;0.9000)
RND_{40}^{6250}	IDFPA	**(0.5025,0.5757,0.5933,0.8833 ;0.9000)**	**(0.4537,0.5242,0.5437,0.8207 ;0.9007)**	**(0.4896,0.5663,0.5857,0.8650 ;0.9000)**
	EFPA	(0.4982,0.5794,0.5975,0.8207 ;0.9007)	(0.4450,0.5121,0.5351,0.7628 ;0.9002)	(0.4847,0.5600,0.5806,0.7786 ;0.9000)
	PSO	(0.4361,0.5209,0.5392,0.6276 ;0.9007)	(0.3842,0.4486,0.4661,0.5307 ;0.9004)	(0.4078,0.4815,0.4989,0.5842 ;0.9000)
	ITL-QCA	(0.4959,0.5695,0.6010,0.7441 ;0.9009)	(0.4719,0.5501,0.5706,0.6894 ;0.9002)	(0.4845,0.5627,0.5829,0.7073 ;0.9001)
RND_{50}^{5000}	IDFPA	**(0.5033,0.5804,0.5941,0.8557 ;0.9001)**	**(0.4519,0.5259,0.5428,0.8026 ;0.9004)**	**(0.4807,0.5566,0.5739,0.8343 ;0.9001)**
	EFPA	(0.5094,0.5828,0.6069,0.7015 ;0.9001)	(0.4459,0.5094,0.5281,0.6420 ;0.9003)	(0.4809,0.5543,0.5730,0.6840 ;0.9000)
	PSO	(0.4281,0.5045,0.5273,0.6038,0.9008)	(0.3827,0.4586,0.4757,0.5493 ;0.9001)	(0.4077,0.4841,0.5030,0.5821 ;0.9000)
	ITL-QCA	(0.4845,0.5680,0.5880,0.6827 ;0.9016)	(0.4450,0.5275,0.5446,0.6486 ;0.9022)	(0.4657,0.5428,0.5618,0.6668 ;0.9000)

For each RND_n^m, the better fuzzy value among the ones of the compared algorithms is in bold.

TABLE 5.8 Composition stability comparisons for five RND_n^ms instances

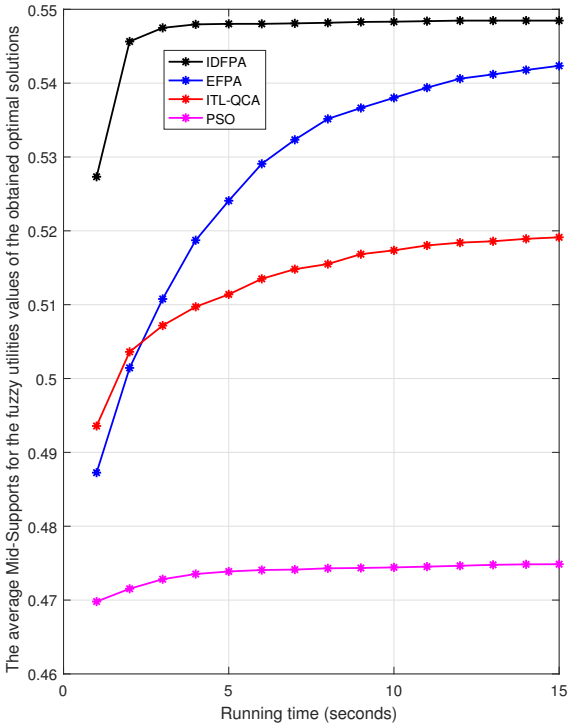
RND_n^m	Algorithm	Fuzzy variance
RND_{10}^{25000}	IDFPA	**(-0.1587,-0.0014,0.0015,0.1591 ;0.9023)**
	EFPA	(-0.1567,-0.0015,0.0017,0.1592 ;0.9001)
	PSO	(-0.1335,-0.0014,0.0018,0.1378 ;0.9001)
	ITL-QCA	(-0.1482,-0.0015,0.0018,0.1514 ;0.9001)
RND_{20}^{12500}	IDFPA	**(-0.1552,-8.7115e$^{-0.4}$,0.0013,0.1588 ;0.9013)**
	EFPA	(-0.1442,-7.2699e$^{-0.4}$,0.0015,0.1506 ;0.9001)
	PSO	(-0.0776,-5.3979e$^{-0.4}$,0.0012,0.0816 ;0.9001)
	ITL-QCA	(-0.1071,-6.2693e$^{-0.4}$,0.0010,0.1123 ;0.9001)
RND_{30}^{8334}	IDFPA	(-0.1475,-5.4027e$^{-0.4}$,0.0012,0.1537 ;0.9002)
	EFPA	(-0.1241,-4.1049e$^{-0.4}$,0.0011,0.1300 ;0.9000)
	PSO	(-0.0551,-2.5838e$^{-0.4}$,0.0011,0.0610 ;0.9000)
	ITL-QCA	**(-0.0721,-3.4219e$^{-0.4}$,7.3102e$^{-0.4}$,0.0757 ;0.9000)**
RND_{40}^{6250}	IDFPA	(-0.1436,-2.4438e$^{-0.4}$,0.0010,0.1515 ;0.9000)
	EFPA	(-0.0883,-2.6465e$^{-0.4}$,0.0011,0.0970 ;0.9000)
	PSO	(-0.0314,-6.4843e$^{-0.5}$,0.0013,0.0377 ;0.9000)
	ITL-QCA	**(-0.0506,-3.7136e$^{-0.4}$,7.2058e$^{-0.4}$,0.0541 ;0.9001)**
RND_{50}^{5000}	IDFPA	(-0.1275,-1.7232e$^{-0.4}$,8.7794eI.$^{-0.4}$,0.1362,0.9001)
	EFPA	(-0.0423,-7.0779e$^{-0.5}$,0.0015,0.0479 ;0.9000)
	PSO	(-0.0309,-2.5464e$^{-0.4}$,8.5049e$^{-0.4}$,0.0345 ;0.9000)
	ITL-QCA	**(-0.0412,-3.0229e$^{-0.4}$,7.3218e$^{-0.4}$,0.0445,0.9000)**

For each RND_n^m, the better fuzzy value among the ones of the compared algorithms is in bold.

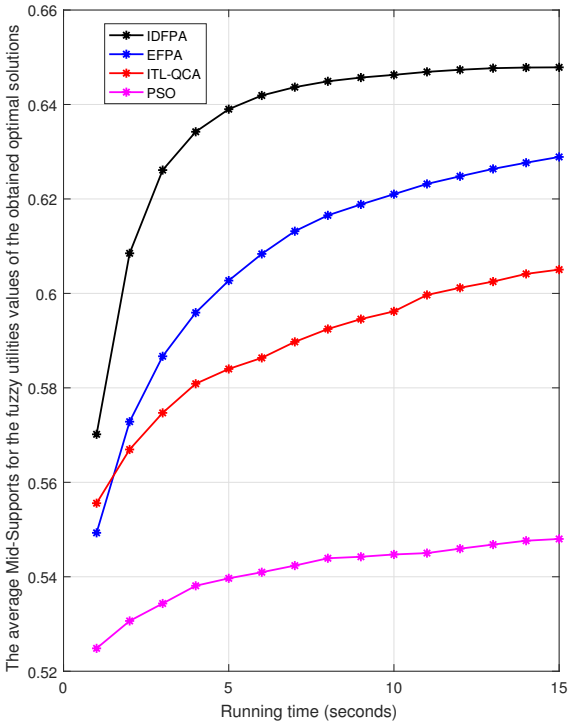
4.2.2 Composition time comparisons

The average composition times taken by each compared algorithm to solve each instance of the five QWS_n^m and the five RND_n^m scenarios over 50 independent runs are illustrated in Fig. 5.3 and Fig. 5.4, respectively.

As evident from the two figures, the composition times of PSO are slightly lower compared to those of IDFPA when solving fuzzy dataset instances with a small number of IoT-services classes n, as indicated by the bar graphs of the QWS_n^m and the RND_n^m with $n \leq 20$. However, as the number of IoT-services classes n increases, illustrated by the bar graphs of the QWS_n^m and the RND_n^m with $n \geq 25$, IDFPA outperforms PSO in terms of speed. One of the key factors contributing to IDFPA's superior speed in solving *ACS*s with large compositions of IoT-services is the implementation of *Discrete Global/Local Pollination* procedures, which employ simple integer operations to update only one position among n positions in each *CS* solution for each iteration cycle of IDFPA. This stands in contrast to PSO, which updates all positions in each iteration. Furthermore, IDFPA exhibits



(a) The QWS_{25}^{100} instance



(b) The RND_{30}^{8334} instance

FIGURE 5.2 The convergence curves associated to the average Mid-Supports of the fuzzy utilities values of the obtained feasible CS_{best}s by the compared algorithms in solving the QWS_{25}^{100} and the RND_{30}^{8334} instances over 50 runs (For colors see online version).

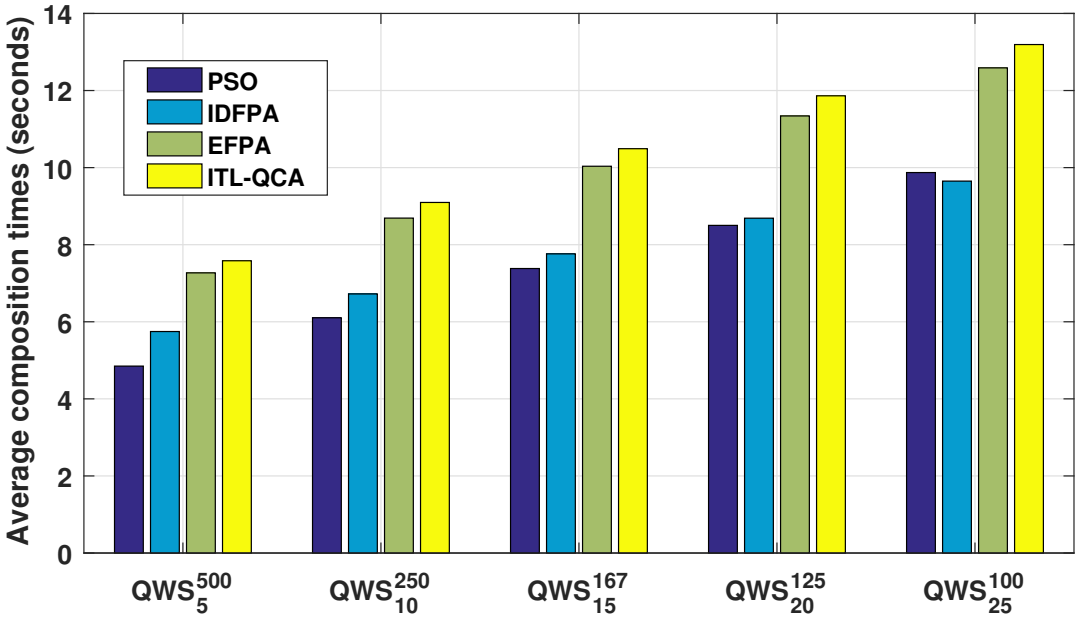


FIGURE 5.3 Composition time comparisons of IDFPA, EFPA, PSO and ITL-QCA algorithms in solving five QWS_n^ms (For colors see online version).

significantly faster performance than both EFPA and ITL-QCA in solving all the fuzzy real and synthetic dataset instances.

From this discussion, we can conclude that IDFPA demonstrates superior composition times compared to those of the compared algorithms, particularly when solving an *ACS* with a substantial number of IoT-services classes.

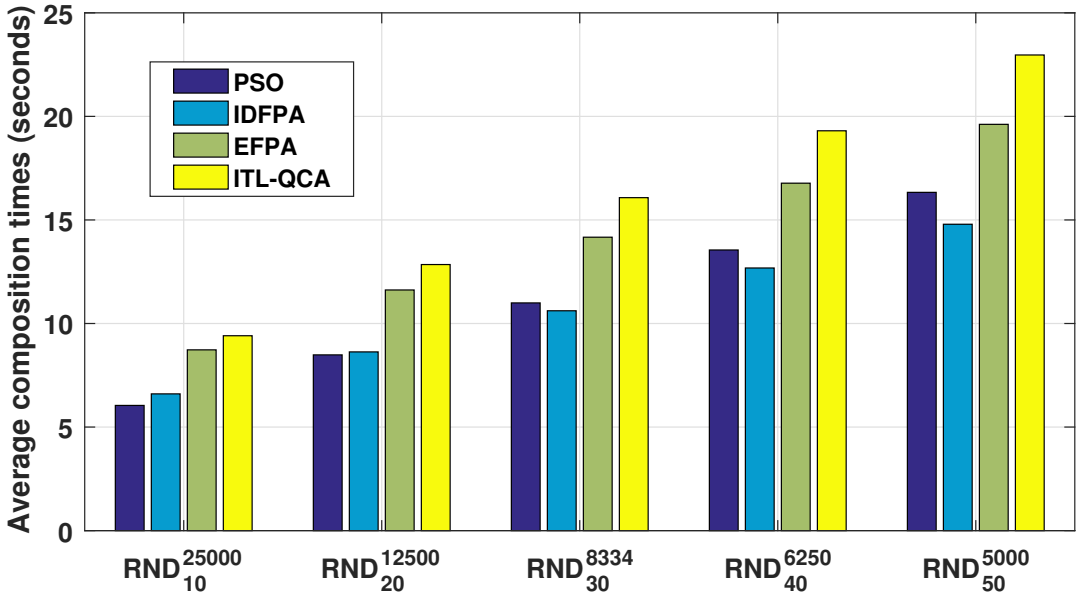


FIGURE 5.4 Composition time comparisons of IDFPA, EFPA, PSO and ITL-QCA algorithms in solving five RND_n^ms (For colors see online version).

5 Conclusion

In this chapter, a novel approach is proposed to tackle the uncertainty in QoS-aware IoT service composition, a problem known to be NP-hard. The approach combines two key modules : a fuzzy skyline-based module and an improved discrete flower pollination algorithm (IDFPA). The Generalized Trapezoidal Fuzzy Numbers are employed to represent the ambiguity inherent in QoS values. The problem is formalized as a fuzzy constrained single-objective optimization model, denoted as QoS-aware Service Composition for IoT (QSCFIoT).

The combined approach, integrating both components, is introduced to effectively address the formulated QSCFIoT. The fuzzy-based skyline module plays a crucial role in reducing the search space by eliminating redundant and dominated IoT services from sets of functionally equivalent ones. The IDFPA is then applied to explore the reduced search space and seek near-optimal composite services for QSCFIoT, leveraging an effective best solution improvement method.

To validate the performance of IDFPA, experiments are conducted using a fuzzy extended version of the public QWS dataset and a random dataset. The results highlight the efficiency of our proposed approach in terms of Composition Time, Composition Optimality, and Composition Stability when compared to existing methods such as the extended flower pollination algorithm, Particle Swarm Optimization, and an improved teaching-learning-based QoS-aware services composition algorithm.

Chapter 6

Collaborative Filtering Techniques for Predicting Web Service QoS Values in Static and Dynamic Environments : A Systematic and Thorough Analysis

1 Introduction

Over the past decades, the rapid expansion of the World Wide Web has driven a significant increase in online services. Research in this area has covered various aspects, including Web Service (WS) selection and composition Zeng et al. [2004], which involves employing optimization algorithms to identify the best services from a set of candidates, forming efficient execution paths for clients. Additionally, studies have explored WS recommendation Zheng et al. [2010a] and WS discovery Ran [2003], the latter focusing on retrieving candidate services that align with the functional requirements of specific workflow tasks. Each WS exhibits both functional and non-functional attributes during invocation. Prior research has predominantly emphasized Quality of Service (QoS) parameters such as response time, throughput, and cost, as these non-functional attributes significantly impact WS performance Jatoth et al. [2015].

QoS attributes are generally categorized into subjective and objective attributes. Subjective attributes—such as cost and reputation—are influenced by user opinions and service evaluations, while objective attributes, including response time and availability, remain independent of user preferences Ma et al. [2015]. Furthermore, QoS parameters can be classified as static or dynamic, with static attributes like cost remaining unchanged, whereas dynamic attributes, such as response time, fluctuate based on system conditions Syu et al. [2017]. In QoS-aware WS composition, objective and dynamic attributes are typically more relevant Ghafouri et al. [2020].

During composition, multiple abstract services can be instantiated by selecting suitable candidates from equivalent functional sets. However, many studies assume that QoS values remain uniform for all users, which is not always realistic. Factors such as internet topology changes, system congestion, and economic policies can cause QoS variations Razian et al. [2020]. As a result, key parameters like reputation, response time, and throughput may differ among users. Evaluating WSs from the user's perspective is therefore crucial. However, relying on service providers for QoS values is often impractical due to inconsistencies caused by network conditions, geographical location, and invocation time. At the same time, user-based evaluation is inefficient, as it is resource-intensive, costly, and unreliable due to fluctuations in QoS values and the limitations of individual user observations Zheng et al. [2010a].

Given these challenges, an effective approach for determining QoS values is essential, making personalized WS QoS prediction (WSQP) increasingly important in QoS-aware WS applications. Collaborative Filtering (CF), a widely used technique in recommender systems, has been extensively applied to WSQP. CF-based QoS prediction techniques leverage user and service interactions to generate personalized QoS value predictions, improving accuracy and adaptability Zheng et al. [2020]. CF approaches are typically classified into memory-based, model-based, and context-aware techniques Zheng et al. [2020], Ghafouri et al. [2020]. The introduction of CF in personalized QoS prediction was pioneered by Shao et al. [2007], and since then, numerous researchers have adopted and expanded upon this methodology for WSQP studies.

As research in this field has grown, several surveys have emerged, analyzing CF-based methods for WSQP Ghafouri et al. [2020], Shao et al. [2007], Ning et al. [2015], Desrosiers and Karypis [2011], Su and Khoshgoftaar [2009], Zheng et al. [2020]. However, the classification of these approaches remains an evolving area, with new techniques continuously being introduced.

CF-based techniques operate by leveraging collaborative data—utilizing user experiences and service behavior—to predict QoS attributes. This survey systematically categorizes CF-based approaches according to their distinct characteristics and methodologies, highlighting their effectiveness in both static and dynamic environments Tong et al. [2021], Herlocker et al. [2004]. Static environments are characterized by relatively stable parameters, whereas dynamic environments involve fluctuating network conditions and evolving user behaviors, both of which significantly impact QoS Yan et al. [2022]. The chapter also examines the role of fuzzy logic tools, which address uncertainty and imprecision in QoS data. These tools enhance WSQP accuracy by incorporating real-world uncertainties, resulting in more robust and adaptive predictions Chen et al. [2020b]. Furthermore, the survey presents an overview of the WS QoS datasets utilized for WSQP evaluation, comparing their characteristics and contributions to state-of-the-art studies.

The remainder of this chapter is structured as follows : Section 2 introduces Collaborative Filtering (CF) and its related subtopics, forming the foundation of our study. Section 3 defines the WSQP problem and outlines the motivation behind our research. Additionally, Section 4 provides a classification of recent and influential CF-based research. The approaches reviewed in Section 4 are then analyzed in detail in Section 5, covering their strengths and limitations. Section 6 discusses

potential improvements and directions for future research. Finally, Section 7 summarizes the key findings and implications of this study.

2 Preliminaries on Collaborative Filtering and Classical Prediction Methods

This section provides a foundational overview of **Collaborative Filtering (CF)**, a critical technique in **Web Service QoS Prediction (WSQP)**. It introduces fundamental concepts related to **Recommendation Systems (RS)**, **Collaborative Filtering**, and the classical prediction methods commonly employed in this field.

2.1 Recommendation Systems (RS)

Recommendation Systems (RSs) are designed to provide **personalized suggestions** by analyzing user preferences and historical interactions. These systems leverage **intelligent algorithms** to predict user interests and recommend relevant items based on item attributes and past user behavior Mashal et al. [2016]. RSs play a crucial role in decision-making processes, such as *product recommendations in e-commerce* or *book suggestions in digital libraries* Adomavicius and Tuzhilin [2011], Ricci et al. [2011].

2.2 Collaborative Filtering (CF)

Collaborative Filtering (CF) is a widely used **technique for predicting user preferences** by analyzing shared behaviors among users. It operates under the assumption that *users with similar interests are likely to prefer similar items* Lo et al. [2012a]. For instance, if a user group shares common preferences for specific items, CF can recommend items they have not yet encountered Tong et al. [2021], Herlocker et al. [2004]. CF approaches are broadly categorized into three main groups : **Memory-based methods**, **Model-based methods**, **Context-based methods**

2.3 Memory-Based Approaches

Memory-based CF methods identify the **top K most similar users** in the dataset and utilize their ratings to predict missing values for an active user. This process involves two fundamental steps :

1. **Identifying similar users or items** based on rating patterns.
2. **Aggregating these ratings** to generate predictions Xue et al. [2005].

Memory-based CF techniques can be further divided into the following categories :

2.3.1 Item-Based Methods

These methods analyze **similarities between items** based on user ratings. The assumption is that if a user has shown interest in a particular item, they are likely to prefer other items with similar characteristics Singh et al. [2020a, 2021b]. For example, in **service recommendations**, an item-based CF approach might suggest web services with functionalities similar to those previously selected by the user.

2.3.2 User-Based Methods

This approach focuses on the relationships between **users rather than items**. By measuring the similarity between users' rating histories, it recommends items favored by users with similar preferences Singh et al. [2020a, 2021b]. This technique assumes that if two users have a history of similar ratings, one user is likely to appreciate items recommended by the other.

2.3.3 Hybrid Filtering (HF) Methods

Hybrid filtering methods **combine item-based and user-based approaches** to enhance prediction accuracy. These methods mitigate issues such as *cold-start problems*, which arise when new users or items lack sufficient interaction history Burke [2007]. By leveraging both user and item similarities, hybrid approaches offer **more reliable** and **adaptive** recommendations.

2.4 Model-Based Approaches

Model-based CF techniques **group users with similar rating behaviors** into predefined classes. These approaches classify the **active user** into an appropriate class and use the ratings from that class to predict unknown preferences Xue et al. [2005]. Instead of relying on direct similarity measurements, model-based approaches leverage **machine learning techniques**, such as **latent factor models** and **neural networks**, to uncover hidden patterns in user interactions.

2.5 Context-Based Approaches

Context-based CF methods make recommendations by considering **a user's historical preferences and contextual factors**. These approaches analyze **item characteristics, user interactions, and service metadata** to generate more relevant recommendations. In **Web Services (WS) selection**, context-based filtering incorporates elements such as **service descriptions, user reviews, QoS parameters, and past interactions** to enhance prediction accuracy Burke [2007], Yu et al. [2023].

2.6 The Role of Collaborative Filtering in WSQP

It is important to note that CF techniques are **not mutually exclusive**, and many methods combine different approaches to **enhance performance**. The diversity in CF-based methods for WSQP reflects the varying strategies used to address the **challenges of QoS prediction** in both **static and dynamic environments**.

2.6.1 Collaborative Filtering in WSQP Research

Studies have identified **CF as a powerful tool** for WSQP. Several reviews have provided comprehensive analyses and classifications of CF-based approaches in WSQP research Ghafouri et al. [2020], Shao et al. [2007], Ning et al. [2015], Desrosiers and Karypis [2011], Su and Khoshgoftaar [2009], Zheng et al. [2020].

2.6.2 Contributions of RSs to WSQP

Recommendation systems contribute significantly to WSQP, particularly in addressing **challenges in static and dynamic service environments**. Their impact can be observed in the following areas :

— **Personalized Service Selection** : RSs provide **tailored recommendations** by analyzing user preferences and historical interactions. By considering unique user needs and prior experiences, RSs predict **which web services** best meet specific **QoS requirements**, assisting users in service selection.

— **QoS Prediction** : RSs leverage multiple QoS attributes—such as **response time, availability, reliability, and throughput**—to assess WS performance. By analyzing **historical QoS data** and **user feedback**, these systems predict QoS values for services that users have not yet invoked.

— **Adaptation to Dynamic Environments** : In **dynamic environments**, QoS values fluctuate due to **network congestion, service load, and user demand**. RSs dynamically **update their predictions** using real-time data, ensuring that recommendations reflect the most **current service conditions**.

— **Feedback-Based Improvements** : RSs collect and analyze **user feedback** on recommended services. This feedback loop is essential for refining future predictions and **enhancing the accuracy of WSQP models** over time.

3 Formulation of the WSQP problem

In this section, we delve deep into the rationale behind employing the WSQP, providing comprehensive textual and mathematical explanations of the QoS prediction problem.

3.1 Motivation of QoS prediction usage

The motivation for addressing the WSQP problem in both static and dynamic environments stems from the need to enhance the performance, reliability, and user satisfaction of web-based applications and services. Accurate WSQP optimizes system performance, leading to better decisions regarding service selection and composition, which in turn reduces response time during WS invocations. Properly predicted QoS values aid in selecting and recommending services that best meet users' end-to-end requirements. Additionally, QoS-aware WS composition is easily achieved by meeting specific local and global criteria, ensuring reliable and efficient service delivery. In dynamic environments, where service invocation conditions may change over time, WSQP becomes crucial for adapting to these changes. Therefore, continuous monitoring and prediction of QoS values are essential.

3.2 Textual description of the WSQP problem

QoS attributes of WSs should be evaluated separately, distinguishing between static and dynamic characteristics. Static attributes, such as price, remain constant and pose no challenges since they are identical for all users. Static QoS prediction methods are relatively straightforward due to the invariant nature of static attributes. Regression analysis and rule-based models are often sufficient for these predictions, as they rely on consistent input-output relationships. For example, a linear regression model can accurately predict a static QoS value based on historical data. Additionally, rule-based approaches can be used to assign static QoS attributes directly from service descriptions, minimizing computational complexity. These methods are computationally efficient and well-suited for environments with minimal variability in QoS parameters. However, dynamic QoS attributes, like response time and availability, vary by user and require careful handling.

To address this issue, a user-service matrix can be employed to evaluate dynamic QoS attributes and predict missing values. This matrix represents users as rows, dynamic QoS attributes as columns, and corresponding QoS values in the cells. Techniques like CF, Matrix Factorization (MF), or Machine Learning (ML) algorithms can predict missing values, enhancing the evaluation of dynamic QoS attributes.

Static QoS attributes are fixed values applicable to all users and need no further analysis. In contrast, dynamic QoS attributes fluctuate based on user interactions and external factors like topological changes and system congestion. Capturing these variations accurately is crucial, and a user-service invocation matrix can help manage and analyze dynamic QoS data efficiently.

By using a matrix-based approach, we can handle dynamic QoS data for multiple users and services, predict missing values accurately, and improve the overall evaluation of dynamic QoS attributes.

3.3 Mathematical Description of the WSQP Problem

Let's consider a QoS prediction framework centered around a *user-service matrix* (U,S) designed to capture dynamic QoS attributes. The system is formally characterized as follows :

— $U = \{u_1, u_2, ..., u_m\}$: Set of m service users where u_i represents an individual user $(1 \leq i \leq m)$

— $S = \{s_1, s_2, ..., s_n\}$: Set of n *functionally equivalent* web services $(1 \leq j \leq n)$, meaning they provide identical operational capabilities but may differ in QoS characteristics

— $M_{m \times n}$: User-Service Invocation Matrix, where

$$M = \{m_{ij}\} \quad \text{, with} \quad m_{ij} = \begin{cases} 1 & \text{if } u_i \text{ invoked } s_j \\ 0 & \text{otherwise} \end{cases} \tag{6.1}$$

Real-world matrices exhibit extreme sparsity (up to 99%) due to limited user-service interactions Guo [2012]. Our goal is to predict missing QoS values using CF techniques while handling this data sparsity challenge.

QoS Parameter Definitions For each observed user-service pair (u_i, s_j), we measure four key QoS attributes :

— R_{ij} : Response time (ms) which represents latency between request and response

— T_{ij} : Throughput (req/s) is defined as requests processed per second

— A_{ij} : Availability (%) is seen as operational uptime probability

— P_{ij} : Reliability (%) represents success rate of request processing

Optimization Objectives Formulation To enable multi-criteria decision making, an objective function is defined for each QoS parameter :

— Response time minimization (lower values preferred) :

$$f_1(u_i, s_j) = R_{ij} \tag{6.2}$$

— Throughput maximization (higher values preferred, thus negated for minimization) :

$$f_2(u_i, s_j) = -T_{ij} \tag{6.3}$$

— Availability maximization (higher values preferred, negated) :

$$f_3(u_i, s_j) = -A_{ij} \tag{6.4}$$

— Reliability maximization (higher values preferred, negated) :

$$f_4(u_i, s_j) = -P_{ij} \tag{6.5}$$

Multi-Objective Optimization Framework The QoS-aware service selection problem is formulated as :

$$\begin{aligned} \text{Minimize} \quad & \mathbf{f}(u_i, s_j) = [f_1(u_i, s_j), f_2(u_i, s_j), f_3(u_i, s_j), f_4(u_i, s_j)] \\ \text{Subject to} \quad & (u_i, s_j) \in U \times S \\ & m_{ij} = 1 \quad \text{(considering only observed interactions)} \end{aligned} \tag{6.6}$$

Pareto Optimality Criterion A solution $(u_i, s_j)^*$ is Pareto optimal if no other solution (u_k, s_l) satisfies :

$$\begin{aligned} \forall i \in \{1,2,3,4\} : f_i(u_k, s_l) \leq f_i((u_i, s_j)^*) \\ \text{with } \exists i \in \{1,2,3,4\} : f_i(u_k, s_l) < f_i((u_i, s_j)^*) \end{aligned} \tag{6.7}$$

This means no alternative service provides equal or better performance across all QoS dimensions without degrading at least one parameter.

Weighted Aggregation Approach To handle multiple objectives, a scalarization technique with user-defined preferences is employed :

$$\text{Minimize} \quad F(u_i, s_j) = \sum_{k=1}^{4} w_k f_k(u_i, s_j) \tag{6.8}$$

Where $w_k \geq 0$ are weights satisfying $\sum w_k = 1$, reflecting relative importance of each QoS attribute.

Parameter Normalization To address scale differences between QoS parameters (e.g., milliseconds vs percentages), min-max normalization is applied :

$$\hat{f}_k(u_i, s_j) = \frac{f_k(u_i, s_j) - \min\limits_{(u,s)} f_k}{\max\limits_{(u,s)} f_k - \min\limits_{(u,s)} f_k} \tag{6.9}$$

Normalized objectives are then combined as :

$$\text{Minimize} \quad \hat{F}(u_i, s_j) = \sum_{k=1}^{4} w_k \hat{f}_k(u_i, s_j) \tag{6.10}$$

Illustrative Example

Let's consider 2 users and 3 services with QoS parameters recorded in Table 6.1. Missing values (marked ' ?') correspond to unobserved interactions in matrix M :

Moreover, the following invocation matrix is considered for these two users and three services. In this matrix, a value of 0 indicates that the service was not invoked by the user, while a value of 1 signifies that the service was invoked. The columns represent the users, and the rows represent the

TABLE 6.1 QoS Parameters with Missing Values (?)

User u_i	Service s_j	R_{ij}(ms)	T_{ij}(req/s)	A_{ij}(%)	P_{ij}(%)
u_1	s_1	200	50	0.95	0.98
u_1	s_2	300	40	0.90	0.95
u_1	s_3	?	?	?	?
u_2	s_1	210	55	0.94	0.99
u_2	s_2	?	?	?	?
u_2	s_3	240	48	0.91	0.95

services.

$$M = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{6.11}$$

The following steps outline the calculated scores for services s_2 and s_3 corresponding to users u_1 and u_2, respectively.

Stp. 1 : **Normalization Calculations :**Using only observed values from Table 6.1 :

— Response Time : $\min R = 200$, $\max R = 300$

— Throughput : $\min T = 40$, $\max T = 55$

— Availability : $\min A = 0.90$, $\max A = 0.95$

— Reliability : $\min P = 0.95$, $\max P = 0.99$

Example normalization for (u_1, s_1) :

$$\hat{R}_{11} = \frac{R_{11} - \min(R_{ij})}{\max(R_{ij}) - \min(R_{ij})} = \frac{200-200}{300-200} = 0$$
$$\hat{T}_{11} = \frac{T_{11} - \min(T_{ij})}{\max(T_{ij}) - \min(T_{ij})} = \frac{50-40}{55-40} = \frac{10}{15} = 0.67$$
$$\hat{A}_{11} = \frac{A_{11} - \min(A_{ij})}{\max(A_{ij}) - \min(A_{ij})} = \frac{0.95-0.90}{0.95-0.90} = 1$$
$$\hat{P}_{11} = \frac{P_{11} - \min(P_{ij})}{\max(P_{ij}) - \min(P_{ij})} = \frac{0.98-0.95}{0.99-0.95} = \frac{0.03}{0.04} = 0.75$$

Stp. 2 : **Weighted Sum Evaluation :** Assuming the weight vector $[0.4, 0.3, 0.2, 0.1]$, which reflects the priorities of the four considered QoS attributes, we obtain :

$F(u_1, s_1) = 0.4(0) + 0.3(0.67) + 0.2(1) + 0.1(0.75) = 0.476$

Full calculations yield comparative scores :

— $F(u_1, s_1) = 0.476$, $F(u_1, s_2) = 0.400$

— $F(u_2, s_1) = 0.600$, $F(u_2, s_3) = 0.359$

Stp. 3 : **Optimal Service Selection :**

— User u_1 : Service s_2 (min score 0.400)

— User u_2 : Service s_3 (min score 0.359)

This example demonstrates how normalized QoS parameters and weighted aggregation enable systematic service selection despite missing data. Contemporary research continues to refine these CF-based approaches to improve prediction accuracy and handling of sparse matrices.

4 Collaborative Filtering techniques

To understand the WSQP problem, the following questions are posed, and their answers could facilitate a clearer understanding of the issue at hand.

— RQ1 : What is the distinction between static and dynamic environments in web services ?

— RQ2 : Why is it important to differentiate between static and dynamic environments when predicting QoS ?

— RQ3 : What are the commonly used prediction models and algorithms in static QoS prediction ?

— RQ4 : How do these models adapt to dynamic QoS prediction ?

To effectively address the aforementioned questions, the initial phase of our research involved conducting a comprehensive literature survey to identify pertinent keywords related to CF techniques for WSQP. These keywords played a crucial role in facilitating targeted searches across multiple academic databases, ensuring the relevance of retrieved publications. We identified 31 keywords, categorized into domains such as static and dynamic environments, paper types, and contextual information (see Table 6.2). This categorization ensured a systematic and exhaustive exploration of the literature.

The initial visualization of keyword co-occurrence network generated using VOS Viewer [1] (See Figure 6.1) highlighted key research themes in WSQP. At this stage, it informed us about further refinement of inclusion criteria and provide an intuitive overview of key research themes before full-text eligibility assessment.Subsequently, it helped to aligning closely with the categorized keywords presented in Table 6.2. Core domain terms such as "Web Services", "Quality of Service (QoS)", and "Prediction Models" appear centrally in the visualization, reflecting their foundational role in the field. Keywords related to static environments, including "Feature Engineering", "Feature Selection", "Historical Data", and "Regression Models" are well-represented, indicating their significance in offline QoS prediction. Similarly, dynamic environment terms such as "Real-time QoS", "Time-Series Prediction", and "Adaptive Models" are interconnected, emphasizing the growing focus on real-time and adaptive QoS forecasting. The network further reveals strong associations with "Deep Learning", "Forecasting", and "Optimization" underscoring the increasing reliance on machine learning techniques for QoS prediction. Subsequently, the connections in the network helped to refine the categorization in Table 6.2, illustrating the interplay between static and dynamic QoS prediction methodologies while reinforcing the field's emphasis on predictive analytics and adaptive modeling. As for the remainder of the selected keywords that do not show in this visualization, they were taken into consideration using Google Scholar due to the fact that the bibliometric datasets used for visualization are limited and do not contain all the needed keywords and information.

Moreover, to systematically retrieve relevant studies on WSQP, we formulated a set of search strings based on the retrived keywords and key concepts. Table 6.3 presents the most relevant search strings used to query academic databases.

1. https ://www.vosviewer.com/getting-started

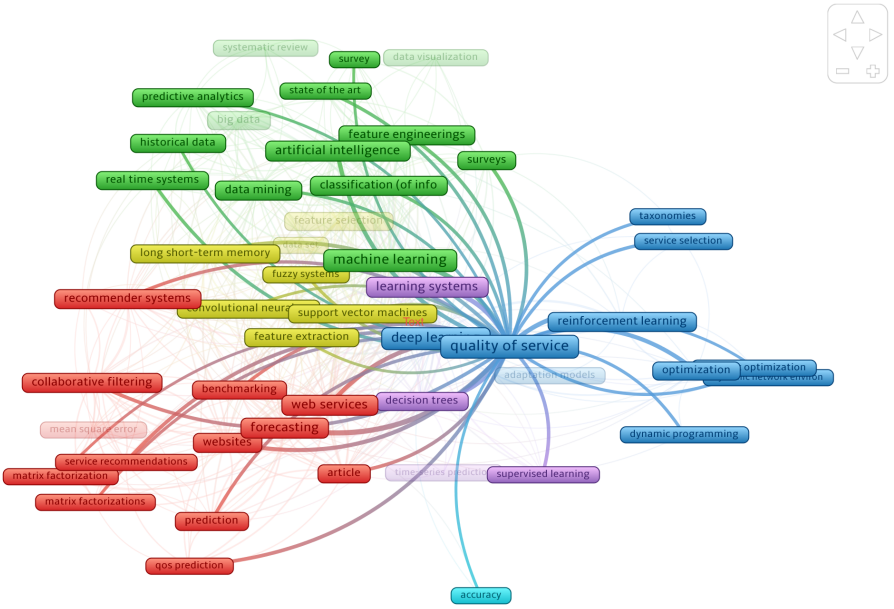


FIGURE 6.1 Keyword Co-Occurrence Network for Web Services QoS Prediction Research

TABLE 6.2 Keywords for Web Services QoS Prediction

Types of keywords	**Lists of keywords**
Domain	Web Service,Quality of Service, QoS Prediction, Dynamic QoS Prediction, Static QoS Prediction, Collaborative Filtering
Static environment	Static QoS, Offline Prediction, Historical Data, Static Analysis, Static Evaluation, Regression Models, Feature Engineering
Dynamic environment	Dynamic QoS Prediction, Real-time QoS, QoS Adaptation, QoS Variation, QoS Dynamics, Online QoS, Temporal QoS, Dynamic Service Quality, Time-Series Prediction, Dynamic Adaptability, Adaptive Models
Paper types	Conference, Proceedings, Review
Other	Web service data-sets, evaluation metrics, contextual information, contextual prediction

TABLE 6.3 Search Strings for Web Services QoS Prediction Across Different Databases

Database/Focus	Search String
IEEE Xplore / ACM Digital Library	("Web Service" OR "Quality of Service" OR "QoS Prediction" OR "Dynamic QoS Prediction" OR "Static QoS Prediction") AND ("Collaborative Filtering" OR "Time-Series Prediction" OR "Machine Learning" OR "Deep Learning") AND ("Static QoS" OR "Offline Prediction" OR "Historical Data" OR "Feature Engineering" OR "Dynamic QoS" OR "Adaptive Models" OR "Real-time QoS" OR "Context-aware QoS")
Scopus (Title, Abstract, Keywords)	TITLE-ABS-KEY(("Web Service" OR "Quality of Service" OR "QoS Prediction" OR "Dynamic QoS Prediction" OR "Static QoS Prediction") AND ("Collaborative Filtering" OR "Time-Series Prediction" OR "Machine Learning" OR "Deep Learning") AND ("Static QoS" OR "Offline Prediction" OR "Historical Data" OR "Feature Engineering" OR "Dynamic QoS" OR "Adaptive Models" OR "Real-time QoS" OR "Context-aware QoS"))
Web of Science (Topic Search)	TS=("Web Service" OR "Quality of Service" OR "QoS Prediction" OR "Dynamic QoS Prediction" OR "Static QoS Prediction" OR "QoS Adaptation" OR "QoS Variation" OR "QoS Dynamics" OR "Online QoS" OR "Temporal QoS" OR "Dynamic Service Quality") AND TS=("Collaborative Filtering" OR "Time-Series Prediction" OR "Machine Learning" OR "Deep Learning" OR "Predictive Analytics" OR "Forecasting" OR "Learning Algorithms") AND TS=("Static QoS" OR "Offline Prediction" OR "Historical Data" OR "Static Analysis" OR "Static Evaluation" OR "Regression Models" OR "Feature Engineering" OR "Feature Extraction" OR "Dynamic QoS" OR "Adaptive Models" OR "Real-time QoS" OR "Context-aware QoS" OR "Contextual Information" OR "Contextual Prediction") AND TS=("Web Service Data-sets" OR "Evaluation Metrics" OR "Benchmarking")
Google Scholar	"Web Service" OR "Quality of Service" OR "QoS Prediction" OR "Dynamic QoS Prediction" OR "Static QoS Prediction" AND "Collaborative Filtering" OR "Time-Series Prediction" OR "Machine Learning" OR "Deep Learning" AND "Static QoS" OR "Offline Prediction" OR "Historical Data" OR "Feature Engineering" AND "Dynamic QoS" OR "Adaptive Models" OR "Real-time QoS" OR "Context-aware QoS"
Focused Search Strings	
Static QoS Prediction	("Static QoS" OR "Offline Prediction" OR "Historical Data" OR "Static Analysis" OR "Regression Models") AND ("Web Service" OR "Quality of Service" OR "QoS Prediction")
Dynamic QoS Prediction	("Dynamic QoS Prediction" OR "Real-time QoS" OR "QoS Adaptation" OR "Time-Series Prediction" OR "Adaptive Models") AND ("Web Service" OR "Quality of Service" OR "QoS Prediction")
Collaborative Filtering in QoS	("Collaborative Filtering" OR "Trust-aware Filtering" OR "User-based QoS Prediction" OR "Matrix Factorization") AND ("Web Service" OR "Quality of Service" OR "QoS Prediction")
Context-Aware QoS Prediction	("Context-aware QoS" OR "Contextual Information" OR "Location-aware QoS" OR "Temporal QoS" OR "Dynamic Adaptability") AND ("Web Service" OR "Quality of Service" OR "QoS Prediction")

To refine our search, we adhered to strict inclusion criteria, focusing on publications that provided substantial insights into methodologies, evaluation strategies, and the application contexts of proposed solutions. A total of 146 peer-reviewed publications (2007-2024) were selected following a rigorous full-text review. Non-peer-reviewed articles, abstracts, and opinion-driven pieces were excluded to maintain the academic rigor of our study. To ensure the inclusion of only relevant and high-quality studies, we applied a set of exclusion criteria to filter out papers that do not align with the scope of WSQP. Table 6.4 presents the key exclusion criteria used in the selection process.

TABLE 6.4 Exclusion Criteria for Selected Studies

Criterion	**Description**
Non-QoS-related studies	Papers that do not explicitly focus on Web Service QoS prediction, evaluation, or adaptation.
Non-predictive approaches	Studies that focus on general QoS measurement, monitoring, or benchmarking without predictive techniques.
Non-collaborative filtering methods	Studies that do not involve collaborative filtering, machine learning, or time-series forecasting for QoS prediction.
Irrelevant domains	Research focusing on non-web-service applications, such as hardware-level optimizations or sensor network QoS.
Non-peer-reviewed sources	Papers from non-peer-reviewed journals, preprints, technical reports, or blog posts.
Limited empirical validation	Studies without real-world datasets, case studies, or proper evaluation metrics (e.g., RMSE, MAE, precision-recall).
Redundant studies	Duplicate papers, extended versions without major contributions, or studies already included in a previous survey.
Non-peer reviewed studies	Studies published in blogs and opinion studies were excluded from the primary selection, as only studies published in high reputed journals and well ranked conferences were included.

The PRISMA flowchart in Fig 6.2 illustrates the step-by-step process for publication selection. The initial 512 records were identified through a comprehensive search of Google Scholar, ACM DL, and Scopus using the predefined search terms indicated in Table 6.2. During the screening phase, 239 records were excluded : 152 did not meet the inclusion criteria, and 87 were duplicates. This left 273 records for title and abstract research screening phase. The exclusion criteria were applied rigorously during this phase, ensuring that only studies meeting the inclusion criteria progressed to the full-text eligibility assessment. From 196 records that were assessed for eligibility, only 146 reports met the criteria, as at the assessment phase 50 records were excluded due to their lack of novelty and their limited empirical studies. Ultimately, all 146 records met the inclusion criteria and were included in the review.

Subsequent data extraction and analysis focused on extracting algorithmic approaches, evaluation metrics, and implementation details. These findings were then categorized to provide a condensed yet comprehensive overview, presented in the following sections. The taxonomy developed in this study

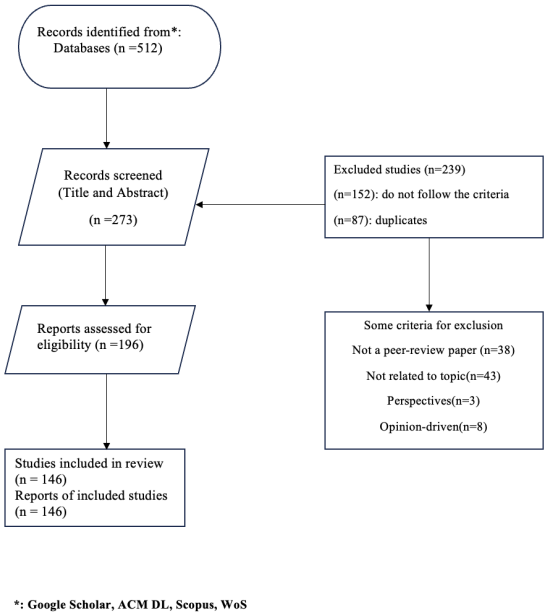


FIGURE 6.2 PRISMA Flowchart for Including Publications

highlights a key contribution : the integration of time-aware and time-series techniques, bridging the gap between temporal dynamics and predictive accuracy. By capturing time-based QoS fluctuations, this taxonomy addresses limitations of traditional WSQP methods. Fig 6.3 showcases this taxonomy and its emphasis on dynamic adaptability.

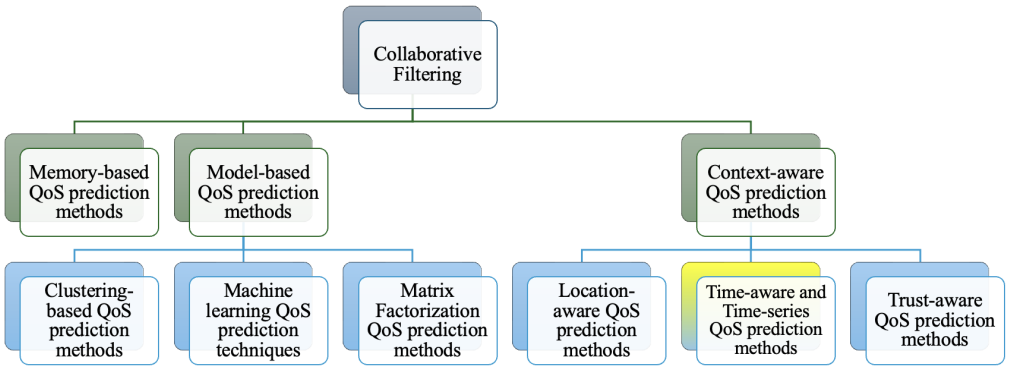


FIGURE 6.3 Web Service QoS Prediction methods Taxonomy

While the methodology provided a structured approach, challenges such as keyword ambiguity and publication overlap were encountered. Future work will focus on refining the inclusion criteria and expanding the scope to cover emerging methodologies.

4.1 Memory-based QoS prediction methods

Memory-based CF techniques are extensively used in WSQP, leveraging the user-service QoS matrix to improve prediction accuracy. These methods emphasize several critical aspects, such as data preprocessing, similarity measurement, selection of similar neighbors, and QoS value estimation Zheng et al. [2020].

Data Pre-processing

Enhancing data quality is essential for improving WSQP accuracy. To address sparse QoS data, various techniques are employed, such as imputing missing values using defaults like the mean or median. Methods like K-means clustering and data smoothing are also applied to mitigate missing data issues Zheng et al. [2010a], Ma et al. [2007]. Additionally, normalizing QoS values across different scales—using approaches like Gaussian normalization or the 3-sigma rule—has been shown to significantly enhance CF prediction accuracy Zhang et al. [2012].

Similarity Computation

plays a vital role in identifying comparable neighbors for collaborative prediction and determining influence weights, making similarity metrics crucial for the effectiveness of CF. Commonly used similarity measures include :

— **Pearson Correlation Coefficient (PCC) :** Measures the linear relationship between two variables Obidallah et al. [2020]. In CF, PCC is widely applied to assess similarity between users or items based on their QoS ratings. A positive correlation value indicates a strong positive relationship, while a negative value signifies an inverse correlation Ma et al. [2007], Yin et al. [2014], Qi et al. [2015].

— **A-Cosine (Adjust-Cosine) similarity :** Enhances cosine similarity by subtracting each service's average QoS value, effectively addressing variations in QoS scales Chen et al. [2019b], Wu et al. [2012].

— **JacMinMax :** Useful in scenarios where QoS values exhibit significant fluctuations. This method utilizes a MinMax ratio to capture the variance in service experiences among users, improving the accuracy of similarity computation Chen et al. [2019b], Zheng et al. [2020].

Similar neighbor selection

Before predicting missing values, it is essential to identify relevant neighborhoods consisting of similar services or users. Effective neighbor selection plays a critical role in ensuring accurate predictions, as incorporating dissimilar neighbors can reduce prediction reliability Singh et al. [2020b, 2021a]. Various approaches are employed for neighbor selection, including :

— **Top-N Filtering :** Identifies the N most similar services or users based on their similarity scores, selecting them as relevant neighbors for prediction Ren and Wang [2018].

— **Negative Filtering :** An enhancement of the Top-N approach, this method excludes neighbors with negative PCC similarities, addressing scenarios where fewer than N similar neighbors are available Zheng and Lyu [2010].

— **Threshold Filtering :** This technique applies a predefined similarity threshold to determine whether a neighbor should be included in the collaborative prediction process. It is particularly useful for evaluating QoS attributes such as response time and reliability Ma et al. [2007].

QoS values prediction

After identifying a set of similar neighbors for each user or service, their QoS values are utilized to estimate the missing QoS values for the target user or service. The prediction approach varies based on the filtering method employed : User-based approaches predict QoS values by leveraging the QoS data from similar users. Item-based approaches estimate QoS values using information from similar web services (WSs) and Hybrid approaches integrate both user-based and item-based methods, with a tunable parameter controlling the balance between the two strategies Zheng et al. [2010a], Yin et al. [2014], Shao et al. [2007], Jiang et al. [2011]. The following works provide a concise overview of the most notable contributions in QoS value prediction as each study addressed specific challenges in web service environments. :

In Zheng et al. [2010a], a hybrid CF mechanism was introduced to improve WSQP by integrating user-based and item-based CF approaches, allowing for the collection of historical QoS data without directly invoking services. While this method effectively mitigates data sparsity by leveraging shared user-service interactions, it heavily relies on historical data, limiting its adaptability to new services and dynamic environments. Similarly, Tong et al. [2021] enhances prediction accuracy by discarding outdated QoS values and incorporating similarity computation across multiple time slices, emphasizing temporal dynamics to ensure the use of relevant data for WSQP. However, despite its improved reliability in dynamic environments, this approach faces scalability challenges due to the computational overhead associated with processing multiple time slices, and its reliance on historical data restricts its ability to accommodate new services. The Colbar framework, introduced in Yin et al. [2014], integrates matrix factorization (MF) with personalized geographical and QoS data to refine prediction accuracy by leveraging latent factor models and spatial information, enabling location-based personalized QoS predictions. However, its susceptibility to outliers and dependence on historical data pose challenges when applied to new or sparsely rated services. In Chen et al. [2010], the **RegionKNN** method was proposed to enhance WSQP by clustering users and identifying region-sensitive WSs, yet its performance is highly dependent on parameters such as the size of the region and the number of neighbors (K), necessitating careful tuning for optimal results, which in turn affects its scalability and applicability. The User Services Matrix Factorization (US-MF) approach, developed in Qi et al. [2015], integrates user networks and service neighborhoods to improve personalized WSQP by

leveraging additional network-based data, yet it remains constrained by its reliance on historical data, reducing its effectiveness in scenarios involving new services or users. The Big Range-Aware Collaborative Filtering (BRACF) model proposed in Chen et al. [2020a] addresses variations in QoS attributes to enhance prediction reliability by tackling data diversity and attribute-specific challenges ; however, while focusing on improving accuracy, it fails to effectively handle scalability and adaptability in dynamic web service environments. The personalized CF-based QoS data collection technique introduced in Shao et al. [2007] tailors WSQP predictions to individual users by utilizing historical QoS data and user-specific preferences, yet its effectiveness diminishes in cases with sparse datasets or insufficient user history. In Zheng et al. [2009], a user-contribution method for QoS data collection was introduced alongside a hybrid CF algorithm combining user-based and item-based filtering, enhancing prediction accuracy by leveraging the strengths of both approaches, though its dependence on historical data and limited adaptability to dynamic QoS conditions remain challenges. The collaborative WSQP approach in Zheng et al. [2012] utilized shared user experiences and a neighborhood-based model to extend WSQP capabilities by leveraging user collaboration and QoS data aggregation for personalized predictions. Addressing the cold-start problem, Qi et al. [2017] introduced the Inverse CF approach (Inverse_CF_Rec), based on Social Balance Theory, which leverages social network structures to predict QoS values for new users or services with minimal historical data, yet its reliance on well-defined and accurate social network data restricts its applicability in cases where such data is unavailable or incomplete. The item-based prediction algorithm analyzed in Sarwar et al. [2001] emphasized scalability for online systems by introducing precomputed item similarity models to improve real-time recommendation efficiency, though this method struggles in dynamic environments where frequent changes render precomputed models obsolete. The pioneering item-based CF approach introduced by Linden et al. [2003] for Amazon.com focused on item-to-item similarity for predicting user preferences, providing a computationally efficient solution for static environments but lacking adaptability to temporal changes or dynamic QoS attributes in WSQP. Similarly, the item-based recommendation algorithm proposed in Deshpande and Karypis [2004] achieved high efficiency and scalability by relying on item similarities for predictions but did not account for temporal variations or user-specific QoS preferences. A trust-based and location-aware WSQP approach, integrating user trust metrics with geographical data to improve prediction accuracy, was introduced in Chen et al. [2017a], yet its dependence on reliable trust data presents challenges, particularly in environments with limited user feedback or conflicting trust evaluations. In White et al. [2018a], the IoT prediction model (IoTPredict) was designed to reduce computational overhead while maintaining accuracy by eliminating direct service invocation, but its effectiveness is constrained in dynamic IoT environments where QoS attributes fluctuate frequently. The LoRec framework proposed in Chen et al. [2013] incorporated location data and QoS attributes into WS recommendations, improving predictions through geographical proximity and service quality considerations, but its applicability is limited to location-aware systems, restricting its generalization to broader contexts. The Highly Accurate Prediction Algorithm (HAPA) introduced in Ma et al. [2015] leveraged MF for WSQP, achieving high accuracy in static environments but struggling to

adapt to dynamic QoS fluctuations and facing difficulties in addressing the cold-start problem. The Personalized Hybrid Collaborative Filtering (PHCF) method developed in Jiang et al. [2011] combined item-based and user-based CF techniques to improve prediction accuracy by integrating personalized user data, yet it requires substantial computational resources for large-scale datasets. The Hybrid User-Based Fuzzy Collaborative Filtering (HU-FCF) method presented in Son [2014] incorporated fuzzy logic for more refined similarity measurements, achieving high accuracy but demanding careful tuning of fuzzy parameters, making it computationally intensive. Lastly, Zhu et al. [2023] introduced the Bi-Subgraph Network based on Graph Contrastive Learning (BGCL), designed to improve WSQP in sparse user-service interaction scenarios by employing an edge dropout strategy for data augmentation and leveraging user and service embeddings. While BGCL effectively addresses cold-start issues and enhances prediction accuracy, its high computational requirements may hinder its scalability in large-scale systems.

4.2 Model-based QoS Prediction Methods

Model-based WSQP methods overcome the drawbacks of traditional memory-based approaches by leveraging user-service invocation datasets to identify patterns within training data, leading to more precise QoS predictions Ghafouri et al. [2020]. The following section highlights key studies in this category, classified into clustering-based, machine learning, and matrix factorization techniques.

4.2.1 Clustering-based QoS Prediction Methods

Clustering-based QoS prediction methods aim to group similar data points while maintaining distinct differences between clusters, improving scalability, reducing computational overhead, and addressing the cold-start problem in WSQP Su and Khoshgoftaar [2009]. Several studies have made significant contributions in this domain. A CF model proposed in AL-Bakri and Hassan [2019] integrates Fuzzy c-means clustering with a fuzzy user-based similarity measure, incorporating both user ratings and fuzzy truthfulness information. While this approach enhances prediction accuracy, its computational efficiency diminishes when applied to large-scale datasets. In Zhu et al. [2012], a landmark-based WSQP framework was introduced, incorporating User-Based Clustering (UBC) and Web Service-Based Clustering (WSBC) algorithms that utilize hierarchical clustering on real-world QoS data from PlanetLab [2]. Though effective for hierarchical datasets, its adaptability to flat data remains limited. Similarly, He et al. [2014] developed a location-based hierarchical matrix factorization (HMF) algorithm, clustering users and services based on location information. While this approach effectively leverages spatial data, it may fail to account for temporal QoS variations. To address data sparsity, Chen et al. [2015] introduced SCQP, a hybrid QoS prediction method that combines web service clustering with user similarity calculations. Despite its robustness, the model suffers from computational complexity. A fuzzy clustering technique known as Possibilistic Fuzzy

2. http ://www.planet-lab.org

C-Means (PFCM) was employed in Zhang et al. [2012] to model QoS as a multi-dimensional object, treating each property as a separate dimension. Although this method enhances multidimensional modeling, handling high-dimensional data presents challenges. Another notable contribution is the ADF approach proposed in Wu et al. [2012], a neighborhood-based CF method that employs a Two-Phase Neighbor Selection (TNS) strategy to refine prediction accuracy. While this enhances scalability, it does not effectively address the cold-start problem. A clustering-based WSQP framework utilizing k-means clustering was introduced in Zhang et al. [2013] to strengthen the relationship between service ratings and user expectations. While simple and effective, its performance is highly sensitive to the choice of initial centroids. The CLUS model, presented in Silic et al. [2013], focuses on predicting atomic web service reliability while addressing scalability through k-means clustering. However, its lack of adaptability to dynamic environments limits its applicability. To improve prediction robustness, Wu et al. [2020] introduced the DCALF model, combining latent factor analysis with density peaks-based clustering to mitigate noise in QoS data. While effective against noisy data, its computational overhead may hinder real-time deployment. A personalized recommendation algorithm proposed in Wu et al. [2014b] addresses inaccuracies due to sparse datasets by utilizing fuzzy clustering to identify the most relevant neighbors for target users. Although this method improves accuracy by mitigating sparsity, it struggles in highly dynamic environments where user preferences change frequently. Additionally, the reliance on fuzzy clustering introduces computational complexity, particularly in large-scale datasets, affecting its real-time efficiency. More recently, Li et al. [2021a] presented QSPC, a deep learning-based QoS prediction model incorporating temporal information and request context to improve accuracy. Evaluations on the WS-DREAM dataset demonstrated its superiority over traditional CF and MF approaches in dynamic environments. The study further employed t-distributed Stochastic Neighbor Embedding (t-SNE) for visualizing request context embeddings, validating their meaningful organization. While QSPC effectively captures temporal patterns, its ability to adapt to sudden, unpredictable QoS variations remains a challenge, particularly in real-time web service environments.

From these studies, it can be concluded that t-SNE plays a crucial role in enhancing WSQP by facilitating the visualization of high-dimensional QoS data, revealing underlying user-service patterns that improve clustering accuracy—complementing traditional methods like k-means. Integrating clustering techniques with t-SNE not only aids in outlier detection for robust data filtering but also enables intuitive visualizations of clustered QoS values, supporting more informed service selection. These advancements collectively contribute to improving prediction interpretability and reliability in dynamic web service environments.

Table 6.5 summerizes the key-findings of the cluster-based WSQP methods.

4.2.2 Machine Learning QoS Prediction Techniques

These techniques have been widely applied to WSQP, effectively addressing undetermined QoS values and capturing nonlinear relationships to enhance prediction performance Ghafouri et al. [2020],

TABLE 6.5 Summary of Clustering-based QoS Prediction Methods

Study	Unique Contribution	Limitation
AL-Bakri and Hassan [2019]	Fuzzy c-means clustering with user-based similarity measures integrating fuzzy truthfulness.	Computational inefficiency for large datasets.
Zhu et al. [2012]	Landmark-based framework with user-based and service-based clustering.	Limited applicability to flat datasets.
He et al. [2014]	Location-based hierarchical matrix factorization (HMF).	Neglects temporal QoS variations.
Chen et al. [2015]	SCQP hybrid method combining clustering and similarity calculations.	High computational complexity.
Zhang et al. [2012]	Fuzzy clustering to model multi-dimensional QoS.	Challenges in handling high-dimensional data.
Wu et al. [2012]	ADF with a two-phase neighbor selection strategy.	Persistent cold-start problem.
Zhang et al. [2013]	k-means clustering to enhance service rating relationships.	Sensitivity to initial centroids.
Silic et al. [2013]	*CLUS* model addressing reliability through clustering.	Limited adaptability for dynamic environments.
Wu et al. [2020]	DCALF combining latent factor analysis with density peaks-based clustering.	Computational overhead for real-time applications.
Li et al. [2021a]	QSPC model combining temporal information and request context	struggle to adapt to sudden and unpredictable changes in QoS values and high computational resources.

Palaios et al. [2023]. Several studies have contributed significantly to this domain. In Luo et al. [2016], a data-driven strategy combining Kernel Least Mean Square (KLMS) and Pearson Correlation Coefficient (PCC) was proposed to extract relevant QoS values from similar users and services, successfully modeling nonlinear relationships but struggling with scalability in large datasets. The Joint CNN-MF (JCM) model introduced in Yin et al. [2020] combines deep features extracted from Convolutional Neural Networks (CNNs) with MF, enhancing neighbor selection accuracy using CNN-based similarity computation, though it requires substantial computational resources. In Luo et al. [2015a], fuzzy neural networks and adaptive dynamic programming (ADP) were employed to model nonlinear and dynamic QoS attributes while ensuring stability through convergence boundedness, yet real-time application remains a challenge. A model incorporating geographical data, user usage patterns, privacy, and retention levels was designed in Anithadevi and Sundarambal [2019], integrating

Neuro Fuzzy Logic (NFL) for precision and the detection of undesirable web services, but its multi-dimensional approach increases implementation complexity. VMCF4SR, a Support Vector Machine (SVM)-based collaborative filtering model, was presented in Ren and Wang [2018], utilizing historical ratings to establish a hyperplane for direct Top-N service recommendations, effectively handling data sparsity but lacking adaptability to dynamic QoS changes. In Wu et al. [2017a], Deviation-based Neighborhood Models (DbNMs) leveraged crowd intelligence and user location for improved IoT and cloud QoS predictions, enhancing location-based accuracy but remaining sensitive to outlier data. Similarly, Wu et al. [2017b] developed EFMPred, which employs embedding vectors and neural networks to capture implicit user-service relationships, improving latent factor modeling but requiring extensive training data. The LANFM (Location-Aware QoS Prediction) model presented in Chen et al. [2019a] utilizes factorization machines and neural networks to transform geographical data into embedding vectors for more precise predictions, though computational intensity remains a concern. To reduce training and request times while maintaining prediction accuracy, White et al. [2019] employed stacked autoencoders in neural networks, effectively minimizing latency but at the cost of interpretability. The STCA-1 and STCA-2 models, introduced in Zhou et al. [2019], incorporate spatio-temporal context awareness, considering invocation time and spatial attributes for WSQP, providing robust performance but being computationally expensive. The LA-LMRBF model, an online QoS prediction approach developed in Zhang et al. [2019a], integrates the Affinity Propagation clustering algorithm with RBF neural networks to predict time-sensitive QoS attributes, achieving high accuracy but requiring precise parameter tuning. In Jin et al. [2019b], NDL (Neighborhood Deep Learning) employed PCC for top-k neighbor extraction and a CNN-based feature extraction mechanism for QoS prediction, demonstrating effectiveness but demanding significant computational resources. The MFDK model introduced in Yan et al. [2022] addresses sparse QoS values by incorporating non-negative matrix decomposition and a Kalman filter for real-time QoS observations, effectively managing sparsity but requiring frequent updates for real-time performance. Deep feature fusion was explored in Ding et al. [2023], integrating environmental preference and multi-class QoS traits through a user-service bipartite graph and a multi-component graph CNN, providing comprehensive feature extraction but remaining computationally intensive for both static and dynamic environments. Similarly, Zhu et al. [2024] introduced LDIF (Location-Aware Deep Interaction Forest), combining location-awareness, scanning interactive structures, and deep forest techniques for QoS prediction, demonstrating innovation but facing scalability limitations due to high computational cost. The Wide and Deep model, proposed in Cheng et al. [2016], combines wide linear models with deep neural networks to capture both low-order and high-order feature interactions, proving effective for diverse feature representations but prone to overfitting on small datasets. In Wang et al. [2017], the Deep and Cross model replaced polynomial regression with a Cross network for feature interaction modeling, enhancing predictive accuracy but requiring extensive training. The MultiFed framework, presented in Xu et al. [2023], employs a cloud-edge collaboration mechanism combined with MF for WSQP, effectively handling heterogeneous data but imposing high computational requirements. Finally, the IRE4DQP framework, introduced in Li et al. [2023], integrates intelligent route estimation (IRE) as

a Markov decision process with neural models for dynamic QoS prediction, offering an innovative approach but requiring careful fine-tuning to maintain stability.

Table 6.6 provides a summary of the key findings from machine learning-based WSQP methods, outlining their strengths, limitations, and specific contributions to enhancing QoS prediction accuracy.

4.2.3 Matrix Factorization QoS Prediction Methods

Matrix Factorization (MF) is widely recognized as a powerful approach for WSQP, effectively addressing challenges such as data sparsity and scalability by reducing the dimensionality of the user-service matrix while preserving key structural relationships Mnih and Salakhutdinov [2007]. Several studies have leveraged MF techniques to improve QoS predictions. Extended Matrix Factorization (EMF) with relational regularization terms was introduced in Lo et al. [2012b] to incorporate user and service neighborhood information, enhancing prediction accuracy through neighborhood relationships, though it may face difficulties in adapting to dynamic data changes. Similarly, the Service Neighborhood Extended Matrix Factorization (SN-EMF) and User Neighborhood Extended Matrix Factorization (UN-EMF) models, proposed in Xu et al. [2013a], integrate historical QoS records and geographical data, making them effective in static environments but potentially limited in real-time applications. Probabilistic Matrix Factorization (PMF) was applied to WSQP in Xu et al. [2013b], where L-PMF and WL-PMF were introduced to incorporate collective intelligence and geographical insights, achieving a balance between precision and scalability while remaining highly dependent on data quality. To address the cold-start problem, the Latent Neighbors Latent Factor Model (LN-LFM) was introduced in Yu et al. [2014], leveraging latent neighbor features for improved prediction accuracy ; however, its computational complexity remains a challenge. Network-aware models such as Network-Aware Matrix Factorization (NAMF) Tang et al. [2016] combine MF with a network map to evaluate user distances, enhancing prediction reliability while requiring significant preprocessing efforts. Similarly, Reputation-based Matrix Factorization (RMF) Xu et al. [2015] incorporates user reputation based on contributed QoS values, improving prediction credibility, though the reliance on reputation metrics may introduce biases. Non-Negative Collaborative Matrix Factorization (NCNMF) Su et al. [2016] integrates memory-based and model-based CF with Expectation-Maximization (EM) for training, effectively tackling data sparsity but at the cost of high computational demands. In Chen et al. [2016], Neighborhood Regularized Matrix Factorization (NRMF) was proposed to integrate **user and service neighborhood data**, improving prediction accuracy for sparse datasets, though it remains sensitive to parameter tuning. Similarly, the Hybrid Asymmetric Correlation Regularized (HACR) MF modelXie et al. [2016] introduces asymmetric correlation for QoS prediction, offering a balance between accuracy and scalability, albeit with computational trade-offs. MF-based models have also been tailored for cloud environments, as seen in CloudPred Zhang et al. [2011a], which integrates neighborhood-based and latent factor approaches to optimize predictions for cloud services, though its applicability to other domains remains limited. More recently, Matrix Factorization Automatic Interaction Network (MFAIN) Zhang

TABLE 6.6 Summary of Machine Learning-based QoS Prediction Methods

Study	Unique Contribution	Limitation
Luo et al. [2016]	Combines kernel least mean square (KLMS) and PCC for QoS prediction.	Scalability issues for large datasets.
Yin et al. [2020]	Joint CNN-MF (JCM) combines CNNs with matrix factorization for improved accuracy.	Computational resource-intensive.
Luo et al. [2015a]	Fuzzy neural networks and ADP for nonlinear and dynamic QoS modeling.	Real-time application challenges.
Anithadevi and Sundarambal [2019]	Integrates geographical data, usage patterns, and Neuro Fuzzy Logic (NFL).	Complexity in implementation.
Ren and Wang [2018]	SVMCF4SR employs SVM for top-N service recommendations.	Limited adaptability for dynamic QoS attributes.
Wu et al. [2017a]	Deviation-based Neighborhood Models (DbNMs) incorporating crowd intelligence.	Sensitivity to outlier data.
Wu et al. [2017b]	EFMPred captures implicit user-service connections using neural networks.	Substantial training data required.
Chen et al. [2019a]	LANFM transforms geographic data into embeddings for QoS prediction.	Computationally intensive.
White et al. [2019]	Stacked auto-encoders reduce training and request times while maintaining precision.	Sacrifices interpretability.
Zhou et al. [2019]	Spatio-temporal context-aware neural models (STCA-1, STCA-2) for WSQP.	High computational demands.
Zhang et al. [2019a]	LA-LMRBF uses RBF neural networks for time-sensitive QoS predictions.	Sensitive to parameter tuning.
Jin et al. [2019b]	NDL extracts features using CNNs and predicts QoS with PCC.	High computational cost.
Yan et al. [2022]	MFDK uses non-negative matrix decomposition and Kalman filter for real-time predictions.	Frequent updates needed for real-time performance.
Ding et al. [2023]	Deep feature fusion with multi-class QoS traits and graph CNNs.	Computational intensity.
Zhu et al. [2024]	LDIF combines location-awareness and deep forests for QoS prediction.	Limited scalability.
Cheng et al. [2016]	Wide and Deep model captures low-order and high-order feature interactions.	Potential overfitting on smaller datasets.
Wang et al. [2017]	Deep and Cross model enhances feature interaction modeling.	High training requirements.
Xu et al. [2023]	MultiFed employs federated learning with cloud-edge collaboration.	High computational requirements.
Li et al. [2023]	IRE4DQP uses intelligent route estimation for dynamic QoS prediction.	Requires fine-tuning for stability.

et al. [2023] combines MF with self-attention mechanisms, demonstrating strong performance in dynamic scenarios, yet requiring substantial computational resources. In Chen et al. [2022], a hybrid QoS prediction model for IoT services was introduced, incorporating factorization machines and deep learning techniques to manage low-order and high-order feature interactions, effectively addressing challenges such as data sparsity and dynamic user-service interactions by leveraging historical QoS data and contextual information. The QoS range model JacMinMax, proposed in Chen et al. [2019b], introduced two neighborhood selection strategies—neighborhood-based and model-based—rooted in MF principles to enhance WSQP, though its inability to adapt to new users and dynamic environments remains a limitation. Additional MF-based models include DNLF, introduced in Luo et al. [2015b], which employs non-negative latent features for WSQP, adapting training processes to individual elements while requiring significant computational power. Wide-Range Aware Matrix Factorization (WRAMF) Chen et al. [2020b] was designed to explicitly handle wide-ranging influences, integrating adaptive learning and efficient parallel stochastic gradient descent, enabling improved adaptability in dynamic environments. Table 6.7 provides a comparative summary of these MF-based WSQP techniques, outlining their contributions in tackling QoS prediction challenges.

TABLE 6.7 Summary of Matrix Factorization-based QoS Prediction Methods

Study	Unique Contribution	Limitation
Lo et al. [2012b]	Relational regularization terms to enhance predictions.	Limited adaptability to dynamic data.
Xu et al. [2013a]	Integrates historical QoS and geographical data.	Ineffective in real-time scenarios.
Xu et al. [2013b]	Introduced L-PMF and WL-PMF for collective intelligence.	Heavily reliant on data quality.
Yu et al. [2014]	Addressed cold-start problem with latent neighbor features.	Computationally intensive.
Tang et al. [2016]	Combines MF with a network map for user distances.	Extensive preprocessing required.
Xu et al. [2015]	Reputation-based MF using user reputation metrics.	Potential bias from reputation metrics.
Su et al. [2016]	Combines memory-based and model-based CF with EM learning.	High computational demands.
Chen et al. [2016]	Neighborhood Regularized MF for sparse datasets.	Sensitive to parameter tuning.
Xie et al. [2016]	Hybrid Asymmetric Correlation Regularized MF.	Computationally demanding.
Zhang et al. [2011a]	Combines neighborhood-based and latent factor approaches.	Limited generalizability.
Zhang et al. [2023]	Integrates MF with self-attention for dynamic scenarios.	High computational resource requirements.

4.3 Context-aware QoS prediction methods

Context-aware approaches acknowledge that various contextual factors, including user preferences, service history, and ambient conditions, can significantly impact QoS attributes Hamzei et al. [2023], Zeng et al. [2023]. By incorporating contextual information into the prediction process, these methods enhance the accuracy and personalization of WSQP, enabling more precise and user-specific QoS estimations Chen et al. [2017a], Wu et al. [2018b].

4.3.1 Location-aware QoS prediction methods

Location-aware QoS prediction methods utilize geographical proximity between users and services to improve prediction accuracy, as QoS attributes like response time, availability, and throughput are directly influenced by network performance and service distance Wu et al. [2018a], Abdullah and Bhaya [2021], Yin et al. [2014]. Several studies have contributed to this field, each employing unique methodologies to refine QoS predictions. In Mezni et al. [2021], Dilated RNNs were utilized to construct a Context-Aware Service Knowledge Graph (C-SKG), which was then mapped to a low-dimensional vector space to enhance computational efficiency. While effective in handling sparse data, the model's complexity may limit its scalability in real-time applications. A location-based regularization framework integrated with Matrix Factorization (MF) was proposed in Lo et al. [2012a] to improve WSQP by leveraging local service connectivity, achieving high accuracy in geographically influenced scenarios but struggling in datasets that lack location metadata. Similarly, the Local Neighborhood MF (LoNMF) method Lo et al. [2015] applies domain knowledge and a two-level neighborhood selection process for improved predictions, though it performs well on small datasets, it faces scalability issues in larger environments. A deep neural model (DNM) for multi-context QoS prediction, integrating location and service history, was introduced in Wu et al. [2018b]. This approach effectively incorporates multiple contextual factors, but its computational intensity presents challenges for large-scale applications. In Liu et al. [2015], a personalized CF method that considers user and WS locations was proposed to refine neighbor selection for WS recommendations, tailoring predictions effectively but lacking adaptability in dynamic temporal contexts. To enhance scalability and prediction quality in dynamic environments, Yu and Huang [2016] introduced time-aware and location-aware CF techniques, which effectively handle geographical and temporal variability but require extensive historical data. A geographical clustering-based MF model was proposed in Chen et al. [2017c] to improve prediction accuracy by clustering neighbors based on location similarities, proving effective in location-aware scenarios, though it remains computationally demanding. Further refining location-aware WSQP, the GNMF framework Chen et al. [2017b] integrates hierarchical clustering and geographical information for personalized predictions, demonstrating effectiveness in hierarchical datasets while performing suboptimally in flat data structures. Ensemble models that incorporate user and service contextual data were introduced in Xu et al. [2016b] to enhance prediction accuracy, though they may be prone to overfitting in sparse datasets. A two-tower deep residual network, NCRL, was developed in Zou et al. [2022] to employ location-aware collaborative

learning for WSQP, excelling in capturing complex relationships but being resource-intensive, making its implementation challenging in dynamic environments. Similarly, GAIN-QoS, designed for edge computing environments, was proposed in Choi et al. [2022], utilizing clustering and Generative Adversarial Imputation Nets (GAIN) to address missing data. While robust against data sparsity, it suffers from computational overhead concerns. A more general context-sensitive matrix factorization approach (CSMF) was introduced in Wu et al. [2018a], integrating user-to-service and environment-to-environment interactions while accounting for both implicit and explicit contextual factors. Although it improves prediction accuracy across diverse scenarios, its computational complexity can be a limitation, especially for large-scale datasets. The Context-Aware Services Recommendation (CASR) method Kuang et al. [2012] employs Bayesian inference to group service invocation records based on context attribute similarity, enabling nuanced relationship modeling and improving probabilistic predictions. However, its reliance on accurate context grouping can result in performance degradation when dealing with noisy or incomplete context data. Additionally, the PSO-USRec approach Chen et al. [2023] improves Particle Swarm Optimization (PSO) by reducing outlier particles and diversifying initial local solutions, enhancing robustness in WSQP. Despite its strengths, scalability challenges arise as the number of users and services increases, necessitating additional optimizations to mitigate computational overhead.

Table 6.8 provides a summary of the key location-aware QoS prediction methods, outlining their methodologies, contributions, and limitations.

4.3.2 Time-aware and Time-series QoS Prediction Methods

Time plays a crucial role in WSQP, as QoS attributes often fluctuate due to network conditions, service load, and temporal variations. To address these changes, time-aware and time-series QoS prediction methods have been developed to incorporate historical trends, temporal dependencies, and real-time fluctuations into QoS prediction.

Time-aware QoS Prediction Methods : The dynamic nature of service conditions and network performance significantly influences QoS attributes such as response time and availability, prompting researchers to explore WSQP methods that integrate **invocation time** to account for temporal variations Syu et al. [2017], Yan et al. [2022], Mezni and Fayala [2018]. To model complex user-service interactions, a temporal service knowledge graph (TSKG) approach was introduced in Mezni [2021], leveraging CNNs to extract top-rated services based on temporal interactions, though its high computational cost presents scalability challenges. Similarly, Mezni et al. [2020] proposed a context-aware WS recommendation framework, combining K-means clustering and PSO with Slope One for missing rating predictions ; while it achieves high accuracy in specific contexts, its generalizability across diverse datasets remains a challenge. In Zhang et al. [2011b], WSPred was developed as a time-aware QoS framework that personalizes QoS values without additional WS invocations, reducing real-time data collection needs but struggling in highly dynamic environments. To address

TABLE 6.8 Summary of Location-Aware QoS Prediction Methods

Study	Unique Contribution	Critical Evaluation
Mezni et al. [2021]	Dilated RNNs to construct C-SKG for computational efficiency.	Handles sparse data well but has scalability limitations.
Lo et al. [2012a]	Location-based regularization with MF for WSQP.	Effective for geographically influenced data; struggles without location metadata.
Lo et al. [2015]	LoNMF with domain knowledge for neighborhood selection.	Strong predictive capabilities but lacks scalability.
Wu et al. [2018b]	DNM integrating multi-context attributes.	Effective multi-context integration but resource-intensive.
Liu et al. [2015]	Location-aware CF for refining neighbor selection.	Tailored recommendations but lacks temporal adaptability.
Yu and Huang [2016]	Time and location-aware CF for dynamic environments.	Handles variability but needs extensive historical data.
Chen et al. [2017c]	Neighborhood clustering using geographical similarities.	Accurate but computationally demanding.
Chen et al. [2017b]	GNMF framework with hierarchical clustering.	Effective for hierarchical data; limited for flat datasets.
Xu et al. [2016b]	Ensemble models using contextual user and service data.	Versatile but prone to overfitting sparse datasets.
Zou et al. [2022]	NCRL with a two-tower residual network.	Excels in complex relationships but resource-intensive.
Choi et al. [2022]	GAIN-QoS combining clustering with imputation nets.	Robust against missing data but computationally expensive.

data sparsity, Hu et al. [2014] proposed a random-walk-based time-aware approach, integrating temporal information into user-service similarity calculations, effectively mitigating sparsity issues but increasing algorithmic complexity with larger datasets. A spatio-temporal WSQP model using sparse representation was introduced in Wang et al. [2016], improving QoS fluctuation modeling but facing limitations in capturing non-linear trends. To enhance real-time adaptability, Zhu et al. [2017] introduced an Adaptive Matrix Factorization (AMF) model, incorporating online learning and adaptive weights, making it highly responsive to real-time changes while being computationally demanding. A time-aware service recommendation system (taSR) in Ding et al. [2018] integrated ARIMA with collaborative filtering, effectively merging statistical and collaborative approaches, though it remains highly dependent on accurate historical data. A Biased Non-Negative Latent Factorization of Tensors (BNLFTs) model, introduced in Luo et al. [2019], utilizes time-varying data for temporal QoS prediction, excelling in time-sensitive contexts but requiring precise parameter tuning across different scenarios. In Jin et al. [2019a], a Two-Phase Dynamic Time-Aware WSQP (TWQP) method was proposed, balancing historical and real-time data for QoS predictions, ensuring robustness but suffering from high computational demands. To further capture temporal patterns, Hu et al. [2022] developed a dynamic graph neural collaborative learning framework, employing GRUs and GCNs to model user-service interactions, excelling in temporal representation but requiring significant computational resources for large-scale applications. Lastly, an outlier-resilient WSQP method in Ye et al. [2021] utilized Cauchy loss to manage temporal anomalies while predicting missing QoS values, demonstrating robustness against outliers but necessitating extensive fine-tuning to adapt to diverse datasets.

Time-series QoS Prediction Methods : are designed to handle sequential data, capturing trends, patterns, and seasonal fluctuations in QoS attributes such as throughput and response time, making them essential for understanding and forecasting dynamic QoS behaviors Syu et al. [2017]. In White et al. [2018b], Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) architectures were explored for future WSQP, effectively capturing long-term dependencies, though their high computational cost remains a challenge. Similarly, Genetic Programming (GP) was applied in Syu et al. [2015] for time-aware WSQP, demonstrating strong predictive capabilities and high adaptability, but requiring significant computational resources for training. In Zadeh and Seyyedi [2010], Time Series Forecasting (TSF) with Neural Networks was introduced to reduce monitoring costs and SOA overhead in WSQP, effectively minimizing operational expenses, though it may struggle to scale across diverse datasets. To address QoS attribute volatility, Amin et al. [2012a] employed a combination of ARIMA and GARCH models, improving forecasting accuracy, but requiring specialized expertise for model configuration. An automated forecasting approach, blending linear and nonlinear models for dynamic QoS attributes, was introduced in Amin et al. [2012b], demonstrating strong adaptability in dynamic environments while posing risks of overfitting in data-limited scenarios. The CI-ANFIS model, combining Clustered IOWA and ANFIS, was proposed in Hussain et al. [2022] for cloud QoS prediction, excelling in handling non-linear datasets, though it

demands significant computational power. More recently, the SCATSF framework was introduced in Zhou et al. [2023], integrating spatial and temporal contexts for WSQP using a SCA-GRU model for time series forecasting, effectively capturing spatio-temporal dynamics but remaining computationally expensive. Table 6.9 summarizes the key findings of Time-Aware and Time-Series QoS Prediction Methods, highlighting their contributions and limitations in dynamic service environments.

TABLE 6.9 Summary of Time-Aware and Time-Series QoS Prediction Methods

Study	Unique Contribution	Critical Evaluation
Mezni [2021]	Temporal knowledge graphs (TSKG) combined with CNNs for service recommendation.	Effective temporal modeling but high computational cost.
Zhu et al. [2017]	Adaptive MF with online learning for WSQP.	Real-time adaptability but resource-intensive.
White et al. [2018b]	LSTM and GRU mechanisms for future WSQP.	Strong sequential modeling but computationally expensive.
Amin et al. [2012a]	ARIMA and GARCH for volatility and accuracy in QoS forecasting.	Handles volatility well but requires expertise in tuning.

4.3.3 Trust-aware QoS Prediction Methods

Trust-aware QoS prediction methods address the issue of unrealistic QoS values recorded by users, incorporating trustworthiness between users and services to enhance WSQP accuracy Xu et al. [2016b]. These methods assume that users develop trust preferences for certain services based on historical interactions or feedback, integrating this trust information into the prediction process to improve reliability Wang et al. [2014]. In Tao et al. [2012], a trustworthy WSQP approach was proposed, extending UDDI and introducing the T_QoS algorithm with PAM clustering to enhance prediction accuracy, though its scalability remains a concern for large datasets. Similarly, a reputation-measuring approach for WS recommendations was introduced in Wang et al. [2014], utilizing the CUSUM control chart to detect malicious feedback ratings and PCC to mitigate user bias. While effective in filtering biased data, this method is computationally intensive for real-time applications. A Reputation-Aware WSQP approach (RAP) Qiu et al. [2013] was designed to rank users based on contributions, filtering untrusted feedback to improve prediction accuracy, though it may struggle in sparse environments. A social network-based service recommendation model was introduced in Deng et al. [2014b], leveraging the Relevant Trust Walker algorithm and MF to assess intra-user trust, improving recommendation precision but demanding high computational resources for large-scale networks. Similarly, Deng et al. [2014c] proposed a Trust-Based Service Recommendation System (TSR), which establishes trust ties between users and WSs to generate personalized recommendations. While excelling in personalization, it requires extensive trust data for accurate predictions. The User-Trust Propagation (UTP) model was presented in Thinh and Tu [2017], integrating trust propagation

and user evaluations with MF training, improving prediction accuracy but facing challenges with data sparsity. A decentralized service discovery technique was proposed in Kalaï et al. [2018], combining trust detection and collaborative service recommendation, effectively merging trust and collaboration, though requiring high computational resources. The Data-Aware Latent Factor (DALF) model, introduced in Wu et al. [2019], incorporated QoS data characteristics and used the DPClust algorithm to identify neighborhoods and untrustworthy data providers, making it robust against untrustworthy data but computationally expensive for large-scale clustering. A trust-aware clustering-based technique (TACF) was proposed in Liu et al. [2015] to personalize cloud manufacturing QoS forecasts, integrating local and global trust values into the k-medoids algorithm. While it balances local and global trust, it may struggle with high-dimensional data. Finally, the Trust-Aware Prediction (TAP) method in Su et al. [2017] incorporated clustering and reputation systems for personalized WSQP, improving reliability through user trust metrics but failing to effectively manage biased or incomplete reputation data. Table 6.10 summarizes the key findings of Trust-Aware QoS Prediction Methods, outlining their contributions, limitations, and effectiveness in trust-based WSQP.

TABLE 6.10 Summary of Trust-Aware QoS Prediction Methods

Study	Unique Contribution	Critical Evaluation
Tao et al. [2012]	Introduced T_QoS algorithm using PAM clustering for trustworthy WSQP.	Improves accuracy but scalability issues exist for large datasets.
Wang et al. [2014]	Reputation measuring approach mitigating feedback bias with PCC.	Strong filtering of biased data but computationally intensive.
Qiu et al. [2013]	RAP calculates user reputation and filters untrusted contributions.	Reliable for untrusted data but struggles with sparse contributions.
Deng et al. [2014b]	Social network-based recommendation using Relevant Trust Walker.	Robust social trust integration but computationally demanding.
Deng et al. [2014c]	TSR investigates trust ties for personalized recommendations.	Excels in personalization but requires extensive trust data.
Thinh and Tu [2017]	UTP predicts QoS using trust propagation and MF.	Accurate but faces challenges with data sparsity.
Kalaï et al. [2018]	Decentralized discovery combining trust detection and collaborative approaches.	Effective but computationally intensive.
Wu et al. [2019]	DALF model identifies untrustworthy data using DPClust.	Strong in handling untrustworthy data but resource-intensive.
Liu et al. [2015]	TACF incorporates task similarity and builds a trust network.	Balances local and global trust effectively but struggles with high-dimensional data.
Su et al. [2017]	TAP uses clustering and reputation systems for WSQP.	Improves reliability but faces challenges with biased reputation data.

5 Classification of the methods and discussion

This section focuses on classifying and analyzing state-of-the-art methods previously summarized in Section 4. The discussion begins with an overview of the evaluation metrics used in WSQP research in both static and dynamic environments, followed by a detailed examination of the datasets employed in these studies. Furthermore, a systematic classification of the methods is provided, considering their performance, applicability in static or dynamic environments, and the significance of fuzzy tools in enhancing predictive model accuracy.

5.1 Evaluation Metrics

QoS prediction studies utilize various evaluation metrics to measure prediction accuracy, with Mean Absolute Error (MAE), Normalized Mean Absolute Error (NMAE), and Root Mean Squared Error (RMSE) being among the most commonly applied criteria Hyndman and Koehler [2006], Goldberg et al. [2001].

— **MAE** : Rather than emphasizing classification accuracy or classification error, the evaluation focuses on computing the average absolute difference between predicted and actual ratings to assess prediction accuracy.

$$MAE = \frac{1}{N} \sum i, j \left| \hat{q}_{i,j} - q_{i,j} \right| \tag{6.12}$$

— **NMAE** : Since recommendation systems (RSs) utilize various numerical rating scales, Normalized Mean Absolute Error (NMAE) normalizes MAE, allowing errors to be expressed as percentages of the full scale.

$$NMAE = \frac{MAE}{r_{max} - r_{min}} \tag{6.13}$$

— **RMSE** : Widely used as a performance metric in movie recommendation systems, Root Mean Squared Error (RMSE) gained particular significance as a key evaluation criterion in the Netflix Prize competition [3].

$$RMSE = \sqrt{\frac{\sum i, j (\hat{q}_{i,j} - q_{i,j})^2}{N}} \tag{6.14}$$

Where :

— $q_{i,j}$ is the actual value

— $\hat{q}_{i,j}$ is the QoS property's predicted value

— N is the number of predicted QoS values

— r_{min} and r_{max} are the upper and the lower bounds of the ratings

While MAE and NMAE are intuitive and straightforward, they fail to penalize larger errors adequately. RMSE, though sensitive to large deviations, may overemphasize outliers. These metrics could be

3. Netflix Prize, http ://www.netflixprize.com/.

supplemented with additional measures like Mean Absolute Percentage Error (MAPE) to offer a more comprehensive evaluation.

The Mean Absolute Percentage Error (MAPE) is calculated using the following equation :

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100 \tag{6.15}$$

where :

— n represents the total number of predictions,

— y_i denotes the actual QoS value,

— $\hat{y}_i$ is the predicted QoS value.

This metric expresses prediction errors as a percentage of actual values, making it particularly useful for comparing errors across different datasets and scales.

5.2 General Data-sets and real-world QoS data-sets

WSQP algorithm research often relies on large-scale real-world WS QoS datasets for comparative analysis, though in some cases, movie-rating datasets have also been used for experimental evaluations. The diversity of datasets employed in these studies has led to the development of a structured categorization based on their characteristics and application domains.

Subjective data-sets consist of values derived from users' subjective evaluations, which can be influenced by cognitive disposition and personal preferences Ma et al. [2015]. A widely used dataset in this category is MovieLens [4] Herlocker et al. [2000], frequently employed in WSQP research Zhang et al. [2013], Ma et al. [2015], Chen et al. [2020a]. This dataset collects user ratings for various movies, incorporating diverse rating scales, as summarized in Table 6.11.

Objective data-sets refer to the values associated with a service's objective attributes, such as response time, throughput, and reliability. These values, although not directly gathered from users, are derived from objective aspects that influence user experience. For example, when a user invokes a web service (WS), its accessibility may depend on objective factors like network traffic and bandwidth, over which the user has no control, as service availability is dictated by these factorsMa et al. [2015]. The most well-known datasets in this category include the following :

— **WSDream** Zheng et al. [2010a] : a real world data-set with over 2.5 million WSs. This data-set collection is publicly available as it was collected from real-world WSs. It comprises static and dynamic QoS data-sets :

4. https ://grouplens.org/datasets/movielens/

TABLE 6.11 Web Services QoS data-set

Reference	Data-set	Number of Services	Number of Users
Herlocker et al. [2000]	MovieLens 100K	1700 movies	1000
GroupLens	MovieLens 25M	62000 movies	162000
GroupLens	MovieLens 1B synthetic data-set	855776 movies	2 197 225
Zheng et al. [2010a]	WSDream	100	150
Zheng et al. [2010b]	WSDream	339	21 358
Zhang et al. [2011b]	WSDream	4 532	142
Al-Masri and Mahmoud [2008]	QWS	2507	1
Shao et al. [2007]	Real-world records	20	136
Silic et al. [2013]	RESTful	49	50

  - **The static ones** include real-world QoS measurements, this data-set Zheng et al. [2010a] has 150 files and each file involves 10000 WS invocations on 100 WSs by a service user. This results in more than 1,5 million WS invocations. The second version of this data-set Zheng et al. [2010b] includes response time and throughput values, it was gathered from 339 users on 5825 WS.
  - **The dynamic one** aggregates real-world QoS measurements from 142 users on 4500 WSs over 64 consecutive time slices with 15-minute interval. This dynamic data-set also contains records of response time and throughput values Zhang et al. [2011b].
- **Quality of Web Service (QWS)** Al-Masri and Mahmoud [2008] : involves a single service user interacting with 2,507 WSs. This data-set has one limits which rests in the fact that various users will observe different QoS for the same WS.
- **Real-world records** Shao et al. [2007] : Records were collected from 136 users for 20 real-world WSs , each user invoked each service an average of 200 times.
- **RESTful** Silic et al. [2013] : This dataset encompasses a range of computational complexities, as services are located across different geographic locations worldwide. It involves 50 users and 49 services.

While CF initially started with subjective data types, QoS data related to WSs, tends to rely on the objective type. This preference arises from the substantial differences between these two types, which can significantly impact the accuracy of predictions, as highlighted in Ma et al. [2015]. For additional details on the QoS datasets, readers can refer to the summary in Table 6.11. Subjective datasets effectively capture user preferences but are susceptible to bias and inconsistencies, while objective datasets offer reliable QoS attributes, though they may lack personalization. Hybrid approaches that combine both data types can improve prediction models by balancing the objectivity of the former with the user-centric insights of the latter.

Relevance of real-world QoS data-sets in static and dynamic predictions

— Objective data-sets play a crucial role in static Web Service QoS Prediction (WSQP) by offering a historical perspective on service performance. Analyzing past performance data enables static prediction models to identify trends, patterns, and correlations, facilitating predictions of future QoS values. In dynamic environments, these datasets serve as a baseline, helping models detect anomalies or deviations from historical performance. By combining objective data with real-time information, dynamic models can adjust predictions to fit the evolving environment.

— Subjective data-sets complement objective data in static predictions by providing insights into user preferences, allowing static models to incorporate user-centric elements. In dynamic environments, however, user feedback becomes a key QoS attribute. Dynamic models leverage sentiment analysis and user ratings to adjust predictions in real-time, reflecting the rapidly changing conditions.

5.3 Discussion of the classified methods

Table 6.12 classifies various methods discussed in Section 4, emphasizing the interconnected nature of these techniques. Many studies integrate multiple approaches within a single framework, utilizing their combined strengths to improve prediction accuracy and adaptability.

TABLE 6.12 Comparative Table of Distinguished CF-Based Prediction Approaches

Reference	Year	Contribution	Dataset	Evaluation Metrics	Prediction Methods
Shao et al. [2007]	2007	User-based similarity	QoS data	Mean Difference Comparison	User-based
Zheng et al. [2009]	2009	Combined user and service similarity	WSRec	MAE, NMAE	User-based and Service-based
Zheng et al. [2010a]	2010	Linear combination of user and service similarity	WSRec	MAE, RMSE	User-based and Service-based
Yin et al. [2014]	2014	Location-aware regularization in MF	WSRec	MAE, RMSE	MF, Location-aware, User-based
Chen et al. [2015]	2015	Clustering using user-QoS patterns	WSRec	MAE, RMSE	Clustering, User-based
Thinh and Tu [2017]	2017	User trust integrated with MF	WSDream	RMSE	Trust-aware, MF

Reference	Year	Contribution	Dataset	Evaluation Metrics	Prediction Methods
Ren and Wang [2018]	2018	SVM for service recommendation	WSDream	Classification Accuracy, Precision, Recall	ML, Clustering
Tong et al. [2021]	2021	Time-aware similarity calculations	WSDream	MAE, RMSE	Time-aware, Model-based
Choi et al. [2022]	2022	Imputation nets for edge environments	WSDream	MAE, RMSE	Clustering, Location-aware
Jiang et al. [2011]	2011	Personalized recommendations for WSs	WSRec	MAE, Precision, Recall, F1-Score	User-based and Service-based
Zhou et al. [2019]	2019	Incorporates spatial attributes with ML	WSRec	MAE, RMSE	ML, Time-aware
Lo et al. [2012a]	2012	Location-based regularization CF	WSDream	MAE, RMSE	Location-aware, Matrix Factorization
Lo et al. [2015]	2015	Local neighborhood with MF	WSDream	RMSE	User-based, Location-aware, MF
Chen et al. [2020a]	2020	Big range-aware CF	WSDream	MAE, RMSE	User-based, Location-aware, Clustering
Chen et al. [2023]	2023	Swarm intelligence techniques for neighbor detection	WSDream	MAE, RMSE	Location-aware, Swarm Intelligence
White et al. [2018a]	2018	Metrics for IoT-based similarity	WSDream	MAE, RMSE, NPRE	User-based, Service-based
Zhu et al. [2024]	2024	Dynamic location-based for user/service updates	WSDream	MAE, RMSE	Location-aware, Dynamic Prediction
Amin et al. [2012a]	2012	Combined ARIMA and GRAPH models	WSRec	MAE, RMSE	Time-series, Static Prediction
Amin et al. [2012b]	2012	Automated forecasting with ARIMA and SETARMA	WSRec	MAE, RMSE	Time-series, Static Prediction
Qiu et al. [2013]	2013	Reputation-aware ranking for QoS values	WSRec	MAE, RMSE	Trust-aware, User-based, Service-based

Reference	Year	Contribution	Dataset	Evaluation Metrics	Prediction Methods
Wang et al. [2014]	2014	CUSUM and bloom filter for malicious detection	WSRec	Trust Metrics	Trust-aware, User-based
Xu et al. [2023]	2023	Federated learning framework for QoS predictions	WSRec	TCR, TCD	ML, MF, Dynamic Prediction

5.3.1 QoS Predictions in Static and Dynamic Environments

In static WSQP scenarios, predictions are made within fixed time slices, relying heavily on contextual factors such as user locations and network information. However, CF techniques face challenges in this setting. For instance, location-aware and neighborhood-based CF algorithms assume stable similarity relationships between users or services, typically relying on historical data of similar users, often derived from past QoS values. In contrast, model-based CF approaches incorporate implicit relationships by applying specific models to WSQP. This approach addresses RQ1.

In real-world scenarios, however, the situation is more complex. Over different time slices, users interact with the same services, but the QoS values for these services fluctuate. Accurately predicting WSQP for future time slices becomes essential to meet user demand for high-quality services, driving the development of dynamic QoS prediction algorithms that use historical QoS data features to forecast future QoS values. This approach addresses RQ2.

As noted by Yan et al. [2022], time-series prediction algorithms often fall short in improving WSQP results due to the need for manual intervention and specialized knowledge in feature extraction. Unlike traditional methods, deep learning-based WSQP approaches—specifically deep NNs—are effective in capturing time-series characteristics, reducing technical complexity, and significantly enhancing prediction accuracy. These dynamic methods leverage historical QoS values to forecast future values by modeling temporal dependencies. However, two main challenges of conventional deep learning-based WSQP techniques include :

— **Limited real-time adaptability** : These models often struggle to adjust to real-time QoS values generated by users during service invocation.

— **Inefficient integration of historical data** : These models encounter difficulties in incorporating past QoS data during training, thus diminishing prediction accuracy.

To address these limitations, advanced methodologies like edge computing and adaptive online learning frameworks are emerging. Edge computing facilitates real-time QoS data processing closer to the source, while adaptive online learning enables dynamic model updates as new data becomes available. These approaches improve the relevance and responsiveness of WSQP systems, thus addressing RQ3.

The classification of prediction methods is crucial for understanding their strengths and weaknesses. Researchers often face the dilemma of choosing between static and dynamic methods, with the choice being influenced by application requirements and dataset characteristics. While static methods offer computational efficiency and simplicity, dynamic approaches are better suited to scenarios where user preferences and service quality change over time. It's important to note that the classification of these methods is not fixed, but rather dependent on their adaptability to changing conditions. Therefore, the decision to use static or dynamic methods depends on factors such as the nature of the problem, data availability, and the level of adaptability required in the WSQP system.

Table 6.13 presents a comprehensive comparison of three distinct categories of prediction methods used in WS QoS prediction : "Memory-based methods," "Model-based methods," and "Context-based methods." Each category is carefully analyzed, outlining its advantages and disadvantages, offering valuable insights to researchers and practitioners in selecting the most suitable method based on their specific challenges and requirements.

Memory-based QoS Prediction Methods use historical data from user-service interactions to make accurate WSQP predictions. However, they face several limitations, especially in static and dynamic environments, such as :

— **Scalability** : With large datasets, the computational complexity of calculating and updating similarities for all users and services can become overwhelming Yu and Huang [2016], Zheng et al. [2020]. Memory-based methods struggle to adapt to dynamic shifts in user preferences and service behaviors over time. Static memory-based models also fail to capture changes in user behavior or service performance accurately.

— **Data Sparsity** : Limited user interactions with services result in sparse data, making it difficult to find sufficient similar neighbors for accurate predictions Wu et al. [2018a], Zheng et al. [2020]. This problem intensifies as the dataset grows, with sparsity causing inaccurate predictions in static environments and making adaptation to new patterns difficult in dynamic environments.

— **Cold Start** : In static environments, the issue when new users or services have limited interaction history Chen et al. [2017c], Zhu et al. [2023]. In dynamic environments, this challenge persists with the continuous introduction of new entities into the system.

— **Data Quality** : Incomplete or unreliable QoS data can skew similarity calculations, leading to inaccurate predictionsWu et al. [2020]. Outliers in the data further impact prediction qualityYe et al. [2021].

— **Similarity Metric Choice** : The selection of an appropriate similarity metric is crucial for accurate predictions. Using an unsuitable metric can degrade performanceWu et al. [2018b], particularly when dealing with diverse or dynamic QoS attributes White et al. [2018a].

— **Data Temporal Dynamics** : Dynamic environments introduce temporal changes in QoS valuesLuo et al. [2019], which can significantly affect predictions. Memory-based methods do

TABLE 6.13 Advantages and Disadvantages of Different Prediction Method Categories

Prediction Method Category	**Advantages**	**Disadvantages**
Memory-based methods	— Simple usage and explainable recommendations — High accuracy when sufficient data is available	— Cold start problem with new users or services — Scalability issues — Low accuracy due to sparse invocation matrices — High computational time
Model-based methods	— High scalability tolerance — Mitigates cold start problem — Models complex features — Reduced computational time for clustering — High accuracy	— Difficult feature extraction — Accuracy dependent on parameter settings — Long computation time for model learning — Need for model reconstruction with new users or services
Context-based methods	— Improved accuracy — Mitigates cold start and data sparsity issues — Considers dynamic nature of QoS — Data validation	— Requires additional parameter configuration — High computational time — Periodic invocation data required

not explicitly model these temporal aspects, potentially leading to outdated predictionsZheng et al. [2020].

Researchers are addressing these challenges by developing techniques that enhance WSQP accuracy Herlocker et al. [2004], Goldberg et al. [1992], Zheng et al. [2020], including :

— Model-based approaches, which outperform memory-based methods by addressing issues like cold start and data sparsity, and incorporating contextual information when new users or services are introduced.

— Clustering techniques, which reduce computational time by grouping similar users and services together, improving scalability and accuracy.

— Incorporating time-aware models to capture the dynamic nature of QoS values and ensure relevance in predictions.

— Online prediction and trustworthiness models, which balance computational efficiency and prediction accuracy.

— Machine learning techniques such as autoencoders and deep learning, which provide enhanced accuracy by modeling complex relationships in QoS data.

Model-based CF Methods Mathematical models are created to capture patterns in user-service interactions, but they encounter several challenges :

— **Overfitting** : Occurs when a model learns noise rather than meaningful patterns, especially in static environments with limited data Yin et al. [2020], Wu et al. [2018b]. It reduces the prediction accuracy.

— **Feature Extraction/Engineering Difficulty** : Model-based methods require complex feature extraction from data. In dynamic environments, shifting data distributions complicate feature selection, making it hard to adapt to emerging patterns Yin et al. [2020], Zou et al. [2022].

— **Parameter Dependency** : The accuracy of model-based methods is highly sensitive to the selection of parameters Wu et al. [2018b], such as latent factors or regularization terms, which require significant fine-tuning.

— **Computation Time** : Training model-based approaches can be computationally expensive, particularly with large datasets Lo et al. [2012a].

— **Model Reconstruction** : In static environments, adding new users or services necessitates retraining or rebuilding the model Wang et al. [2016], which is time-consuming.

To improve WSQP accuracy, researchers are combining model-based techniques with context-based filtering, hybrid models, and reinforcement learning, leveraging the strengths of each approach to overcome their limitations.

Hybrid Approaches Multiple models are combined to improve accuracy and adaptability. For instance, a memory-based method may be enhanced by a model-based approach to alleviate cold-start issues, or deep learning models could be used alongside simpler algorithms to capture both long-term and short-term patterns in QoS data. Emerging hybrid techniques combining traditional methods with ML models hold great promise for improving QoS prediction in both static and dynamic environments. As research progresses, more adaptive, accurate, and scalable models are anticipated. Subsequently, this answers the RQ4.

The classification of these integrated methods as static or dynamic depends on their implementation and the extent to which they incorporate temporal or contextual factors in their predictions, as illustrated in Table 6.14.

TABLE 6.14 Classification of Methods in Selected Studies

Reference	Static Methods	Dynamic Methods	Final Evaluation
Shao et al. [2007]	User-based		Static prediction
Zheng et al. [2009]	User-based and service-based		Static prediction
Zheng et al. [2010a]	User-based and service-based		Static prediction
Yin et al. [2014]	MF and user-based	Adapted location-aware	Dynamic prediction
Chen et al. [2015]	Clustering and user-based		Static prediction
Chen et al. [2017c]	User-based, service-based, clustering, MF	location-aware	Dynamic prediction
Ren and Wang [2018]	Clustering and Machine Learning		Static prediction
Thinh and Tu [2017]	Matrix Factorization	Trust-aware	Dynamic prediction
Tong et al. [2021]	Model-based	time-aware	Dynamic prediction
Chen et al. [2023]		location-aware, swarm intelligence techniques	Dynamic prediction
Choi et al. [2022]	Clustering	Adapted location-aware	Dynamic prediction
Lo et al. [2012a]	Location-based regularization CF		Static prediction
Lo et al. [2015]	Matrix Factorization	Adapted location aware	Dynamic prediction
Qi et al. [2015]	Matrix Factorization	Adapted user network and service neighborhood	Dynamic prediction
Zhou et al. [2019]	Machine learning	time-aware	Dynamic prediction
Jiang et al. [2011]		Adapted user-based and service-based	Dynamic prediction
Wu et al. [2012]	Location-aware and Clustering		Static prediction
Amin et al. [2012a]	ARIMA and GRAPH models		Static prediction
Amin et al. [2012b]	ARIMA and SETARMA models		Static prediction
Qiu et al. [2013]	User-based, service-based, trust-aware		Static prediction
Wang et al. [2014]	trust and reputation metrics		Static prediction
Zhu et al. [2024]		Location-aware for user movements or service updates	Dynamic prediction

Based on the existing literature, only a few studies tackle the ambiguity in QoS values. Consequently, the focus has shifted toward developing fuzzy personalized WSQP systems. These systems utilize various features, including individual user characteristics, trustworthiness, available services, environmental factors, response times, and contextual information such as location. The goal of this approach is to better capture the dynamic and uncertain nature of QoS data in real-world scenarios.

5.3.2 Importance of Fuzzy Tools in WSQP

The need to address uncertainty in WSQP has gained increasing attention within the academic community. A notable study by Yera and Martinez [2017] explored the integration of fuzzy logic tools into recommender systems (RS) to handle the ambiguity and variability of QoS values, improving prediction accuracy and adaptability. Several fuzzy tools have gained widespread application in this context, including :

— **Fuzzy C-Means Clustering (FCM) :** An unsupervised learning algorithm that partitions a dataset into C clusters, allowing data points to belong to multiple clusters with varying degrees of membership, thus enhancing flexibility in dealing with uncertain data.

— **Fuzzy Logic-based Recommendation :** Utilizes fuzzy logic principles to manage uncertainty and imprecision in user preferences and item characteristics, enabling recommendations based on vague or subjective user input.

— **Fuzzy Inference System (FIS) :** A framework for reasoning with fuzzy logic, using fuzzy sets, rules, and inference methods to map inputs to outputs, making it effective in capturing complex relationships.

— **Fuzzy k-Nearest Neighbors (FkNN) :** Extends the k-NN algorithm by incorporating fuzzy logic to address uncertainty in the classification of data points.

— **Fuzzy Association Rules :** Discovers interesting relationships between variables in datasets where data is uncertain or imprecise.

Fuzzy Logic and Uncertainty Conventional WSQP models often treat QoS values as deterministic, overlooking their inherent variability due to factors such as network congestion, service provider changes, and user behaviors. Fuzzy logic addresses this limitation by representing QoS values as ranges rather than fixed points, offering a more accurate model for the dynamic and ambiguous nature of real-world data Seghir and Khababa [2021a].

Traditional QoS Prediction vs. Fuzzy Approach Traditional WSQP models aim for accuracy based on historical data but typically fail to account for the inherent fuzziness of QoS values, compromising prediction accuracy, particularly in dynamic environments. Fuzzy logic improves this by :

— Capturing imprecise QoS attributes, thus enhancing prediction reliability.

— Offering personalized predictions that align with users' subjective preferences.

These advancements underscore the indispensable role of fuzzy tools in handling the complexity and uncertainty of WSQP, enabling more accurate, personalized, and adaptive predictions.

5.3.3 WSQP Techniques in Practical Situations

WSQP plays a crucial role in service-oriented architectures and cloud environments, where performance, reliability, and scalability are essential. In cloud service selection, QoS prediction helps identify the optimal provider based on metrics such as response time, throughput, and availability. For example, e-commerce platforms He and He [2021] may switch cloud providers during peak traffic to maintain optimal performance. In service composition, QoS prediction aids in selecting the best combination of services for tasks such as booking systems that integrate flight, hotel, and payment services to reduce latency and improve reliability.

WSQP is also critical for real-time service monitoring, especially in high-stakes systems like financial platforms, healthcare, and telecommunications, where performance lags can trigger proactive service switching. For example, in healthcare systems, QoS prediction anticipates delays in critical services, ensuring timely responses. Additionally, QoS prediction enhances recommendations, helping users or developers select the most efficient services based on anticipated performance. It also aids in dynamic load balancing by forecasting server loads and optimizing traffic routing. In content delivery networks, QoS prediction ensures faster access by predicting response times.

6 Future Directions and Open Issues

In this section, we highlight key open issues and suggest potential directions for future research.

— **Incorporation of Fuzzy Logic :**
  — Hybrid systems that combine fuzzy logic with neural networks or reinforcement learning can further enhance adaptability and scalability, aligning predictions more closely with the dynamic nature of web services and user expectations. This area remains critical for innovation.
  — Despite increasing adoption, many studies still overlook the importance of addressing uncertainty in QoS values.
  — One challenge is the computational complexity involved in integrating fuzzy logic with deep learning, especially in real-time scenarios.
  — Standardized evaluation frameworks are needed to assess the impact of fuzzy systems on QoS prediction accuracy.

— **Emerging Trends in WSQP :**
  — **Advanced ML and AI Techniques :** Deep learning (DL) and reinforcement learning are employed to capture complex patterns and dynamically optimize QoS predictions.

However, the computational cost and data requirements of these models remain significant challenges.

- **Big Data Analytics :** Frameworks such as Hadoop and Spark facilitate efficient processing and analysis of large-scale QoS datasets, but ensuring data integration and consistency across different sources is crucial for leveraging these technologies effectively.
- **Hybrid Approaches :** Combining multiple predictive techniques is increasingly common to overcome the limitations of single-method approaches. For instance, integrating memory-based and model-based methods can improve scalability and accuracy.
- **Real-Time Monitoring :** Adaptive prediction systems are gaining importance for maintaining relevant predictions under changing network conditions. However, achieving low latency while maintaining prediction accuracy remains an open issue.

— **Security and Privacy in WSQP :**

- **Privacy-Preserving Techniques :** Federated learning is being adopted to address security concerns in QoS predictions. Challenges include maintaining model performance while minimizing data leakage risks.
- **User Feedback and Crowdsourcing :** These approaches enhance the quality of QoS datasets but raise concerns about data reliability and trustworthiness. Incorporating blockchain technology could help ensure data integrity.

— **Technological Enhancements :**

- **Edge and Fog Computing :** These technologies enable faster processing and real-time updates by placing models closer to data sources. However, issues such as resource constraints and load balancing require further investigation.
- **Blockchain Technology :** Blockchain ensures the integrity and authenticity of QoS data, fostering trust in web service environments. Key challenges to its widespread adoption include scalability and energy consumption.

— **Interdisciplinary Research :**

- Combining insights from human-computer interaction and cognitive science can improve the usability and interpretability of WSQP systems.
- Cross-domain applications, such as integrating WSQP into IoT ecosystems or healthcare services, represent promising but underexplored areas.

— **Evaluation and Benchmarking :**

- Developing standardized metrics and benchmarking frameworks to evaluate new WSQP techniques is essential for their adoption.
- Comparative studies across different datasets and real-world environments can provide valuable insights into the scalability and robustness of various methods.

These advancements are shaping the next generation of intelligent and user-centric WS quality assessments, while addressing the outlined challenges will be crucial for their successful implementation.

7 Conclusions

This chapter has provided a systematic review of collaborative filtering (CF) techniques for predicting web service Quality of Service (QoS) in both static and dynamic environments. Following PRISMA guidelines, we screened 512 records and conducted in-depth analysis on 146 studies. Our findings revealed the strengths and limitations of traditional CF methods, which are effective in static environments but struggle with data sparsity, scalability, and high bias/variance when applied to dynamic scenarios.

A key insight from the review is that incorporating temporal information and context-aware models leads to significant improvements in predictive performance. Hybrid models that combine time-series analysis, deep learning, and contextual filtering demonstrate considerable reductions in error metrics, effectively bridging the gap between static and dynamic prediction approaches. This validates our proposed taxonomy, which emphasizes the integration of time-aware techniques to address fluctuating QoS values.

Furthermore, our analysis highlights the adaptability and robustness of dynamic models that utilize real-time data, compared to static models reliant on historical data. This shift underscores the importance of developing more flexible and scalable prediction techniques in the face of modern web service complexities.

Despite these advancements, several challenges remain, including computational inefficiencies, limited access to large-scale datasets, and the need for improved model interpretability. Moving forward, research should focus on scalable, privacy-preserving techniques such as federated learning and distributed machine learning, while also exploring promising methods like graph neural networks, reinforcement learning, and self-supervised learning. These innovations hold potential to enhance predictive accuracy and broaden the applicability of web service QoS prediction models in real-world scenarios.

Although this chapter marks the final section of the thesis, it sets the stage for future work. We will implement a dynamic, time-aware web service QoS prediction method based on the insights derived from this review. This practical implementation aims to advance the field by integrating the discussed approaches into a robust, adaptive system capable of accurately predicting QoS in real-time, dynamic environments.

Chapter 7

General conclusion and Future Work

This thesis introduces three significant contributions to the QoS-aware service composition problem (QSC), which is recognized as an NP-complete problem. While the QSC has been acknowledged as NP-complete, there has been relatively limited research aimed at resolving the QSC considering imprecise or uncertain QoS values. The majority of existing contributions have primarily focused on formulating and solving the problem with precise QoS values incorporated into the optimization processes.

1 Contributions and research summary

This section aims to provide a comprehensive overview summarizing the diverse contributions presented in our research work. The thesis encompasses three principal contributions, each contributing significantly to the advancement of the field :

— An extended artificial bee colony with local search for solving the Skyline-based web services composition under interval QoS properties.

— An Improved Discrete Flower Pollination Algorithm for Fuzzy QoS-aware IoT Services Composition Based on Skyline Operator.

— Collaborative Filtering Techniques for Predicting Web Service QoS Values in Static and Dynamic Environments : A Systematic and Thorough Analysis

1.1 An extended artificial bee colony with local search for solving the Skyline-based web services composition under interval QoS properties

In this thesis, the simple interval number is applied to represent the ambiguity of the QoS values in solving the QoS uncertainty-aware web service composition problem. This later is modeled as an

interval constrained single-objective optimization (IQSC) model, while a new approach combining two components : skyline operator and an interval extended version of the basic artificial bee colony (EABC) algorithm, is shaped to address the formulated IQSC. The first component (skyline operator) is utilized to reduce the search space of IQSC by pruning the redundant and dominated web services from their sets of functionally equivalent ones. Whereas, the second component which is (EABC) is performed to obtain the optimal/near-optimal composite service of IQSC in a reduced search space. The experimental results, which have been performed on an interval extended version of the public QWS dataset, of comparing our proposed approach to an existing skyline-based PSO, an efficient discrete gbest-guided artificial bee colony and a recently provided Harris Hawks optimization with an elite evolutionary strategy algorithms, demonstrate and validate both the performance superiority and the efficiency of our introduced approach.

1.2 An Improved Discrete Flower Pollination Algorithm for Fuzzy QoS-aware IoT Services Composition Based on Skyline Operator

In this thesis, the generalized trapezoidal fuzzy number is applied to represent the ambiguity of the QoS values in solving the QoS uncertainty-aware IoT service composition problem. We have proposed an new approach that combines two modules : (1) a fuzzy skyline-based module and (2) an improved discrete flower pollination algorithm (IDFPA) to solve with rapidity and efficiency the QoS-aware IoT services composition problem under GTrFN-based QoS parameters (QSCFIoT), which is formulated as a fuzzy constrained single-objective optimization model. In the proposed approach, the fuzzy-skyline-based module is utilized to reduce the search space of QSCFIoT, whereas the IDFPA is employed to quickly look for near-optimal CSs with high-quality solutions in the reduced search space of QSCFIoT. Compared to some existing QoS-aware services composition algorithms (flower pollination algorithm, Particle Swarm Optimization and an improved teaching-learning-based QoS-aware services composition algorithm), experimental results, which have been performed on both synthetic and real datasets for the QSCFIoT with different scales, show that the proposed approach is efficient and obtains better CSs solutions.

In the QoS-aware IoT-services composition problem, the dynamic demand of users and the geographical states of the IoT devices will change over time, which may affect the composition quality in terms of QoS properties of the IoT-services composition algorithms. In future work, we further plan to investigate these hard constraints in the performance evaluation of such algorithms in dynamic IoT environments.

1.3 Collaborative Filtering Techniques for Predicting Web Service QoS Values in Static and Dynamic Environments : A Systematic and Thorough Analysis

In this thesis, a systematic literature review (SLR) was constructed to comprehensively examine collaborative filtering (CF) techniques for predicting Quality of Service (QoS) values in web services, with a focus on both static and dynamic environments. As the prediction is a crucial step for the QSC problem. Following the PRISMA guidelines, the study systematically screened 512 records and conducted full-text evaluations of 146 studies. The synthesis of these studies provided a consolidated view of the prevailing algorithmic approaches, evaluation metrics, benchmark datasets, and reported performance outcomes.

The findings reveal that while traditional CF methods demonstrate effectiveness in static contexts, they are often hindered by issues such as data sparsity, limited scalability, and susceptibility to high bias and variance, particularly in dynamic scenarios. In contrast, approaches that incorporate temporal and contextual information—such as time-aware models, context-based filtering, and hybrid frameworks integrating deep learning and time-series analysis—consistently outperform conventional models. These hybrid and context-enriched strategies have shown notable reductions in error metrics, underscoring the performance benefits of dynamic modeling in fluctuating environments.

The review supports a growing consensus that bridging the gap between static and dynamic prediction requires models that are both adaptive and aware of temporal and contextual changes. This observation validates the proposed taxonomy, which emphasizes time-aware modeling as a unifying framework for future research. Furthermore, it is evident that leveraging real-time, dynamic data yields greater adaptability and robustness compared to static models reliant on historical data. Visual analysis using VOSviewer further reinforces this classification, highlighting strong co-occurrence of emerging concepts in adaptive and predictive analytics with domain-specific keywords.

Nonetheless, several challenges persist. These include computational inefficiencies in complex models, the predominance of small-scale experimental datasets, and ongoing concerns related to model transparency and interpretability. To address these gaps and advance the field of Web Service QoS Prediction (WSQP), future research should prioritize the development of scalable and privacy-preserving methods, such as federated and distributed learning. In parallel, novel approaches including graph neural networks, reinforcement learning, and self-supervised learning offer promising directions for enhancing both prediction accuracy and real-world applicability.

In conclusion, this review not only synthesizes the current state-of-the-art in collaborative filtering for QoS prediction but also outlines a clear and structured roadmap for future inquiry. By promoting the integration of dynamic, context-aware, and scalable methodologies, the field can better meet the evolving challenges of modern web service environments.

2 Future Works and Perspectives

Following the notable contributions mentioned earlier, there remain several potential avenues for future contributions to the field of research. These perspectives are pivotal and could significantly contribute to the advancement of science and technology. Some of these perspectives include :

— **QoS Uncertainty-Aware Service Composition in Cloud and Fog Computing** : Future plans include extending the scope of solving QoS uncertainty-aware web service composition, specifically focusing on Cloud and Fog computing environments. This expansion involves considering more specialized QoS parameters and context-aware information, such as the geographical location of users and available web services.

— **Utilization of Contextual Information for Web Service Recommendation** : Incorporating contextual information like geographical location and the distribution of users and web service providers through personalized collaborative filtering techniques for web service recommendation Shao et al. [2007] is an upcoming area of exploration.

— **Development of Multi-Objective Versions of IQSC and QSCFIoT Models** : A focus will be on developing efficient multi-objective versions of the introduced IQSC and QSCFIoT models to handle various objectives simultaneously, providing a more comprehensive solution.

— **Exploration of Probabilistic Methods for QoS Uncertainty** : The current thesis primarily focused on interval and fuzzy numbers to represent QoS attribute uncertainty. Future investigations will delve into probabilistic methods to represent the impreciseness of QoS attributes in solving the QoS-uncertainty-aware service composition problem.

— **Implementation of Collaborative Web Services Selection Platform** : A planned endeavor involves the implementation and validation of an effective collaborative platform for web service selection and composition, validating its functionality in both synthetic and real environments.

— **Addressing Data Sparsity in Real-World QoS Datasets** : In consideration of the reference Mezni [2021], the interrelation between service selection and recommendation has been recognized as a significant element within service filtering procedures. Service selection involves identifying the most suitable candidate services based on their QoS (Quality of Service) and contextual criteria. Conversely, service recommendation refines selection outcomes by integrating additional factors like feedback, ratings, user similarities, service similarities, and more. The QWS (Quality of Web Services) dataset is static and lacks empty entries, differing from real-world datasets like the WSDream dataset in Zheng et al. [2009], which are characterized by sparsity and multiple missing values. Subsequently, future research endeavors will address this data sparsity challenge by developing a precise model capable of efficiently predicting missing QoS values.

Bibliography

M. N. Abdullah and W. S. Bhaya. Predicting qos for web service recommendations based on reputation and location clustering with collaborative filtering. In *2021 7th International Conference on Contemporary Information Technology and Mathematics (ICCITM)*, pages 19–25. IEEE, 2021.

A. N. Abosaif and H. S. Hamza. Quality of service-aware service selection algorithms for the internet of things environment : A review paper. *Array*, 8 :100041, 2020.

G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer, 2011.

M. I. Ahmad, M. H. Ab. Rahim, R. Nordin, F. Mohamed, A. Abu-Samah, and N. F. Abdullah. Ionizing radiation monitoring technology at the verge of internet of things. *Sensors*, 21(22) :7629, 2021.

M. Ahmed and Z. Jaaz. Wireless sensor networks for metropolitan scale explosives detection. *European Scientific Journal*, 10(15), 2014.

N. F. AL-Bakri and S. Hassan. A proposed model to solve cold start problem using fuzzy user-based clustering. In *2019 2nd Scientific Conference of Computer Sciences (SCCS)*, pages 121–125. IEEE, 2019.

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

A. Alphonsa and G. Ravi. Earthquake early warning system by iot using wireless sensor networks. In *2016 International conference on wireless communications, signal processing and networking (WiSPNET)*, pages 1201–1205. IEEE, 2016.

M. Alrifai, D. Skoutas, and T. Risse. Selecting skyline services for qos-based web service composition. In *Proceedings of the 19th international conference on World wide web*, pages 11–20, 2010.

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

A. AlSedrani and A. Touir. Web service composition processes : a comparative study. *International Journal on Web Service Computing (IJWSC)*, 7(1) :1–21, 2016.

J. Alston, D. Hess, and R. Ruggieo. Uddi (universal description, discovery, and integration) and web services : Perspective', 2002.

A. Amin, A. Colman, and L. Grunske. An approach to forecasting qos attributes of web services based on arima and garch models. In *2012 IEEE 19th International Conference on Web Services*, pages 74–81, 2012a.

A. Amin, L. Grunske, and A. Colman. An automated approach to forecasting qos attributes based on linear and non-linear time series modeling. In *2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, pages 130–139. IEEE, 2012b.

R. Amiri, J. M. Sardroud, and B. G. de Soto. Bim-based applications of metaheuristic algorithms to support the decision-making process : Uses in the planning of construction site layout. *Procedia Engineering*, 196 :558–564, 2017.

N. Anithadevi and M. Sundarambal. A design of intelligent qos aware web service recommendation system. *Cluster Computing*, 22(6) :14231–14240, 2019.

H. Arasteh, V. Hosseinnezhad, V. Loia, A. Tommasetti, O. Troisi, M. Shafie-khah, and P. Siano. Iot-based smart cities : A survey. In *2016 IEEE 16th international conference on environment and electrical engineering (EEEIC)*, pages 1–6. IEEE, 2016.

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

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

P. Asghari, A. M. Rahmani, and H. H. S. Javadi. Privacy-aware cloud service composition based on qos optimization in internet of things. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–26, 2020.

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

B. Benatallah and H. R. Motahari Nezhad. Service oriented architecture : Overview and directions. *Advances in Software Engineering : Lipari Summer School 2007, Lipari Island, Italy, July 8-21, 2007, Revised Tutorial Lectures*, pages 116–130, 2008.

D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical programming*, 98(1-3) :49–71, 2003.

A. K. Bhunia and S. S. Samanta. A study of interval metric and its application in multi-objective optimization with interval objectives. *Computers & Industrial Engineering*, 74 :169–178, 2014.

A. D. Birrell and B. J. Nelson. Implementing remote procedure calls. *ACM Transactions on Computer Systems (TOCS)*, 2(1) :39–59, 1984.

D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard. Web services architecture-w3c working group note 11 february 2004. *World Wide Web Consortium, article available from : http ://www. w3. org/TR/ws-arch*, page 13, 2004.

S. Borzsony, D. Kossmann, and K. Stocker. The skyline operator. In *Proceedings 17th international conference on data engineering*, pages 421–430. IEEE, 2001.

T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau, and J. Cowan. Extensible markup language (xml) 1.0 w3c recommendation 04 february 2004. *Available via the World Wide Web at http ://www. w3. org/TR/2004/REC-xml-20040204*, 2004.

N. Brown and C. Kindel. Distributed component object model protocol–dcom/1.0, 1998. *Redmond, WA*, 1996.

R. Burke. The adaptive web : Methods and strategies of web personalization. lncs, vol. 4321, 2007.

G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani. An approach for qos-aware service composition based on genetic algorithms. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 1069–1075. ACM, 2005.

J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut. Quality of service for workflows and web service processes. *Journal of web semantics*, 1(3) :281–308, 2004.

Z.-y. Chai, M.-m. Du, and G.-z. Song. A fast energy-centered and qos-aware service composition approach for internet of things. *Applied Soft Computing*, 100 :106914, 2021.

S. Chattopadhyay and A. Banerjee. Qscas : Qos aware web service composition algorithms with stochastic parameters. In *2016 IEEE International Conference on Web Services (ICWS)*, pages 388–395. IEEE, 2016.

F. Chen, S. Yuan, and B. Mu. User-qos-based web service clustering for qos prediction. In *2015 IEEE international conference on web services*, pages 583–590. IEEE, 2015.

J. Chen, C. Mao, and W. W. Song. Qos prediction for web services in cloud environments based on swarm intelligence search. *Knowledge-Based Systems*, 259 :110081, 2023.

K. Chen, H. Mao, X. Shi, Y. Xu, and A. Liu. Trust-aware and location-based collaborative filtering for web service qos prediction. In *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 143–148. IEEE, 2017a.

L. Chen, F. Xie, Z. Zheng, and Y. Wu. Predicting quality of service via leveraging location information. *Complexity*, 2019, 2019a.

S.-J. Chen and S.-M. Chen. Fuzzy risk analysis based on similarity measures of generalized fuzzy numbers. *IEEE Transactions on fuzzy systems*, 11(1) :45–56, 2003.

X. Chen, X. Liu, Z. Huang, and H. Sun. Regionknn : A scalable hybrid collaborative filtering algorithm for personalized web service recommendation. In *2010 IEEE international conference on web services*, pages 9–16. IEEE, 2010.

X. Chen, Z. Zheng, Q. Yu, and M. R. Lyu. Web service recommendation via exploiting location and qos information. *IEEE Transactions on Parallel and distributed systems*, 25(7) :1913–1924, 2013.

Y. Chen, P. Yu, Z. Zheng, J. Shen, and M. Guo. Modeling feature interactions for context-aware qos prediction of iot services. *Future Generation Computer Systems*, 137 :173–185, 2022.

Z. Chen, L. Shen, D. You, and F. Li. A user dependent web service qos collaborative prediction approach using neighborhood regularized matrix factorization. In *2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 316–321, 2016.

Z. Chen, L. Shen, and F. Li. Exploiting web service geographical neighborhood for collaborative qos prediction. *Future Generation Computer Systems*, 68 :248–259, 2017b.

Z. Chen, L. Shen, F. Li, and D. You. Your neighbors alleviate cold-start : On geographical neighborhood influence to collaborative web service qos prediction. *Knowledge-Based Systems*, 138 : 188–201, 2017c.

Z. Chen, L. Shen, and F. Li. Your neighbors are misunderstood : On modeling accurate similarity driven by data range to collaborative web service qos prediction. *Future Generation Computer Systems*, 95 :404–419, 2019b.

Z. Chen, L. Shen, F. Li, Y. Dianlong, and M. J. P. Buanga. Web service qos prediction : when collaborative filtering meets data fluctuating in big-range. *World Wide Web*, 23(3) :1715–1740, 2020a.

Z. Chen, Y. Sun, D. You, F. Li, and L. Shen. An accurate and efficient web service qos prediction model with wide-range awareness. *Future Generation Computer Systems*, 109 :275–292, 2020b.

H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.

J. Choi, J. Lee, D. Ryu, S. Kim, and J. Baik. Gain-qos : A novel qos prediction model for edge computing. *Journal of Web Engineering*, pages 27–52, 2022.

M. Cremene, M. Suciu, D. Pallez, and D. Dumitrescu. Comparative analysis of multi-objective evolutionary algorithms for qos-aware web service composition. *Applied Soft Computing*, 39 : 124–139, 2016.

F. Curbera, W. Nagy, and S. Weerawarana. Web services : Why and how. In *Workshop on Object-Oriented Web Services-OOPSLA*, volume 2001, 2001.

F. Dahan, K. El Hindi, and A. Ghoneim. Enhanced artificial bee colony algorithm for qos-aware web service selection problem. *Computing*, 99(5) :507–517, 2017. doi : 10.1007/s00607-017-0547-8.

S. De Deugd, R. Carroll, K. Kelly, B. Millett, and J. Ricker. Soda : Service oriented device architecture. *IEEE Pervasive Computing*, 5(3) :94–96, 2006.

K. Deb. An efficient constraint handling method for genetic algorithms. *Computer methods in applied mechanics and engineering*, 186(2-4) :311–338, 2000.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm : Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2) :182–197, 2002.

S. Deng, L. Huang, D. Hu, J. L. Zhao, and Z. Wu. Mobility-enabled service selection for composite services. *IEEE Transactions on Services Computing*, 9(3) :394–407, 2014a.

S. Deng, L. Huang, and G. Xu. Social network-based service recommendation with trust enhancement. *Expert Systems with Applications*, 41(18) :8075–8084, 2014b.

S.-G. Deng, L.-T. Huang, J. Wu, and Z.-H. Wu. Trust-based personalized service recommendation : A network perspective. *Journal of computer science and technology*, 29(1) :69–80, 2014c.

M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1) :143–177, 2004.

C. Desrosiers and G. Karypis. A comprehensive survey of neighborhood-based recommendation methods. *Recommender systems handbook*, pages 107–144, 2011.

L. Ding, J. Liu, G. Kang, Y. Xiao, and B. Cao. Joint qos prediction for web services based on deep fusion of features. *IEEE Transactions on Network and Service Management*, 2023.

S. Ding, Y. Li, D. Wu, Y. Zhang, and S. Yang. Time-aware cloud service recommendation using similarity-enhanced collaborative filtering and arima model. *Decision Support Systems*, 107 : 103–115, 2018.

Z. Ding, J. Liu, Y. Sun, C. Jiang, and M. Zhou. A transaction and qos-aware service selection approach based on genetic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, 45 (7) :1035–1046, 2015.

T. Downing and R. Java. Idg books worldwide. *Inc., USA*, 1998.

M. Eusuff, K. Lansey, and F. Pasha. Shuffled frog-leaping algorithm : a memetic meta-heuristic for discrete optimization. *Engineering optimization*, 38(2) :129–154, 2006.

A. M. Fathollahi-Fard, M. Hajiaghaei-Keshteli, and R. Tavakkoli-Moghaddam. Red deer algorithm (rda) : a new nature-inspired meta-heuristic. *Soft Computing*, pages 1–29, 2020.

J. Feng and L. Kong. A fuzzy multi-objective genetic algorithm for qos-based cloud service composition. In *2015 11th International Conference on Semantics, Knowledge and Grids (SKG)*, pages 202–206. IEEE, 2015.

D. M. Gay. The ampl modeling language : An aid to formulating and solving optimization problems. In *Numerical analysis and optimization*, pages 95–116. Springer, 2015.

S. H. Ghafouri, S. M. Hashemi, and P. C. Hung. A survey on web service qos prediction methods. *IEEE Transactions on Services Computing*, 2020.

M. Ghobaei-Arani and A. Souri. Lp-wsc : a linear programming approach for web service composition in geographically distributed cloud environments. *The Journal of Supercomputing*, 75(5) :2603–2628, 2019.

D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12) :61–70, 1992.

K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste : A constant time collaborative filtering algorithm. *information retrieval*, 4(2) :133–151, 2001.

N. Gondchawar, R. Kawitkar, et al. Iot based smart agriculture. *International Journal of advanced research in Computer and Communication Engineering*, 5(6) :838–842, 2016.

N. Gunantara. A review of multi-objective optimization : Methods and its applications. *Cogent Engineering*, 5(1) :1502242, 2018.

G. Guo. Resolving data sparsity and cold start in recommender systems. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 361–364. Springer, 2012.

Y. Guo, S. Wang, K.-S. Wong, and M. H. Kim. Skyline service selection approach based on qos prediction. *International Journal of Web and Grid Services*, 13(4) :425–447, 2017.

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

M. Hamzei, S. Khandagh, and N. Jafari Navimipour. A quality-of-service-aware service composition method in the internet of things using a multi-objective fuzzy-based hybrid algorithm. *Sensors*, 23 (16) :7233, 2023.

L. He and Y. He. Cross-border e-commerce logistics development model based on data mining and qos. In *2021 6th International Conference on Communication and Electronics Systems (ICCES)*, pages 1376–1379. IEEE, 2021.

P. He, J. Zhu, Z. Zheng, J. Xu, and M. R. Lyu. Location-based hierarchical matrix factorization for web service recommendation. In *2014 IEEE international conference on web services*, pages 297–304. IEEE, 2014.

J. L. Herlocker, J. A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 241–250, 2000.

J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1) :5–53, 2004.

S. Hu, G. Zou, B. Zhang, S. Wu, S. Lin, Y. Gan, and Y. Chen. Temporal-aware qos prediction via dynamic graph neural collaborative learning. In *Service-Oriented Computing : 20th International Conference, ICSOC 2022, Seville, Spain, November 29–December 2, 2022, Proceedings*, pages 125–133. Springer, 2022.

Y. Hu, Q. Peng, and X. Hu. A time-aware and data sparsity tolerant approach for web service recommendation. In *2014 IEEE international conference on web services*, pages 33–40. IEEE, 2014.

Y. Huo, Y. Zhuang, J. Gu, S. Ni, and Y. Xue. Discrete gbest-guided artificial bee colony algorithm for cloud service composition. *Applied Intelligence*, 42(4) :661–678, 2015.

W. Hussain, J. M. Merigó, M. R. Raza, and H. Gao. A new qos prediction model using hybrid iowa-anfis with fuzzy c-means, subtractive clustering and grid partitioning. *Information Sciences*, 584 :280–300, 2022.

R. J. Hyndman and A. B. Koehler. Another look at measures of forecast accuracy. *International journal of forecasting*, 22(4) :679–688, 2006.

C. Jatoth, G. Gangadharan, and R. Buyya. Computational intelligence based qos-aware web service composition : a systematic literature review. *IEEE Transactions on Services Computing*, 10(3) : 475–492, 2015.

X. Ji, H. Ye, J. Zhou, Y. Yin, and X. Shen. An improved teaching-learning-based optimization algorithm and its application to a combinatorial optimization problem in foundry industry. *Applied Soft Computing*, 57 :504–516, 2017.

X. Jian, Q. Zhu, and Y. Xia. An interval-based fuzzy ranking approach for qos uncertainty-aware service composition. *Optik-International Journal for Light and Electron Optics*, 127(4) :2102–2110, 2016.

Y. Jiang, J. Liu, M. Tang, and X. Liu. An effective web service recommendation method based on personalized collaborative filtering. In *2011 IEEE International Conference on Web Services*, pages 211–218, 2011.

H. Jin, X. Yao, and Y. Chen. Correlation-aware qos modeling and manufacturing cloud service composition. *Journal of Intelligent Manufacturing*, 28(8) :1947–1960, 2017.

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

Y. Jin, W. Guo, and Y. Zhang. A time-aware dynamic service quality prediction approach for services. *Tsinghua Science and Technology*, 25(2) :227–238, 2019a.

Y. Jin, K. Wang, Y. Zhang, and Y. Yan. Neighborhood-aware web service quality prediction using deep learning. *EURASIP Journal on Wireless Communications and Networking*, 2019(1) :1–10, 2019b.

A. Kalaï, C. A. Zayani, I. Amous, W. Abdelghani, and F. Sèdes. Social collaborative service recommendation approach based on user's trust and domain-specific expertise. *Future Generation Computer Systems*, 80 :355–367, 2018.

D. Karaboga and B. Basturk. A powerful and efficient algorithm for numerical function optimization : artificial bee colony (abc) algorithm. *Journal of global optimization*, 39(3) :459–471, 2007.

S. Karmakar and A. K. Bhunia. A comparative study of different order relations of intervals. *Reliable Computing*, 16(2) :38–72, 2012.

G. Khababa, F. Seghir, and S. Bessou. An extended artificial bee colony with local search for solving the skyline-based web services composition under interval qos properties. *Journal of Intelligent & Fuzzy Systems*, 42(4) :3855–3870, 2022.

G. Khababa, S. Bessou, F. Seghir, N. H. Harun, A. S. Almazyad, P. Jangir, and A. W. Mohamed. Collaborative filtering techniques for predicting web service qos values in static and dynamic environments : A systematic and thorough analysis. *IEEE Access*, 2025.

M. M. Khan, T. M. Alanazi, A. A. Albraikan, and F. A. Almalki. Iot-based health monitoring system development and analysis. *Security and Communication Networks*, 2022, 2022.

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

M. E. Khanouche, N. Atmani, and A. Cherifi. Improved teaching learning-based qos-aware services composition for internet of things. *IEEE Systems Journal*, pages 1–10, 2020b. doi : 10.1109/JSYST.2019.2960677.

M. Kim, B. Oh, J. Jung, and K.-H. Lee. Outlier-robust web service selection based on a probabilistic qos model. *International Journal of Web and Grid Services*, 12(2) :162–181, 2016.

A. Klein, F. Ishikawa, and S. Honiden. Efficient heuristic approach with improved time complexity for qos-aware service composition. In *2011 IEEE International Conference on Web Services*, pages 436–443. IEEE, 2011.

L. Kuang, Y. Xia, and Y. Mao. Personalized services recommendation based on context-aware qos prediction. In *2012 IEEE 19th International Conference on Web Services*, pages 400–406. IEEE, 2012.

E. L. Lawler. A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. *Management science*, 18(7) :401–405, 1972.

E. L. Lawler and M. Bell. A method for solving discrete optimization problems. *Operations Research*, 14(6) :1098–1112, 1966.

G. Lee, J. Park, et al. The iot—concept and problem statement. *IETF Standard draft-lee-iot-problem-statement-05*, 2012.

B. Li, C. Ye, X. Yu, H. Zhou, and C. Huang. Qos prediction based on temporal information and request context. *Service Oriented Computing and Applications*, 15 :231–244, 2021a.

C. Li, J. Li, H. Chen, and A. A. Heidari. Memetic harris hawks optimization : Developments and perspectives on project scheduling and qos-aware web service composition. *Expert Systems with Applications*, 171 :114529, 2021b.

J. Li, X.-L. Zheng, S.-T. Chen, W.-W. Song, and D.-r. Chen. An efficient and reliable approach for quality-of-service-aware service composition. *Information Sciences*, 269 :238–254, 2014.

J. Li, H. Wu, Q. He, Y. Zhao, and X. Wang. Dynamic qos prediction with intelligent route estimation via inverse reinforcement learning. *IEEE Transactions on Services Computing*, 2023.

S. Li, L. D. Xu, and S. Zhao. The internet of things : a survey. *Information systems frontiers*, 17 : 243–259, 2015.

J. Liao, Y. Liu, X. Zhu, and J. Wang. Accurate sub-swarms particle swarm optimization algorithm for service composition. *Journal of Systems and Software*, 90 :191–203, 2014.

G. Linden, B. Smith, and J. York. Amazon. com recommendations : Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1) :76–80, 2003.

J. Liu, M. Tang, Z. Zheng, X. Liu, and S. Lyu. Location-aware and personalized collaborative filtering for web service recommendation. *IEEE Transactions on Services Computing*, 9(5) :686–699, 2015.

W. Lo, J. Yin, S. Deng, Y. Li, and Z. Wu. Collaborative web service qos prediction with location-based regularization. In *2012 IEEE 19th international conference on web services*, pages 464–471. IEEE, 2012a.

W. Lo, J. Yin, S. Deng, Y. Li, and Z. Wu. An extended matrix factorization approach for qos prediction in service selection. In *2012 IEEE Ninth International Conference on Services Computing*, pages 162–169, 2012b.

W. Lo, J. Yin, Y. Li, and Z. Wu. Efficient web service qos prediction using local neighborhood matrix factorization. *Engineering Applications of Artificial Intelligence*, 38 :14–23, 2015.

X. Luo, Y. Lv, R. Li, and Y. Chen. Web service qos prediction based on adaptive dynamic programming using fuzzy neural networks for cloud services. *IEEE Access*, 3 :2260–2269, 2015a.

X. Luo, M. Zhou, Y. Xia, Q. Zhu, A. C. Ammari, and A. Alabdulwahab. Generating highly accurate predictions for missing qos data via aggregating nonnegative latent factor models. *IEEE transactions on neural networks and learning systems*, 27(3) :524–537, 2015b.

X. Luo, J. Liu, D. Zhang, and X. Chang. A large-scale web qos prediction scheme for the industrial internet of things based on a kernel machine learning algorithm. *Computer Networks*, 101 :81–89, 2016.

X. Luo, H. Wu, H. Yuan, and M. Zhou. Temporal pattern-aware qos prediction via biased non-negative latent factorization of tensors. *IEEE transactions on cybernetics*, 50(5) :1798–1809, 2019.

Y.-s. Luo, Y. Qi, D. Hou, L.-f. Shen, Y. Chen, and X. Zhong. A novel heuristic algorithm for qos-aware end-to-end service composition. *Computer Communications*, 34(9) :1137–1144, 2011.

H. Ma, I. King, and M. R. Lyu. Effective missing data prediction for collaborative filtering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 39–46, 2007.

Y. Ma, S. Wang, P. C. Hung, C.-H. Hsu, Q. Sun, and F. Yang. A highly accurate prediction algorithm for unknown web service qos values. *IEEE Transactions on Services Computing*, 9(4) :511–523, 2015.

S. Mahato and A. Bhunia. Interval-arithmetic-oriented interval computing technique for global optimization. *Applied Mathematics Research Express*, 2006, 2006.

M. Masdari and H. Khezri. Service selection using fuzzy multi-criteria decision making : a comprehensive review. *Journal of Ambient Intelligence and Humanized Computing*, 12(2) :2803–2834, 2021.

M. Masdari, M. Nozad Bonab, and S. Ozdemir. Qos-driven metaheuristic service composition schemes : a comprehensive overview. *Artificial Intelligence Review*, 54(5) :3749–3816, 2021.

I. Mashal, O. Alsaryrah, T.-Y. Chung, C.-Z. Yang, W.-H. Kuo, and D. P. Agrawal. Choices for interaction with things on internet and underlying issues. *Ad Hoc Networks*, 28 :68–90, 2015.

I. Mashal, O. Alsaryrah, and T.-Y. Chung. Performance evaluation of recommendation algorithms on internet of things services. *Physica A : Statistical Mechanics and its Applications*, 451 :646, 2016.

H. Mezni. Temporal knowledge graph embedding for effective service recommendation. *IEEE Transactions on Services Computing*, 2021.

H. Mezni and M. Fayala. Time-aware service recommendation : Taxonomy, review, and challenges. *Software : Practice and Experience*, 48(11) :2080–2108, 2018.

H. Mezni, S. A. Arab, D. Benslimane, and K. Benouaret. An evolutionary clustering approach based on temporal aspects for context-aware service recommendation. *Journal of Ambient Intelligence and Humanized Computing*, 11(1) :119–138, 2020.

H. Mezni, D. Benslimane, and L. Bellatreche. Context-aware service recommendation based on knowledge graph embedding. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

B. L. Miller, D. E. Goldberg, et al. Genetic algorithms, tournament selection, and the effects of noise. *Complex systems*, 9(3) :193–212, 1995.

S. Mirjalili. Introduction to evolutionary single-objective optimisation. In *Evolutionary Algorithms and Neural Networks*, pages 3–14. Springer, 2019.

A. Mishra. Metrics to evaluate your machine learning algorithm. *Towards data science*, pages 1–8, 2018.

A. Mnih and R. R. Salakhutdinov. Probabilistic matrix factorization. *Advances in neural information processing systems*, 20, 2007.

M. V. Moreno, M. A. Zamora, and A. F. Skarmeta. An iot based framework for user–centric smart building services. *International Journal of Web and Grid Services*, 11(1) :78–101, 2015.

A. Naseri and N. Jafari Navimipour. A new agent-based method for qos-aware cloud service composition using particle swarm optimization algorithm. *Journal of Ambient Intelligence and Humanized Computing*, 10(5) :1851–1864, 2019.

A. G. Neiat, A. Bouguettaya, T. Sellis, and Z. Ye. Spatio-temporal composition of sensor cloud services. In *2014 IEEE International Conference on Web Services*, pages 241–248. IEEE, 2014.

E. Newcomer. *Understanding Web Services : XML, Wsdl, Soap, and UDDI*. Addison-Wesley Professional, 2002.

X. Ning, C. Desrosiers, and G. Karypis. A comprehensive survey of neighborhood-based recommendation methods. *Recommender systems handbook*, pages 37–76, 2015.

S. Niu, G. Zou, Y. Gan, Y. Xiang, and B. Zhang. Towards the optimality of qos-aware web service composition with uncertainty. *International Journal of Web and Grid Services*, 15(1) :1–28, 2019.

W. J. Obidallah, B. Raahemi, and U. Ruhi. Clustering and association rules for web service discovery and recommendation : a systematic literature review. *SN Computer Science*, 1 :1–33, 2020.

A. Palaios, C. L. Vielhaus, D. F. Külzer, C. Watermann, R. Hernangómez, S. Partani, P. Geuer, A. Krause, R. Sattiraju, M. Kasparick, et al. Machine learning for qos prediction in vehicular communication : Challenges and solution approaches. *IEEE Access*, 11 :92459–92477, 2023.

M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-oriented computing : State of the art and research challenges. *Computer*, 40(11) :38–45, 2007.

J. Pasley. How bpel and soa are changing web services development. *IEEE Internet computing*, 9(3) : 60–67, 2005.

C. Pennington. Introduction to web services. In *Semantic Web Services : Theory, Tools and Applications*, pages 134–154. IGI Global, 2007.

K. Qi, H. Hu, W. Song, J. Ge, and J. Lü. Personalized qos prediction via matrix factorization integrated with neighborhood information. In *2015 IEEE International Conference on Services Computing*, pages 186–193. IEEE, 2015.

L. Qi, W. Dou, and X. Zhang. An inverse collaborative filtering approach for cold-start problem in web service recommendation. In *Proceedings of the Australasian Computer Science Week Multiconference*, pages 1–9, 2017.

W. Qiu, Z. Zheng, X. Wang, X. Yang, and M. R. Lyu. Reputation-aware qos value prediction of web services. In *2013 IEEE International Conference on Services Computing*, pages 41–48. IEEE, 2013.

N. S. Raghavan and T. Waghmare. Dpac : an object-oriented distributed and parallel computing framework for manufacturing applications. *IEEE transactions on robotics and automation*, 18(4) : 431–443, 2002.

A. Ramírez, J. A. Parejo, J. R. Romero, S. Segura, and A. Ruiz-Cortés. Evolutionary composition of qos-aware web services : a many-objective perspective. *Expert Systems with Applications*, 72 : 357–370, 2017.

S. Ran. A model for web services discovery with qos. *ACM Sigecom exchanges*, 4(1) :1–10, 2003.

R. V. Rao. Introduction to multiple attribute decision-making (madm) methods. *Decision Making in the Manufacturing Environment : Using Graph Theory and Fuzzy Multiple Attribute Decision Making Methods*, pages 27–41, 2007.

R. V. Rao, V. J. Savsani, and D. Vakharia. Teaching–learning-based optimization : an optimization method for continuous non-linear large scale problems. *Information sciences*, 183(1) :1–15, 2012.

M. Razian, M. Fathian, and R. Buyya. Arc : Anomaly-aware robust cloud-integrated iot service composition based on uncertainty in advertised quality of service values. *Journal of Systems and Software*, 164 :110557, 2020.

M. Razian, M. Fathian, R. Bahsoon, A. N. Toosi, and R. Buyya. Service composition in dynamic environments : A systematic review and future directions. *Journal of Systems and Software*, page 111290, 2022.

L. Ren and W. Wang. An svm-based collaborative filtering approach for top-n web services recommendation. *Future Generation Computer Systems*, 78 :531–543, 2018.

F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

R. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1(2) :127–190, 1999.

S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv :1609.04747*, 2016.

B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.

S. Sefati and N. J. 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*, 8 (20) :15620–15627, 2021.

F. 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*, 167 :114413, 2021.

F. Seghir and G. Khababa. Fuzzy teaching learning based optimization approach for solving the qos-aware web service selection problem in uncertain environments. *J Ambient Intell Human Comput*, 2021a. URL https://doi.org/10.1007/s12652-020-02879-y.

F. Seghir and G. Khababa. Fuzzy teaching learning based optimization approach for solving the qos-aware web service selection problem in uncertain environments. *Journal of Ambient Intelligence and Humanized Computing*, 12(12) :10667–10697, 2021b.

F. Seghir and G. Khababa. An improved discrete flower pollination algorithm for fuzzy qos-aware iot services composition based on skyline operator. *The Journal of Supercomputing*, 79(10) : 10645–10676, 2023.

F. Seghir, A. Khababa, and F. Semchedine. An interval-based multi-objective artificial bee colony algorithm for solving the web service composition under uncertain qos. *The Journal of Supercomputing*, 2019. URL https://doi.org/10.1007/s11227-019-02814-9.

A. Shahzad, Y.-G. Kim, and A. Elgamoudi. Secure iot platform for industrial control systems. In *2017 International Conference on Platform Technology and Service (PlatCon)*, pages 1–6. IEEE, 2017.

L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei. Personalized qos prediction for web services via collaborative filtering. In *Ieee international conference on web services (icws 2007)*, pages 439–446. IEEE, 2007.

A. Sharma, P. K. Singh, and Y. Kumar. An integrated fire detection system using iot and image processing technique for smart cities. *Sustainable Cities and Society*, 61 :102332, 2020.

Q. She, X. Wei, G. Nie, and D. Chen. Qos-aware cloud service composition : A systematic mapping study from the perspective of computational intelligence. *Expert Systems with Applications*, 2019.

M. Silic, G. Delac, and S. Srbljic. Prediction of atomic web services reliability based on k-means clustering. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, pages 70–80, 2013.

P. K. Singh, P. K. D. Pramanik, and P. Choudhury. Collaborative filtering in recommender systems : Technicalities, challenges, applications, and research trends. In *New Age Analytics*, pages 183–215. Apple Academic Press, 2020a.

P. K. Singh, S. Setta, P. K. D. Pramanik, and P. Choudhury. Improving the accuracy of collaborative filtering-based recommendations by considering the temporal variance of top-n neighbors. In *International Conference on Innovative Computing and Communications : Proceedings of ICICC 2019, Volume 1*, pages 1–10. Springer, 2020b.

P. K. Singh, R. Ahmed, I. S. Rajput, and P. Choudhury. A comparative study on prediction approaches of item-based collaborative filtering in neighborhood-based recommendations. *Wireless Personal Communications*, 121 :857–877, 2021a.

P. K. Singh, P. K. D. Pramanik, A. K. Dey, and P. Choudhury. Recommender systems : an overview, research trends, and future directions. *International Journal of Business and Systems Research*, 15 (1) :14–52, 2021b.

I. Smith et al. Rfid and the inclusive model for the iot. *CASAGRAS Partnership Rep., West Yorkshire, UK, Final Rep*, pages 10–12, 2009.

L. H. Son. Hu-fcf : a hybrid user-based fuzzy collaborative filtering method in recommender systems. *Expert Systems with Applications : An International Journal*, 41(15) :6861–6870, 2014.

P. Spiess, S. Karnouskos, D. Guinard, D. Savio, O. Baecker, L. M. S. De Souza, and V. Trifa. Soa-based integration of the internet of things in enterprise services. In *2009 IEEE international conference on web services*, pages 968–975. IEEE, 2009.

J. A. Stankovic. Research directions for the internet of things. *IEEE Internet of Things Journal*, 1(1) : 3–9, 2014.

K. Su, L. Ma, B. Xiao, and H. Zhang. Web service qos prediction by neighbor information combined non-negative matrix factorization. *Journal of Intelligent & Fuzzy Systems*, 30(6) :3593–3604, 2016.

K. Su, B. Xiao, B. Liu, H. Zhang, and Z. Zhang. Tap : A personalized trust-aware qos prediction approach for web service recommendation. *Knowledge-Based Systems*, 115 :55–65, 2017.

X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.

Y. Syu, Y.-Y. Fanjiang, J.-Y. Kuo, and S.-P. Ma. Applying genetic programming for time-aware dynamic qos prediction. In *2015 IEEE International Conference on Mobile Services*, pages 217–224. IEEE, 2015.

Y. Syu, J.-Y. Kuo, and Y.-Y. Fanjiang. Time series forecasting for dynamic quality of web services : an empirical study. *Journal of Systems and Software*, 134 :279–303, 2017.

M. Tang, Z. Zheng, G. Kang, J. Liu, Y. Yang, and T. Zhang. Collaborative web service quality prediction via exploiting matrix factorization and network map. *IEEE Transactions on Network and Service Management*, 13(1) :126–137, 2016.

F. Tao, D. Zhao, Y. Hu, and Z. Zhou. Resource service composition and its optimal-selection based on particle swarm optimization in manufacturing grid system. *IEEE Transactions on industrial informatics*, 4(4) :315–327, 2008.

Q. Tao, H.-y. Chang, C.-q. Gu, and Y. Yi. A novel prediction approach for trustworthy qos of web services. *Expert Systems with Applications*, 39(3) :3676–3681, 2012.

L.-V. Thinh and T.-D. Tu. Qos prediction for web services based on user-trust propagation model. *New Review of Hypermedia and Multimedia*, 23(4) :277–291, 2017.

E. Tong, W. Niu, and J. Liu. A missing qos prediction approach via time-aware collaborative filtering. *IEEE Transactions on Services Computing*, 2021.

H. Tran-Dang, N. Krommenacker, P. Charpentier, and D.-S. Kim. The internet of things for logistics : Perspectives, application review, and challenges. *IETE Technical Review*, 39(1) :93–121, 2022.

P. Valsalan, T. A. B. Baomar, and A. H. O. Baabood. Iot based health monitoring system. *Journal of critical reviews*, 7(4) :739–743, 2020.

W. Viriyasitavat, L. Da Xu, Z. Bi, D. Hoonsopon, and N. Charoenruk. Managing qos of internet-of-things services using blockchain. *IEEE Transactions on Computational Social Systems*, 6(6) : 1357–1368, 2019.

R. Wang, B. Fu, G. Fu, and M. Wang. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, pages 1–7. 2017.

S. Wang, Q. Sun, H. Zou, and F. Yang. Particle swarm optimization with skyline operator for fast cloud-based web service composition. *Mobile Networks and Applications*, 18(1) :116–121, 2013.

S. Wang, Z. Zheng, Z. Wu, M. R. Lyu, and F. Yang. Reputation measurement and malicious feedback rating prevention in web service recommendation systems. *IEEE Transactions on Services Computing*, 8(5) :755–767, 2014.

X. Wang, J. Zhu, Z. Zheng, W. Song, Y. Shen, and M. R. Lyu. A spatial-temporal qos prediction approach for time-aware web service recommendation. *ACM Transactions on the Web (TWEB)*, 10 (1) :1–25, 2016.

G. White, A. Palade, C. Cabrera, and S. Clarke. Iotpredict : collaborative qos prediction in iot. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10, 2018a.

G. White, A. Palade, and S. Clarke. Forecasting qos attributes using lstm networks. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018b.

G. White, A. Palade, C. Cabrera, and S. Clarke. Autoencoders for qos prediction at the edge. In *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom*, pages 1–9. IEEE, 2019.

D. Wu, X. Luo, M. Shang, Y. He, G. Wang, and X. Wu. A data-aware latent factor model for web service qos prediction. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 384–399. Springer, 2019.

D. Wu, X. Luo, M. Shang, Y. He, G. Wang, and X. Wu. A data-characteristic-aware latent factor model for web services qos prediction. *IEEE Transactions on Knowledge and Data Engineering*, 34(6) :2525–2538, 2020.

H. Wu, K. Yue, C.-H. Hsu, Y. Zhao, B. Zhang, and G. Zhang. Deviation-based neighborhood model for context-aware qos prediction of cloud and iot services. *Future Generation Computer Systems*, 76 :550–560, 2017a.

H. Wu, K. Yue, B. Li, B. Zhang, and C.-H. Hsu. Collaborative qos prediction with context-sensitive matrix factorization. *Future Generation Computer Systems*, 82 :669–678, 2018a.

H. Wu, Z. Zhang, J. Luo, K. Yue, and C.-H. Hsu. Multiple attributes qos prediction via deep neural model with contexts. *IEEE Transactions on Services Computing*, 2018b.

J. Wu and W. Tan. Method towards service composition optimization on cost-effective using mixed flower pollination algorithm. In *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 37–42. IEEE, 2021.

J. Wu, L. Chen, Y. Feng, Z. Zheng, M. C. Zhou, and Z. Wu. Predicting quality of service for selection by neighborhood-based collaborative filtering. *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, 43(2) :428–439, 2012.

Q. Wu and Q. Zhu. Transactional and qos-aware dynamic service composition based on ant colony optimization. *Future Generation Computer Systems*, 29(5) :1112–1119, 2013.

Q. Wu, Q. Zhu, and M. Zhou. A correlation-driven optimal service selection approach for virtual enterprise establishment. *Journal of Intelligent Manufacturing*, 25(6) :1441–1453, 2014a.

Y. Wu, F. Xie, L. Chen, C. Chen, and Z. Zheng. An embedding based factorization machine approach for web service qos prediction. In *International Conference on Service-Oriented Computing*, pages 272–286. Springer, 2017b.

Z. Wu, Y. Chen, and T. Li. Personalized recommendation based on the improved similarity and fuzzy clustering. In *2014 International Conference on Information Science, Electronics and Electrical Engineering*, volume 2, pages 1353–1357. IEEE, 2014b.

x. wang, X. Xu, Q. Z. Sheng, Z. Wang, and L. Yao. Novel artificial bee colony algorithms for qos-aware service selection. *IEEE Transactions on Services Computing*, 12(2) :247–261, 2019. doi : 10.1109/TSC.2016.2612663.

W.-L. Xiang and M.-Q. An. An efficient and robust artificial bee colony algorithm for numerical optimization. *Computers & Operations Research*, 40(5) :1256–1265, 2013.

Q. Xie, S. Zhao, Z. Zheng, J. Zhu, and M. R. Lyu. Asymmetric correlation regularized matrix factorization for web service recommendation. In *2016 IEEE International Conference on Web Services (ICWS)*, pages 204–211. IEEE, 2016.

J. Xu, Z. Zheng, and M. R. Lyu. Web service personalized quality of service prediction via reputation-based matrix factorization. *IEEE transactions on reliability*, 65(1) :28–37, 2015.

J. Xu, L. Guo, R. Zhang, H. Hu, F. Wang, and Z. Pei. Qos-aware service composition using fuzzy set theory and genetic algorithm. *Wireless Personal Communications*, 102(2) :1009–1028, 2018.

J. Xu, J. Lin, Y. Li, and Z. Xu. Multifed : A fast converging federated learning framework for services qos prediction via cloud–edge collaboration mechanism. *Knowledge-Based Systems*, 268 :110463, 2023.

X. Xu, Q. Z. Sheng, Z. Wang, L. Yao, et al. Novel artificial bee colony algorithms for qos-aware service selection. *IEEE Transactions on Services Computing*, 12(2) :247–261, 2016a.

Y. Xu, J. Yin, and W. Lo. A unified framework of qos-based web service recommendation with neighborhood-extended matrix factorization. In *2013 IEEE 6th International Conference on Service-Oriented Computing and Applications*, pages 198–205. IEEE, 2013a.

Y. Xu, J. Yin, W. Lo, and Z. Wu. Personalized location-aware qos prediction for web services using probabilistic matrix factorization. In *International Conference on Web Information Systems Engineering*, pages 229–242. Springer, 2013b.

Y. Xu, J. Yin, S. Deng, N. N. Xiong, and J. Huang. Context-aware qos prediction for web service recommendation and selection. *Expert Systems with Applications*, 53 :75–86, 2016b.

G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen. Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 114–121, 2005.

Y. Yan, P. Sun, J. Zhang, Y. Ma, L. Zhao, and Y. Qin. Dynamic qos prediction algorithm based on kalman filter modification. *Sensors*, 22(15) :5651, 2022.

X.-S. Yang. Flower pollination algorithm for global optimization. In *International conference on unconventional computing and natural computation*, pages 240–249. Springer, 2012.

F. Ye, Z. Lin, C. Chen, Z. Zheng, and H. Huang. Outlier-resilient web service qos prediction. In *Proceedings of the Web Conference 2021*, pages 3099–3110, 2021.

R. Yera and L. Martinez. Fuzzy tools in recommender systems : A survey. *International Journal of Computational Intelligence Systems*, 10(1) :776–803, 2017.

J. Yin, W. Lo, S. Deng, Y. Li, Z. Wu, and N. Xiong. Colbar : A collaborative location-based regularization framework for qos prediction. *Information Sciences*, 265 :68–84, 2014.

Y. Yin, L. Chen, Y. Xu, J. Wan, H. Zhang, and Z. Mai. Qos prediction for service recommendation with deep feature learning in edge computing environment. *Mobile networks and applications*, 25 (2) :391–401, 2020.

H. Ying and Z. Jiande. A nonlinear service composition method based on the skyline operator. *Journal of Systems Engineering and Electronics*, 31(4) :743–750, 2020.

C. Yu and L. Huang. A web service qos prediction approach based on time-and location-aware collaborative filtering. *Service Oriented Computing and Applications*, 10(2) :135–149, 2016.

D. Yu, Y. Liu, Y. Xu, and Y. Yin. Personalized qos prediction for web services using latent factor models. In *2014 IEEE international conference on services computing*, pages 107–114. IEEE, 2014.

T. Yu, H. Liu, L. Zhang, and H. Liu. Msrdl : Deep learning framework for service recommendation in mashup creation. *Scientific Reports*, 13(1) :7641, 2023.

M. H. Zadeh and M. A. Seyyedi. Qos monitoring for web services by time series forecasting. In *2010 3rd International Conference on Computer Science and Information Technology*, volume 5, pages 659–663. IEEE, 2010.

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

Y. Zeng, J. Xu, Y. Li, C. Chen, Q. Dai, and Z. Du. Towards highly-efficient and accurate services qos prediction via machine unlearning. *IEEE Access*, 2023.

H. Zhang, D. Wang, W. Zhang, L. Tan, G. Kibalya, P. Zhang, and K. K. Igorevich. Qos prediction in intelligent edge computing based on feature learning. *Journal of Cloud Computing*, 12(1) :1–16, 2023.

M. Zhang, X. Liu, R. Zhang, and H. Sun. A web service recommendation approach based on qos prediction using fuzzy clustering. In *2012 IEEE ninth international conference on services computing*, pages 138–145. IEEE, 2012.

P. Zhang, H. Jin, H. Dong, W. Song, and L. Wang. La-lmrbf : Online and long-term web service qos forecasting. *IEEE Transactions on Services Computing*, 14(6) :1809–1823, 2019a.

S. Zhang, Y. Xu, W. Zhang, and D. Yu. A new fuzzy qos-aware manufacture service composition method using extended flower pollination algorithm. *Journal of Intelligent Manufacturing*, pages 1–15, 2017.

S. Zhang, S. Xu, W. Zhang, D. Yu, and K. Chen. A hybrid approach combining an extended bbo algorithm with an intuitionistic fuzzy entropy weight method for qos-aware manufacturing service supply chain optimization. *Neurocomputing*, 272 :439–452, 2018.

S. Zhang, Y. Xu, W. Zhang, and D. Yu. A new fuzzy qos-aware manufacture service composition method using extended flower pollination algorithm. *Journal of Intelligent Manufacturing*, 30(5) : 2069–2083, 2019b.

S. Zhang, W. Yang, W. Zhang, and M. Chen. A collaborative service group-based fuzzy qos-aware manufacturing service composition using an extended flower pollination algorithm. *Nonlinear Dynamics*, pages 1–24, 2019c.

X. Zhang, Z. Wang, X. Lv, and R. Qi. A clustering-based qos prediction approach for web service selection. In *2013 International Conference on Information Science and Cloud Computing Companion*, pages 201–206. IEEE, 2013.

Y. Zhang, Z. Zheng, and M. R. Lyu. Exploring latent features for memory-based qos prediction in cloud computing. In *2011 IEEE 30th international symposium on reliable distributed systems*, pages 1–10, 2011a.

Y. Zhang, Z. Zheng, and M. R. Lyu. Wspred : A time-aware personalized qos prediction framework for web services. In *2011 IEEE 22nd international symposium on software reliability engineering*, pages 210–219. IEEE, 2011b.

H. Zheng, J. Yang, and W. Zhao. Probabilistic qos aggregations for service composition. *ACM Transactions on the Web (TWEB)*, 10(2) :1–36, 2016.

Z. Zheng and M. R. Lyu. Collaborative reliability prediction of service-oriented systems. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, pages 35–44, 2010.

Z. Zheng, H. Ma, M. R. Lyu, and I. King. Wsrec : A collaborative filtering based web service recommender system. In *2009 IEEE International Conference on Web Services*, pages 437–444. IEEE, 2009.

Z. Zheng, H. Ma, M. R. Lyu, and I. King. Qos-aware web service recommendation by collaborative filtering. *IEEE Transactions on services computing*, 4(2) :140–152, 2010a.

Z. Zheng, Y. Zhang, and M. R. Lyu. Distributed qos evaluation for real-world web services. In *2010 IEEE International Conference on Web Services*, pages 83–90. IEEE, 2010b.

Z. Zheng, H. Ma, M. R. Lyu, and I. King. Collaborative web service qos prediction via neighborhood integrated matrix factorization. *IEEE Transactions on Services Computing*, 6(3) :289–299, 2012.

Z. Zheng, X. Li, M. Tang, F. Xie, and M. R. Lyu. Web service qos prediction via collaborative filtering : A survey. *IEEE Transactions on Services Computing*, 15(4) :2455–2472, 2020.

J. Zhou, D. Ding, Z. Wu, and Y. Xiu. Spatial context-aware time-series forecasting for qos prediction. *IEEE Transactions on Network and Service Management*, 20(2) :918–931, 2023.

Q. Zhou, H. Wu, K. Yue, and C.-H. Hsu. Spatio-temporal context-aware collaborative qos prediction. *Future Generation Computer Systems*, 100 :46–57, 2019.

J. Zhu, Y. Kang, Z. Zheng, and M. R. Lyu. A clustering-based qos prediction approach for web service recommendation. In *2012 IEEE 15th international symposium on object/component/service-oriented real-time distributed computing workshops*, pages 93–98. IEEE, 2012.

J. Zhu, P. He, Z. Zheng, and M. R. Lyu. Online qos prediction for runtime service adaptation via adaptive matrix factorization. *IEEE Transactions on Parallel and Distributed Systems*, 28(10) : 2911–2924, 2017.

J. Zhu, B. Li, J. Wang, D. Li, Y. Liu, and Z. Zhang. Bgcl : Bi-subgraph network based on graph contrastive learning for cold-start qos prediction. *Knowledge-Based Systems*, 263 :110296, 2023.

S. Zhu, J. Ding, and J. Yang. Location-aware deep interaction forest for web service qos prediction. *Applied Sciences*, 14(4) :1450, 2024.

G. Zou, S. Wu, S. Hu, C. Cao, Y. Gan, B. Zhang, and Y. Chen. Ncrl : Neighborhood-based collaborative residual learning for adaptive qos prediction. *IEEE Transactions on Services Computing*, 2022.

Appendix A

Proof of the Theorem 1's proprieties

($\mathscr{P}1$) Let $A = [a^l, a^u]$ be an arbitrary interval-number. As the two order relations $<_{\min}$ and $>_{\max}$ are not held to rank the closed interval-number A with itself, i.e, $(a^l = a^l)$ and $(a^u = a^u)$, then A is considered equivalent to itself, i.e., $A \sim A$, which implies $A \leq_{\min} (\geq_{\max}) A$. Therefore, the reflexivity priority is satisfied.

($\mathscr{P}2$) Here, the anti-symmetry is trivial by Definitions 15 and 16.

($\mathscr{P}3$) Let $A = [a^l, a^u]$, $B = [b^l, {}^b u]$ and $C = [c^l, c^u]$ be three arbitrary interval-numbers. Given $A \leq_{\min} B$. Hence, according to the Definitions 13 and 15, we have :

1. $(a^l = b^l)$ and $(a^u = b^u)$;
2. or $(a^l < b^l)$ and $(a^u < b^u)$;
3. or $(a^l - b^l) < (b^u - a^u)$;
4. or $\big((a^l - b^l) = (b^u - a^u)\big)$ and $(a^u < b^u)$

Similarly for $B \leq_{\min} C$,

1. $(b^l = c^l)$ and $(b^u = c^u)$;
2. or $(b^l < c^l)$ and $(b^u < c^u)$;
3. or $(b^l - c^l) < (c^u - b^u)$;
4. or $\big((b^l - c^l) = (c^u - b^u)\big)$ and $(b^u < c^u)$

This implies that,

1. $\big[(a^l = b^l)$ and $(a^u = b^u)$ and $(b^l = c^l)$ and $(b^u = c^u)\big] \Rightarrow (a^l = c^l)$ and $(a^u = c^u)$;
2. or $\big[(a^l = b^l)$ and $(a^u = b^u)$ and $(b^l < c^l)$ and $(b^u < c^u)\big] \Rightarrow (a^l < c^l)$ and $(a^u < c^u)$;
3. or $\big[(a^l = b^l)$ and $(a^u = b^u)$ and $(b^l - c^l) < (c^u - b^u)\big] \Rightarrow (a^l - c^l) < (c^u - a^u)$;
4. or $\big[(a^l = b^l)$ and $(a^u = b^u)$ and $\big((b^l - c^l) = (c^u - b^u)\big)$ and $(b^u < c^u)\big] \Rightarrow$ $\big((a^l - c^l) = (c^u - a^u)\big)$ and $(a^u < c^u)$;
5. or $\big[(a^l < b^l)$ and $(a^u < b^u)$ and $(b^l = c^l)$ and $(b^u = c^u)\big] \Rightarrow (a^l < c^l)$ and $(a^u < c^u)$;
6. or $\big[(a^l < b^l)$ and $(a^u < b^u)$ and $(b^l < c^l)$ and $(b^u < c^u)\big] \Rightarrow (a^l < c^l)$ and $(a^u < c^u)$;

7. or $\left[(a^l < b^l) \text{ and } (a^u < b^u) \text{ and } (b^l - c^l) < (c^u - b^u)\right] \Rightarrow (a^l + a^u) < (b^l + b^u) < (c^l + c^u) \Rightarrow (a^l + a^u) < (c^l + c^u) \Rightarrow (a^l - c^l) < (c^u - a^u)$;
8. or $\left[(a^l < b^l) \text{ and } (a^u < b^u) \text{ and } \left((b^l - c^l) = (c^u - b^u)\right) \text{ and } (b^u < c^u)\right] \Rightarrow (a^l + a^u) < (b^l + b^u) = (c^l + c^u) \Rightarrow (a^l + a^u) < (c^l + c^u) \Rightarrow (a^l - c^l) < (c^u - a^u)$;
9. or $\left[(a^l - b^l) < (b^u - a^u) \text{ and } (b^l = c^l) \text{ and } (b^u = c^u)\right] \Rightarrow (a^l - c^l) < (c^u - a^u)$
10. or $\left[(a^l - b^l) < (b^u - a^u) \text{ and } (b^l < c^l) \text{ and } (b^u < c^u)\right] \Rightarrow (a^l + a^u) < (b^l + b^u) < (c^l + c^u) \Rightarrow (a^l + a^u) < (c^l + c^u) \Rightarrow (a^l - c^l) < (c^u - a^u)$;
11. or $\left[(a^l - b^l) < (b^u - a^u) \text{ and } (b^l - c^l) < (c^u - b^u)\right] \Rightarrow (a^l + a^u) < (b^l + b^u) < (c^l + c^u) \Rightarrow (a^l + a^u) < (c^l + c^u) \Rightarrow (a^l - c^l) < (c^u - a^u)$;
12. or $\left[(a^l - b^l) < (b^u - a^u) \text{ and } \left((b^l - c^l) = (c^u - b^u)\right) \text{ and } (b^u < c^u)\right] \Rightarrow (a^l + a^u) < (b^l + b^u) = (c^l + c^u) \Rightarrow (a^l + a^u) < (c^l + c^u) \Rightarrow (a^l - c^l) < (c^u - a^u)$;
13. or $\left[\left((a^l - b^l) = (b^u - a^u)\right) \text{ and } (a^u < b^u) \text{ and } (b^l = c^l) \text{ and } (b^u = c^u)\right] \Rightarrow \left((a^l - c^l) = (c^u - a^u)\right) \text{ and } (a^u < c^u)$
14. or $\left[\left((a^l - b^l) = (b^u - a^u)\right) \text{ and } (a^u < b^u) \text{ and } (b^l < c^l) \text{ and } (b^u < c^u)\right] \Rightarrow (a^l + a^u) = (b^l + b^u) < (c^l + c^u) \Rightarrow (a^l + a^u) < (c^l + c^u) \Rightarrow (a^l - c^l) < (c^u - a^u)$;
15. or $\left[\left((a^l - b^l) = (b^u - a^u)\right) \text{ and } (a^u < b^u) \text{ and } (b^l - c^l) < (c^u - b^u)\right] \Rightarrow (a^l + a^u) = (b^l + b^u) < (c^l + c^u) \Rightarrow (a^l + a^u) < (c^l + c^u) \Rightarrow (a^l - c^l) < (c^u - a^u)$;
16. or $\left[\left((a^l - b^l) = (b^u - a^u)\right) \text{ and } (a^u < b^u) \text{ and } \left((b^l - c^l) = (c^u - b^u)\right) \text{ and } (b^u < c^u)\right] \Rightarrow (a^l + a^u) = (b^l + b^u) = (c^l + c^u) \text{ and } (a^u < b^u < c^u) \Rightarrow (a^l + a^u) = (c^l + c^u) \text{ and } (a^u < c^u) \Rightarrow \left((a^l - c^l) = (c^u - a^u)\right) \text{ and } (a^u < c^u)$

This implies that,

1. $(a^l = c^l) \text{ and } (a^u = c^u)$;
2. or $(a^l < c^l) \text{ and } (a^u < c^u)$
3. or $(a^l - c^l) < (c^u - a^u)$;
4. or $\left((a^l - c^l) = (c^u - a^u)\right) \text{ and } (a^u < c^u)$

Therefore $A \leq_{\min} C$. Similar to the transitivity proof of $\leq_{\min}$, it is easy to check that if $A \geq_{\max} B$ and $B \geq_{\max} C$ then $A \geq_{\max} C$.

Appendix B

Proof of the proprieties $(\mathscr{P}1)$, $(\mathscr{P}2)$ and $(\mathscr{P}3)$.

$(\mathscr{P}1)$ Let $\tilde{A} = (a1, a2, a3, a4; \omega_A)$ be an arbitrary GTrFN. Now, $\mathscr{MS}(\tilde{A}) = \mathscr{MS}(\tilde{A})$, $\mathscr{MK}(\tilde{A}) = \mathscr{MK}(\tilde{A})$, $\mathscr{SW}(\tilde{A}) = \mathscr{SW}(\tilde{A})$, $\mathscr{KW}(\tilde{A}) = \mathscr{KW}(\tilde{A})$ and $\omega_A = \omega_A$, i.e., $\tilde{A} \sim \tilde{A}$, which implies $\tilde{A} \preccurlyeq_{min} (\succcurlyeq_{max}) \tilde{A}$; as result, the reflexivity priority is satisfied.

$(\mathscr{P}2)$ Here, the anti-symmetry is trivial by Definitions 10 and 11 as well as $(O2)$.

$(\mathscr{P}3)$ Let $\tilde{A} = (a1, a2, a3, a4; \omega_A)$, $\tilde{B} = (b1, b2, b3, b4; \omega_B)$ and $\tilde{C} = (c1, c2, c3, c4; \omega_C)$ be three arbitrarily GTrFNs. Given $\tilde{A} \preccurlyeq_{min} \tilde{B}$, hence, according to the Definition 10 and $(O2)$, we have :

(1) $\mathscr{MS}(\tilde{A}) < \mathscr{MS}(\tilde{B})$

(2) Or $(\mathscr{MS}(\tilde{A}) = \mathscr{MS}(\tilde{B}))$ and $(\mathscr{MK}(\tilde{A}) < \mathscr{MK}(\tilde{B}))$

Or $(\mathscr{MS}(\tilde{A}) = \mathscr{MS}(\tilde{B}))$ and $(\mathscr{MK}(\tilde{A}) = \mathscr{MK}(\tilde{B}))$ and $(\mathscr{SW}(\tilde{A}) < \mathscr{SW}(\tilde{B}))$

(3) Or $(\mathscr{MS}(\tilde{A}) = \mathscr{MS}(\tilde{B}))$ and $(\mathscr{MK}(\tilde{A}) = \mathscr{MK}(\tilde{B}))$ and $(\mathscr{SW}(\tilde{A}) = \mathscr{SW}(\tilde{B}))$ and $(\mathscr{KW}(\tilde{A}) < \mathscr{KW}(\tilde{B}))$

(4) Or $(\mathscr{MS}(\tilde{A}) = \mathscr{MS}(\tilde{B}))$ and $(\mathscr{MK}(\tilde{A}) = \mathscr{MK}(\tilde{B}))$ and $(\mathscr{SW}(\tilde{A}) = \mathscr{SW}(\tilde{B}))$ and $(\mathscr{KW}(\tilde{A}) = \mathscr{KW}(\tilde{B}))$ and $(\omega_A < \omega_B)$

(5) Or $(\mathscr{MS}(\tilde{A}) = \mathscr{MS}(\tilde{B}))$ and $(\mathscr{MK}(\tilde{A}) = \mathscr{MK}(\tilde{B}))$ and $(\mathscr{SW}(\tilde{A}) = \mathscr{SW}(\tilde{B}))$ and $(\mathscr{KW}(\tilde{A}) = \mathscr{KW}(\tilde{B}))$ and $(\omega_A = \omega_B)$

Similarly for $\tilde{B} \preccurlyeq_{min} \tilde{C}$,

(1) $\mathscr{MS}(\tilde{B}) < \mathscr{MS}(\tilde{C})$

(2) Or $(\mathscr{MS}(\tilde{B}) = \mathscr{MS}(\tilde{C}))$ and $(\mathscr{MK}(\tilde{B}) < \mathscr{MK}(\tilde{C}))$

(3) Or $(\mathscr{MS}(\tilde{B}) = \mathscr{MS}(\tilde{C}))$ and $(\mathscr{MK}(\tilde{B}) = \mathscr{MK}(\tilde{C}))$ and $(\mathscr{SW}(\tilde{B}) < \mathscr{SW}(\tilde{C}))$

(4) Or $(\mathscr{MS}(\tilde{B}) = \mathscr{MS}(\tilde{C}))$ and $(\mathscr{MK}(\tilde{B}) = \mathscr{MK}(\tilde{C}))$ and $(\mathscr{SW}(\tilde{B}) = \mathscr{SW}(\tilde{C}))$ and $(\mathscr{KW}(\tilde{B}) < \mathscr{KW}(\tilde{C}))$

(5) Or $(\mathscr{MS}(\tilde{B}) = \mathscr{MS}(\tilde{C}))$ and $(\mathscr{MK}(\tilde{B}) = \mathscr{MK}(\tilde{C}))$ and $(\mathscr{SW}(\tilde{B}) = \mathscr{SW}(\tilde{C}))$ and $(\mathscr{KW}(\tilde{B}) = \mathscr{KW}(\tilde{C}))$ and $(\omega_B < \omega_C)$

(6) Or $(\mathscr{M}\mathscr{S}(\tilde{B}) = \mathscr{M}\mathscr{S}(\tilde{C}))$ and $(\mathscr{M}\mathscr{K}(\tilde{B}) = \mathscr{M}\mathscr{K}(\tilde{C}))$ and $(\mathscr{S}\mathscr{W}(\tilde{B}) = \mathscr{S}\mathscr{W}(\tilde{C}))$ and $(\mathscr{K}\mathscr{W}(\tilde{B}) = \mathscr{K}\mathscr{W}(\tilde{C}))$ and $(\omega_B = \omega_C)$

This implies that,

(1) $\mathscr{M}\mathscr{S}(\tilde{A}) < \mathscr{M}\mathscr{S}(\tilde{B}) < \mathscr{M}\mathscr{S}(\tilde{C})$

(2) Or $(\mathscr{M}\mathscr{S}(\tilde{A}) = \mathscr{M}\mathscr{S}(\tilde{B}) = \mathscr{M}\mathscr{S}(\tilde{C}))$ and $(\mathscr{M}\mathscr{K}(\tilde{A}) < \mathscr{M}\mathscr{K}(\tilde{B}) < \mathscr{M}\mathscr{K}(\tilde{C}))$

(3) Or $(\mathscr{M}\mathscr{S}(\tilde{A}) = \mathscr{M}\mathscr{S}(\tilde{B}) = \mathscr{M}\mathscr{S}(\tilde{C}))$ and $(\mathscr{M}\mathscr{K}(\tilde{A}) = \mathscr{M}\mathscr{K}(\tilde{B}) = \mathscr{M}\mathscr{K}(\tilde{C}))$ and $(\mathscr{S}\mathscr{W}(\tilde{A}) < \mathscr{S}\mathscr{W}(\tilde{B}) < \mathscr{S}\mathscr{W}(\tilde{C}))$

(4) Or $(\mathscr{M}\mathscr{S}(\tilde{A}) = \mathscr{M}\mathscr{S}(\tilde{B}) = \mathscr{M}\mathscr{S}(\tilde{C}))$ and $(\mathscr{M}\mathscr{K}(\tilde{A}) = \mathscr{M}\mathscr{K}(\tilde{B}) = \mathscr{M}\mathscr{K}(\tilde{C}))$ and $(\mathscr{S}\mathscr{W}(\tilde{A}) = \mathscr{S}\mathscr{W}(\tilde{B}) = \mathscr{S}\mathscr{W}(\tilde{C}))$ and $(\mathscr{K}\mathscr{W}(\tilde{A}) < \mathscr{K}\mathscr{W}(\tilde{B}) < \mathscr{K}\mathscr{W}(\tilde{C}))$

(5) Or $(\mathscr{M}\mathscr{S}(\tilde{A}) = \mathscr{M}\mathscr{S}(\tilde{B}) = \mathscr{M}\mathscr{S}(\tilde{C}))$ and $(\mathscr{M}\mathscr{K}(\tilde{A}) = \mathscr{M}\mathscr{K}(\tilde{B}) = \mathscr{M}\mathscr{K}(\tilde{C}))$ and $(\mathscr{S}\mathscr{W}(\tilde{A}) = \mathscr{S}\mathscr{W}(\tilde{B}) = \mathscr{S}\mathscr{W}(\tilde{C}))$ and $(\mathscr{K}\mathscr{W}(\tilde{A}) = \mathscr{K}\mathscr{W}(\tilde{B}) = \mathscr{K}\mathscr{W}(\tilde{C}))$ and $(\omega_A < \omega_B < \omega_C)$

(6) Or $(\mathscr{M}\mathscr{S}(\tilde{A}) = \mathscr{M}\mathscr{S}(\tilde{B}) = \mathscr{M}\mathscr{S}(\tilde{C}))$ and $(\mathscr{M}\mathscr{K}(\tilde{A}) = \mathscr{M}\mathscr{K}(\tilde{B}) = \mathscr{M}\mathscr{K}(\tilde{C}))$ and $(\mathscr{S}\mathscr{W}(\tilde{A}) = \mathscr{S}\mathscr{W}(\tilde{B}) = \mathscr{S}\mathscr{W}(\tilde{C}))$ and $(\mathscr{K}\mathscr{W}(\tilde{A}) = \mathscr{K}\mathscr{W}(\tilde{B}) = \mathscr{K}\mathscr{W}(\tilde{C}))$ and $(\omega_A = \omega_B = \omega_C)$

Hence $\tilde{A} \preccurlyeq_{min} \tilde{C}$. Similar to the transitivity proof of $\preccurlyeq_{min}$, it is easy to check that if $\tilde{A} \succcurlyeq_{max} \tilde{B}$ and $\tilde{B} \succcurlyeq_{max} \tilde{C}$ then $\tilde{A} \succcurlyeq_{max} \tilde{C}$.