VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui

School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

Comprehensive Quality-Aware Automated Semantic Web Service Composition

Chen Wang

Supervisors: Dr. Hui Ma and Dr. Aaron Chen

September 6, 2017

Submitted in partial fulfilment of the requirements for PhD.

Abstract

Automated Web service composition is an NP-hard problem and it has been raising much attention in the research community due to the computational challenge and real-world applicability. Existing works either optimize QoS or semantic matchmaking quality, or are semi-automated approaches. The focus of our studies on service composition is to find effective and efficient approaches to comprehensive quality-aware semantic Web service composition, which aims to optimize semantic matchmaking quality and Quality of service (QoS) simultaneously. We will address this problem by achieving the following objectives: (1) developing EC-based approaches that explicitly support the comprehensive quality, (2) developing multi-objective approaches for optimizing the quality criteria involved in the comprehensive quality, (3) developing EC-based composition approaches for dynamic service composition while handling various changes of composition environment, and (4) developing EC-based approaches for semantic Web service composition supporting precondition and effects. In our preliminary work, we developed two EC-based approaches to single-objective comprehensive quality-aware automated semantic web service composition.

Contents

Figures

Chapter 1

Introduction

1.1 Problem Statement

Service-oriented computing (SOC) is a novel computing paradigm that employs services as fundamental elements to achieve the agile development of cost-efficient and integrable enterprise applications in heterogeneous environments [77, 78]. SOC aims to be platform neural and language agnostic to enable integrable and seamless communication among those existing or newly built independent services. *Service Oriented Architecture* (SOA) could abstractly implement service-oriented paradigm of computing. This accomplishment has been contributing to the reuse of software components, from the concept of functions to units and from units to services during the ongoing development of SOA [10, 74]. One of the most typical implementation of SOA is *Web service*, which is designated as "modular, self-describing, self-contained applications that are available on the Internet" [21]. Several standards play a significant role in registering, enquiring and grounding Web services across the Web, such as UDDI [20], WSDL [52] and SOAP [34]. Since an atomic Web can not always satisfy users' requirements, *Web service composition* aims to loosely couple a set of Web services to provide a value-added composite service that accommodates complex requirements of service users.

The first notable challenges for Web service composition are ensuring interoperability of services [34]. *Interoperability* of Web services presents challenges in syntactic and semantic dimensions. On the one hand, the syntactic dimension is covered by the XML-based technologies [119], such as $WSDL$ and $SOAP$. In this dimension, most services are composed together merely based on the matching of input-output parameters. On the other hand, the semantic dimension enables better collaboration through ontology-based semantics [73], in which many standards have been established, such as OWL-S [66], Web service Modeling Ontology (WSMO) [53], SAWSDL [49], and Semantic Web services Ontology (SWSO) [81]. Sometimes, this dimension brings around some other underlying functionality of services (i.e., precondition and postcondition) that could affect the execution of Web services and their composition. The interoperability challenge gives birth to *Semantic Web services composition*, as a different technique from traditional service composition methods (i.e., only syntactic dimension is presented in Web services). The interoperability of services is presented as a semantic matchmaking relationship between provided information (i.e. one service output) and required information (i.e., another service input), which do not often perfectly match with each other based on their semantic descriptions. Therefore, the quality of matchmaking becomes a major aspect of the quality concern for users, and also raises researchers' interests.

The second notable challenges for Web service composition are achieving Quality of Service (QoS) optimization [34]. QoS represents non-functional attributes of service composition (e.g, cost, time, reliability and availability). Often, a global search method is employed

to minimize the cost and maximize the reliability of composite services. This challenge gives birth to *QoS-aware service composition* that aims to find composition solutions with optimized QoS. Furthermore, QoS-aware service composition problem is facilitated by *Service Level Agreement (SLA)* [89], i.e., binding constraints on QoS. This results in the *SLA-aware Web service composition*, e.g., constraints on cost, execution time, availability and reliability are separately specified with both lower and upper bounds. In addition, when QoS preference is provided by users, optimization problem often returns a single near-optimal solution, and it is considered as a *single-objective optimization* problem. Otherwise, a set of composition solutions are returned when QoS preference is not clearly by users. This problem is considered as a *multi-objective optimization problem*.

Apart from the two notable challenges discussed above, the service composition problem is often *dynamic* in the real world, rather than *static*. For example, QoS of services are fluctuating over time. These services chosen at the planning stage may not be available during the runtime, it may due to imprecise interface description [44] or hardware failures [39]. New services can also be registered in the service repository from time to time. Existing techniques for *static Web service composition* cannot cope with such a changing environment. Therefore, *dynamic Web service composition* becomes a very important research field with growing research interest and practical value. Particularly, some mechanisms are required to automatically detect the changes or recover from the faults [17]. Additionally, in the context of semantic Web service composition, dynamic Web service composition must also handle the changes in the ontology.

Different service composition approaches [24, 27, 40, 54, 64, 83, 88, 106, 121] have been proposed to cope with those composition challenges discussed above and they can be classified into two main categories: *semi-automated Web service composition* and *fully automated Web service composition*. The first category of composition approaches require human beings to manually create abstract service workflows. It is generally assumed that a pre-defined abstract workflow is given a priori. The optimization problem in this approach turns to selecting concrete services with the best possible quality for each abstract service in the given workflow. Due to a tremendous growth in industries and enterprise applications, the number of Web services has increased dramatically at an unprecedented pace. The process of designing abstract workflows manually is fraught with difficulties in effectively and efficiently solving composition problems [54], such as QoS-aware service composition problem. Therefore, full automation techniques have been introduced to enable Web service composition with less human intervention, less time consumption, and high productivity [85]. The differences in fully automated approach lie in the fact that an abstract workflow is not provided, but instead created automatically with the service discovery and service selection.

Generating composition plans automatically in discovering and selecting suitable Web services is an NP-hard problem [69], which means the near-optimal composition solution is not likely to be found in a reasonable computation time. On the one hand, traditional approaches, such as Integer Linear Programming (ILP) and Dynamic Programming, often suffer scalability problems when the number of services increases and/or the size of ontology increases. On the other hand, to effectively and efficiently solve this problem, *Artificial Intelligence (AI) planning-based approaches*, *Evolutionary Computation (EC) techniques* and hybrid techniques have been introduced [80, 105]. AI planning is utilized to solve the automated Web service composition problems as a plan making process, from an initial state to a set of actions to a desired goal state, where services are considered as actions triggered by one state (i.e., inputs) and resulted in another state (i.e., outputs). In the second approach, heuristics have been employed to generate near-optimal solutions using a variety of EC techniques, e.g., Genetic Algorithms (GA), Genetic Programming (GP) and Particle Swarm Optimization (PSO). EC-based techniques have been demonstrated to be highly promising

in solving the *QoS-aware Web service composition* problems. The representation utilized in EC allows to represent a solution to a specific problem, and same problem can use different representations. Many representations have been carefully investigated in QoS-aware Web service composition problems, since they could significantly affect the performance of fully automated service composition approaches. In the third approach, a hybridization of AI planning-based approaches and EC-based approaches [24, 64] have been proposed to simultaneously ensure the correctness of service compositions and optimize the quality of composition solutions (e.g., QoS). These methods are considered to be more effective in finding more near-optimal solutions. As summarized here, a few challenges previously addressed, but not solved for fully automated service compositions lies in jointly optimizing QoS and semantic matchmaking quality, dealing with multi-objective semantic Web service composition for simultaneously optimizing different quality aspects, handling environment changes (i.e., changes in QoS, ontology and service repository) in dynamic service composition, and composing semantic Web services supporting precondition and effects while optimizing QoS and quality of semantic matchmaking. Those challenges are addressed by existing works, but with some key limitations discussed in Section 1.2.

The overall goal of this thesis is to propose effective and efficient approaches to comprehensive quality-aware automated Web service composition, which aims to jointly optimize semantic matchmaking quality and QoS. Meanwhile, this new approach also tackles several service composition problems, such as single-objective Web service composition, multi-objective Web service composition, dynamic semantic Web service composition and Web service composition supporting preconditions and postconditions.

1.2 Motivations

The motivations of this proposed research lies in five key aspects that simultaneously account for. 1. *Various techniques of combining AI planning and EC for automatic service composition*. 2. *Simultaneous handling the semantic matchmaking quality and QoS*. 3. *Multi-objective optimization*. 4. *Dynamic semantic service composition*. 5. *Semantic service composition supporting preconditions and effects*. These key aspects will be discussed in more detail below.

Various hybridized methods for Web service composition

Various techniques have been utilized to solve service composition problems, such as AI planning, local searching and EC-based techniques [33, 79, 83, 106]. AI planning is a prominent technique for handling Web service composition problems while always ensuring the correctness of the solution (i.e., all the inputs of involved services are satisfied) [106]. Local search techniques exhaustively search for solutions to optimization problems. In local search, solutions keep moving to the neighbor solutions, driven by some local maximization criterion until an optimal solution found or a time bound reached [79]. However, this technique has the shortage of suffering scalability and efficiency. On the other hand, EC-based techniques are outstanding at solving combinatorial optimization problems and are less prone to premature convergence in complex search spaces. To take the benefits from various techniques, various techniques of hybridization allows escaping local optima easily and improving the rate of convergence rate [87].

Various techniques have been employed to handle service composition problems in the literature [33, 65, 79, 80, 83, 106]. Many researchers investigated AI planning techniques for service composition problems using classical planning algorithms (e.g., GraphPlan [9]), where inputs, outputs, preconditions and effects are well defined along with the actions (i.e., services) [65, 80]. On the one hand, AI planning ensures both the correctness of functionality

and satisfactory of constraints, but it is always considered to be less efficient and less scalable, incapable of solving complicated optimization problems [79]. On the other hand, some researchers combined AI planning algorithms and local search algorithms to handle optimization problems, e.g., a combination of Graphplan and Dijkstras algorithm is proposed by [33] to find a correct solution with optimized QoS. EC techniques have been utilized to handle service composition problems, due to their promising performance in handling combinatorial optimization problems in a huge searching space. A few researchers also combined both local search and EC-based techniques for efficiently finding composition solution with optimized QoS [26, 79]. From the techniques discussed above, they are problem-specific for either optimizing QoS or the number of services. In this thesis, more complicated and realistic service composition problems are addressed. To cope with this composition problems featured in all the motivations discussed below (i.e., comprehensive quality of semantic Web service composition, multi-objective optimization, dynamic semantic service composition, and composing semantic Web services supporting preconditions and postconditions), EC-based techniques, optionally combing with AI planning or local search, are motivated to be proposed in our problems. For example, one heuristic in an EC-based technique is initially utilized for global search, while another heuristic is additionally designed for local search, which determining neighborhood of composition solutions. This hybrid method may lead to more effective performance.

Comprehensive Quality of Semantic Web service Composition

Web service compositions often focus on optimizing non-functional attributes (i.e., QoS). In the domain of semantic Web services, often, the provided information (i.e., outputs from proceeding services and from service requests) does not perfectly match the required information (i.e., inputs) according to their semantic descriptions [56]. Then, to achieve optimized quality of these matches (i.e., quality of semantic matchmaking) become one part of the goal of service compositions [54]. Therefore, a comprehensive quality model will be proposed to simultaneously consider both QoS and semantic matchmaking quality as a novel combinatorial optimization problem. From a practical perspective, many different service compositions can meet a user request but differ significantly in terms of QoS and semantic matchmaking quality. For example, in a classical travel planning context, some component services must be composed to generate a *travel map*. Suppose that two services can be considered for this purpose. One service S provide a *street map* at a *price* of 6.72. The other service S' can provide a *tourist map* at a *price* of 16.87. According to the corresponding service ontology, a *tourist map* is more desirable than a *street map*, S' clearly enjoys better semantic matchmaking quality than S but has negative impact on the QoS of the service composition (i.e., the price is much higher). One can easily imagine that similar challenges frequently occur when looking for service compositions. Hence, a good balance between QoS and semantic matchmaking quality is called for.

Existing works on service composition mainly address either QoS or semantic matchmaking quality. For the semantic matchmaking quality, existing works mainly focused on the discovery of atomic services, i.e., one-to-one matching of user requirements to a single service. Some works [6, 11, 68] on semantic service composition often optimize the number of services or the size of a graph representation for Web service composition. These approaches do not always guarantee optimized overall QoS, so huge efforts have been devoted to studying QoS-aware Web service composition [22, 27, 40, 64, 83, 121]. Some of these works do consider different semantic matchmaking types (e.g., Exact and Plugin matches [76]) when they compose services, but do not evaluate the quality of semantic matchmaking. It is not sufficient to only consider one quality aspect for optimizing service composi-

tion. However, only very few works address both semantic matchmaking quality and QoS for the Web service composition problems. To the best of our knowledge, [32, 54, 82] reported about first attempts that consider both aspects together, but only semi-automated approaches are proposed in their works. For these reasons, there is a need to device a comprehensive quality model for jointly consider the two quality aspects. Apart from that, new fully automated approaches need to be proposed to handle this optimization problem and explicitly support the comprehensive quality.

Multi-Objective Optimization for Service Composition

Depending on different goals of optimization, existing approaches for handling Web service composition problems fall into two groups, i.e., a single-objective or multiple-objective methods. In single-objective service compositions, one composition solution is always returned for a composition task, where the preferences of each quality component within the single objective (e.g., a weighted sum of different quality criteria) are provided by users. However, users do not always have clear preferences for many quality criteria in advance, and trade-offs presented by two or more conflicting criteria are also taken into optimal decisions. Therefore, multi-objective optimization is a natural approach to generate a set of trade-off solutions that concern about the conflicting and independent quality criteria. For example, premium users do not care cost as much as price-sensitive users do. Therefore, premium users may prefer a composition solution with the lowest execution time and the highest cost. To suit users with different preferences, multi-objective Web service composition approaches are proposed for producing a set of solutions, for the situation when the preferences of different quality are not clear and hard to determine in advance, and when single-objective optimization using weighted sum method cannot reach solutions in the non-convex regions [46].

Existing research on the automated Web service composition mainly concentrates on single objective problems for QoS-aware Web service compositions [16, 22, 23, 54, 60, 62, 64, 82, 88, 96, 121]. That is, only one solution is generated by a unified QoS ranking score to the users. To our best knowledge, existing works on multi-objective service composition [61, 100, 114, 115] are only approached by semi-automated methods to handle the different quality criteria in QoS. Simultaneously constructing workflows and selecting proper component services for optimizing multiple objectives is a very challenging work. Some constraints defined SLA have also been taken into account by some approaches [100, 115] and specify an official commitment that prevails between service providers and service users. For example, each quality of QoS are agreed between the providers and the users. These constraints raise the complexity of absolute Preto priority relation [37].

Apart from SLA constraints, users often have vague thoughts about what solutions are preferred. For example, a maximally acceptable trade-off for service execution cost and reliability can be one vague idea about what solutions are preferred. These partial preferences should be used for searching the most preferred regions. Many preference articulation techniques for multi-objective optimization have been studied [13, 14, 15, 19, 38], these approaches can be classified into three groups based on the articulation of preferences [97]. The first group utilizes articulation before optimization process, i.e., priori approaches. The second group utilizes articulation during optimization process, i.e., interactive approaches. The third group utilizes articulation after optimization process, i.e., posteriori approaches. As argued in [38], the third approaches are capable of generating more desired solutions in the preferred regions using biased crowding distance [14] or reference vectors [19]. However, all these approaches have not been used for solving service composition problem supporting user preferences on different quality aspect within a comprehensive quality. In par-

ticular, user preference models on comprehensive quality-aware service composition are required along with useful modifications of posteriori methods for effective cope with our service composition problems. From above discussion, these is a lack of fully automated approaches to multi-objective Web service composition problems for QoS-aware Web service composition abiding by constraints on SLA. Meanwhile, the insufficiency of handling only non-functional attributes (i.e., QoS) gives rise to considering additional semantic matchmaking quality as one objective for the multi-objective service composition. Further, user preference articulation techniques have not been studied in service composition problem so far.

Dynamic Semantic Web service Composition

In a dynamic environment, QoS of atomic services in a service repository is fluctuating over time [108]. Static service composition solution is no longer enough, and remedy actions must be taken if the original composition solution changes in QoS or is no longer executable due to any service involved that goes offline. In addition, the ontology used for describing services may also change. Apart from that, new services may be registered from time to time, and might be used to improve the overall QoS or quality of semantic matchmaking. Therefore, dynamic semantic Web service composition will be proposed to effectively and efficiently update composition solutions when their performance and executability cannot be guaranteed any further [58].

Most approaches on dynamic service composition aim to design effective and efficient methods for dynamic adaption or service re-selection for each component services in a service composition solution, which do not allow any changes to the service composition structure (workflow). Apart from that, the cost of initial planning is ignored and separated from the adaption of dynamic environment. On the one hand, some techniques [4, 7, 48] endeavor to update outdated or incorrect compositions, and they allow for dynamic adaptation of the solutions based on implementation of variability constructs at the language level. For example, a composition language extending a typical WS-BPEL [4] is proposed for supporting the dynamic adaption using ECA (Event Condition Action) rules, which is utilized for guiding the operations for self-reconfiguration. This approach is difficult to manage and error-prone. Based on and by extending the previous approaches, variability model-based approaches [3] have been proposed to support the adaption of service composition solutions. Meanwhile, some other works [70, 90, 101, 116] utilize decision tree learning, reinforcement learning and Rtree query techniques for handling dynamic service composition problem. Those non-EC based approaches focus on developing efficient methods for service re-selection to replace services once negative QoS changes and/or service failures are detected. They leave the problem on handing changes in the ontology and newly registered service in the service repository aside.

On the other hand, While showing promise, existing EC-based approaches to Web service composition have only been studied in static scenarios, rather than dynamic ones. Although some EC-based approaches [33, 61] claim that their approaches fit the dynamic problems due to the natural features of their adopted algorithm (i.e., a continuous optimization process), there is no discussion about how their approaches can be applied to dynamic service composition problems. However, EC-based techniques can overcome two limitations. That is, allowing the changes of composition structure and reducing re-planning cost through re-using known plans (composite services) obtained prior. First, a proper number of individuals (i.e., service composition solutions) stored could be used for retrieving an alternative composition solution in the case of service changes. This is a save of computation cost for re-planning. Second, the stored individuals could be further evolved to generate a

new composite service with changes in component services or composition structure. Therefore, due to the nature of EC, EC-based approaches to dynamic Web service composition can be explored to keep previously generated population, which can be further evolved using evolutionary operators in a timely manner. Given above discussions, it is very advisable to investigate EC techniques to a dynamic semantic composition problem.

Automated Semantic Web service Composition Supporting Preconditions and Effects

Since conditions and effects do not apply to most of Web services, it is considered a good strategy to achieve service composition based mainly on inputs and outputs. However, Web services can include conditions and effects occasionally, these conditional constraints also determine the executability of services, which lead to more complex service composition constructs (i.e., choice and loop) to be considered. For example, an online book shopping system [106] provides a service for purchasing books. Users expect a purchasing outcome (e.g., receipts and book deliveries) returned if books and customer details (e.g., title, author, customer information) are given. In this case, users may have specific constraints. If the customer has enough money to pay for the book in full amount, they would like to do so. Otherwise, the customer would like to pay by installments. Therefore, the constraints on their current account balance needs to be handled during the execution of the service composition.

Most of the existing approaches to automated Web service composition do not consider conditional constraints [75]. Many EC-based approaches have shown their promise in solving service composition problems represented only by input and outputs, leave preconditions and effects aside. However, a few approaches [6, 12] have been explored to achieve compositions that consider precondition and effects using AI planning, since AI planning ensures both the correctness of functionality and satisfaction on the conditional constraints (i.e., preconditions and effects). These constraints are automated constructed with different composition constructs, such as choice and loop. Meanwhile, AI planning approaches are utilized with local search for tackling optimization problems. These methods suffer hugely in terms of efficiency, scalability, and computation cost. Therefore, efficient methods are needed to for semantic Web service composition supporting preconditions and effects.

1.3 Research Goals

The overall goal of this thesis is to *develop new and effective EC-based hybrid approaches for comprehensive quality-aware automated semantic Web service composition*. More specifically, the research focus will be on: (1) developing EC-based approaches that could explicitly support a proposed comprehensive quality model that jointly optimizes QoS and semantic matchmaking quality, (2) developing multi-objective approaches for optimizing multiple quality criteria involved in the comprehensive quality model, (3) developing EC-based composition techniques for handling various changes in composition environment, and (4) developing EC-based approaches to semantic Web service composition supporting precondition and effects. This research aims to develop EC-based approaches, optionally combined with local search and/or AI planning techniques, for effectively and efficiently handling the research focus above. The research goal described above can be achieved by completing the following set of objectives:

1. **To develop EC-based approaches to comprehensive quality-aware automated semantic Web service composition that simultaneously optimizes both QoS and semantic matchmaking quality**. Particularly, we extend existing works on QoS-aware

service composition by considering jointly optimizing both quality aspects, which will be formalized collectively through a comprehensive quality model. Meanwhile, representations of the composition solutions are a key aspect of the approaches, since they must maintain all the information required for the evaluation. Therefore, we will investigate the following sub-objectives to handle this objective.

(a) *To propose a comprehensive quality model that addresses QoS and semantic matchmaking quality simultaneously to reach a desirable compromise.* We aim to establish a quality model with a simple calculation and a good performance. This model is utilized for evaluating QoS and quality of semantic matchmaking of service composition solutions. Meanwhile, to enable an empirical experiment, this model must support evaluations on most of existing benchmark datasets, e.g., Web service Challenge 2009 (WSC09)[47] and OWLS-TC [51].

(b) *To propose indirect solution representations for comprehensive quality-aware Web service composition.* The *indirect representations* do not present the final service composition solutions directly, they must be decoded to produce executable service composition solutions. Previous studies have shown their good performances in searching an optimal solution for QoS-aware Web service composition [26, 27]. However, the decoding process could increase the overall computation time. Apart from that, the indirect representations potentially increase the searching space as discussed in [27]. They also do not preserve semantic matchmaking information utilized for the evaluation of semantic matchmaking quality. To overcome these disadvantages, we will propose an effective indirect presentation for comprehensive quality-aware semantic Web service composition.

(c) *To propose direct solution representations for comprehensive quality-aware Web service composition.* Graph-based and tree-based representations are widely used for directly representing service composition solutions. Graph-based representations are capable of presenting all the semantic matchmaking relationships as edges [54, 103], but hardly supporting some composition constructs (e.g., loop and choice). Tree-based representations could be more ideal for practical use, since they can present all composition constructs as inner nodes of trees. However, they could hardly maintain all the edge-related relationships supported by graphs, and their size is often very large with multiple isomorphic copies. To overcome the limitations of tree-based representations, we aim to propose a tree-like representation. In particular, any isomorphic copy in the traditional tree-based representations is removed, and an edge is inserted to the root of the copy. Meanwhile, specific genetic operators will be developed without breaking the functionality of the inserted edge.

(d) *To propose EC-based hybrid methods to effectively and efficiently handle the problem for comprehensive quality-aware automated Web service composition.* The reasons of utilizing hybrid techniques are briefly discussed in Subsection 1.2, and they aim to efficiently and effectively handle service composition problems. Herein, memetic approaches are developed for supporting both the proposed indirect and indirect representations. In addition, new and effective genetic operators are to be developed for EC-based algorithms. The introduced local search techniques will be developed with properly designed structures of neighboring solutions for easily escaping from local optimal. Apart from that, these approaches must explicitly support the comprehensive quality model. Further more, AI planning techniques will be incorporated into the newly memetic approaches to expedite the convergence speed.

2. **To develop multi-objective approaches to fully automated comprehensive quality-aware semantic service composition**. In practice, many quality criteria proposed in our comprehensive quality model are often conflicting in natural, such as an amount of money paid for service execution and a possibility of service availability. Existing multi-objective approaches [18, 110, 115, 61, 118, 124] to Web service composition mainly concentrate on semi-automated QoS-aware Web service composition. Therefore, research must be carried out not only for a better understanding of inherent tradeoffs among different objectives (e.g., quality of semantic matchmaking and QoS are naturally considered as two conflicting objectives), but also for developing fully automated approaches that reduce human involvement and improve composition quality. It is approached by utilizing cutting-edge multi-objective optimization algorithms (e.g, NSGA-II [30], SPEA2 [128] and MOEA/D [123]). These algorithms are needed for finding a Pareto front of evolved solutions. Meanwhile, different representations may not perform equally well, so a study on improving the effectiveness of different representations along with different Evolutionary Multi-Objective Optimization (EMO) techniques also arouses researchers' interest. Apart from that, SLA consideration needs to be taken into account to satisfy a QoS-level contract between service providers and users. It is also necessary to consider customized matchmaking levels to bring the flexibility for meeting different requirements from segmented users (e.g., platinum users, gold users and normal users). Further more, preferences articulation techniques for EMO have been shown their promise in finding most preferred regions of the Pareto Front [13, 14, 15, 19, 38], it is very motivated to investigate these techniques in EMO-based semantic Web service composition.

   (a) *To develop a EMO-based approaches for multi-objective fully automated semantic Web service composition*. Herein we will propose useful modifications of multi-objective EC algorithms (for example, NSGA-II [30], SPEA2 [128] and MOEA/D [123]) simultaneously to improve the effectiveness and efficiency of our service composition approaches. This sub-objective is also established for an in-depth investigation of each quality criteria based on our proposed comprehensive quality model in Objective 1. In particular, both quality of semantic matchmaking and QoS must be optimized separately, since they can conflict each other. It would be interesting to examine different tradeoffs among the service composition solutions with respect to the different quality criterion. Apart from that, fully automated approaches are also developed to achieve high practical usefulness without replying on given abstract workflows.

   (b) *To develop EMO-based hybrid approaches for multi-objective fully automated semantic Web service composition*. Once we achieve the sub-objective 2(a), the effectiveness of the EMO-based approaches is the next focus. The EMO-based approaches should be extended by introducing local search. In particular, we aim to design an effective mechanism to decide how often, how long, which solutions that we apply the local search for targeting the neighboring solutions. For example, a local search is designed to target the neighborhood of the individuals of each current population. The possibility of performing local search is determined by the fitness value of each individual — the higher fitness value, the higher possibility.

   (c) *To develop EMO-based approaches for multi-objective fully automated semantic Web service composition subject to constraints on SLA and customized matchmaking level*. In the real world, sometimes, satisfaction on given SLA constraints is required in addition to optimizing QoS, i.e., multilevel constraints with lower and upper bounds for different individual QoS criterion [115]. Meanwhile, different cus-

tomized semantic matchmaking levels (e.g., exact matchmaking level and less strict matchmaking level) are also preferred by different users. Therefore, methods should be developed to cope with both SLA constraints on QoS and different accepted matchmaking level. We aim to propose a vector that represents QoS and semantic matchmaking level constraints, which is modeled as a set of lower and upper bounds for each quality criterion. To effectively handle the constraints, the vector could be utilized in the fitness function for penalizing the individuals that breach the constraints.

(d) *To develop EMO-based approaches for multi-objective fully automated semantic Web service composition subject to preference articulation*. The articulation of user preferences could be achieved either before (i.e., a priori), during (i.e., interactive), or after (i.e., a posteriori) an EMO process. As argued in [38], a posteriori preference articulation often results in more and more desired solutions on the preferred regions of a Pareto Front. We aim to develop fully automated multi-objective semantic Web service composition approach integrating posteriori preference articulation methods. In particular, an effective preference model will be created to cope with full or partial preference information according to users' preferences. Meanwhile, posteriori articulation techniques should be modified to explicitly support this preference model and guild the solutions towards both true Pareto Front and the most preferred regions.

3. **To develop EC-based techniques to achieve dynamic semantic Web service composition effectively**. Objectives 1 and 2 are proposed for solving service composition problems in static composition environments. In this proposal, composition environment refers to the registered services in the service repository, non-functional attributes advertised by service providers, and the ontology utilized for describing the properties of Web services. On the one hand, traditional approaches [72, 90, 101, 116] work on efficient methods for service re-selection by replacing component services, where only changes in QoS and service failures are handled. Technically, their approaches do not allow changes in service composition structure, and cannot reduce re-planning cost through re-using known plans (composite services) obtained prior. On the other hand, EC-based approaches are motived to be developed as they can easily overcome the two limitations above. Some factors that significantly impact the execution of the plan are to be handled in this objective, such as changes in QoS, ontology and service repository. In particular, three studies are conducted as three sub-objectives below:

   (a) *To develop EC-based techniques to re-optimize solution candidates in response to changes in QoS and ontology.* Traditionally, an initial population is created with solution candidates that are further evolved for searching optimal solutions. After obtaining a near-optimal solution, most of service candidates are discarded. Those discarded solution candidates may be promising for good candidate solutions due to the changes in services repository or the ontology. These candidates preserve both diversity and elitism, which motivate us to continue the evolutionary process through re-using these candidates (composite services). Therefore, we will propose an EC-based approach, which take the previously computed population as an initialization and design efficient evolutionary operators to generate a replacement in a timely manner.

   (b) *To develop EC-based hybrid techniques to re-optimize solution candidates for changes in QoS and ontology.* Once the EC-based techniques to re-optimize solution candi-

dates for changes in QoS and ontology are achieved, it should be further studied for developing more effective approach to handle this problem. We aim to propose an adaptive and hybrid approach to this dynamic problem. On the one hand, we firstly assign a higher priority to a group of services with changes. This priority is utilized in service selection for building up new individuals (i.e., service composition solutions), and must be adaptively handled with a properly decreasing rate with respect to each service. That is, the assigned priority decreases when the services are to be selected in the future. On the other hand, the hybrid refers to a memetic approach that combines an EC-based approach and a local search strategy, where the local search is developed to accelerate the convergency and decrease the number of evaluation times. In this sub-objective, we aim to achieve more effective performance compared to the EC-based approaches in Objective 3(a).

(c) *To develop EC-based techniques for handling service failure and new service registration using updated candidates in the population.* Apart from the changes in the QoS and the ontology, occasionally, existing services may fail and/or new service may become available. For the case of new service registrations, efficient methods will be proposed to update the plan with new services in a timely manner. When new services are registered in the service repository, we discard a portion of the current population with relatively low fitness values. Then, we replenish population from an updated services repository. For the case of service failure, efficient approaches need to be proposed to either mutate on an un-invokable component service of an individual, or a proposed parent structure of the component service, or effectively re-generate whole solutions using invokable services in the service repository.

4. **To develop EC-based approaches to semantic Web service compositions supporting preconditions and effects (Optional)**. We plan to extend the service composition approaches (i.e., satisfactions on inputs and outputs) to include preconditions and effects. These conditional constraints also necessitate the study of various of composition constructs for automated semantic Web service composition, e.g., loop and choice. Since AI planning ensures both the correctness of functionality and satisfactions on the conditional constraints (i.e., preconditions and effects). Meanwhile, EC techniques are considered to be more flexible and more efficient. Therefore, Given the benefits of both AI planning and EC-based techniques, they are motivated to be collectively explored for automated Web service composition based on preconditions and effects. Herein three sub-objectives have been proposed as our targets as follow.

   (a) *To develop EC-based techniques for semantic Web service composition supporting preconditions and effects*. In this objective, inputs and outputs are not everything for the execution of Web services, our composition problem is extended by considering not only inputs and outputs, but also preconditions and effects. In particular, we need to establish a general matchmaking mechanism for satisfactions on preconditions and effects, since there is no systematic studying in existing works. We aim to model inputs and outputs as variable names and further referenced in preconditions and effects, which can be distinguished as different instances from a knowledge base. In particular, preconditions are assigned to the description of requirements on inputs using logic formulas. The effects are restricted to the description of constraints on returned outputs, relationships between inputs and outputs, and changes caused by the service in the knowledge bases. Based

on the mechanism, two simple composition constructs (i.e., sequence and parallel) are automatically constructed for supporting preconditions and effects in this sub-objective. Meanwhile, we aim to develop EC-based approaches to effectively handle this problem with new representations to copy with preconditions and effects, new evolutionary operators will be investigated to meet the requirements of the proposed representations.

(b) *To develop EC-based hybrid techniques for semantic Web service composition supporting preconditions and effects*. Once the EC-based techniques to semantic Web service composition supporting preconditions and effects are proposed. More effective techniques should be developed. In particular, we aim to create a memetic approach that utilize an EC-based approach and a local search strategy to avoid being trapped in a local optimal. This memetic approach is expect to benefit in an accelerated convergency rate and a decreasing number of evaluation times.

(c) *To develop EC-based techniques for semantic Web service composition supporting preconditions and effects for loops and choice*. We initially extend the matchmaking mechanism of satisfactions on preconditions and effects to include loops and choice composition constructs supports. To extensively cope with these two constructs, new and effective representations must be studied since the old representation only consider sequence and parallel constructs. Apart from that, EC-based approaches will be developed to effectively solve this problem.

1.4 Published Papers

During the initial stage of this research, the preliminary work was carried out on establishing the comprehensive quality model. Afterwards, some studies on the direct and indirect representations are completed for one part of Objective 1, but the earlier works focus on static Web service composition using single-objective optimization technique. The following are the publications made from the preliminary studies:

- WANG, C., MA, H., CHEN, A., AND HARTMANN, S. "Comprehensive Quality-Aware Automated Semantic Web service Composition". *AI 2017: Advances in Artificial Intelligence: 30th Australasian Joint Conference*. 2017, pp. 195-207.
- WANG, C., Ma, H., CHEN, A., AND HARTMANN, S.: "GP-Based Approach to Comprehensive quality-aware automated semantic Web service composition". In: SEAL2017: International Conference on Simulated Evolution and Learning(To appear)

1.5 Organization of Proposal

The remainder of the proposal is organized as follows: Chapter 2 provides a fundamental definition of the Web service composition problem and performs a literature review covering a range of works in this field; Chapter 3 presents a formal definition for our service composition problem, and discusses the preliminary work explores direct and indirect representations for comprehensive quality-aware semantic Web service composition using a hybridization of AI planning techniques and EC-based techniques; Chapter 4 presents a plan detailing this project's intended contributions, a project timeline, and a thesis outline.

Chapter 2

Literature Review

In this chapter, we firstly introduce the background knowledge of Web service in Subsection 2.1.1 and Web service composition in Subsection 2.1.2. Afterwards, EC techniques will be briefly introduced in Subsection 2.1.3. Following that, Section 2.2 reviews single-objective service composition using EC and non-EC based techniques in Subsection 2.2.1. Subsection 2.2.2 reviews existing works on multi-objective approaches, many-objective approaches, and preference articulation techniques for multi-objective approaches. Dynamic Web service composition is covered in Subsection 2.2.3. Subsection 2.2.4 discusses service composition supporting preconditions and effects. Lastly, Subsection 2.2.5 summarizes some key reviews and limitations in the literature review.

2.1 Background

2.1.1 Web Service

Web services are self-describing, self-contained software modules available over the internet [31]. As described in [31], Web services are loosely coupled software modules, where service interfaces enable applications to work cooperatively and defined in a neutral way regardless the platforms, operation systems and programming languages. Besides that, Web services are distributed via internet and communicate via internet protocol, such as HTTP, and are described by standard description languages, e.g., WSDL. These description languages are mainly used to describe functional properties of Web services in terms of service inputs and service outputs, and provide mechanisms to allow users to search desired services.

Web services are classified into two groups based on their functionalities: *information-providing services* and *world-altering services* [67]. The first type of services expect some data returned by giving inputs or nothing. For example, an air velocity transducer Web service reads the wind speed and returns the velocity at the time. This service does not require any inputs. On the other side, a city weather Web service requires a city name and returns weather information for that city. Information-providing services do not produce any side effect to the world. The functionalities of these services are only described as inputs and outputs. The second type of services not only provide data information but also alters the status of the world by producing side effects. For example, a PayPal service will cause a deduction in the balance of users' bank account. *In this proposal, we mainly focus on the first type of services for first three objective. Later on, an extensive study is optionally carried for the second type of services*

As demonstrated above, the functional attributes determine what service really does, while the non-functional attributes often refer to some quality criteria, which is considered for raking services [2]. When several services provide the same functionality, users would

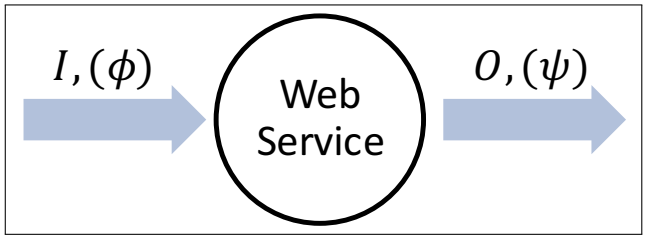


Figure 2.1: Functional Properties of a Web Service.

prefer a service with a lower cost. Herein we will demonstrate both the functional and non-functional properties in following subsections.

Functional Properties of Web Services

The operational characteristics of Web services are related to the functional properties, which demonstrate the behaviors of Web services, i.e., what information is needed to successfully invoke a service and what information will be returned after execution. In other words, a set of inputs I is required by a service and a set of output O is returned by a service. Sometimes, preconditions ϕ are required and effects φ are produced along with the outputs. For *semantic Web services*, an ontology is employed to semantically describe the properties of Web services. Therefore, a knowledge base is created to enable a better interoperability of the functional properties. Sometimes, the preconditions ϕ must be hold in the knowledge base before service execution, and they must remain constant while passing the input I. Side effects are also created as effects φ along with outputs O. All functional properties of Web services are demonstrated in Fig. 2.1.

Nonfunctional Properties of Web Services

Apart from the functional properties of Web services discussed above, the non-functional properties of Web services also play an important part in composing services. One important type of non-functional property is QoS. For example, customers always prefer a web service with the lowest execution cost and the highest response time and reliability. According to [122], four most often considered QoS parameters are as follows:

- *Response time* (T) measures the expected delay in seconds between the moment when a request is sent and the moment when the results are received.
- *Cost* (C) is the amount of money that a service requester has to pay for executing the Web service
- *Reliability* (R) is the probability that a request is correctly responded within the maximum expected time frame.
- *Availability* (A) is the probability that a Web service is accessible.

Service level agreement (SLA) is another important type of nonfunctional property of Web services. It serves as a declarative contract, which clarifies a QoS level that is agreed by service providers and service users [125]. For example, an SLA is clarified as end-to-end QoS constraints on a business process (that is achieved by Web services), which presents how a certain business goal is achieved [99].

Web Service Discovery

To generate a service composition solution, mechanisms must be provided for discovering required services. Therefore, service discovery becomes a fundamental technique to be considered in all service composition approaches. Agarwal et al. [1] discuss three mechanisms of semantic Web service discovery: classification-based approach, functionality-based approach and hybrid approach. Those service discovery techniques are further demonstrated below.

The first service discovery technique makes use of classes provided by service semantic annotation in WSMO-Lite language. Therefore, service requesters can use class names to express a goal, which is a straightforward discovery from a set of classes. However, classes without clear semantic definition could lead to an issue of incomprehensibility of Web service discovery. For example, several classes may be declared in either different terms for the same content, or the same term for the different contents.

The second service discovery technique does not take classes into account, but consider functional properties of a Web service. In particular, a desired functionality is defined by the functional properties of Web services. A discovery algorithm must be developed to handle a match for different input, output, precondition and effects with associated concepts and relations in a given domain knowledge base. The key idea of the matchmaking is to check whether services accept all the inputs provided by users and whether the desired outputs is delivered by services. In addition, this discovery technique also checks for the satisfactions of implications that actual precondition and actual effects must imply the desired precondition and desired effects, respectively. The strength of this technique is that it potentially meets the demands of all the comprehensible discovery, but lacks of efficiency and scalability.

The third service discovery technique is based on a hybridization of classification and functionality-based discovery. A classification hierarchy is proposed to achieve automatic semantic reasoning regarding the functionalities of classes, inputs and outputs provided by different services. For example, a class can be associated with super classes and sub classes, where more general functionality and more specific functionality are presented, respectively. Meanwhile, to make a consistency of classification hierarchy, all the inputs, outputs, preconditions and effects of a functionality class must satisfy the conditions that contains all the inputs, outputs, precondition and effects of all the classes it is subsumed by. The advantage of this approach is to achieve better performance combining strengths of the previous two pure classification-based and functionality-based approaches. However, the classification hierarchy demands lots of work to remain consistent when a new Web service is published or updated.

As discussed above, the first and third approach is considered either less effective or demands researchers to build up an ontology to support class hierarchy along with functional properties. It is not the focus of our research for building up any ontology. *We propose to use the second service discovery technique for meeting a comprehensible discovery. That is, ontology reasoning is utilized to approach a semantic matchmaking as a fundamental part of service composition algorithm.*

2.1.2 Web Service Composition

Since an atomic Web service may not satisfy or fully satisfy users' complex functional requirements, Web service composition collectively composes existing Web services to produce a complex functionality to meet the requirements. On the one hand, manual service composition is very time-consuming and less productive. Therefore, many approaches have been developed to achieve semi-automated or fully automated service composition.

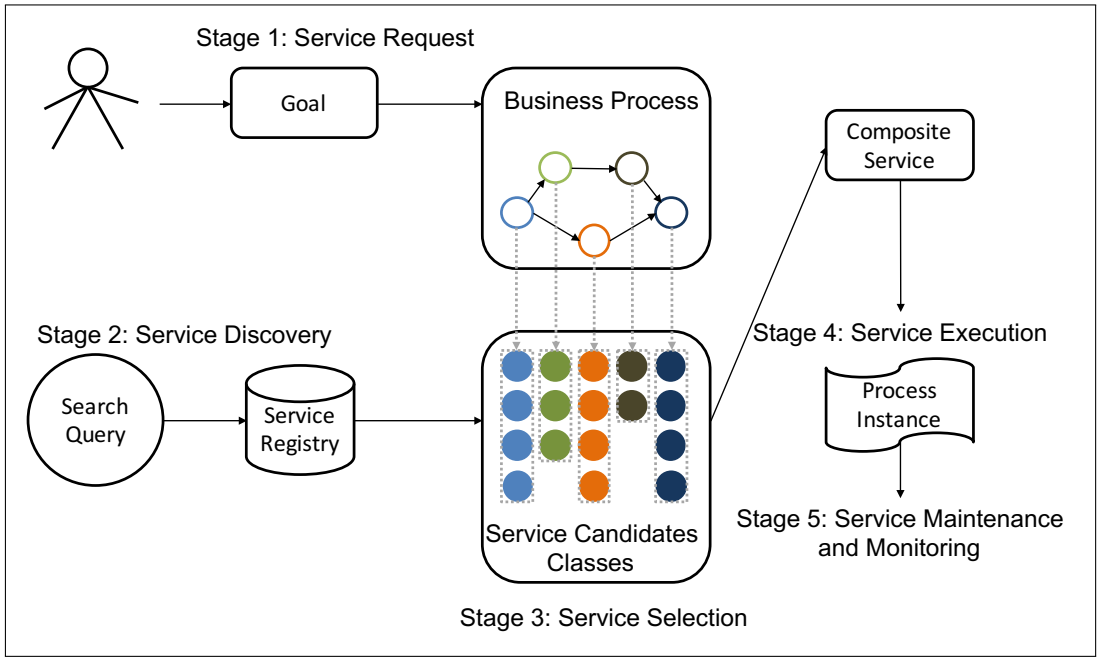


Figure 2.2: Semi-automated Web service composition lifecycle [69].

The *semi-automated service composition* is inspired by the business process that requires prior knowledge to build up abstract workflows [69]. These workflows consist of abstract services, each of which is defined by input-output pairs. The details of these steps are further discussed below. On the other hand, when we are composing services, the interoperability of services is very important in both semi-automated and fully automated Web service composition. In particular, several problems are simultaneously taken into account. That is I/O matchmaking (i.e., a mechanism for ensuring the interoperability), service discovery for finding services with required functionalities, and service selection for optimizing the quality of service composition.

Semi-Automated Web Service Composition

We firstly demonstrate semi-automated Web service composition, where several steps are shown in Fig. 2.2. The detail of the service composition lifecycle is discussed as follows:

1. *Service request:* The first step in service composition is to collect users' requirements for composition goal that comprises of the functional (i.e., correct data flow) and non-functional sides (i.e., QoS). This step is achieved by building up an abstract workflow including a series of service discovery tasks (i.e, abstract services) with clearly defined functionalities. Those tasks could be completed by selecting proper concrete services to reach desired QoS. *Service request* is different from what we discussed in the fully automated service composition, which does not include an abstract workflow.

2. *Service discovery:* Once the goal is clearly specified, concrete Web services will be selected for each task regarding its functional requirement. Often, more than one concrete Web service is likely to be found to match one service discovery task. However, those matched Web services are always different in QoS.

3. *Service selection:* At this stage, many techniques have been studied to select Web services to best match each task for the satisfying functional requirement and overall QoS

of the business process (i.e., composite service). Therefore, a plan of service composition is created ahead of execution.

4. *Service execution:* the process instance is monitored for any changes or services failures during service execution. In this stage, some actions are to be taken for adapting the changes.

Fully Automated Web Service Composition

There is a distinction between semi-automated and fully automated approaches. In context of semi-automated service composition, for a given service request and a service repository, the aim of service composition is to find concrete services for each abstract service of a workflow. It is impractical to manually design an optimal workflow. Hence, fully automated service composition is needed to generate a workflow of service composition. The service request (also called composition task) for fully automated service composition approaches only consists of task inputs and task outputs, which specify the information gathered from users and information expected to be returned respectively. Often, *fully automated service composition* rely on some algorithms (e.g., Graphplan algorithm [9]) to achieve service composition, where service workflow is gradually built up along with service discovery and service selection.

Fig. 2.3 shows a popular Web service composition example from a travel domain. In this scenario, the agency provides customers with a serial of services to satisfy their requirements for booking flights, accommodations and buses, and generating tourist maps for the conference city. In Fig. 2.3, the task inputs (i.e., $TravelDepartureDate$, $TravelReturnDate$, $HomeCity$ and $ConferenceCity$) are gathered from customers, and task outputs (i.e., $BusTicket$, $Flight\ Ticket$, $HotelReservation$ and $StreepMap$) are expected to be returned. The component services are gradually discovered and selected to construct a workflow from $TaskInput$ node to $TaskOut$ node, if the inputs of selected services are fulfilled with $TaskInput$ or the outputs of previously selected services. Therefore, all the inputs of all the component services are satisfied. In this example, we begin by executing the FlightBooking service and GenerateMap service, the FlightBooking service books the flights and determines a $arrivalDate$. Then, using the $arrivalDate$ together with other given data, we can book the hotel and bus, and generate a Map for the conference city. Together, these four services produce all required outputs for customers.

In our thesis, we will concentrate on developing methods for fully automated service composition, where service discovery and service selection are considered as interrelated tasks that are interleaved with the composition algorithm.

Functional Properties of Web Service Composition

From the example demonstrated above, two characteristics are addressed for the functional properties of Web service composition. One is that all the inputs of component services must be matched by given task inputs and the outputs of proceeding component services, so service composition solutions can be successfully executed. Another is that task output must be a subset of all the outputs of component Web services, so a service request for composing services is achieved.

Semantic Web Service Composition

Semantic Web service composition is achieved by composing semantic Web services through reasoning about their semantically described properties. Often, the inputs of a component

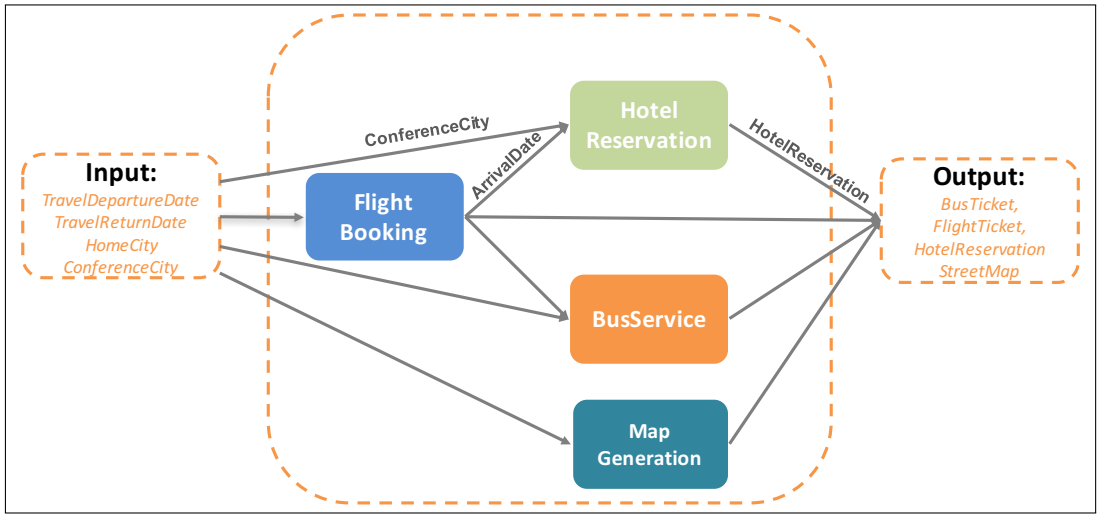


Figure 2.3: An example of service composition for a travel agency.

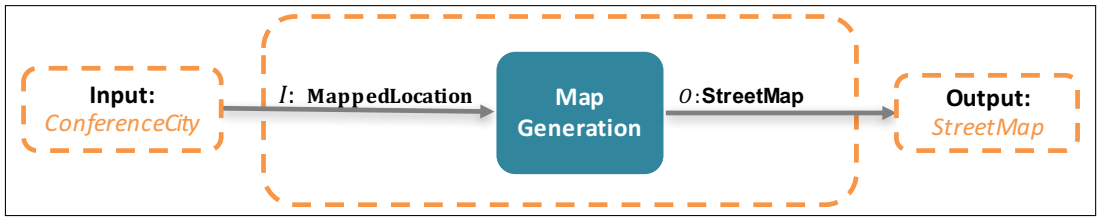


Figure 2.4: An example for demonstrating semantic matchmaking quality.

service cannot be perfectly matched by the task inputs and the outputs of the proceeding services. For example, in Fig. 2.4, a MapGeneration service requires inputs $MappedLocation$ and produces output $StreetMap$. Given that $instance-of(ConferenceCity, City)$, $instance-of(MappedLocation, Location)$ and $City \sqsubseteq Location$. Based on the ontology-based description, $ConferenceCity$ does not perfectly match $MappedLocation$. Therefore, quality of semantic matchmaking between the given information $ConferenceCity$ and the required input $MappedLocation$ is relatively low.

In semantic web service composition, substantial works [6, 54, 55, 57, 86] utilize description logic (DL) reasoning between input and output parameters of Web services for matchmaking, where different matchmaking types are considered associated with *different matchmaking qualities*. Therefore, exploring an effective mechanism to measure the quality of semantic matchmaking for semantic service composition is one of crucial tasks in our research.

Firstly, we demonstrate different *matchmaking types* discussed above here. Given two concepts a, b in ontology $\mathcal{O}$, four commonly used matchmaking types are often used to describe the level of matches [76]:

- *exact* returned, if a and b are equivalent ($a \equiv b$),
- *plugin* returned, if a is a sub-concept of b ($a \sqsubseteq b$),
- *subsume* returned, if a is a super-concept of b ($a \sqsupseteq b$),
- *fail* returned, if none of previous matchmaking types is returned.

Secondly, to our best knowledge, three methods [55, 82, 91] are utilized for measuring

the quality of semantic matchmaking in the domain of service composition. We will demonstrate these three methods below.

The first method measures the quality of matchmaking regarding the different matchmaking types. In [55], the quality of semantic matchmaking is measured by the two quality criterion, matchmaking types and common description rate. They additionally consider *interaction* matchmaking type ($a \sqcap b$), i.e., if the intersection of a and b is satisfiable. In their work, a causal link $sl_{i,j} \doteq \langle S_i, Sim_T(Out\_s_i, In\_s_j), S_j \rangle$ is created between inputs of service S_i and outputs of S_j. In particular, both *exact* match and *plugin* match are presented as robust causal links, while both *subsume* match and *intersection* match are presented as valid casual links. However, valid casual links are not specific enough to be utilized as the input of another Web service. Thus the output requires Extra Description, denoted as $In\_s_x \setminus Out\_s_y$, to enable proper service composition using Eq. 2.1. As a result, *subsume* and *intersection* are transferred to be *exact* and *plugin* respectively to formulate a robust link, so common description rate, $q_{cd}(sl_{i,j})$is calculated based on Extra Description and Least common subsume, denoted as $lcs(In\_s_x, Out\_s_y)$, using Eq. (2.1).

$$In\_s_x \setminus Out\_s_y \doteq \min_{\preceq d}\{B | B \sqcap Out\_s_y \equiv In\_s_x\}, since\ Out\_s_y \sqsupseteq In\_s_x \tag{2.1}$$

$$q_{cd}(sl_{i,j}) = \frac{lcs(In\_s_x, Out\_s_y)}{In\_s_x \setminus Out\_s_y + lcs(In\_s_x, Out\_s_y)} \tag{2.2}$$

The second method for measuring the quality of semantic matchmaking utilizes a similarity measurement from information retrieval. In [82], the similarity is calculated by an average value of $F\_Measure(S_i.out_k, S_j.in_k)$. $F\_Measure(S_i.out_k, S_j.in_k)$ measures the similarity between two matched output $S_i.out_k$ and input $S_j.in_k$, and it is calculated based precision and recall between these two matched output and input.

The third method for measuring the quality of semantic matchmaking utilize semantic similarity in [91]. For concepts a, b in $\mathcal{O}$, semantic similarity, denoted as $sim(a, b)$, is calculated based on an edge counting method in a taxonomy like WorldNet or Ontology using Eq. (2.3) [91]. This method has the advantages of a simple calculation and a good performance . In Eq. (2.3), N_a, N_b and N_c measure the distances from concept a, concept b, and the closest common ancestor c of a and b to the top concept of the ontology $\mathcal{O}$, respectively. L is the shortest distance between the two concepts, a and b, while D is the depth of ontology tree. Also, λ equals 1 for neighborhood concepts or 0 for concepts from same hierarchy.

$$sim(a, b) = \frac{2N_c \cdot e^{-\lambda L/D}}{N_a + N_b} \tag{2.3}$$

In this paper we are only interested in robust compositions, where only exact and plugin matches are considered, and we suggest to consider the semantic similarity of concepts when comparing different plugin matches. As argued in [54] *plugin* matches are less preferable than *exact* matches due to the overheads associated with data processing

Nonfunctional Properties of Web Service Composition

The nonfunctional properties of Web service composition is determined by the QoS of all the component Web services in the solutions. The aggregation value of QoS for Web services composition varies with respect to different constructs, which determines how services are associated with each other in a composite service [122].

- *Sequence construct*: service composition executes each atomic service associated with a sequence construct in a definite sequence order. The aggregated time (T) and execution cost (C) is as a sum of time and cost of Web services involved respectively. The aggregated availability and reliability in a sequence construct are calculated by multiplying the availability and reliability of each component Web service according to the probability theory. This construct is shown in Fig. 2.5.

- *Parallel construct*: Web services in a parallel construct are executed concurrently. The aggregated execution cost, availability and reliability are calculated in the same way as these for the sequence construct while the aggregated time (T) is determined by the most time-consuming path in the composite flow of the solution. This construct is presented in Fig. 2.6.

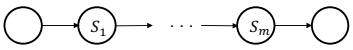
- *Choice construct*: Only one service path is executed in a choice construct depending on the satisfaction of the conditions on each path. In Fig. 2.7, assuming the choice construct has n branches, $p_1, \ldots, p_n$ with $\sum_{k=1}^{d} p_k = 1$ denote the probabilities of the different branches being selected. For example, the aggregated total cost C is the multiplication of the cost of each branching service and the possibility p of the branch.

- *Loop construct*: Web services composed with a loop construct are executed repeatedly until a certain condition is satisfied. In Fig. 2.8, assuming the average number of iterations is ℓ, and t, c, r, and a are corresponding aggregated value of a composite service. Therefore, For a loop construct, aggregated response time (T) and execution cost (C) are $p_n \cdot t$ and $p_n \cdot c$ respectively while aggregated availability A and reliability R are the ℓ^{th} power of the value of one iteration, i.e., $A = a^{\ell}$ and $R = r^{\ell}$.

2.1.3 An Overview of Evolutionary Computation Techniques

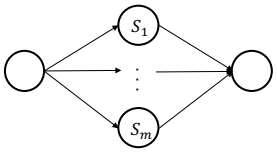
Evolution Computing (EC) techniques are founded based on the principles of Darwin natural selection. The nature evolution and selection of individuals in a population are automated simulated in EC techniques. In particular, a population of individuals is initialized for directly or indirectly presenting the solutions. Those individual candidates are evolved and evaluated using a fitness function to evaluate the degree of how good (or bad) of each individual. Therefore, it is possible to reach solution with a near-optimal fitness value. EC techniques have been shown their promise in solving combinatorial optimization problems [5]. This is due to their flexibility in encoding the problems for the representation and their good performances in many problem domains. In particular, they have effective methods to manage the constraints in the combinatorial optimization problems, i.e., coding, penalty functions, repair algorithms, indirect methods of representation and multi-objective optimization [35]. In the context of service composition. Many EC techniques, e.g., Genetic Algorithms (GA) [109], Genetic Programming (GP) [50], Particle Swarm Optimization (PSO) [45], and Clonal Section Algorithm [29], have been utilized for handling optimization problems for Web service composition. These techniques are briefly introduced here.

GP is considered as a particular application of GA with a set of different encoded genes. In particular, the representation of GA is commonly represented as a linear structure. However, In GP, each individual is commonly represented as a tree structure, which has a terminal set and a function set. In addition, the tree structure is considered be efficiently evaluated recursively. Three genetic operators consisting of reproduction, crossover, and mutation are involved in to generate next generation for both GA and GP. Reproduction operator retains



$$T = \sum_{n=1}^{m} t_n \qquad C = \sum_{n=1}^{m} c_n \qquad A = \prod_{n=1}^{m} a_n \qquad R = \prod_{n=1}^{m} r_n$$

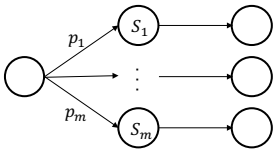
Figure 2.5: Sequence construct and calculation of its QoS [121].



$$T = MAX\{t_n | n \in \{1, \ldots, m\}\}$$

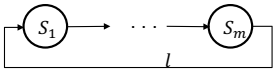
$$C = \sum_{n=1}^{m} c_n \qquad A = \prod_{n=1}^{m} a_n \qquad R = \prod_{n=1}^{m} r_n$$

Figure 2.6: Parallel construct and calculation of its QoS [121].



$$T = \sum_{n=1}^{m} p_n \cdot t_n \qquad C = \sum_{n=1}^{m} p_n \cdot c_n \qquad A = \prod_{n=1}^{m} p_n \cdot a_n \qquad R = \prod_{n=1}^{m} p_n \cdot r_n$$

Figure 2.7: Choice construct and calculation of its QoS [121].



$$T = \ell \cdot t \qquad C = \ell \cdot c \qquad A = a^{\ell} \qquad R = r^{\ell}$$

Figure 2.8: Loop construct and calculation of its QoS [121].

the elite individuals without any changes. Crossover operator replaces one node of one individual with another node of another individual. Mutation operator replaces a randomly selected node in an individual. The whole evaluation process won't stop unless an optimal solution found or a pre-defined number of generation reached.

PSO is one of swarm intelligence (SI) techniques that is based on the behavior of decentralized, self organized system. PSO algorithm is initialized by a group of random particles, which direct or indirect present the solutions. Those particles explores for the optimization position, which is approached by repeating the process of transferring particles position according to both their own best-known position and global best position.

Artificial immune system (AIS) has been studied for performing machine learning, pattern recognition and solving optimization problems. Clonal Section Algorithm (CSA) is one of AIS for handling optimization problems, and the principles of CSA lie in the features of immune memory, affinity maturation. In particular, the antigen is considered as a fitness function instead of the explicit antigen population. A proportion of antibody, rather than the best affinity antibody, are chosen to proliferation. Further more, speed and accuracy of the immune responses grow higher and higher after each infection even confronting cross-reactive response. Apart from that, hypermutation and receptor editing contribute to

avoiding local optimization and selecting optimized solution respectively.

2.2 Related Work

2.2.1 Single-Objective Web Service Composition Approaches

In this section, approaches to QoS-aware Web service composition will be discussed in two distinct groups: EC-based approaches, which mainly rely on the EC techniques to identify good solutions, and non-EC based approaches, which do not utilize any bio-inspired methods. However, most of the existing works treat service composition problems as a single-objective optimization problems, where a united QoS score is computed using a simple additive weighting (SAW) technique [42].

EC-based Composition Approaches

EC-based Web service composition mainly relies on evolutionary computation algorithms for searching optimal solutions. These algorithms are inspired by the behavior of human, animals or even T-cells. To use different EC algorithms, proper representations need to be designed. These representations can directly or indirectly represent the service composition solutions. Herein we mainly discuss some promising research works on QoS-aware Web service composition using Genetic Algorithm (GA), Genetic Programming (GP), Particle Swarm optimization (PSO), and Clonal Selection Algorithm (CSA).

Genetic Algorithm. GA is a very reliable and powerful technique for solving combinatorial optimization problems [95]. It has been applied to solve optimization problems for QoS-aware Web service composition [105]. [16] develop a GA-based approach for semi-automated QoS-aware service composition, where an abstract workflow is given. In their work, the proposed GA method are compared to a LIP-based method. The experimental finding reveals GA method is preferred when the size of service candidates increases exponentially. [96] propose a hybrid approach utilizing GA and local search. In particular, a local optimizer is developed and only recalled in the initial population for improving QoS value. This local search contributes to a better overall performance compared to the one without utilizing local search. In [54], a semi-automated service composition approach is developed for optimizing the quality of semantic matchmaking and some quality criteria of QoS. In particular, the quality of matchmaking is transferred to measure the quality of semantic links, which is measured by two quality aspects: matchmaking type and degree of similarity.

Genetic Programming. Tree-based representations could be more ideal for practical use, since they can present all composition constructs as inner nodes of trees. GP technique is utilized for handling tree-based representations. [88] relies on GP utilizing a context-free grammar for population initialization, and uses a fitness function to penalize invalid individuals throughout evolutionary process. This method is considered to be less efficient as it represents a low rate of fitness convergence. To overcome the disadvantages of [88], [121] proposes a GP-based approach employing the standard GP to bypass the low rate of convergence and premature convergence. The idea of this paper is to increase the mutation rate while encountering low diversity in the population, and to adopt a higher crossover probability while trapped in local optimization. During the evolutionary process, the elitism strategy is adopted, in which the best individual produced is reproduced to next generation directly without crossover and mutation. [64] proposes a hybrid approach combining GP and a greedy algorithm. In particular, a set of DAGs that represent valid solutions are initialized by a random greedy search and transferred into trees using the graph unfolding

technique. In each individual, terminal nodes are considered as task inputs, root node as outputs, and all the inner nodes as atomic Web services. During the reproduction process, a randomly selected node on one individual is replaced with a new subtree generated by a greedy search to perform mutation while same atomic inner nodes in two random chosen individuals are swapped to perform crossover. However, [24] proposes a different transformation algorithm to present composition constructs as the functional nodes of trees. On the whole, all these GP-based approaches [64, 88, 24, 121] consistently ignore the semantic matchmaking quality, and their representations do not preserve semantic matchmaking information and composition constructs simultaneously.

Graph-Based Genetic Programming. A graph-evolutionary approach is introduced in [22] with graph-based genetic operators, which are utilized to evolve individuals represented by graph-based representations. Although graph-based representations are capable of presenting all the matchmaking relationships as edges, they hardly present some composition constructs (e.g., loop and choice). [25] investigate Directed Acyclic Graph with branches using GraphEvol approach [22] to find near-optimal QoS for Web service composition. This approach is compared to the GP-based approach in [23]. The experiment results reveal a significant improvement in execution time with a slightly tradeoff in the fitness value. However, the branches handling in their approach has a big limitation. That is, any nested choice composition construct is not supported.

Particle Swarm optimization. PSO is considered to be a simple and effective approach for solving combinatorial optimization problems with few parameters settings [62]. Liu et al. [60] propose a hybrid Genetic Particle Swarm optimization Algorithm (GPSA). In particular, GA is employed with only crossover operator to produce new individuals with n_1 iterations while PSO is only utilized for local searching (i.e., C_2 parameter is set to 0 in the standard velocity updating functional) with n_2 iterations. This approach achieves a good balance of global and local optimization through a mechanism based on two thresholds, which determine the values of n_1 and n_2. However, this work [60] only handles semi-automated service composition problems. On the other hand, [27] proposes a PSO-based fully automated approach to generate a composition solution from an optimized service queue. The idea is to translate the particle location into a service queue as an indirect representation of a composition solution, so finding the best composite solution is to discover the optimized location of the particle in the search space. However, only QoS is optimized in their approach.

Clonal Selection Algorithm. Yan et al. [113] introduce a novel Web service composition approach using an immune algorithm for QoS optimization with a constraint on cost. As a given abstract graph could be broken into several single pipelines, the optimization problem is transferred into getting an optimal executing plan for each single pipeline. In a pipeline, each abstract task could be slotted with several alternative Web services with QoS values labelled to their edges so that a weighted multistage graph is established for optimal paths selection. In the immune algorithm, the service composition problem is encoded using a binary string as an antibody for evaluating the affinity value of the antigen (i.e.,fitness function), and the antibody with low concentration will be selected for crossover and mutation. However, the efficiency of creating the weighted multistage graph is to very low. Pop et al. [82] introduce an immune-inspired Web service composition approach combining an enhancing planning graph (EPG) and a clonal selection algorithm to solve optimization problem considering both semantic link quality and QoS. The EPG model is characterized with actions and layers involved in multiple stages, where each action represents clustered Web services, and each layer represents input s or outputs grouped in concepts. During the clonal selection process, the antigen is represented as a fitness function, and the antibody is represented as a binary alphabet to encode EPG. The remaining steps are standard computation procedure for CLONALG, which is a repeated procedure consisting of antibody

selection, affinity maturation process, low-affinity antibody replacement.

AI Based Approaches

AI Planning techniques have been widely employed for service composition [65, 80]. The main idea of these techniques considers services as actions that are defined with functional properties (i.e., inputs, outputs, preconditions and effects) using classic planning algorithms.

Various AI planning approaches [33, 41, 84, 102, 106] have been presented to solve semantic Web service composition problems using the Graphplan algorithm [9]. [106] employs the Graphplan to secure the correctness of overall functionality, which enables atomic Web services to be concretely selected and accurately matched for achieving desired functionality. In addition, conditional branch structure is also handled. The pitfalls of this approach are procuring only linear sequences of actions, and it is hard to deal with QoS optimization. In paper [33], service-dependent QoS is modeled for QoS-aware Web service composition. This dependent QoS model is formed in three cases: a default QoS attribute, a partially dependent QoS attribute, and a completely dependent QoS attribute. They are used for the dependency checking base on a backwards Graph building with a breadth-first strategy. However, computation of service dependencies is very intensive for initialization and updating. Some approaches [55, 94] rely some frameworks supported by particular agent programming languages (e.g., Golog [94] and SHOP2 [93]) to composite Web services. In [94], a service composition framework supported by Golog is used to describe the properties of services and users' preferences. Therefore, Golog can effectively perform a search to reach a terminating situation as a service composition solution. As summarized here, AI planning techniques are considered to be less efficient, and not capable of dealing with optimization solutions for service composition (e.g., generate either optimal QoS or number of services) independently. In addition, they may suffer scalability issues when large service repositories are given.

Other Approaches

The non-EC based approaches do not rely on bio-inspired approaches. They target the optimized service composition solutions by some other methods, such as, integer programming, exhaustive search, local search and so on.

Integer Linear Programming (ILP). ILP methods are utilized for achieving Web service composition. Generally, an ILP model is created with three inputs: a set of decision variables, an objective function and a set of constraints. The outputs are values of maximized/minimized objective function and decision variables. ILP is flexible for handling QoS constraints and optimizing problems for QoS-aware service composition. Gao et al. [36] define a zero-one IP model for Web service composition based on an abstract service workflow, where services with same functionality but different QoS are classified into a same class. Yoo et al. [117] formulate Web service composition problem based on the model introduced in [36]. They simultaneously take both QoS and constraints on QoS into account. However, due to an increase in the number of decision variables, ILP may lead to exponentially increased complexity and cost in computation [59]. The resulted huge delay is not allowed in the real world scenarios. In addition, if non-linear function is utilized in ILP, the scalability is also a big problem.

Dynamic Programming Approach. Dynamic programming is an effective method for solving problems that can be broken into subproblems, which present many repetitions and substructures. In [41], an efficient pruning approach is developed combing three techniques

— a forward filtering algorithm for searching task-related services, a modified dynamic programming approach for dealing a subproblem of service composition (i.e., a problem on satisfaction of each concept pool of every graph layer), and a backward-search method for searching optimal composition solutions. Xu et al. [111] forge a problem on solving large-scale service composition efficiently with QoS guarantee, where a dynamic programming algorithm named QDA is developed to optimize every subproblem (i.e., service paths). In particular, best-known QoS are recorded and updated for all added Web services. To derive an execution plan, a depth-first search is utilized to traceback paths with maximum QoS. However, the global best QoS cannot be reached due to a trade-off in the efficiency at an expense of optimal.

ER Model-Based Approach (ILP). Most of the small and medium business rely on ER database to process information and data. Xu et al. [112] employ ER model to construct domain ontology and semantic Web services. Therefore, semantic Web service composition problem is transferred to reason composite service based on a link path between entities in the ER model. However, loop and switch constructs cannot be effectively handled in their approach, and they do not deal with optimization problem at all.

2.2.2 Multi-objective and Many-objective Web Service Composition Approaches

Maximizing or minimizing a single objective function is a most commonly used way to handle optimizing problems in automated Web service composition. That is a Simple Additive Weighting (SAW) [42] technique, which presents a utility function for all the individual quality criteria as a whole. This technique optimizes and ranks each Web service composition using a single value for each solution. However, the limitation of this technique lies in not handing the conflicting quality criteria. Those conflicting quality criteria are always presented as trade-offs. To overcome this limitation, a set of objectives corresponding to different independent quality criteria are optimized simultaneously. Consequently, a set of promising solutions that presents many quality criteria trade-offs are returned.

Multi-objective approaches

Many multi-objective techniques [61, 124, 118, 115, 110, 18] have been investigated to extensively study QoS-aware Web service composition problems. A set of optimized solutions is ranked based on a set of independent objectives, i.e., different QoS attributes. In particular, solutions are compared according to their relationship for domination. That is, figure out solutions that clearly dominate the others. For example, given two service composition solutions that are compared based on execution cost c and execution time t, solution one, $wsc_1(c = 10, t = 1)$ and solution two, $wsc_2(c = 13, t = 1)$. In our context, wsc_1 dominate wsc_2 as wsc_1 has a same execution time, but a lower execution cost. If given $wsc_3(c = 10, t = 2)$, wsc_2 is a *non-dominant* solution in the relation to wsc_3 because of its longer execution time and cheaper execution cost. Therefore, non-dominant solutions are globally produced among both the dominant and non-dominant solutions, i.e., they do not dominate themselves. These solutions are called a *Pareto Front*, which provide a set of non-dominant solutions for users to choose.

Multi-objective techniques with GA. Many approaches to multi-objective Web service composition employs GA [61]. Liu et al. [61] employ a service composition model, called MCOOP (i.e., multi-constraint and multi-objective optimal path) as Web service composition solutions with only sequence composition construct supported. In the model, different paths are selected from a service composition graph that includes N service group. In each group, services present same functionality with different QoS. They developed a

strategy GODSS (Global Optimization of Dynamic Web Service Selection) based on multi-objective Genetic Algorithm, where two points crossover and mutation are applied to their approach to speed up the astringency of the algorithm. Wada et al. [100] investigate a semi-automated approach to SLA-aware Web service composition. Each vector-based representation presents three service composition solutions for three group users. The individuals are randomly initialized, and further evaluated and optimized based on objectives from all the possible combinations of throughput, latency, cost and user category. Two multi-objective genetic algorithms are developed in the work: E-MOGA and Extreme-E. E-MOGA is proposed to search a set of solutions that are equally distributed in the searching space using a fitness function, where a production of domination value, Manhattan distance to the worst point and sparsity is assigned to feasible individuals as fitness values, and $SLA\ violation/domination\ value$ is assigned to infeasible solutions. On the other hand, Extrem-E produces extreme solutions using a fitness function, where weights use a term 1/exp(p-1), where p is the number of objectives and is assigned to the p^{th} objective.

Multi-objective techniques with PSO. Yin et al. [115] combine genetic operators and particle swarm optimization algorithm together to tackle the multi-objective SLA-aware Web service composition problems. A discrete PSO-based method is proposed for considering different scare of cases. In particular, the updates of particle's velocity and position are achieved by the crossover operator, where both velocity and position of new individual are updated in accordance with positions of *pbest*, *gbest*, and current velocity. On the other hand, mutation strategy is introduced to increase the diversity of particle, and it is performed on the *gbest* particle, if the proposed swarm diversity indicator is below a certain value.

Multi-objective techniques with ACO. Generally, ACO simulates foraging behaviors of a group of ants for optimizing the traversed foraging path, where the strength of pheromones is taken account for. Zhang et al. [124] turn the service composition problem into path selection problem, where a given abstract workflow is given with different service candidate sets. This work employs a different strategy of "divide and conquer" for decomposing a given workflow. That is, two or more abstract execution paths are decomposed from the workflow with no overlapped abstract services. This decomposing strategy results in a much smaller length of the execution paths compared to the decomposition method in [120]. Also, a new ACO algorithm is proposed to handle the multi-objective QoS-aware service composition problem. In particular, the phenomenon is presented as a k-tuple for k objectives, rather than a single value. Apart from that, a different phenomenon updating rule is proposed by considering an employment of a proposed utility function as a global criterion. Wang et al. [104] introduce nonfunctional attributes of Web services to include trust degree according to the execution log. A novel adaptive ant colony optimization algorithm is proposed to overcome the slow convergence presented from the traditional ant colony optimization algorithm. In particular, the pheromone strength coefficient is adjusted dynamically to control both the updating and evaporation of pheromone.

Many-objective approaches

More than three objectives in multi-objective problems (MOPs) are often considered as many-objective problems. Ishibuchi et al. [43] present an analysis of the multi-objective algorithm for handling optimization problems with more than 3 objectives. However, they address that the searching ability is deteriorating while the number of objectives increase exponentially. This is due to a very large size of non-dominated solutions, which make it harder to move solutions towards the Pareto Front.

De et al. [28] employ NSGA-II to deal with five different quality criteria (i.e., runtime, price, reputation, availability and reliability) for semi-automated Web service composition

problem. To decrease the deterioration, two preference relations proposed by [8] are applied to NSGA-II: Maximum Ranking (MR) and Average Ranking methods (AR). In particular, MR is the best of all the ranking scores from all the objectives, and AR is a sum of all the ranking scores from all the objectives. Therefore, three algorithms (NSGA-II, NSGA-II with MR and NSGA-II with AR) are compared for studying the five different performance metrics (i.e., hypervolume [126], Generational Distance [98], Spread and Coverage [127], and pseudo Pareto front (i.e., a combination of all non-dominated solutions)). The experiment shows NSGA-II with AR outperforms others in both GD and Spread (i.e., more balanced solutions). However, only a certain region of Pareto Front is generated by NSGA-II, rather than a wider distribution one. NSGA-II with MR performs intermediately compared to the other two algorithms. On the whole, this work firstly takes two preference relations into account for solving many-objective service composition problem, and contribute to finding better solutions with many performance metrics. However, this work is only approached in a semi-automated way.

Preferences Articulation Techniques for Multi-and Many-objective Approaches

Most of the EMO algorithms focus on generating evenly distributed non-dominated Pareto solutions. Often, an indispensable decision must be made for choosing a small number of solutions, which satisfy customers' preferences. However, in some practical scenarios, some issues in EMO algorithms need to be handled when no or few solutions are gained in the preferred areas. To solve these issues, some user preferences-based approaches have been proposed to search only the space that is preferred by users, and to increases the density of solutions in that space. These approaches can be classified into three groups based on the articulation of preferences [97]. The first group is prior approaches, where the articulation is done before optimization process; the second group is interactive approaches, where the articulation is done during optimization process; the third group is posteriori approaches, where the articulation is done after optimization process. As argued in [38], the third approaches are capable of generating more desired solutions in the preferred region.

Most of the multi-objective service composition assumes that the user preferences are not known in advance. However, often, a vague preference is provided by users at least, according to which preferred solutions are roughly specified. These vague preferences should be taken into decision making and guide us to search for the most interesting areas on the Pareto front. Branke et al. [14] proposed two effective and efficient methods for finding the most relevant parts of Pareto solutions according to users' preferences. In particular, vague trade-offs between different objectives are taken into account when users have some rough thoughts about trade-offs. The first user articulation method is based on a guided multi-objective evolutionary algorithm in [15], and the second user articulation method is approached by a new biased crowding operator. In the first method, the dominance is specified for a maximally acceptable trade-off for each pair of objectives, which is appropriately transferred into two auxiliary objectives. The second method works on finding a biased distribution on the Pareto solutions. The idea of this method is to introduce a biased crowding distance, which controls the expansion of solutions according to their location projected on a proposed hyper plane, i.e., users' specified direction vector. The experiment shows that both methods can effectively and efficiently find relevant Pareto solutions for users. In [19], a posteriori articulation method is proposed to find Pareto solutions in the preferred areas using reference vectors, which are created based on the observation of given Pareto solutions. The reference vectors divide the objective space into subspaces, where one individual is selected from each subspace and put into the next generation. Therefore, the newly generated individuals move towards optimal Pareto Front and reference vectors. The experiments show

that this method could handle preferences effectively and efficiently. On the whole, none of existing preference articulation techniques focus on Web service composition problems,

let alone applicability for comprehensive quality-aware semantic Web service composition.

2.2.3 Dynamic Web Service Composition Approaches

All the previously discussed approaches can be classified into one group that assumes the composition environment is static. The rest approaches can be classified into another group that does not make a closed world assumption. Instead, a real world scenario takes a dynamic composition environment into account. For example, nonfunctional properties of Web services may fluctuate over time or services are failed/newly registered. To address this problem [72], a suitable mechanism for effectively and efficiently handle this problem raise a significant challenge.

Dynamic Web Service Composition Approaches for Handling Changes in QoS

Service selection is one of the crucial steps when we compose services. In the context of static Web service composition, we always select services based on the QoS advertised by service providers. However, QoS is fluctuating over the time according to the instances of a service at the run time. This dynamic QoS is formally modeled as an uncertain QoS model. Based on this model, some studies [108] have been addressed recently. To efficient provide a relatively small set of services for selection based on the uncertain QoS model, the data structure of R-tree is introduced for spatial query on multidimensional data (i.e., many dimension of QoS attributes) since it can significantly reduce the searching space. This space index technique is one of the key contributions in the paper for efficiently storing and retrieving services for service selection.

Reinforcement learning (RL) is one technique of machine learning for solving sequential decision-making problems to maximize some long-term rewards. RL is utilized to deal with how actions are taken in an uncertain environment. In our context, this uncertain environment is related to QoS. Mostafa et al. [71] combine multi-objective optimization and reinforcement techniques to solve multi-objective service composition problem in an uncertain and dynamic environment. In particular, we service composition is modeled based on Partially Observable Markov Decision Process (POMDP), and solutions to services composition are considered to be a set of decision policies. Each decision policy is considered as a procedure of service selection (i.e., a single workflow). They proposed a method to learn an optimal selection policy to reach optimal solutions.

Nasridinov et al. [72] propose a QoS-aware performance prediction for a self-healing Web service composition system. This system consists of three main phases: monitoring, diagnosis and repair. The monitoring phase is to detect degradation, diagnosis is to identify the source of degradation, and repair is to re-select desired services. To minimize the number of re-selection in the phase of repair, decision tree learning is used to the prediction of the performance based on the QoS. This technique outperforms other classification techniques (i.e., back propagation neural network, support vector machine probabilistic neural network, and group method of data handling and regression tree [70].

Dynamic Web Service Composition Approaches for Handling Unexpected Service Failure

Traditionally, re-selection of failed service is one of widely used approaches to re-ensure the service composition plan. The idea of this traditional approach is to restore many alterna-

tive service candidates for each component service involved in the composition solution. If a component service confronts a failure, the alternatives are used for the replacement. A framework, WS-Replication is introduced in [90], which mainly address service failure using the idea of the traditional method. However, those approaches [90] suffer a huge cost in computation. To overcome the disadvantage of traditional resection approach, [101] proposes a method to cluster services as a back up for service failures. This method determines a set of backup services based on their functional properties as different clusters. In particular, a set of backup services is available in both the clusters in and the sub-clusters, from which we can effectively select a suitable service.

As summarized here, existing approaches handle either changes in QoS or service failures. None of these approach handles services newly registered and changes in the ontology of services. Technically, all these approaches do not allow the changes of composition structure, and they mainly focus on efficient re-section techniques for desired services.

2.2.4 Web Service Composition Approaches Supporting Preconditions and Effects

Apart from the consideration of satisfactions on inputs and outputs of Web services, more complex chaining strategies for semantic services are proposed to handle preconditions and effects. To cope with these complicated requirements, preconditions and effects are well formulated by ontology-based techniques (e.g., DL) for supporting semantic matchmaking of semantic Web services [106].

Most of the existing service composition is approached merely based on input and outputs, but preconditions and effects are not considered. Therefore, it is hard to generate service composition solutions with some composition constructs (e.g., choice and loop). A generalized semantic Web service composition is introduced in [6], preconditions and effects are effectively presented in a conditional directed acyclic graph where conditional nodes created with two outgoing edges representing a satisfied and a unsatisfied case at the run time. They filter the solutions based on a trust rate using Centrality Measure of Social Networks to find trusted Web services. They also implement a semantic Web service composition engine for automated conditional service composition. In [107], an extensive Graphplan technique is proposed to support preconditions and effects in a direct representation. In particular, a two-level directed graph, called planning graph (PG) is utilized and comprises of two kinds of nodes (i.e., proposition and action ones) and three kinds of edges (i.e., precondition-edges, add-edges, and delete-edges). The proposition and action level is alternated between each other in PG. By utilizing this presentation, the branch structure is supported in the service composition solution. On the whole, these two approaches do not support loop construct, and leave optimization problems aside.

2.2.5 Summary and Limitations

An overview of recent related works on Web service composition is presented in this chapter. The first related research area is **single-objective Web service composition**, which mainly focuses on generating composition solutions with optimal QoS or number of services based on pre-defined objective functions. On the one hand, EC-based approaches have been widely used for effectively and efficiently solving QoS-aware service composition. On the other hand, non-EC based approaches, such as AI planning approaches, Integer Linear Programming, Dynamic Programming Approaches are employed in Web service composition, but they may suffer scalability due to an increase in the size of searching space or leave

combinatorial optimization for service composition aside. None of existing approaches simultaneously optimize QoS and quality of semantic matchmaking in a fully automated way.

The second related research area is **EMO-based service composition**, in which different quality criteria of QoS are simultaneously optimized and a high quality Pareto Front is produced. Existing works on EMO-based approaches focus on QoS-aware semantic service composition in a semi-automated way. Sometimes, SLA constraints are considered. However, none of these approaches take QoS and quality of semantic matchmaking simultaneously in a fully automated way. Apart from that, none of existing preference articulation techniques are investigated in the domain of Web service composition.

Dynamic Web service composition aims at effectively handling a dynamic service composition environment. On the one hand, existing dynamic service composition mainly works on various techniques to efficiently handling changes in QoS and service failures, such as R-tree, decision tree learning and RL. These approaches do not allow the changes of service composition structure. Apart from that, the cost of initial planning is ignored and separated from an adaption of dynamic environment. On the other hand, EC-based approaches have not been utilized in dynamic service composition so far and their natures allow the changes of composition structure and avoid re-planning cost through re-using known plans (composite services) obtained prior.

The last related research area focuses on **semantic Web service composition**, which explores a more complex service composition. Most approaches on Web service composition support precondition and effects. However, loop composition construct is not handled and combinatorial optimization problems is left alone. On the whole, none of existing works achieve web service composition supporting precondition and effects for all the composition constructs while simultaneously optimizing QoS and quality of semantic matchmaking.

Chapter 3

Preliminary Work

This chapter exhibits the two initial works for comprehensive quality-aware semantic Web service composition approach, which combines a hybridization of various techniques for optimising the quality of semantic matchmaking and quality of service. In particular, One direct representation and another indirect representation are proposed in two premilitary works using a PSO-based approach and a GP-based approach respectively. Apart from that, composition solutions are represented in a DAG or a tree-like representation, respectively, in the two approaches. These two works are discussed in the following sections.

3.1 Problem Formalisation

We consider a *semantic Web service* (*service*, for short) as a tuple $S = (I_S, O_S, QoS_S)$ where I_S is a set of service inputs that are consumed by S, O_S is a set of service outputs that are produced by S, and $QoS_S = \{t_S, c_S, r_S, a_S\}$ is a set of non-functional attributes of S. The inputs in I_S and outputs in O_S are parameters modelled through concepts in a domain-specific ontology $\mathcal{O}$. The attributes t_S, c_S, r_S, a_S refer to the response time, cost, reliability, and availability of service S, respectively. These four QoS attributes are most commonly used [122].

A *service repository* $\mathcal{SR}$ is a finite collection of services supported by a common ontology $\mathcal{O}$. A *service request* (also called *composition task*) over $\mathcal{SR}$ is a tuple $T = (I_T, O_T)$ where I_T is a set of task inputs, and O_T is a set of task outputs. The inputs in I_T and outputs in O_T are parameters described by concepts in the ontology $\mathcal{O}$.

Matchmaking types are often used to describe the level of a match between outputs and inputs [76]: For concepts a, b in $\mathcal{O}$ the *matchmaking* returns *exact* if a and b are equivalent ($a \equiv b$), $plugin$ if a is a sub-concept of b ($a \sqsubseteq b$), *subsume* if a is a super-concept of b ($a \sqsupseteq b$), and $fail$ if none of previous matchmaking types is returned. In this paper we are only interested in robust compositions where only *exact* and $plugin$ matches are considered, see [54]. As argued in [54] $plugin$ matches are less preferable than *exact* matches due to the overheads associated with data processing. We suggest to consider the semantic similarity of concepts when comparing different $plugin$ matches.

Robust causal link [56] is a link between two matched services S and S', noted as $S \rightarrow S'$, if an output a ($a \in O_S$) of S serves as the input b ($b \in O_{S'}$) of S' satisfying either $a \equiv b$ or $a \sqsubseteq b$. For concepts a, b in $\mathcal{O}$ the *semantic similarity* $sim(a, b)$ is calculated based on the edge counting method in a taxonomy like WorldNet or Ontology [91]. This method has the advantages of simple calculation and good performance [91]. Therefore, the *matchmaking*

type and *semantic similarity* of a robust causal link can be defined as follow:

$$type_{link} = \begin{cases} 1 & \text{if } a \equiv b \ (exact \text{ match}) \\ p & \text{if } a \sqsubseteq b \ (plugin \text{ match}) \end{cases}, \qquad sim_{link} = sim(a,b) = \frac{2N_c}{N_a + N_b} \tag{3.1}$$

with a suitable parameter $p, 0 < p < 1$, and with N_a, N_b and N_c, which measure the distances from concept a, concept b, and the closest common ancestor c of a and b to the top concept of the ontology $\mathcal{O}$, respectively. However, if more than one pair of matched output and input exist from service S to service S', $type_{link}$ and sim_{link} will take on their average values.

The *semantic matchmaking quality* of the service composition can be obtained by aggregating over all robust causal links as follow:

$$MT = \prod_{j=1}^{m} type_{link_j}, \qquad SIM = \frac{1}{m} \sum_{j=1}^{m} sim_{link_j} \tag{3.2}$$

We consider two special atomic services $Start = (\emptyset, I_T, \emptyset)$ and $End = (O_T, \emptyset, \emptyset)$ to account for the input and output requirements given by the composition task T, and add them to $\mathcal{SR}$.

Given a service request $T = (I_T, O_T)$, we represent a service composition solution for T with services $S_1, \ldots, S_n$ by a weighted DAG, $\mathcal{G} = (V, E)$ with node set $V = \{Start, S_1, S_2, \ldots, S_n, End\}$ and edge set $E = \{link_1, link_2, ...link_m\}$. Each edge $link$ is a *robust causal link*.

We also use formal expressions as in [63] to represent service compositions. We use the constructors $\bullet$, $\|$, $+$ and $*$ to denote sequential composition, parallel composition, choice, and iteration, respectively. The set of *composite service expressions* is the smallest collection $\mathcal{SC}$ that contains all atomic services and that is closed under sequential composition, parallel composition, choice, and iteration. That is, whenever $C_0, C_1, \ldots, C_d$ are in $\mathcal{SC}$ then $\bullet(C_1, \ldots, C_d)$, $\| \ (C_1, \ldots, C_d)$, $+(C_1, \ldots, C_d)$, and $*C_0$ are in $\mathcal{SC}$, too. Let C be a composite service expression. If C denotes an atomic service S then its QoS is given by QoS_S. Otherwise the QoS for C can be obtained inductively as summarized in Table 3.1. Herein, $p_1, \ldots, p_d$ with $\sum_{k=1}^{d} p_k = 1$ denote the probabilities of the different options of the choice $+$, while ℓ denotes the average number of iterations.

Table 3.1: QoS calculation for a composite service expression C

$C =$	$r_C =$	$a_C =$	$c_C =$	$t_C =$
$\bullet(C_1, \ldots, C_d)$	$\prod_{k=1}^{d} r_{C_k}$	$\prod_{k=1}^{d} a_{C_k}$	$\sum_{k=1}^{d} c_{C_k}$	$\sum_{k=1}^{d} t_{C_k}$
$\| \ (C_1, \ldots, C_d)$	$\prod_{k=1}^{d} r_{C_k}$	$\prod_{k=1}^{d} a_{C_k}$	$\sum_{k=1}^{d} c_{C_k}$	$MAX\{t_{C_k} \mid k \in \{1, ..., d\}\}$
$+(C_1, \ldots, C_d)$	$\prod_{k=1}^{d} p_k \cdot r_{C_k}$	$\prod_{k=1}^{d} p_k \cdot a_{C_k}$	$\sum_{k=1}^{d} p_k \cdot c_{C_k}$	$\sum_{k=1}^{d} p_k \cdot t_{C_k}$
$*C_0$	${r_{C_0}}^{\ell}$	${a_{C_0}}^{\ell}$	$\ell \cdot c_{C_0}$	$\ell \cdot t_{C_0}$

When multiple quality criteria are involved in decision making, the fitness of a solution can be defined as a weighted sum of all individual criteria using Eq. (3.3), assuming the preference of each quality criterion is provided by users.

$$Fitness = w_1 \hat{MT} + w_2 \hat{SIM} + w_3 \hat{A} + w_4 \hat{R} + w_5(1 - \hat{T}) + w_6(1 - \hat{C}) \tag{3.3}$$

with $\sum_{k=1}^{6} w_k = 1$. We call this objective function the *comprehensive quality model* for service composition. The weights can be adjusted according to users' preferences. $\hat{MT}$, $\hat{SIM}$, $\hat{A}$, $\hat{R}$,

$\hat{T}$, and $\hat{C}$ are normalised values calculated within the range from 0 to 1 using Eq. (3.4). To simplify the presentation we also use the notation $(Q_1, Q_2, Q_3, Q_4, Q_5, Q_6) = (MT, SIM, A, R, T, C)$. Q_1 and Q_2 have minimum value 0 and maximum value 1. The minimum and maximum value of Q_3, Q_4, Q_5, and Q_6 are calculated across all task-related candidates in the service repository $\mathcal{SR}$ using the greedy search in [64, 24].

$$\hat{Q}_k = \begin{cases} \frac{Q_k - Q_{k,min}}{Q_{k,max} - Q_{k,min}} & \text{if } k = 1, \ldots, 4 \text{ and } Q_{k,max} - Q_{k,min} \neq 0, \\ \frac{Q_{k,max} - Q_k}{Q_{k,max} - Q_{k,min}} & \text{if } k = 5, 6 \text{ and } Q_{k,max} - Q_{k,min} \neq 0, \\ 1 & \text{otherwise.} \end{cases} \tag{3.4}$$

To find best possible solution for a given composition task T, our goal is to maximise the objective function in Eq. (3.3).

3.2 PSO-based Approach to Comprehensive Quality-Aware Automated Semantic Web service Composition

3.2.1 An Overview of our PSO-based Approach

As PSO has shown promise in solving combinatorial optimisation problems, we propose a PSO-based approach to comprehensive quality-aware automated semantic Web service composition. Fig. 3.1 shows an overview of our approach consisting of four steps:

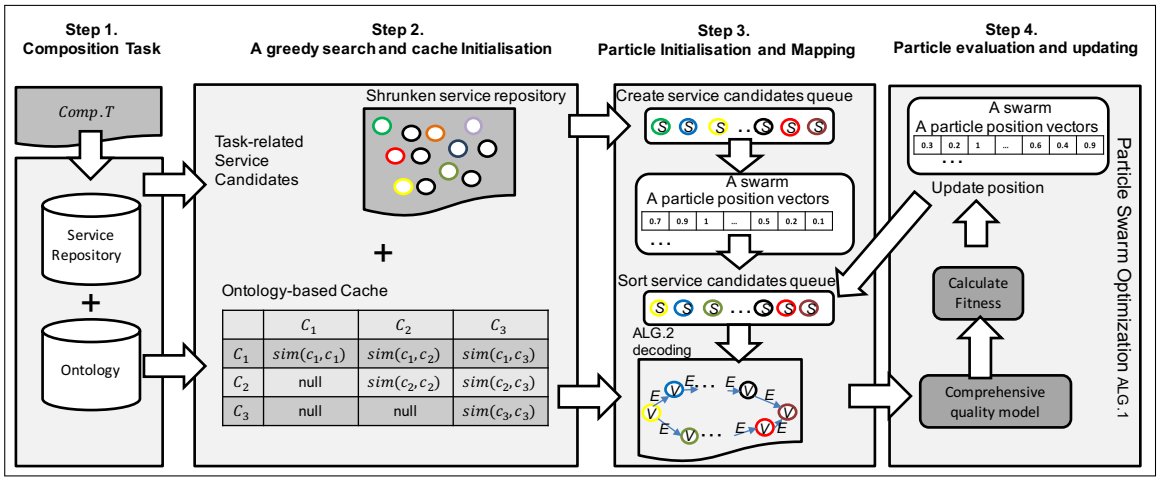


Figure 3.1: An overview of our PSO-based approach to comprehensive quality-aware automated semantic Web service composition.

Step 1: The composition process is triggered by a composition task, which is clearly defined in Section 3.1.

Step 2: The composition task is used to discover all task-related service candidates using a greedy search algorithm adopted from [64], which contributes to a shrunken service repository. This greedy search algorithm keeps adding outputs of the invoked services as available outputs (initialised with I_T) , and these available outputs are used to discover task-related services from a service repository and updated with the outputs of these discovered services. This operation is repeated until no service is satisfied by the available outputs. During the greedy search, an ontology-based cache (*cache*) is initialised, which stores the concept similarities of matched inputs and outputs of task-related candidates. This *cache* is

also used to discover services by checking whether *null* is returned by given two output-related and input-related concepts.

Step 3 and Step 4: These two steps follow the standard PSO steps [92] except for some differences in particles mapping and decoding processes. In particular, these two differences are related to sorting a created service queue using service-to-index mapping for a particle' position vectors and evaluating the fitness of a particle after decoding this service queue into a $\mathcal{G}$ respectively. Those differences are further addressed in Algorithms 1 and 2 in Section 3.2.2.

3.2.2 The Algorithms for our PSO-based Approach

The overall algorithm investigated here is made up of a PSO-based Web service composition technique (Algorithm 1) and a $\mathcal{G}$ creating technique from a service queue (Algorithm 2). In Algorithm 1, the steps 4, 5, 6 and 7 are different from those of standard PSO: In step 4, the size of task-related service candidates generated by a greedy search determines the size of each particle's position. Each service candidate in a created service candidates queue is mapped to an index of a particles position vectors, where each vector has a weight value between 0.0 and 1.0. In step 5, service candidates in the queue are sorted according to their corresponding weight values in descending order. In step 6, this sorted queue is used as one of the inputs of the forward decoding Algorithm 2 to create a $\mathcal{G}$. In step 7, the fitness value of the created $\mathcal{G}$ is the fitness value of the particle calculated by the comprehensive model discussed in Section 3.1.

ALGORITHM 1. Steps of PSO-based service composition technique [27].

```
1:  Randomly initialise each particle in the swarm;
2:  while max. iterations not met do
3:      foreach particle in the swarm do
4:          Create a service candidates queue and map service candidates to a particle's
            position vectors;
5:          Sort the service queue by position vectors' weights;
6:          Use Algorithm 2 to create a G from the service queue;
7:          Calculate the G fitness value;
8:          if fitness value better than pBest then
9:              Assign current fitness as new pBest;
10:         else
11:             Keep previous pBest;
12:     Assign best particle's pBest value to gBest, if better than gBest;
13:     Calculate the velocity of each particle;
14:     Update the position of each particle;
```

Algorithm 2 is a forward graph building algorithm extended from [9]. This algorithm takes one input, a sorted service queue from step 5 of Algorithm 1. Note that different service queues may lead to different $\mathcal{G}$s. In addition. I_T, O_T and *cache* are also taken as the inputs. Firstly, *Start* and *End* are added to V of $\mathcal{G}$ as an initialisation, and $OutputSet$ is also created with I_T. The following steps are repeated until O_T can be satisfied by $Outputset$ or the service queue is *null*. If all the inputs I_S of the first popped S from *queue* can be satisfied by provided outputs from $OutputSet$, this S is added to V and its outputs are added to $OutputSet$, and S is removed from *queue*. Otherwise, the second popped S from *queue* is considered for these operations. Meanwhile, *link* is created with $type_{link}$ and sim_{link} if S is

added, and calculated using information provided from *cache*. This forward graph building technique could lead to more services and edges connected to the $\mathcal{G}$, these redundancies should be removed before $\mathcal{G}$ is returned.

ALGORITHM 2. Create a $\mathcal{G}$ from a sorted service queue.

Input : I_T, O_T, *queue*, *cache*
Output: $\mathcal{G}$
1: $\mathcal{G} = (V, E)$;
2: $V \leftarrow \{Start, End\}$;
3: $OutputSet \leftarrow \{I_T\}$;
4: **while** O_T *not satisfied by* $OutputSet$ **do**
5: **foreach** S *in queue* **do**
6: **if** I_S *satisfied by* $OutputSet$ **then**
7: insert S into V;
8: adjoin O_S to $OutputSet$;
9: *queue*.remove S;
10: $link \leftarrow$ calculate $type_{link}$, sim_{link} using *cache*;
11: insert $link$ into E;
12: remove *dangling nodes* and *edges* from $\mathcal{G}$;
13: **return** $\mathcal{G}$;

3.3 Experiment Study for PSO-based Approach

In this section, we employ a quantitative evaluation approach with a benchmark dataset used in [64, 24], which is an augmented version of Web service Challenge 2009 (WSC09) including QoS attributes. Two objectives of this evaluation are to: (1) evaluate the effectiveness of our PSO-based approach, see comparison test in Section 3.3.1. (2) evaluate the effectiveness of our proposed comprehensive quality model to achieve a desirable balance on semantic matchmaking quality and QoS, see comparison test in Section 3.3.2.

The parameters for the PSO are chosen from the settings from [92], In particular, PSO population size is 30 with 100 generations. We run 30 times independently for each dataset. We configure the weights of fitness function to properly balance semantic matchmaking quality and QoS. Therefore, w_1 and w_2 are set equally to 0.25, and w_3, w_4, w_5, w_6 are all set to 0.125. The p of $type_{link}$ is set to 0.75 ($plugin$ match) according to [54]. In general, weight settings and parameter p are decided according to users' preferences.

3.3.1 GP-based vs. PSO-based approach

To evaluate the effectiveness of our proposed PSO-based approach, we compare our PSO-based method with one recent GP-based approach [64] using our proposed comprehensive quality model. We extend this GP-based approach by measuring the semantic matchmaking quality between parent nodes and children nodes. To make a fair comparison, we use the same number of evaluations (3000 times) for these two approach. We set the parameters of that GP-based approach as 30 individuals and 100 generations, which is considered to be proper settings referring to [23].

The first column of Table 3.2 shows five tasks from WSC09. The second and third column of Table 3.2 show the original service repository size and the shrunk service repository size after the greedy search respectively regarding the five tasks. This greedy search helps

Table 3.2: Mean fitness values for comparing GP-based approach

WSC09	Original $\mathcal{SR}$	Shrunken $\mathcal{SR}$	PSO-based approach	GP-based approach
Task 1	572	80	$0.5592 \pm 0.0128 \uparrow$	0.5207 ± 0.0208
Task 2	4129	140	$0.4701 \pm 0.0011 \uparrow$	0.4597 ± 0.0029
Task 3	8138	153	0.5504 ± 0.0128	$0.5679 \pm 0.0234 \uparrow$
Task 4	8301	330	$0.4690 \pm 0.0017 \uparrow$	0.4317 ± 0.0097
Task 5	15211	237	$0.4694 \pm 0.0008 \uparrow$	0.2452 ± 0.0369

reducing the original repository size significantly, which contributes to a reduced searching space. The fourth and fifth column of Table 3.2 show the mean fitness values of 30 independent runs accomplished by two methods. We employ independent-samples T tests to test the significant differences in mean fitness value. The results show that the PSO-based approach outperforms the existing GP-based approach in most cases except Task 3. Note that all p-values are consistently smaller than 0.01. Using our PSO-based approach, small changes to sorted queues (particles in PSO) could lead to big changes to the composition solutions. This enables the PSO-based approach to escape from local optima more easily than the GP-based approach.

3.3.2 Comprehensive Quality Model vs. QoS Model

Recently, a QoS Model, $Fitness = w_1\hat{A} + w_2\hat{R} + w_3(1-\hat{T}) + w_4(1-\hat{C})$, where $\sum_{i=1}^{4} w_i = 1$, is widely used for QoS-aware Web service composition [64, 27, 22]. To show the effectiveness of our proposed comprehensive quality model, we compare the best solutions found by this QoS model and our comprehensive model using our PSO-based approach. We record and compare the mean values of both SM ($SM = 0.5\hat{MT} + 0.5\hat{SIM}$) and QoS($QoS = 0.25\hat{A} + 0.25\hat{R} + 0.25(1-\hat{T}) + 0.25(1-\hat{C})$) of best solutions over 30 independent runs. To make the comparison informative, all these recorded values have been normalised from 0 to 1, and compared using independent-samples T tests, see Table 3.3. Note that p-values are consistently smaller than 0.001 in the results indicating significant differences in performance.

In Table 3.3, the mean values of QoS using QoS model are significantly higher than those using comprehensive quality model for Tasks 2, 3, 4 and 5. However, the mean value of SM using the comprehensive quality model are significantly higher than those using the QoS model, while a slight trade-off in QoS are observed in all tasks. In addition, our comprehensive model achieves a consistently higher comprehensive quality in terms of a combination of SM and QoS, which is significantly better in Tasks 1, 2, 3 and 4.

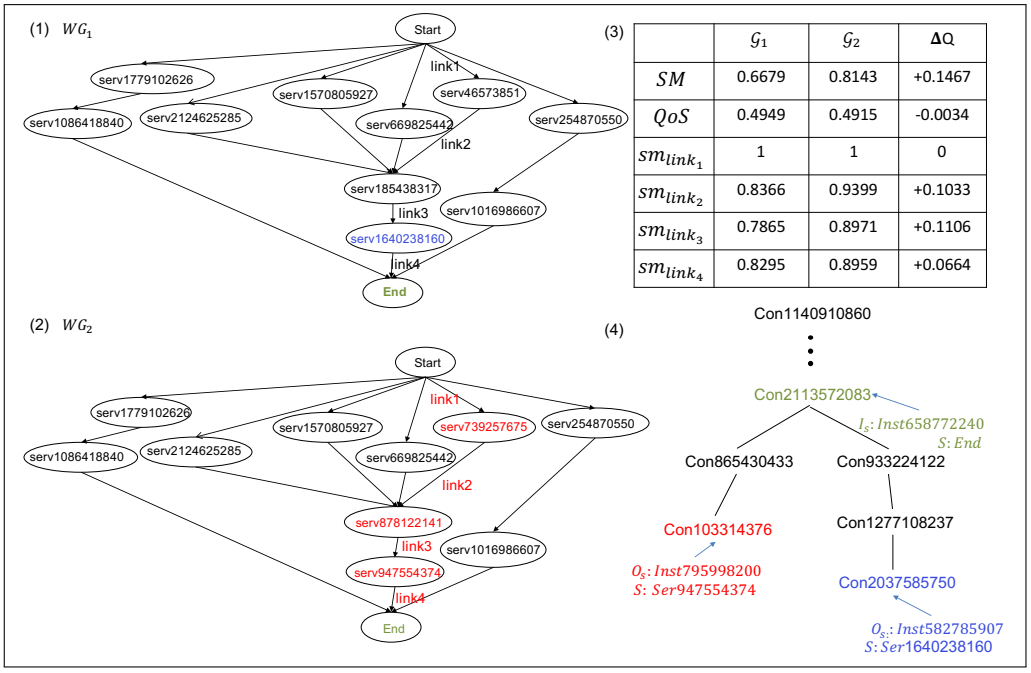
3.3.3 Further Discussion

To analyse the effectiveness of achieving a good comprehensive quality at the expense of slightly reduced QoS, we demonstrate two best solutions produced using Task 3 as an example. Fig. 3.2 (1) and (2) show two weighted DAGs, $\mathcal{G}_1$ and $\mathcal{G}_2$, which have been obtained as the best service compositions solutions based on the QoS model and on the comprehensive quality model, respectively. Both $\mathcal{G}$s have exactly the same service workflow structure, but some service vertices and edges denoted in red are different. To better understand these differences, we list the overall semantic matchmaking quality SM, overall QoS and semantic matchmaking quality sm_{link_n} associated to these different edges in $\mathcal{G}_1$ and $\mathcal{G}_2$. (Note: $sm_{link_n} = 0.5type_{link_n} + 0.5sim_{e_n}$), where ΔQ reveals the gain (positive ΔQ) or a loss (negative ΔQ) of the listed qualities for our comprehensive quality model. Therefore, we achieve

Table 3.3: Mean values of SM, QoS and sum of SM and QoS for QoS model and comprehensive quality model using PSO-based approach

WSC09		QoS Model	Comprehensive Quality Model
Task1	SM	0.5373 ± 0.0267	$0.5580 \pm 0.0094 \uparrow$
	QoS	0.5574 ± 0.0156	0.5604 ± 0.0164
	$SM + QoS$	1.0947 ± 0.0423	$1.1184 \pm 0.0258 \uparrow$
Task2	SM	0.4549 ± 0.0033	$0.4630 \pm 0.0042 \uparrow$
	QoS	$0.4800 \pm 0.0012 \uparrow$	0.4772 ± 0.0025
	$SM + QoS$	0.9349 ± 0.0045	$0.9402 \pm 0.0067 \uparrow$
Task3	SM	0.5538 ± 0.0082	$0.6093 \pm 0.0054 \uparrow$
	QoS	$0.4940 \pm 0.0013 \uparrow$	0.4913 ± 0.0009
	$SM + QoS$	1.0478 ± 0.0095	$1.1006 \pm 0.0063 \uparrow$
Task4	SM	0.4398 ± 0.0037	$0.4604 \pm 0.0000 \uparrow$
	QoS	$0.4845 \pm 0.0010 \uparrow$	0.4734 ± 0.0044
	$SM + QoS$	0.9243 ± 0.0047	$0.9338 \pm 0.0044 \uparrow$
Task5	SM	0.4580 ± 0.0065	$0.4639 \pm 0.0013 \uparrow$
	QoS	$0.4764 \pm 0.0005 \uparrow$	0.4750 ± 0.0007
	$SM + QoS$	0.9344 ± 0.0070	0.9389 ± 0.0020

a comprehensive quality gain (+0.1433), a result of a gain in semantic matchmaking quality (+0.1467) and a loss in QoS (-0.0034). To understand the improvement of semantic matchmaking quality from these numbers, we pick up $link_4$ that is associated with the smallest ΔQ. The $link_4$ of $\mathcal{G}_1$ and $\mathcal{G}_2$ has two different source nodes, $Ser1640238160$ and $Ser947554374$, and two the same End nodes. $Ser1640238160$ and $Ser947554374$ are services with output parameters $Inst582785907$ and $Inst795998200$ corresponds to two concepts $Con2037585750$ and $Con103314376$ respectively in the given ontology shown in Fig. 3.2 (4). As $Inst658772240$ is a required parameter of End, and related to concept $Con2113572083$, $Inst795998200$ is closer to the required output $Inst658772240$ than $Inst582785907$. Therefore, $Ser947554374$ is selected with a better semantic matchmaking quality compared to $Ser1640238160$.



	$\mathcal{G}_1$	$\mathcal{G}_2$	ΔQ
SM	0.6679	0.8143	+0.1467
QoS	0.4949	0.4915	-0.0034
sm_{link_1}	1	1	0
sm_{link_2}	0.8366	0.9399	+0.1033
sm_{link_3}	0.7865	0.8971	+0.1106
sm_{link_4}	0.8295	0.8959	+0.0664

Figure 3.2: An example for the comparison of the best solutions obtained based on the QoS model and on the comprehensive quality model for Task 3.

3.4 Summary for our PSO-based Approach

In PSO-based approach, we propose an effective PSO-based approach to comprehensive quality-aware semantic Web service composition, which also has shown promise in achieving a better comprehensive quality in terms of a combination of semantic matchmaking quality and QoS compared to existing works.

3.5 GP-based Approach to Comprehensive Quality-Aware Automated Semantic Web service Composition

In this section, we first introduce the tree-like representation that will be used in our approach, and then discuss the differences to the most widely used tree-based representations for GP-based service composition in the literature [40, 24, 121]. Finally, we present our GP-based approach with newly designed genetic operation methods.

3.5.1 A New Tree-like Representation for Web service Composition

Let $\mathcal{G} = (V, E)$ be a weighted DAG representation of a service composition. Let S be a service in $\mathcal{G}$, and let $S_1, \ldots, S_d$ be its successors in $\mathcal{G}$. We define the composite service expression relative to S as follows:

$$C_S = \begin{cases} \bullet(S, \| \ (C_{S_1}, \ldots, C_{S_d})), & \text{if } d \geq 2, \\ \bullet(S, C_{S_1}), & \text{if } d = 1, \\ S, & \text{if } d = 0, \end{cases} \tag{3.5}$$

which can be evaluated inductively starting with $Start$ which has no incoming edges in $\mathcal{G}$. The resulting expression C_{Start} is a composite service expression that is equivalent to $\mathcal{G}$.

Example 1. *Consider the composition task $T = (\{a, b, e\}, \{i\})$. Fig. 3.3 shows an example of a composition solution. It involves four atomic services $S_1 = (\{a, b\}, \{c, d, j\}, QoS_{S_1})$, $S_2 = (\{c\}, \{f, g\}, QoS_{S_2})$, $S_3 = (\{d\}, \{h\}, QoS_{S_3})$, and $S_4 = (\{f, g, h\}, \{i\}, QoS_{S_4})$. The two special services $Start = (\emptyset, \{a, b, e\}, \emptyset)$ and $End = (\{i\}, \emptyset, \emptyset)$ are defined by the given composition task T. The corresponding service composition expression is:*
$C_{Start} = \bullet(Start, \bullet(S_1, \| \ (\bullet(S_2, \bullet(S_4, End)), \bullet(S_3, \bullet(S_4, End)))))$.

Formal expressions can be visualized by expressions trees. For a composite service expression C let $\mathcal{T}$ denote the corresponding expression tree. Every leaf node in $\mathcal{T}$ is labelled by the corresponding atomic service, while every internal node in $\mathcal{T}$ is labelled by the corresponding composition constructor. For the sake of brevity we only consider $\bullet$ and $\|$ here, but our approach can easily be extended to $+$ and $*$, too. If a subtree of $\mathcal{T}$ (except for End) has an isomorphic copy in $\mathcal{T}$ then we remove it, label its root with a special symbol q, and insert an edge to the root of the copy. As a result we obtain a tree-like representation of a service composition. An example is shown in Fig. 3.3.

Fig. 3.3 shows for every atomic service S its sets of (least required) inputs $\mathcal{I}_S$ and outputs $\mathcal{O}_S$. Moreover, the set of available inputs $\mathcal{PI}_S$ is shown which is just the union of the input sets of all (direct and indirect) predecessors of S in the DAG. This can be easily generalized to composite service expressions. For a parallel composition $C = \| \ (C_1, \ldots, C_d)$ we define $\mathcal{I}_C = \cup_{k=1}^{d} \mathcal{I}_{C_k}$, and $\mathcal{O}_C = \cup_{k=1}^{d} \mathcal{O}_{C_k}$, and $\mathcal{PI}_C = \cup_{k=1}^{d} \mathcal{PI}_{C_k}$. For a sequential composition $C = \bullet(S, C')$ we define $\mathcal{I}_C = \mathcal{I}_S \cup (\mathcal{I}_{C'} - \mathcal{O}_S)$, and $\mathcal{O}_C = \mathcal{O}_S \cup \mathcal{O}_{C'}$, and $\mathcal{PI}_C = \mathcal{PI}_S$.

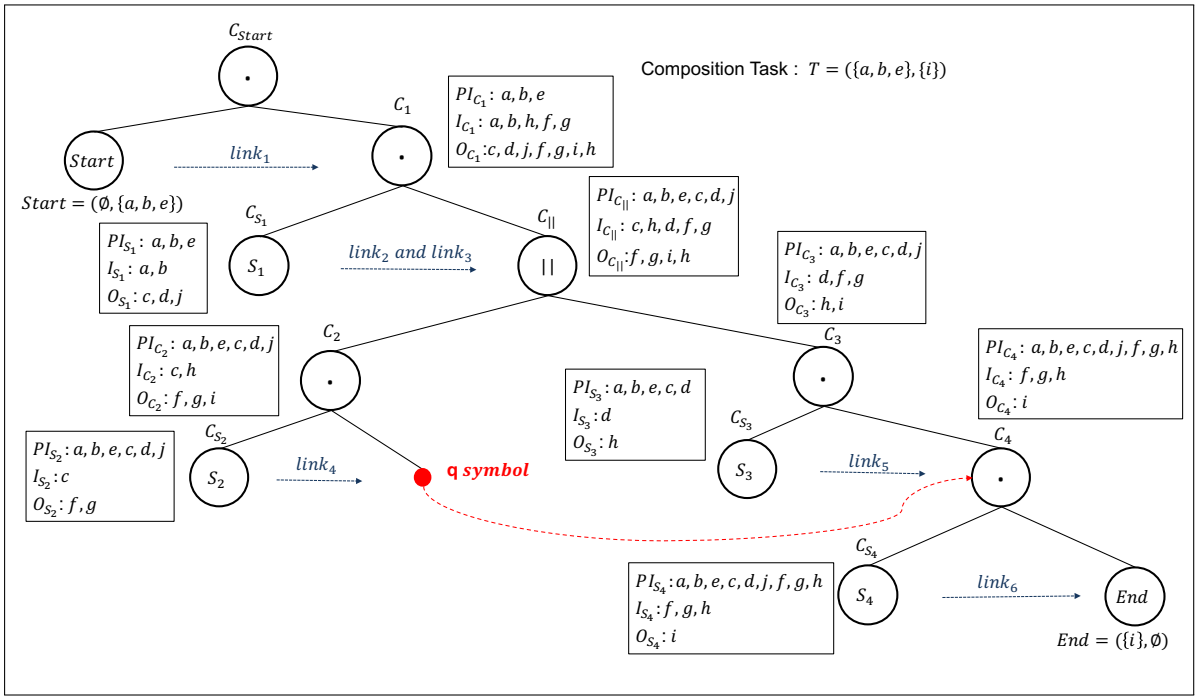


Figure 3.3: Example of a tree-like representation

Example 2. *Consider the sequential composition $C_4 = \bullet(S_4, End)$ which is shown in the rightmost position in Fig. 3.3. We obtain $\mathcal{I}_{C_4} = \{f, g, h\}$ which represents the (least required) inputs for this composition, and $\mathcal{O}_{C_4} = \{i\}$ which represents the outputs produced by this composition, and $\mathcal{PI}_{C_4} = \{a, b, e, c, d, j, f, g, h\}$ which represents the union of the input sets of all (direct or indirect) predecessors of S_4 in the DAG (i.e., S_1, S_2 and S_3).*

Our representation supports composition constructs that are available in commonly used composition languages, such as BPEL4WS or OWL-S. Note that our representation is different from the most widely used tree-based representations in [40, 24, 121]. These differences are as follows.

1. *Start* and *End* are included in $\mathcal{T}$, as they are related to measuring the semantic matchmaking qualities regarding I_T and O_T.
2. $\mathcal{I}_C$, $\mathcal{O}_C$, $\mathcal{PI}_C$, QoS_C are attributes, defined as a tuple $(\mathcal{I}_C, \mathcal{O}_C, \mathcal{PI}_C, QoS_C)$ for any C_S in $\mathcal{T}$. These attributes must be updated after population initialisation and genetic operations described in Sect. 3.5.2.
3. $\mathcal{T}$ preserves all the semantic matchmaking information, which can be easily used for computing robust casual links.

To compute semantic matchmaking quality, we need to retrieve all the robust causal links on $\mathcal{T}$. This is performed by retrieving robust causal links for every sequential composition $C = \bullet(S, C')$. For example, in Fig 3.3, two robust causal links ($link_2 : S_1 \rightarrow S_2$ and $link_3 : S_1 \rightarrow S_3$) are retrieved from $C_1 = \bullet(S_1, C_{||})$, because outputs $O_{S_1} = \{c, d, j\}$ match inputs $I_{C_{||}} = \{c, h, d, f, g\}$.

3.5.2 GP-Based Algorithm

Now we present our GP-based approach for service composition, see Algorithm 3. To begin with the algorithm, we generate the initial population P_0, which is then evaluated using our

comprehensive quality model. The iterative part of the algorithm comprises lines 3 to 7, which will be repeated until the maximum number of generations is reached or the best solution is found. During each iteration, we use tournament selection to select individuals, on which crossover and/or mutation are performed to evolve the polulation. These steps correspond to the standard GP steps [50] except for some particularities that will be discussed below.

ALGORITHM 3. GP-based algorithm for service composition.

Input : $T, \mathcal{SR}, \mathcal{O}$
Output: an optimal composition solution
1: Initialise population P_0 (using a 3-step method);
2: Evaluate each individual in population P_0 (using our comprehensive quality model);
3: **while** *max.populations or max.fitness not yet met* **do**
4: Select the fittest individuals for evolution;
5: Apply crossover and mutation to the selected individuals;
6: Evaluate each new individual;
7: Replace the individuals with the smallest fitness in the population by the new individuals;
8: Find the individual with the highest fitness in the final population;

Population initialisation. The initial population is created by generating a set of service compositions in form of DAGs, and then transforming them into their tree-like representations (*the individuals*). The initialisation is performed as follows:

STEP 1. Greedy search is performed to randomly generate a set of DAGs, each representing a (valid) service composition for the given composition task T. For this, a simple forward graph building algorithm is applied starting with the node $Start$ and the inputs I_T of the composition task T. Details of this algorithm can be found in [64]. An example of a generated DAG is shown in Fig. 3.4 with seven robust casual links marked on.

STEP 2. The DAGs can be simplified by removing some redundant edges and service nodes. While this step is not compulsory, it can help to notably reduce the size of the DAG and, consequently, the corresponding tree-like representation.

STEP 3. We transform each DAG into its tree-like representation using an algorithm modified from [24] to satisfy the particular requirements of our proposed approach. For example, Fig. 3.3 shows an example of a tree-like individual corresponding to the DAG shown in Fig. 3.4.

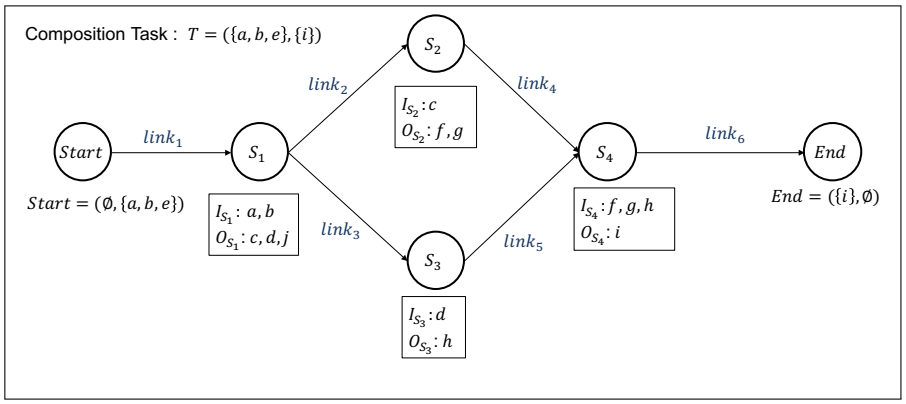


Figure 3.4: Example of a weighted DAG used for transferring it into tree-like representation

Crossover and Mutation. During the evolutionary process, the correctness of the representation is maintained by crossover and mutation.

A crossover operation exchanges a subtree of a selected individual (its attributes noted as $C_1(\mathcal{I}_{C_1}, \mathcal{O}_{C_1}, \mathcal{PI}_{C_1}, QoS_{C_1})$) with the subtree of another selected individual (its attributes noted as $C_2(\mathcal{I}_{C_2}, \mathcal{O}_{C_2}, \mathcal{PI}_{C_2}, QoS_{C_2})$) if they represent the same functionality (i.e. $\mathcal{I}_{C_1} = \mathcal{I}_{C_2}$ and $\mathcal{O}_{C_1} = \mathcal{O}_{C_2}$). That is, at the root nodes of both subtrees, we have identical inputs and identical outputs. A crossover operation is performed in two cases: crossover on two functional nodes or on two terminal nodes. We never exchange a functional node with an terminal node, since the two associated subtrees cannot be equivalent in this case. For example, *End* must appear in the subtree associated with any functional node, but not for any selected terminal node (atomic services).

A mutation operation replaces a subtree of the selected individual (its attributes noted as $C_1(\mathcal{I}_{C_1}, \mathcal{O}_{C_1}, \mathcal{PI}_{C_1}, QoS_{C_1})$) with a newly generated subtree satisfying the least required functionality. To do this, a subtree C_1 must be selected from the selected individual, and a new composition task $T = (\{\mathcal{PI}_{C_1}\}, \{\mathcal{O}_{C_1} \cap O_T\})$ or $T' = (\{\mathcal{PI}_{C_1}\}, \{\mathcal{O}_{C_1}\})$ is used to generate a tree in the same way as the 3-step method performed during the population initialisation. We utilise the available inputs and least required outputs for mutation, because it potentially bring more possibilities in generating more varieties of subtrees. The mutation is performed in two cases: mutation on a functional node with T or on a service node with T', two examples shown in Fig 3.5 (a) and (b). In Fig 3.5 (a), a functional node $C_{||}$ is selected for mutation, the whole subtree is replaced with the generated subtree excluding its head (i.e., *Start* and its parent node •). In Fig 3.5 (b), a atomic service S_1 is selected for mutation, the branch of the selected node (i.e., S_1 and its parent node •) is replaced with the generated subtree excluding both its head (i.e., *Start* and its parent node •) and its tail (i.e., *End*).

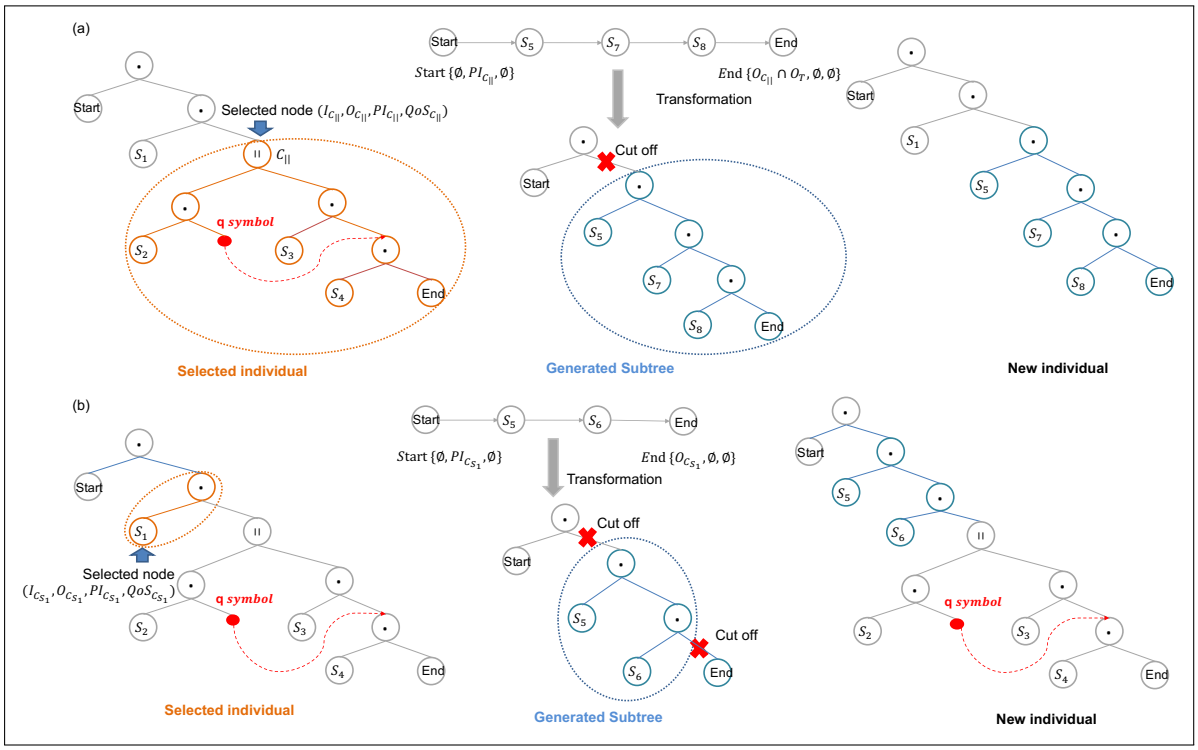


Figure 3.5: Examples of two mutations on terminal and functional nodes

Note: The set of available nodes considered for crossover and mutation do not include *Start* and *End*, and their parent nodes, because these nodes remain the same for all indi-

viduals. In addition, the nodes selected for crossover and mutation must not break the functionality of q symbols. For example, in Fig. 3.3, both sequential composition C_2 and C_3 are not considered for crossover and mutation as they break the edge of q symbol, but the parallel composition $C_{\|}$ can be considered for genetic operations, as it may bring a new fully functional q symbol or a subtree without q symbol involved. The pointed subtree C_4 could also be selected for genetic operations.

3.6 Experiment Study for GP-based Approach

We have conducted experiments to evaluate our proposed approach. For our experiments we have used the benchmark datasets originating from OWLS-TC [51] , which have been extended with real-world QoS attributes and five composition tasks [64]. To explore the effectiveness and efficiency of our proposed GP-based approach, we compare it against one recent GP-based approach [64]. For that we have extended the later approach by our proposed comprehensive quality model, so that semantic matchmaking quality can be computed based on the parent-child relationship in the underlying tree representations.

To assure a fair comparison we have used exactly the same parameter settings as in [64]. In particular, the GP population size has been set to 200, the number of generations to 30, the reproduction rate to 0.1, the crossover rate to 0.8, and the mutation rate to 0.1. We have run every experiment with 30 independent repetitions. Without considering any users' true service composition preferences, the weights in fitness function in Eq. (3.3) have been are configured simply to balance semantic matchmaking quality and QoS. Particularly, w_1 and w_2 are both set to 0.25, while w_3, w_4, w_5, and w_6 are all set to 0.125. The parameter p of $type_{link}$ is set to 0.75 ($plugin$ match) in accordance with the recommendation in [54].

Our experiments indicate that our method can work consistently well under valid weight settings and parameter p to be decided by users' preferences in practice.

3.6.1 Comparison against a previous GP-based approach

Table 3.4 shows the fitness values obtained by the two GP-based approaches. To compare the results, an independent-samples T-test over 30 runs has been conducted. The results show that our GP-based approach outperforms the previous GP-based approach [64] in finding more optimized composition solutions for Tasks 3 and 4. (Note: the P-values are lower than 0.0001). For Tasks 1, 2 and 5, both approaches achieve the same fitness. Therefore, the overall effectiveness of our proposed approach is considered to be better.

Table 3.4: Mean fitness values for our approach in comparison to [64]
(Note: the higher the fitness the better)

Task	Our GP-based approach	Ma et al. approach [64]
OWL-S TC1	0.923793 ± 0.000000	0.923793 ± 0.000000
OWL-S TC2	0.933026 ± 0.000000	0.933026 ± 0.000000
OWL-S TC3	$0.870251 \pm 0.000000 \uparrow$	0.832306 ± 0.008241
OWL-S TC4	$0.798137 \pm 0.007412 \uparrow$	0.760146 ± 0.005044
OWL-S TC5	0.832998 ± 0.000000	0.832998 ± 0.000000

Table 3.5 shows the execution times observed for the two GP-based approaches. Again an independent-samples T-test over 30 runs has been conducted. For Tasks 1 and 2 both approaches need about the same time, while for Tasks 3,4, and 5 our approach needs slightly more time. (Note: the P-values are lower than 0.0001). However, even in the worst case it

exceeds [64] by no more than 1 second, which is acceptable for most real-word scenarios. Hence, in terms of efficiency our approach is comparable to [64].

Table 3.5: Mean execution time (in ms) for our approach in comparison to [64] (Note: the shorter the time the better)

Task	Our GP-based approach	Ma et al. approach [64]
OWL-S TC1	7396.366667 ± 772.408168	7310.866667 ±952.701775
OWL-S TC2	2956.133333 ± 761.350965	3036.966667 ± 777.121101
OWL-S TC3	1057.266667 ± 174.405183	763.800000 ± 221.241232 ↓
OWL-S TC4	4479.466667 ± 519.767172	3068.800000 ± 472.013106 ↓
OWL-S TC5	6276.533333 ± 1075.102328	5030.200000 ± 991.863812 ↓

The experiments confirm that there is a trade-off between fitness and execution time in GP-based service composition. It can be argued that our proposed approach achieves a better balance as the computed solutions observe a significantly higher fitness while there is a moderate increase in execution time compared to [64].

3.6.2 Further Discussion

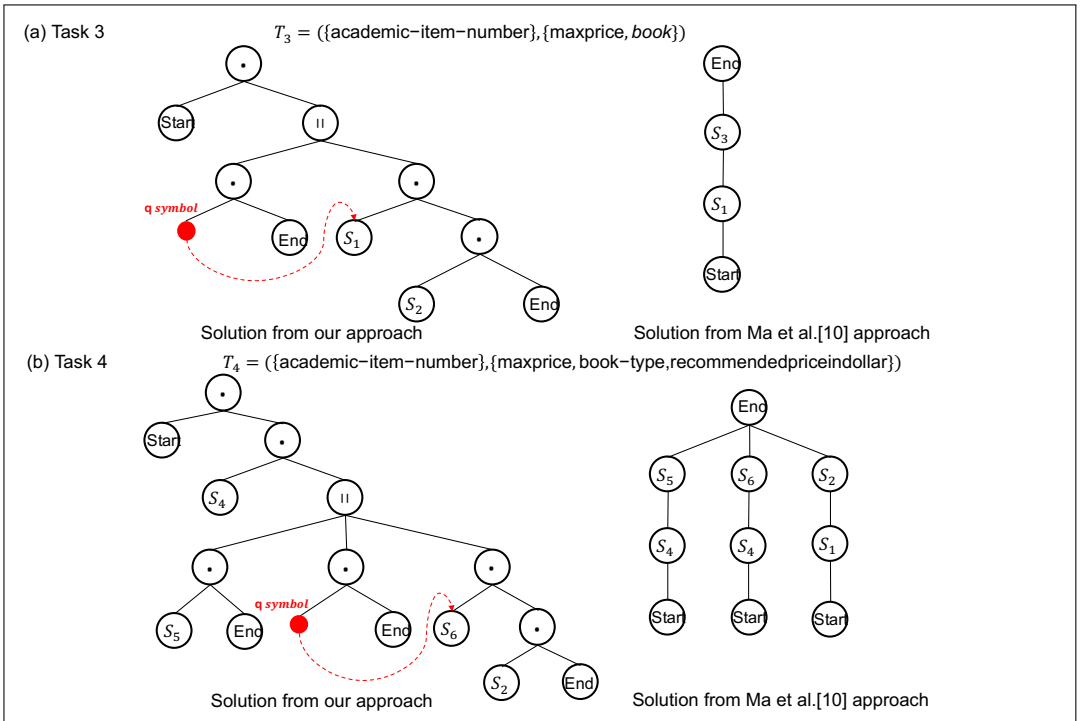
For Tasks 3 and 4, the optimized composition solutions obtained by the two approaches are shown in Fig. 3.6(a) and Fig. 3.6(b), respectively. The functional and nonfunctional descriptions of all services involved in these solutions are listed in Fig. 3.6(c).
For Task 3 the composition task is $T_3 = (\{academic\text{-}item\text{-}number\}, \{book, maxprice\})$. The best composition solutions obtained by the two approaches are different, see Fig. 3.6. Both solutions have the same semantic matchmaking quality as the matchmaking type of all links is *exact* match. However, both solutions differ in their QoS. This is due to the different services that are involved: S_2 versus S_3. The QoS of S_2 is much better that of S_3. Consequently, the best composition solution obtained by our approach has higher fitness according to our quality model. It is interesting to observe that the best composition solution obtained by our approach for Task 3 can be evolved from the best composition solution in [64] just by a single mutation on S_3 using available inputs.

For Task 4 the composition task is $T_4 = (\{academic\text{-}item\text{-}number\}, \{maxprice, book - type, recommendedpriceindollar\})$. The best solutions obtained by the two approaches are also different, see Fig. 3.6. Note that solution generated by our approach is composed of four atomic services (S_4, S_5, S_6 and S_2) while the solution generated by approach [64] is composed of five atomic services (S_4, S_5, S_6, S_1 and S_2). Both solutions have the same semantic matchmaking quality as the matchmaking type of all links is *exact* match. However, the overall QoS of our approach is better. This is due to the additional S_1 in their approach, which has a significant negative impact on QoS.

We observe from above examples that our approach is able to produce better solutions because our proposed representation keeps available inputs and least required outputs of each node on the tree which unlocks more opportunities for mutation and crossover rather than restricting them to the previously used inputs and outputs only.

3.7 Summary for our GP-based Approach

In this work, we introduces a novel GP-based approach to comprehensive quality-aware semantic Web service composition. In particular, a tree-like representation is proposed to direct cope with the evaluation of semantic matchmaking quality. Meanwhile, crossover



(c) Services used in the solutions above

S.	Atomic web service	Inputs	Outputs	QoS
S_1	academic-item-number_bookauthor_service	academic-item-number	auther, book	{ t:1765.91, c:6.61, r:0.51,a:0.83 }
S_2	author_publicationmaxprice_service	author	maxprice, publication	{ t:828.17, c:1.44, r:0.8, a:0.92 }
S_3	author_bookmaxprice_service	author	maxprice, book	{t:1912.14, c:1.39, r:0.78, a:0.5}
S_4	academicbooknumbersearch	academic-item-number	book	{ t:3280.67, c:9.72, r:0.49 a:0.66 }
S_5	tizonbook_recommendedpriceindollar_service	book	recommendedpriceindollar	{ t:3025.16, c:3.97r:0.43, a:0.88 }
S_6	book_authorbook-type_service	book	book-type, author	{ t:2481.87, c:1.47, r:0.47, a:0.68 }

Figure 3.6: Example of best solutions using the two GP-based approaches

and mutation methods are proposed to maintain the correctness of individuals. The experiment shows that our proposed approach could effectively produce better solutions in both semantic matchmaking quality and QoS than the existing approach.

3.8 Conclusion

In the preliminary work, we propose and formalize a novel and effective comprehensive quality model for simultaneously optimize QoS and quality of semantic matchmaking. Apart from that, two novel and effective EC-based methods are proposed for comprehensive quality-aware semantic Web service composition. In particular, indirect and direct representations are designed for studying their effectiveness and efficiency along with newly developed evolutionary operators. Overall experimental studies show both two methods reach good performances comparing to existing works. We also find that our PSO-based approach utilizing an indirect representation outperforms our GP-based approach utilizing a direct representation for finding more effective solutions. These findings motivate us to work on a more effective indirect presentation along with PSO-based memetic method.

Chapter 4

Proposed Contributions and Project Plan

In the previous chapter, some preliminary works have been done to investigate the performances of direct representation and indirect representation in the proposed PSO-based approach and GP-based approach respectively. We have shown that those two approaches outperform some recent existing EC-based approaches using our proposed comprehensive quality evaluation model. To conduct a further research on the remaining objectives outlined in the proposal, we present the proposed contribution, project plan, timeline and thesis outlines in this chapter.

4.1 Proposed Contributions

This thesis will contribute to the field of semantic Web service composition by considering several key composition problems simultaneously, and to the field of Evolutionary Computation techniques by proposing more novel representations and genetic operators. The proposed contributions of this project are listed below:

1. Develop fully automated semantic Web service composition approaches for simultaneously optimizing the quality of semantic matchmaking and QoS. We expect more effective and efficient methods to be developed for handling this novel combinatorial optimization problem, which is done by developing effective representations, a comprehensive quality model and EC-based methods with local search or AI planning techniques.

2. Develop fully automated EMO-based approaches to effectively and efficiently explore the Pareto Front to comprehensive quality-aware service composition. Meanwhile, conditional constraints on SLA and customized matchmaking level are expected to be effectively and efficiently handled. Apart from that, posteriori preference articulation techniques for user preference on comprehensive quality are also to be developed for reaching the most preferred Pareto Front. Each of these achievements is not completed in any past research.

3. Develop fully automated EC-based approaches for dynamic Web service composition. These approaches is expected to effectively and efficiently handle composition environment changes. That is, changes in QoS, ontology and service repository (i.e., service failure and new service registration). These dynamic problems are not fully studied in the past research, and will be firstly solved using EC-based techniques.

Table 4.1: The milestones of PhD project plan.

Phase	Description	Duration (months)
1	Reviewing literature, developing initial EC-based composition approach, and writing the proposal	9 (Complete)
2	Develop EC-based approaches to comprehensive quality-aware automated semantic Web service composition	6 (In progress)
3	Develop EMO-based approaches to comprehensive quality-aware automated semantic Web service composition	7
4	Develop EC-based techniques to support dynamic semantic Web service composition	8
5	Develop EC-based approaches to comprehensive quality-aware automated semantic Web service composition supporting preconditions and effects	3
6	Writing the thesis	6

4. Develop fully automated EC-based approaches to comprehensive quality-aware semantic Web service composition supporting preconditions and postconditions. We expected a general mechanism to be developed for satisfactions on preconditions and effects, and for supporting loop and choice composition constructs. Various composition constructs, preconditions and effects support and comprehensive quality optimization are not simultaneously handled in any existing research.

4.2 Overview of Project Plan

Six milestones for PhD project are defined in the initial research plan shown in Table 4.1. The first phase of this plan has been completed, which comprising of literature view, two initial works on comprehensive quality-aware semantic Web service composition and a research proposal writing. The second phase is related to the first objective of this proposal that is currently in progress and covered in Chapter 3 of the proposal. The remaining phases are expected to be completed as planned.

4.3 Project Timeline

Table 4.2 provide an estimated timeline comprising of the minor goals and milestones, which is expected to serve as a guide and be completed through the PhD project.

4.4 Thesis Outline

The following is an outline of the PhD thesis, in which Chapter 6 might be replaced by Chapter 7 since it is an optionally objective.

- *Chapter 1: Introduction*
  This chapter covers a problem statement, motivations, research goals, contributions, and organization of the thesis.

- *Chapter 2: Literature Review*
  This chapter initially provides some fundamental concepts for demonstrating the background of service composition. Followed, a comprehensive understanding and analyzing of existing work on the Web service composition. In particular, four research

Table 4.2: Project timeline for the remaining 24 months.

		Time in Months											
Phase	Task	2	4	6	8	10	12	14	16	18	20	22	24
n/a	Updating the literature review	x	x	x	x	x	x	x	x	x	x	x	x
2	Developing indirect representations utilize in a hyper-heuristics methods	x	x										
3	Investigating unconstrained EMO-based approaches for comprehensive quality-aware service composition		x	x									
3	Improving performance of EMO-based approaches by integrating other techniques			x	x								
3	Extending EMO-based approaches to handle constraints on SLA and semantic match-making level				x	x							
3	Extending EMO-based approaches to integrate preference articulation techniques					x	x						
4	Develop EC-based approaches to handle changes in QoS and ontology						x	x					
4	Develop EC-based approaches to handle service failures and new service registration							x	x				
5	Develop EC-based approaches to handle preconditions and effects.								x	x			
6	Writing the first thesis draft										x	x	
6	Editing the final draft											x	x

directions for Web service composition are investigated: single-objective approaches, multi-objective approaches, dynamic service composition and Web service composition supporting preconditions and effects.

- *Chapter 3: EC-based Approaches to Comprehensive Quality-Aware Web service Composition*
  This chapter will introduce new EC-based approaches that combine local search and/or AI planing methods. These approaches are developed to effective and efficient handle comprehensive quality-aware fully automated semantic service composition problems. Apart from that, the effectiveness of different direct/indirect representations are also investigated here.

- *Chapter 4: EMO-based Approaches to Comprehensive Quality-Aware Semantic Service Composition*
  This chapter will demonstrate our fully automated EMO-based approaches, optionally combining local search and/or AI planing methods, which optimizes different quality criteria within the comprehensive quality. Constraints on SLA and customized matchmaking levels are also handled in the multi-objective approaches. Apart from that, preference articulation techniques for multi-objective comprehensive quality-aware semantic Web service composition are also demonstrated here.

- *Chapter 5: EC-based Approaches to Support Dynamic Semantic Web service Composition*
  This chapter will discuss effective and efficient EC-based methods for handling dynamic service composition problems regarding the changes in QoS and Ontology and service repository (i.e., service failure new service registration). Those approaches are compared with existing dynamic service composition approaches, which do not utilize EC-based techniques.

- *Chapter 6: EC-based Approaches for Semantic Web service Compositions Supporting Preconditions and Effects*
  This chapter will discuss service composition supporting preconditions and effects. We firstly introduce a proposed matchmaking mechanism for preconditions and effects, which fully support different composition constructs, such as sequence, parallel, loop and choice. New and effective representations are introduced here to cope with preconditions and effects and utilized in our EC-based approaches, which are optionally combined with local search or AI planning to simultaneously handles various composition constructs and comprehensive quality optimization.

- *Chapter 7: Conclusions and Future Work*
  This chapter concludes all the contributions made by our works completed for each objective. In addition, limitations are also pointed out along with the future research directions.

4.5 Resources Required

4.5.1 Computing Resources

This research mainly utilizes an experimental approach. Due to the high computation of the experiment execution, ECS Grid computing facilities are required to complete these experiments.

4.5.2 Library Resources

The related literature of this research can be found online using the resources provided by Victoria University of Wellington. Apart from that, useful textbook and lecture notes can be also found in university's library.

4.5.3 Conference Travel Grants

Publications to relevant venues in this field are expected throughout this project, therefore travel grants from Victoria University of Wellington are required for key conferences.

Bibliography

[1] AGARWAL, S., JUNGHANS, M., FABRE, O., TOMA, I., AND LORRE, J.-P. D5. 3.1 first service discovery prototype. *Deliverable D5 3* (2009).

[2] AGARWAL, S., LAMPARTER, S., AND STUDER, R. Making web services tradable: A policy-based approach for specifying preferences on web service properties. *Web Semantics: Science, Services and Agents on the World Wide Web 7*, 1 (2009), 11–20.

[3] ALFÉREZ, G. H., PELECHANO, V., MAZO, R., SALINESI, C., AND DIAZ, D. Dynamic adaptation of service compositions with variability models. *Journal of Systems and Software 91* (2014), 24–47.

[4] ANDREWS, T., CURBERA, F., DHOLAKIA, H., GOLAND, Y., KLEIN, J., LEYMANN, F., LIU, K., ROLLER, D., SMITH, D., THATTE, S., ET AL. Business process execution language for web services, 2003.

[5] BACK, T., HAMMEL, U., AND SCHWEFEL, H.-P. Evolutionary computation: Comments on the history and current state. *IEEE transactions on Evolutionary Computation 1*, 1 (1997), 3–17.

[6] BANSAL, S., BANSAL, A., GUPTA, G., AND BLAKE, M. B. Generalized semantic web service composition. *Service Oriented Computing and Applications 10*, 2 (2016), 111–133.

[7] BARESI, L., AND GUINEA, S. Self-supervising bpel processes. *IEEE Transactions on Software Engineering 37*, 2 (2011), 247–263.

[8] BENTLEY, P. J., AND WAKEFIELD, J. P. Finding acceptable solutions in the pareto-optimal range using multiobjective genetic algorithms. *Soft computing in engineering design and manufacturing 5* (1997), 231–240.

[9] BLUM, A. L., AND FURST, M. L. Fast planning through planning graph analysis. *Artificial intelligence 90*, 1 (1997), 281–300.

[10] BOOTH, D., HAAS, H., MCCABE, F., NEWCOMER, E., CHAMPION, M., FERRIS, C., AND ORCHARD, D. Web services architecture. w3c working note. *W3C Working Notes* (2004).

[11] BOUSTIL, A., MAAMRI, R., AND SAHNOUN, Z. A semantic selection approach for composite web services using owl-dl and rules. *Service Oriented Computing and Applications 8*, 3 (2014), 221–238.

[12] BOUSTIL, A., MAAMRI, R., AND SAHNOUN, Z. A semantic selection approach for composite web services using OWL-DL and rules. *Service Oriented Computing and Applications 8*, 3 (2014), 221–238.

[13] BRANKE, J., CORRENTE, S., GRECO, S., SŁOWIŃSKI, R., AND ZIELNIEWICZ, P. Using choquet integral as preference model in interactive evolutionary multiobjective optimization. *European Journal of Operational Research 250*, 3 (2016), 884–901.

[14] BRANKE, J., AND DEB, K. Integrating user preferences into evolutionary multiobjective optimization. In *Knowledge incorporation in evolutionary computation*. Springer, 2005, pp. 461–477.

[15] BRANKE, J., KAUSSLER, T., AND SCHMECK, H. Guidance in evolutionary multiobjective optimization. *Advances in Engineering Software 32*, 6 (2001), 499–507.

[16] CANFORA, G., DI PENTA, M., ESPOSITO, R., AND VILLANI, M. L. An approach for qos-aware service composition based on genetic algorithms. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation* (2005), ACM, pp. 1069–1075.

[17] CHAN, K., BISHOP, J., STEYN, J., BARESI, L., AND GUINEA, S. A fault taxonomy for web service composition. In *Service-oriented computing-ICSOC 2007 Workshops* (2009), Springer, pp. 363–375.

[18] CHEN, Y., HUANG, J., AND LIN, C. Partial selection: An efficient approach for qos-aware web service composition. In *Web Services (ICWS), 2014 IEEE International Conference on* (2014), IEEE, pp. 1–8.

[19] CHENG, R., OLHOFER, M., AND JIN, Y. Reference vector based a posteriori preference articulation for evolutionary multiobjective optimization. In *Evolutionary Computation (CEC), 2015 IEEE Congress on* (2015), IEEE, pp. 939–946.

[20] CURBERA, F., DUFTLER, M., KHALAF, R., NAGY, W., MUKHI, N., AND WEERAWARANA, S. Unraveling the web services web: an introduction to soap, wsdl, and uddi. *IEEE Internet computing 6*, 2 (2002), 86–93.

[21] CURBERA, F., NAGY, W., AND WEERAWARANA, S. Web services: Why and how. In *Workshop on Object-Oriented Web Services-OOPSLA* (2001), vol. 2001.

[22] DA SILVA, A., MA, H., AND ZHANG, M. Graphevol: A graph evolution technique for web service composition. In *Database and Expert Systems Applications*, vol. 9262. Springer International Publishing, 2015, pp. 134–142.

[23] DA SILVA, A. S., MA, H., AND ZHANG, M. A gp approach to qos-aware web service composition including conditional constraints. In *Evolutionary Computation (CEC), 2015 IEEE Congress on* (2015), IEEE, pp. 2113–2120.

[24] DA SILVA, A. S., MA, H., AND ZHANG, M. Genetic programming for qos-aware web service composition and selection. *Soft Computing* (2016), 1–17.

[25] DA SILVA, A. S., MA, H., ZHANG, M., AND HARTMANN, S. Handling branched web service composition with a qos-aware graph-based method. In *International Conference on Electronic Commerce and Web Technologies* (2016), Springer, pp. 154–169.

[26] DA SILVA, A. S., MEI, Y., MA, H., AND ZHANG, M. A memetic algorithm-based indirect approach to web service composition. In *Evolutionary Computation (CEC), 2016 IEEE Congress on* (2016), IEEE, pp. 3385–3392.

[27] DA SILVA, A. S., MEI, Y., MA, H., AND ZHANG, M. Particle swarm optimisation with sequence-like indirect representation for web service composition. In *European Conference on Evolutionary Computation in Combinatorial Optimization* (2016), Springer, pp. 202–218.

[28] DE CAMPOS, A., POZO, A. T., VERGILIO, S. R., AND SAVEGNAGO, T. Many-objective evolutionary algorithms in the composition of web services. In *Neural Networks (SBRN), 2010 Eleventh Brazilian Symposium on* (2010), IEEE, pp. 152–157.

[29] DE CASTRO, L. N., AND VON ZUBEN, F. J. Learning and optimization using the clonal selection principle. *IEEE transactions on evolutionary computation 6*, 3 (2002), 239–251.

[30] DEB, K., PRATAP, A., AGARWAL, S., AND MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation 6*, 2 (2002), 182–197.

[31] ERL, T. *Service-oriented architecture: a field guide to integrating XML and web services*. Prentice hall, 2004.

[32] FANJIANG, Y.-Y., AND SYU, Y. Semantic-based automatic service composition with functional and non-functional requirements in design time: A genetic algorithm approach. *Information and Software Technology 56*, 3 (2014), 352–373.

[33] FENG, Y., NGAN, L. D., AND KANAGASABAI, R. Dynamic service composition with service-dependent qos attributes. In *Web Services (ICWS), 2013 IEEE 20th International Conference on* (2013), IEEE, pp. 10–17.

[34] FENSEL, D., FACCA, F. M., SIMPERL, E., AND TOMA, I. *Semantic web services*. Springer Science & Business Media, 2011.

[35] FLEMING, P. J., AND PURSHOUSE, R. C. Evolutionary algorithms in control systems engineering: a survey. *Control engineering practice 10*, 11 (2002), 1223–1241.

[36] GAO, A., YANG, D., TANG, S., AND ZHANG, M. Web service composition using integer programming-based models. In *e-Business Engineering, 2005. ICEBE 2005. IEEE International Conference on* (2005), IEEE, pp. 603–606.

[37] GAREY, M. R., AND JOHNSON, D. S. A guide to the theory of np-completeness. *WH Freemann, New York 70* (1979).

[38] GIAGKIOZIS, I., AND FLEMING, P. J. Pareto front estimation for decision making. *Evolutionary computation 22*, 4 (2014), 651–678.

[39] GUINARD, D., TRIFA, V., SPIESS, P., DOBER, B., AND KARNOUSKOS, S. Discovery and on-demand provisioning of real-world web services. In *Web Services, 2009. ICWS 2009. IEEE International Conference on* (2009), IEEE, pp. 583–590.

[40] GUPTA, I. K., KUMAR, J., AND RAI, P. Optimization to quality-of-service-driven web service composition using modified genetic algorithm. In *Computer, Communication and Control (IC4), 2015 International Conference on* (2015), IEEE, pp. 1–6.

[41] HUANG, Z., JIANG, W., HU, S., AND LIU, Z. Effective pruning algorithm for qos-aware service composition. In *2009 IEEE Conference on Commerce and Enterprise Computing* (2009), IEEE, pp. 519–522.

[42] HWANG, C.-L., AND YOON, K. Lecture notes in economics and mathematical systems. *Multiple Objective Decision Making, Methods and Applications: A State-of-the-Art Survey 164* (1981).

[43] ISHIBUCHI, H., TSUKAMOTO, N., AND NOJIMA, Y. Evolutionary many-objective optimization: A short review. In *IEEE congress on evolutionary computation* (2008), pp. 2419–2426.

[44] ISHIKAWA, F., KATAFUCHI, S., WAGNER, F., FUKAZAWA, Y., AND HONIDEN, S. Bridging the gap between semantic web service composition and common implementation architectures. In *Services Computing (SCC), 2011 IEEE International Conference on* (2011), IEEE, pp. 152–159.

[45] KENNEDY, J., AND EBERHART, R. Particle swarm optimization. In *IEEE International Conference on Neural Networks* (1995), IEEE, pp. 1942–1948.

[46] KIM, I. Y., AND DE WECK, O. Adaptive weighted sum method for multiobjective optimization: a new method for pareto front generation. *Structural and multidisciplinary optimization 31*, 2 (2006), 105–116.

[47] KONA, S., BANSAL, A., BLAKE, M. B., BLEUL, S., AND WEISE, T. Wsc-2009: a quality of service-oriented web services challenge. In *2009 IEEE Conference on Commerce and Enterprise Computing* (2009), IEEE, pp. 487–490.

[48] KONING, M., SUN, C.-A., SINNEMA, M., AND AVGERIOU, P. Vxbpel: Supporting variability for web services in bpel. *Information and Software Technology 51*, 2 (2009), 258–269.

[49] KOPECKÝ, J., VITVAR, T., BOURNEZ, C., AND FARRELL, J. Sawsdl: Semantic annotations for wsdl and xml schema. *IEEE Internet Computing 11*, 6 (2007).

[50] KOZA, J. R. *Genetic programming: on the programming of computers by means of natural selection*, vol. 1. MIT press, 1992.

[51] KÜSTER, U., KÖNIG-RIES, B., AND KRUG, A. Opossum-an online portal to collect and share sws descriptions. In *Semantic Computing, 2008 IEEE International Conference on* (2008), IEEE, pp. 480–481.

[52] LAUSEN, H., AND FARRELL, J. Semantic annotations for wsdl and xml schema. *W3C recommendation, W3C* (2007), 749–758.

[53] LAUSEN, H., POLLERES, A., AND ROMAN, D. W3c member submission-web service modeling ontology (wsmo). *W3C. Available at¡ URL: http://www. w3. org/Submission/WSMO* (2005).

[54] LÉCUÉ, F. Optimizing qos-aware semantic web service composition. In *International Semantic Web Conference* (2009), Springer, pp. 375–391.

[55] LÉCUÉ, F., AND DELTEIL, A. Making the difference in semantic web service composition. In *Proceedings of the National Conference on Artificial Intelligence* (2007), vol. 22, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, p. 1383.

[56] LÉCUÉ, F., DELTEIL, A., AND LÉGER, A. Optimizing causal link based web service composition. In *ECAI* (2008), pp. 45–49.

[57] LÉCUÉ, F., AND LÉGER, A. A formal model for semantic web service composition. In *International Semantic Web Conference* (2006), Springer, pp. 385–398.

[58] LI, G., LIAO, L., SONG, D., AND ZHENG, Z. A fault-tolerant framework for qos-aware web service composition via case-based reasoning. *International Journal of Web and Grid Services 10*, 1 (2014), 80–99.

[59] LI, J., YAN, Y., AND LEMIRE, D. Full solution indexing for top-k web service composition. *IEEE Transactions on Services Computing* (2016).

[60] LIU, J., LI, J., LIU, K., AND WEI, W. A hybrid genetic and particle swarm algorithm for service composition. In *Advanced Language Processing and Web Information Technology, 2007. ALPIT 2007. Sixth International Conference on* (2007), IEEE, pp. 564–567.

[61] LIU, S., LIU, Y., JING, N., TANG, G., AND TANG, Y. A dynamic web service selection strategy with qos global optimization based on multi-objective genetic algorithm. In *International Conference on Grid and Cooperative Computing* (2005), Springer, pp. 84–89.

[62] LONG, J., AND GUI, W. An environment-aware particle swarm optimization algorithm for services composition. In *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on* (2009), IEEE, pp. 1–4.

[63] MA, H., SCHEWE, K.-D., THALHEIM, B., AND WANG, Q. A formal model for the interoperability of service clouds. *Service Oriented Computing and Applications 6*, 3 (2012), 189–205.

[64] MA, H., WANG, A., AND ZHANG, M. A hybrid approach using genetic programming and greedy search for qos-aware web service composition. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems XVIII*. Springer, 2015, pp. 180–205.

[65] MARKOU, G., AND REFANIDIS, I. Non-deterministic planning methods for automated web service composition. *Artificial Intelligence Research 5*, 1 (2015), 14.

[66] MARTIN, D., BURSTEIN, M., HOBBS, J., LASSILA, O., MCDERMOTT, D., MCILRAITH, S., NARAYANAN, S., PAOLUCCI, M., PARSIA, B., PAYNE, T., ET AL. Owl-s: Semantic markup for web services. *W3C member submission 22* (2004), 2007–04.

[67] MCILRAITH, S. A., SON, T. C., AND ZENG, H. Semantic web services. *IEEE intelligent systems 16*, 2 (2001), 46–53.

[68] MIER, P. R., PEDRINACI, C., LAMA, M., AND MUCIENTES, M. An integrated semantic web service discovery and composition framework.

[69] MOGHADDAM, M., AND DAVIS, J. G. Service selection in web service composition: A comparative review of existing approaches. In *Web Services Foundations*. Springer, 2014, pp. 321–346.

[70] MOHANTY, R., RAVI, V., AND PATRA, M. R. Web-services classification using intelligent techniques. *Expert Systems with Applications 37*, 7 (2010), 5484–5490.

[71] MOSTAFA, A., AND ZHANG, M. Multi-objective service composition in uncertain environments. *IEEE Transactions on Services Computing* (2015).

[72] NASRIDINOV, A., BYUN, J.-Y., AND PARK, Y.-H. A qos-aware performance prediction for self-healing web service composition. In *Cloud and Green Computing (CGC), 2012 Second International Conference on* (2012), IEEE, pp. 799–803.

[73] O'LEARY, D. Review: Ontologies: A silver bullet for knowledge management and electronic commerce. *The Computer Journal 48*, 4 (2005), 498–498.

[74] OVERDICK, H. The resource-oriented architecture. In *Services, 2007 IEEE Congress on* (2007), IEEE, pp. 340–347.

[75] PALIWAL, A. V., SHAFIQ, B., VAIDYA, J., XIONG, H., AND ADAM, N. Semantics-based automated service discovery. *IEEE Transactions on Services Computing 5*, 2 (2012), 260–275.

[76] PAOLUCCI, M., KAWAMURA, T., PAYNE, T. R., AND SYCARA, K. Semantic matching of web services capabilities. In *International Semantic Web Conference* (2002), Springer, pp. 333–347.

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

[78] PAPAZOGLOU, M. T. P., dustdar, s., leymann, f.. service-oriented computing. research roadmap, 2006.

[79] PAREJO, J. A., FERNANDEZ, P., AND CORTÉS, A. R. Qos-aware services composition using tabu search and hybrid genetic algorithms. *Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos 2*, 1 (2008), 55–66.

[80] PEER, J. Web service composition as ai planning-a survey. *University of St. Gallen* (2005).

[81] PETRIE, C. J. *Web Service Composition*. Springer, 2016.

[82] POP, C. B., CHIFU, V. R., SALOMIE, I., AND DINSOREANU, M. Immune-inspired method for selecting the optimal solution in web service composition. In *International Workshop on Resource Discovery* (2009), Springer, pp. 1–17.

[83] QI, L., TANG, Y., DOU, W., AND CHEN, J. Combining local optimization and enumeration for qos-aware web service composition. In *Web Services (ICWS), 2010 IEEE International Conference on* (2010), IEEE, pp. 34–41.

[84] RAO, J., DIMITROV, D., HOFMANN, P., AND SADEH, N. A mixed initiative approach to semantic web service discovery and composition: Sap's guided procedures framework. In *2006 IEEE International Conference on Web Services (ICWS'06)* (2006), IEEE, pp. 401–410.

[85] RAO, J., AND SU, X. A survey of automated web service composition methods. In *International Workshop on Semantic Web Services and Web Process Composition* (2004), Springer, pp. 43–54.

[86] RAO, J., AND SU, X. Semantic web services and web process composition, volume 3387 of lncs, chapter a survey of automated web service composition methods, 2005.

[87] RENDERS, J.-M., AND FLASSE, S. P. Hybrid methods using genetic algorithms for global optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 26*, 2 (1996), 243–258.

[88] RODRIGUEZ-MIER, P., MUCIENTES, M., LAMA, M., AND COUTO, M. I. Composition of web services through genetic programming. *Evolutionary Intelligence 3*, 3-4 (2010), 171–186.

[89] SAHAI, A., MACHIRAJU, V., SAYAL, M., VAN MOORSEL, A., AND CASATI, F. Automated sla monitoring for web services. In *International Workshop on Distributed Systems: Operations and Management* (2002), Springer, pp. 28–41.

[90] SALAS, J., PEREZ-SORROSAL, F., PATIÑO-MARTÍNEZ, M., AND JIMÉNEZ-PERIS, R. Ws-replication: a framework for highly available web services. In *Proceedings of the 15th international conference on World Wide Web* (2006), ACM, pp. 357–366.

[91] SHET, K., ACHARYA, U. D., ET AL. A new similarity measure for taxonomy based on edge counting. *arXiv preprint arXiv:1211.4709* (2012).

[92] SHI, Y., ET AL. Particle swarm optimization: developments, applications and resources. In *evolutionary computation, 2001. Proceedings of the 2001 Congress on* (2001), vol. 1, IEEE, pp. 81–86.

[93] SIRIN, E., PARSIA, B., WU, D., HENDLER, J., AND NAU, D. Htn planning for web service composition using shop2. *Web Semantics: Science, Services and Agents on the World Wide Web 1*, 4 (2004), 377–396.

[94] SOHRABI, S., PROKOSHYNA, N., AND MCILRAITH, S. A. Web service composition via the customization of golog programs with user preferences. In *Conceptual Modeling: Foundations and Applications*. Springer, 2009, pp. 319–334.

[95] SRINIVAS, M., AND PATNAIK, L. M. Genetic algorithms: A survey. *computer 27*, 6 (1994), 17–26.

[96] TANG, M., AND AI, L. A hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition. In *Evolutionary Computation (CEC), 2010 IEEE Congress on* (2010), IEEE, pp. 1–8.

[97] VAN VELDHUIZEN, D. A., AND LAMONT, G. B. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary computation 8*, 2 (2000), 125–147.

[98] VAN VELDHUIZEN, D. A., AND LAMONT, G. B. On measuring multiobjective evolutionary algorithm performance. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on* (2000), vol. 1, IEEE, pp. 204–211.

[99] WADA, H., CHAMPRASERT, P., SUZUKI, J., AND OBA, K. Multiobjective optimization of sla-aware service composition. In *Services-Part I, 2008. IEEE Congress on* (2008), IEEE, pp. 368–375.

[100] WADA, H., SUZUKI, J., YAMANO, Y., AND OBA, K. E^3: A multiobjective optimization framework for sla-aware service composition. *IEEE Transactions on Services Computing 5*, 3 (2012), 358–372.

[101] WAGNER, F., ISHIKAWA, F., AND HONIDEN, S. Robust service compositions with functional and location diversity. *IEEE Transactions on Services Computing 9*, 2 (2016), 277–290.

[102] WANG, A., MA, H., AND ZHANG, M. Genetic programming with greedy search for web service composition. In *International Conference on Database and Expert Systems Applications* (2013), Springer, pp. 9–17.

[103] WANG, C., MA, H., CHEN, A., AND HARTMANN, S. Comprehensive quality-aware automated semantic web service composition. In *Advances in Artificial Intelligence: 30th Australasian Joint Conference* (2017), Springer, pp. 195–207.

[104] WANG, D., HUANG, H., AND XIE, C. A novel adaptive web service selection algorithm based on ant colony optimization for dynamic web service composition. In *Algorithms and Architectures for Parallel Processing*. Springer, 2014, pp. 391–399.

[105] WANG, L., SHEN, J., AND YONG, J. A survey on bio-inspired algorithms for web service composition. In *Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on* (2012), IEEE, pp. 569–574.

[106] WANG, P., DING, Z., JIANG, C., AND ZHOU, M. Automated web service composition supporting conditional branch structures. *Enterprise Information Systems 8*, 1 (2014), 121–146.

[107] WANG, P., DING, Z., JIANG, C., ZHOU, M., AND ZHENG, Y. Automatic web service composition based on uncertainty execution effects. *IEEE Transactions on Services Computing 9*, 4 (2016), 551–565.

[108] WEN, S., TANG, C., LI, Q., CHIU, D. K., LIU, A., AND HAN, X. Probabilistic top-k dominating services composition with uncertain qos. *Service Oriented Computing and Applications 8*, 1 (2014), 91–103.

[109] WHITLEY, D. A genetic algorithm tutorial. *Statistics and computing 4*, 2 (1994), 65–85.

[110] XIANG, F., HU, Y., YU, Y., AND WU, H. Qos and energy consumption aware service composition and optimal-selection based on pareto group leader algorithm in cloud manufacturing system. *Central European Journal of Operations Research 22*, 4 (2014), 663–685.

[111] XU, B., LUO, S., YAN, Y., AND SUN, K. Towards efficiency of qos-driven semantic web service composition for large-scale service-oriented systems. *Service Oriented Computing and Applications 6*, 1 (2012), 1–13.

[112] XU, C., LIANG, P., WANG, T., WANG, Q., AND SHEU, P. C. Semantic web services annotation and composition based on er model. In *Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC), 2010 IEEE International Conference on* (2010), IEEE, pp. 413–420.

[113] YAN, G., JUN, N., BIN, Z., LEI, Y., QIANG, G., AND YU, D. Immune algorithm for selecting optimum services in web services composition. *Wuhan University Journal of Natural Sciences 11*, 1 (2006), 221–225.

[114] YAO, Y., AND CHEN, H. Qos-aware service composition using nsga-ii 1. In *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human* (2009), ACM, pp. 358–363.

[115] YIN, H., ZHANG, C., ZHANG, B., GUO, Y., AND LIU, T. A hybrid multiobjective discrete particle swarm optimization algorithm for a sla-aware service composition problem. *Mathematical Problems in Engineering 2014* (2014).

[116] YIN, Y., ZHANG, B., AND ZHANG, X. Qos-driven transactional web service reselection for reliable execution. In *Information Science and Management Engineering (ISME), 2010 International Conference of* (2010), vol. 2, IEEE, pp. 79–82.

[117] YOO, J. J.-W., KUMARA, S., LEE, D., AND OH, S.-C. A web service composition framework using integer programming with non-functional objectives and constraints. In *2008 10th IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services* (2008), IEEE, pp. 347–350.

[118] YU, Q., AND BOUGUETTAYA, A. Efficient service skyline computation for composite service selection. *Knowledge and Data Engineering, IEEE Transactions on 25*, 4 (2013), 776–789.

[119] YU, Q., LIU, X., BOUGUETTAYA, A., AND MEDJAHED, B. Deploying and managing web services: issues, solutions, and directions. *The VLDB JournalThe International Journal on Very Large Data Bases 17*, 3 (2008), 537–572.

[120] YU, T., ZHANG, Y., AND LIN, K.-J. Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Transactions on the Web (TWEB) 1*, 1 (2007), 6.

[121] YU, Y., MA, H., AND ZHANG, M. An adaptive genetic programming approach to qos-aware web services composition. In *2013 IEEE Congress on Evolutionary Computation* (2013), IEEE, pp. 1740–1747.

[122] ZENG, L., BENATALLAH, B., DUMAS, M., KALAGNANAM, J., AND SHENG, Q. Z. Quality driven web services composition. In *Proceedings of the 12th international conference on World Wide Web* (2003), ACM, pp. 411–421.

[123] ZHANG, Q., AND LI, H. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation 11*, 6 (2007), 712–731.

[124] ZHANG, W., CHANG, C. K., FENG, T., AND JIANG, H.-Y. Qos-based dynamic web service composition with ant colony optimization. In *Computer Software and Applications Conference (COMPSAC), 2010 IEEE 34th Annual* (2010), IEEE, pp. 493–502.

[125] ZHAO, X., SHEN, L., PENG, X., AND ZHAO, W. Toward sla-constrained service composition: An approach based on a fuzzy linguistic preference model and an evolutionary algorithm. *Information Sciences 316* (2015), 370–396.

[126] ZITZLER, E. Evolutionary algorithms for multiobjective optimization: Methods and applications.

[127] ZITZLER, E., DEB, K., AND THIELE, L. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation 8*, 2 (2000), 173–195.

[128] ZITZLER, E., LAUMANNS, M., THIELE, L., ET AL. Spea2: Improving the strength pareto evolutionary algorithm, 2001.