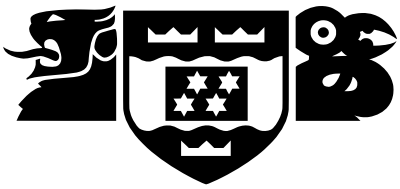
VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui



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

July 24, 2017

Submitted in partial fulfilment of the requirements for PhD.

Abstract

.....

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 [72, 73]. One of the primary purposes of SOC is to overcome conflicts due to diverse platforms and programming languages to make integrable and seamless communication among those existing or newly built independent services. *Service Oriented Architecture* (SOA) could abstractly realise service-oriented paradigm of computing. This accomplishment has been contributing to reuse of software components, from the concept of functions to units and from units to services during the evolution of development in SOA [11, 69]. 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" [18]. Several standards play a significant role in registering, enquiring and grounding web services across the web, such as UDDI [17], WSDL [47] and SOAP [29]. *Web service composition* aims to loosely couple a set of web services to provide a value-added composite service that accommodates complex functional and non-functional requirements of service users.

Two most notable challenges for web service composition are ensuring interoperability of services and achieving Quality of Service (QoS) optimisation [29]. *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 [111], 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 a better collaboration through ontology-based semantics [68], in which many standards have been established, such as OWL-S [61], Web Service Modeling Ontology (WSMO) [48], SAWSDL [44], and Semantic Web Services Ontology (SWSO) [76]. This dimension brings around some other underlying functionality of services (i.e., precondition and postcondition) that could effect 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). Since the quality of semantic matchmaking is always used to measure the goodness of interoperability while composing services. Another challenge is related to QoS optimisation, where QoS represents non-functional attributes of service composition (e.g, cost, time, reliability and availability). Often, a global search is employed to minimise the cost and maximise the reliability of composite services. This challenge gives birth to *QoS-aware service composition* that aims to find composition solutions with optimised QoS. Furthermore, QoS-aware service composition problem is facilitated by *Service Level Agreement (SLA)* [84], 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.

Apart from the two notable challenges discussed above, the environment of service composition is changing in the real world, rather than *static*. E.g, QoS values of services being composed of are fluctuating over time. Services chosen at the planning stage may not available to be invoked at the runtime, it may due to imprecise interface description [39] or hardware failures [34]. New services can also be registered in the service repository from time to time. Existing techniques for *static web service composition* can not support such a changing environment. Therefore, *Dynamic web service composition* become a very demanding research field with a growing interest and a practical value. Particularly, some mechanisms are required to be developed to automatically detect the changes or recover from the faults [15]. Additionally, in context of semantic web service composition, semantics of web services can make the problem of dynamic web service composition more complicated due to the changes in the ontology.

Different approaches [21, 24, 35, 49, 59, 78, 83, 112, 98] have been proposed to solve those composition problems discussed above and they can be classified into two main categories: *semi-automated web service composition* and *fully automated web service composition*. The first composition problem requires human beings to manually create abstract workflows. Generally, researchers assume the pre-defined abstract workflow is given and provided by the users. The optimisation problem in this approach turns to selecting the concrete services with the best possible quality to each abstract service slot in a given workflow. Due to a tremendous growth in industries and enterprise applications, the number of web services has increased dramatically and unprecedentedly. The process of conducting abstract workflows manually is fraught with difficulties in effectively and efficiently solving composition problems, such as QoS-aware service composition problem. Therefore, full automation of composition process is introduced in web service composition for less human intervention, less time consumption, and high productivity [80]. The differences in fully automated approach is that an abstract workflow is not provided, but generated while service are being selected.

Generating composition plans automatically in discovering and selecting suitable web services is a NP-hard problem [64], which means the composition solution is not likely to be found with reasonable computation time in a large searching space. *Artificial Intelligence (AI) planning-based approaches*, *Evolutionary Computation (EC) techniques* and hybrid techniques have been introduced to handle this problem. AI planning problem is utilised to solve automated web service composition problems as a plan making process, from initial states to a set of actions to desired goal states-composite web services, 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 have been used in this context, e.g., Genetic Algorithms (GA), Genetic Programming (GP) and Particle Swarm Optimisation (PSO). EC-based techniques have been effectively developed to solve *QoS-aware web service composition* problems with different structures for solution 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 systems. In the third approach, a hybrid of AI planning-based approaches and EC-based approaches [21, 59] have been proposed to fulfil the correctness in constructing workflows under users' constraints, while the quality of composition solutions (e.g., QoS) are also optimised according to users' requirements. From the literature, hybrid approaches generally outperform EC-based methods for finding more optimal solutions in the domain of automated QoS-aware web service

composition. It becomes a very promising approach to investigate. As summarised here, a few previously addressed challenges for fully automated service compositions lies in jointly optimising QoS and semantic matchmaking quality, dealing with multi-objective optimisation, handling environment changes in dynamic service composition, and composing semantic web services. Those challenges are addressed with key limitations in Section 1.2.

The overall goal of this thesis is to propose hybrid approaches to comprehensive-quality aware automated web service composition. This comprehensive quality aims to jointly optimise semantic matchmaking quality and QoS. Meanwhile, this new approach also tackles several service composition problems, such as multi-objective optimisation, dynamic web service composition and semantic web service composition based on preconditions and postconditions.

1.2 Motivations

The motivations of this proposed research lies in the requirements from five key aspects that simultaneously account for. 1. *Various techniques of hybridisation*. 2. *Hybridisation of quality of semantic matchmaking and quality of service*. 3. *Muti-objective optimisation*. 4. *Dynamic semantic service composition*. 5. *Service composition based on preconditions and effects*. Herein these requirements are explicitly discussed below.

Various Techniques of Hybridisation

Various techniques have been utilised to solve service composition problems, such as AI planning, local searching and EC-based techniques [28, 74, 78, 98]. 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) [98]. Local search is a exhaustive search technique for solving optimisation problems. In local search, solutions keep moving to the neighbour solutions, driven by some local maximisation criterion until near-optimal solution found [74]. However, this technique has the shortage of being trapped in local optimal. On the other hand, EC-based techniques are outstanding at solving combinatorial optimisation problems for finding the globally optimised solutions in large searching spaces. To take the benefits from various techniques, various techniques of hybridisation allows escaping local optima easily and improving the rate of convergence rate [82].

Traditionally, various techniques are employed to handle service composition problem independently in existing works. Many researchers investigated AI planning techniques for service composition problems using classical planning algorithm, where inputs, outputs, preconditions and effects are well defined along with the actions (i.e, services) [60, 75]. On the one hand, AI planning ensures both the correctness of functionality and satisfactory of constrains, but it is always considered to be less efficient and less scalable, incapable of solving complicated optimisation problems [74]. On the other hand, some researchers combined AI planning and local search to handle optimisation problems, e.g., a combination of Graphplan [10] and Dijkstras algorithm is proposed by [28] to achieve a correct solution with optimised QoS. Yet, many EC techniques have been utilised to handle service composition problems for global optimal solutions. A few researchers also combine both local search and EC-based techniques for efficiently finding composition solution with optimised QoS [74]. From the techniques discussed above, they are problem-specific for either optimising QoS or number of services. In this thesis, more complicated and realistic service composition problems are addressed. To with cope this composition problems featured in all the motivations

discussed below (i.e comprehensive quality-aware service composition, multi-objective optimisation, dynamic service composition, and composing semantic web services based on preconditions and postconditions), new hybrid methods must be proposed to adapt in the problems specified in the thesis. For example, a hybrid method for jointly optimising both semantic matchmaking quality and QoS.

Hybridisation of Quality of Semantic Matchmaking and Quality of Service

Web service compositions are optimised by the well known non-functional attributes (i.e, QoS), when services are composed together based on outputs provided by one service and inputs required by another service. In the domain of semantic web services, often, the provided information (i.e., outputs) does not perfectly match the required information (i.e., inputs) according to their semantic descriptions [51]. The quality of the matches (i.e., quality of semantic matchmaking) are one part of the goal for achieving service compositions [49]. Therefore, a hybridisation of QoS and semantic matchmaking quality becomes a combinatorial optimisation problem in web service composition. One motivated example from the practical perspective is explained here: many different service compositions can meet a user request but differ significantly in terms of QoS and semantic matchmaking quality. For example, in the classical travel planning context, some component service must be employed to obtain a travel map. Suppose that two services can be considered for this purpose. One service S can 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. Because in our context a tourist map is more desirable than a street map, S' clearly enjoys better semantic matchmaking quality than S but will have 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 focus mainly on addressing only one quality aspect discussed above. For the semantic matchmaking quality, it is mainly addressed in works that focus on the discovery of atomic services, i.e., one-to-one matching of user requirements to a single service. Some works [7, 12, 63] on semantic service composition utilise semantic descriptions of web services (e.g., description logic) to ensure the interoperability of web services, but the goal of the composition is often to minimise the number of services or the size of a graph representation for a web service composition. These approaches do not guarantee an optimised QoS of service compositions. On the other hand, huge efforts have been devoted to studying QoS-aware web service composition [19, 24, 35, 59, 78, 112]. Some of these works do consider the semantic matchmaking while composing solutions, but do not recognise the importance of semantic matchmaking quality for optimising service composition. Therefore, it is not sufficient to only consider one quality aspect for optimising service composition. For these reasons, there is a need to device an comprehensive quality model for jointly optimising the two quality aspects. Apart from that, existing solutions representations may not effectively and efficiently handle this new optimisation problem, so new solution representations need to be proposed to maintain the required information for the optimisation of the two quality aspects using hybrid methods.

Multi-Objective Composition Optimisation

EC-based approaches for handling web service composition problems fall into two groups, depending on the different goals of optimisation for either a single objective or multiple objectives. In single-objective service compositions, one composition solution is always returned by a composition task, where the preferences of each quality component within the

single objective (e.g., a weighted sum of different quality criteria) is known by users. However, users do not always have clear preferences when many quality criteria are presented. Therefore, multi-objective is a natural requirements from users to provide a set of trade-off solutions that concern about the conflicting and independent quality criteria. E.g., Premium users do not care cost as much as price-sensitive users do, so premium users usually may prefer a composition solution with lowest execution time, rather than one with a relatively lower execution time without exceeding a budget. Therefore, a multi-objective fully automated service composition approach is very demanding for providing a set of solutions due to the following two reasons: first, the preferences of different quality is not clear and hard to determine in advance; second, single-objective optimisation using weighted sum method can not reach solutions in the non-convex regions [41].

Existing research on the automated web service composition mainly concentrates on single objective problems for QoS-aware web service compositions. For example, there is only one solution promoted by a unified QoS ranking score to the users. However, to our best knowledge in multi-objective context, works [56, 93, 106, 107] on service composition problems are only approached by semi-automated methods to handle the conflicting QoS attributes independently, where the workflow structure is assumed to be pre-existing. On the one hand, simultaneously constructing workflows and selecting proper services for optimising multi-objectives is a very challenge work to complete. On the other hand, some constraints on SLA are also employed to some of these approaches to reach the solutions with desirable level. These constrains raised the complexity of absolute Preto priority relation [33]. 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 constrains on SLA. Moreover, the insufficiency of handling only non-functional attributes (i.e., QoS) has given rise to adding semantic matchmaking quality into simultaneous consideration.

Dynamic Semantic Web Service Composition

In a dynamic environment, QoS of the atomic services in service repository is fluctuating over time. Static service composition solution is no longer enough, and requisite actions must be taken if the original composition solution changes in QoS or is not be executable due to any service involved goes offline. Apart from that, newly registered services could also could alter the composition plan as it could significantly contribute to the overall QoS or quality of semantic matchmaking. Therefore, dynamic web service composition is proposed to effectively and efficiently update composition solutions when they are not presented as optimal and/or executable solutions [53].

Most approaches work on effective and efficient methods for service re-selection for each services employed, which do not allow the changes of service composition structure. Apart from that, the cost of initial planning is ignored and separated from the adaption of dynamic environment. Some techniques [5, 8, 43] endeavoured 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 [5] is proposed for supporting the dynamic adaption using ECA (Event Condition Action) rules, which is utilised 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 [4] is proposed to support the adaption. On the other hand, EC-based techniques have been showing their confidence in its behaviour for handling dynamic web service composition to overcome the limitations due to the following reasons: a proper amount of individuals stored could be used for retrieving

an alternative composition solution in the case of failure for saving computation cost in the initial planning stage; the stored individuals could be further evolved while taking changes into account and leads to either changes on a concrete service or composition structure. These two advantages of using EC support the adaption of a dynamic environment. Existing EC-based approaches to web service composition have been studied in static scenarios, rather than dynamic ones. Although some works [28, 56] points out their approaches fit the dynamic problems since the natural features of ACO algorithm (i.e., a continuous optimisation process), there is no dynamic problems defined and further discussed in their papers. Therefore, a lack of research in this field. Given above discussion, it is very advisable to study the effectiveness of EC approaches in a dynamic composition context.

Automated Web Service Composition Based on Preconditions and Effects

Apart from considering the satisfactory inputs and production of outputs, some conditional constraints also determine the executability of services. These conditional constraints lead to multiple possible paths for execution when services are composed together, since inputs and outputs are not everything required for service execution. For example, in the scenario of an online book shopping system [98], services are composed to provide an operation for book shopping. Users expect purchasing outcome (e.g. receipt) returned if books and customer details (e.g. title, author, customer id) are given. In this case, the users may have specific constraints. If the customer has enough money to pay for the book in full amount, then they would like to do so. Otherwise, the customer would like to pay by instalments. 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 are approached through services represented by only inputs and outputs. However, the underlying functional knowledge base of services (i.e., in the form of preconditions and effects) is not covered [70]. On the one hand, a few approaches [7, 13] have been explored to achieve compositions that consider precondition and effects using AI planning, since AI planning ensures both the correctness of functionality and satisfactory of constrains. Meanwhile, Exhaustive methods are always utilised with AI planning for tackling optimisation problems. These methods suffer hugely in terms of efficiency, scalability, and computation cost. On the other hand, EC techniques (i.e., heuristic methods) are considered to be more flexible and more efficient. Given the benefits from both AI planning and EC-based techniques, they are motivated to be collectively explored for automated web service composition based on preconditions and effects.

1.3 Research Goals

The overall goal of this thesis is to *develop new and effective hybrid approaches for comprehensive quality-aware automated semantic web service composition*. More specifically, the focus will be on developing hybrid approaches that could explicitly support a comprehensive quality model that jointly optimises QoS and semantic matchmaking quality using single-objective method, developing multi-objective approaches for optimising the quality criteria that involved in the decision making of composition solutions selection, and developing hybrid composition techniques for dynamic service composition and handling various changes of composition environment, and developing approaches for semantic web service composition, particularly, considering precondition and effects. This research aim to develop a hybridisation of various composition techniques for effectively handling the several service

composition problems discussed above. The research goal described above can be achieved by completing the following set of objectives:

1. **Hybrid approaches to comprehensive quality-aware automated web service composition that simultaneously optimises both QoS and semantic matchmaking quality**. Particularly, we extend existing works on QoS-aware service composition by considering jointly optimising the both quality aspects, which is proposed as a comprehensive quality model. Meanwhile, representations of the composition solutions are the key aspect of the approaches, and they must maintain all the required information for the evaluation. Therefore, we will investigate the following sub-objectives to handle this objective.

   (a) *Propose a comprehensive quality model that addresses QoS and semantic matchmaking quality simultaneously with a desirable balance on both sides.* We aim to establish a quality model with a simple calculation and good performance for the evaluation of our proposed comprehensive quality. Meanwhile, to enable a better evaluation on our approaches, it must support most of existing benchmark datasets, e.g., Web Service Challenge 2009 (WSC09)[42] and OWLS-TC [46].

   (b) *Propose direct and indirect 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, 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. To take advantage of the graph-based and tree-based representations, we aim to propose a tree-like representation representation. In particular, any isomorphic copy in the traditional tree-based representations is removed and labeled with a special symbol q, and insert an edge to the root of the copy. Meanwhile, particular genetic operators are developed without breaking the functionality of symbol q.

   The *indirect representations* do not present the final service composition solutions, they must be decoded to executable service composition. Previous studies have shown their good performances in searching optimal solution for QoS-aware web service composition [23, 24]. However, the decoding process could increase the computation time. Apart from that, the indirect representation potentially increases the searching space, due to the changes of the indirect representation may result in the same solutions as discussed in the work [24]. To overcome these disadvantages, it is advisable to propose more efficient indirect representations.

   (c) *Propose hybrid methods to effectively and efficiently handle the problem for comprehensive quality-aware automated web service composition.* The reasons of utilising hybrid techniques are briefly discussed in the first motivation. Herein, hybrid approaches are suggested to be developed for supporting both the proposed indirect and indirect representations, as well as the comprehensive quality model. In particularly, we aim to propose hybrid heuristics strategies to provide fast convergence of fitness value and avoid being trapped by the local optimal.

2. **Develop multi-objective approaches to optimise the comprehensive quality for fully automated service composition**. In practice, many quality criteria proposed in our comprehensive quality model are often conflicting in natural. Existing works [16,

102, 107, 56, 110, 115] mainly concentrated on semi-automated QoS-aware web service composition. Therefore, a study needs to be carried out not only for better understanding of inherent trade-offs 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 by utilising cutting-edge multi-objective optimisation algorithms (e.g, NSGA-II [27], SPEA2 [118] and MOEA/D [114]). These algorithms are needed for finding a Pareto front of evolved solutions that comprehensively cover users' real interests. Meanwhile, different representations may not perform equally well, so a study on improving the performances of different representations with different fully automated approaches also arouses researchers' interest. Apart from that, SLA consideration needs to be taken into account. It is also necessary to consider customised matchmaking levels to bring the flexibility in meeting different requirements of segmented users (e.g., platinum users, gold users and normal users). The development of this approach can be divided into the following three sub-objectives:

(a) *Develop a EC-based approaches for multi-objective fully automated semantic web service composition*.

Here we develop a multi-objective optimisation approach by using effective multi-objective EC-based algorithms. (e.g, NSGA-II [27], SPEA2 [118] and MOEA/D [114]). We will study different representations and useful modifications of multi-objective EC algorithms simultaneously to improve the effectiveness and efficiency of our service composition system. This sub-objective is also established for 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 optimised independently, since they may represent conflicting interests. 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 overcome the limitation (i.e., semi-automated approaches) in existing works assuming an abstract workflow is given .

(b) *Develop hybrid approaches for multi-objective fully automated semantic web service composition*. Once we achieve the sub-objective 2a, the effectiveness and efficiency are the next focus. The EC-based approaches should be extended by introducing some local search. In particular, some efforts could be made for simultaneously considering the improvements on representations themselves and the combinations with a fast local search. For example, an exhaustive search for the neighbourhood of best the individual within the current population is performed with a relatively higher priority for service selection.

(c) *Develop hybrid approaches for multi-objective fully automated semantic web service composition subject to constraints on SLA and customised matchmaking level*. In real world, satisfaction on given SLA constraints is required in addition to optimising QoS. Therefore, this sub-objective should be further extended to consider some additional constraints on QoS (i.e., multilevel constraints with lower and upper bound for different individual QoS criterion [107]). Meanwhile, to satisfy the customised different semantic matchmaking levels (e.g., exact matchmaking level and less strict matchmaking level), extensive methods are also required to cope with the constraints on the different accepted matchmaking level.

3. **Develop hybrid techniques to support dynamic semantic web service composition effectively**. Objectives 1 and 2 are proposed assuming the environment of service composition is static. In our context, composition environment refers to the registered services in the service repository, non-functional attributes advertised by service providers, and the ontology utilised for describing the resources of web services. On the one hand, existing EC-based approaches are only executed once to generate a composition plan from a given composition task, some factors could significantly impact the execution of the plan. For example, QoS values of services being composed of are fluctuating over time, service chosen at the planning stage may not available to be invoked at the runtime, or newly registered service may need to be considered for reconstructing a better plan. On the other hand, existing non-EC based approaches [67, 85, 94, 108] work on effective and efficient methods for service re-selection for repairing each service employed. Technically, their approaches do not allow the changes of service composition structure. Also, the cost of initialising a composition plan is ignored and separated from their approaches. To effectively handle the two limitations above, three studies are performed as three sub-objectives as follows:

   (a) *Develop EC-based techniques to re-optimise solution candidates for changes in QoS and Ontology.* Traditionally, initial population is created with solution candidates that are further evolved for searching optimal solutions. During the evolutionary computation process, most of service candidates are discarded except the best service candidate identified. Those discarded solution candidates may be promising, since some of them could turn to be alternative best due to the changes in services or ontology. Therefore, instead of discarding the solution candidates, we aim to propose a new and effective EC-based approach to re-optimise these maintained candidates for further use since these candidates preserve both diversity and elitism.

   (b) *Develop hybrid techniques to re-optimise solution candidates for changes in QoS and Ontology.* Once the EC-based techniques to re-optimise solution candidates for changes in QoS and Ontology are achieved, it should be further studied in developing more efficient and effective approach to handle this problem. We aim to propose an adaptive and hybrid approach to this dynamic problem. Our initial idea is to assign a higher priority to a group of services with changes and a lower priority to a group of services without changes, respectively, for considering of service selection based on a local search during the evolutionary process. The higher priority must be adaptively handled with a proper decreasing rate with respect to each service in the first group. We aim to achieve more effective and efficient performance compared to the EC-based approaches in Objective 3a.

   (c) *Develop hybrid techniques for handling service failure and new service registration using updated candidates in the population.* Apart from the changes in the QoS and Ontology, occasionally existing service may fail and/or new service may be registered. For the case of service failure, some methods must be proposed to replace the un-invokable services or update the plan with new services. We aim to propose some approaches using direct representations, where we could either efficiently mutate the solutions candidates partially on un-invokable atomic services, or its involved parent composition components, or effectively re-generate whole solutions using invokable services in the service repository. For the case of new service registration, giving a priority for newly registered services should be properly considered for service selection. We could discard a portion of the current population (e.g 50 percentage), and then replenish population based on

updated services repository.

4. **Develop hybrid approaches for semantic web service compositions based on preconditions and effects. (Optional)** We plan to extend most service composition approaches (i.e., satisfactory on inputs and outputs) to include preconditions and effects. These conditional constrains also necessitates the study of various of composition constructs for automated semantic web service composition, e.g., loop and choice. Therefore, three sub-objectives have been proposed as our targets as follow.

   (a) *Develop EC-based techniques for semantic web service composition based on preconditions and effects*. In the problem stated above, inputs and outputs are everything of web services for handling some web services. An initial task is required to be completed. That is, service composition problem is re-modelled by further considering the preconditions and effects. In particular, we need to establish a general matchmaking mechanism of satisfaction on preconditions and effects. Based on the mechanism, sequence and parallel composition constructs are automatically constructed. We aim to develop EC-based approaches to effectively handle this problem. In particular, new representations are needed to be proposed for coping with the newly modelled problem.

   (b) *Develop hybrid techniques for semantic web service composition based on preconditions and effects*. Once the EC-based techniques to semantic web service composition based on preconditions and effects are proposed. more effective and efficient techniques shall be developed. In particular, we aim to create hybrid techniques that utilise a hybridisation of various techniques for improving the performances of EC-based techniques for semantic web service composition based on preconditions and effects.

   (c) *Develop EC-based techniques for semantic web service composition based on preconditions and effects for supporting loops and choice*. We initially extend the matchmaking mechanism of satisfaction on preconditions and effects to support loops and choice composition constructs. To extensively cope with these two constructs, new and effective representations must be studied. Apart from that, EC-based approaches can 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 optimisation 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., 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 Organisation of Proposal

The remainder of the proposal is organised 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 discusses the preliminary work explores direct and indirect representations for comprehensive quality-aware semantic web service composition using a hybridisation of AI planning techniques and EC-based techniques; Chapter **??** presents a plan detailing this project's intended contributions, a project timeline, and a thesis outline.

Chapter 2

Literature Review

In this chapter, we first introduce the background knowledge of web service composition in Section 2.1, i.e., functional and nonfunctional properties of web service and web service composition in Subsections 2.1.1 and 2.1.2 respectively. Evolutionary Computation techniques are also briefly introduced in Subsection 2.1.3. Followed that Section 2.2 reviews the single-objective service composition for both EC and non-EC based approaches in Subsection 2.2.1. Subsection 2.2.2 reviews existing works in multi-objective approaches and many-objective approaches. Dynamic web service composition is covered in Subsection 2.2.3. Subsection 2.2.4 discussed semantic service composition based on preconditions and postconditions. Lastly, Subsection 2.2.5 summarises some key reviews and limitations in the literature review.

2.1 Background

2.1.1 Web Service

We first introduce the concept of web services and a formal model from [2], where both the functional and non-functional attributes are captured uniformly. Further, A Labelled Transition System [3] is addressed with its abstract and updated model for demonstrating the behaviours of web services in Subsection 2.1.1, which emphasises on side of the functional attributes. After demonstrating these models, we will discuss about the nonfunctional properties of web services in Subsection 2.1.1. Apart from these, we will dicuss three main service discover mechanisms in 2.1.1.

Web services are self-describing modules offering functionalities over the internet. The functionalities of web services are often specified by their functional attributes, which satisfy users' functional requirements and provide mechanisms to allow users to search desired service. Web services are classified into two groups based on their functionalities: *information-providing services* and *world-altering services* [62]. The first type of services expect some data returned by giving inputs or nothing. For example, a service for air velocity transducer reads the wind speed and return the velocity at the time. This service does not require any inputs. On the other side, a service for city weather requires given city name to return the weather information for that specific city. Information-providing services do not produce any side effect to the world. The functionalities of these services are only 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 the first type of services for first three objective. Later on, an extensive study is optimally carried for the second type of services*

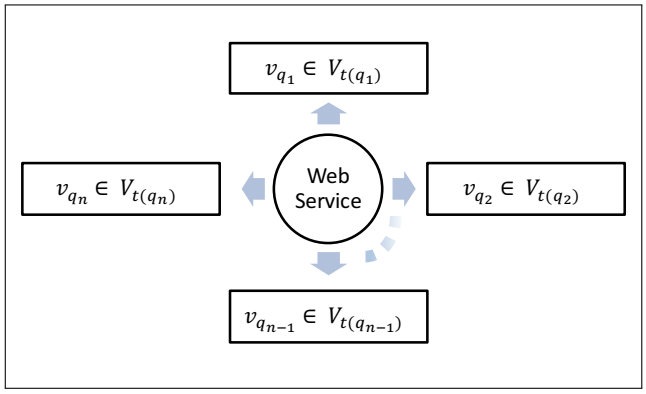


Figure 2.1: Property-Based Web Service Formal Model [2].

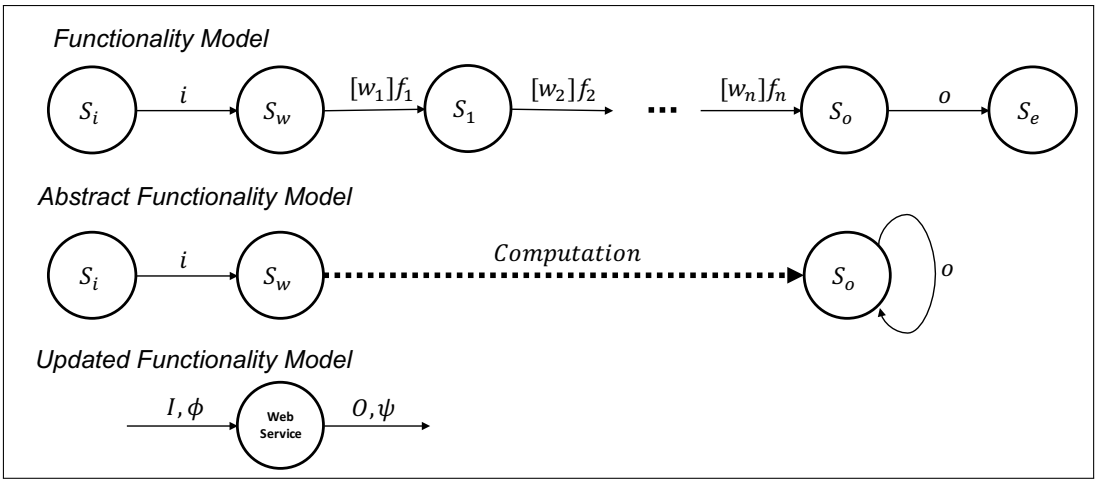


Figure 2.2: The functionality of a Web Service [2].

In realistic scenarios, the non-functional are also important. For example, users may not prefer a service with higher cost with the same functionality provided by another one. As demonstrated above, the functional attributes determine what service really does, while the non-functional attributes often refers to some quality criteria, which is considered for raking services [3]. *A Formal Model of Web Service* is demonstrated here: given a finite set $\wp$ of property types of web services and a finite set ϑ of values sets. Each property type is associated with a value set. We view a Web service as a finite set Q of property instances with each property instance $q \in Q$ being of a property type$t(q) \in \wp$ that is associated with a value $v_q \in \vartheta_{t(q)}$. See Fig. 2.1

Functional Properties of Web services

A Functionality Model is demonstrated for descriping all the functionalities of web service. This mode is represented as a Labelled Transition System (LTS): $L = (S, T, \rightarrow)$ comprises a set S of states, a set T of transition labels and a labeled transition relation $\rightarrow \subseteq S \times T \times S$. This transition system is established to present the actual behaviour of web services, which consists of a series of states. See Fig. 2.2.

- S_i start state includes knowledge available to web service before the web service is invoked by inputs;
- S_w state includes the inputs additional to the knowledge in 1;

- A series of state S_1 to S_n proceed with corresponding actions f_n that only occurred if each related condition $[w_i]$ is approved to be true.
- S_o state contains all the outputs and all the changes performed in the knowledge base.
- S_e is the end state, which is equivalent to S_o, since the knowledge base is not changed.

The first functionality model presented above is not completely demonstrated for the internal actions, as the services provider does not want reveal all the internal functions, and it is not feasible to list a global set of property name. Therefore, *An Abstract Functionality Model* is modelled by eliminating all the intermediate properties. In the abstract model of a web service, the functional properties of the web service could be identified as inputs i, pre-state S_i, outputs o and post-state S_o. These four properties are mapped to a set of input I, preconditions ϕ, a set of outputs O and postcondition φ respectively in third updated functionality model demonstrated below.

In *A Updated functionality Model*. , a set of inputs I is required by a service and a set of output O is returned after the successful execution. Apart from that, the precondition ϕ must be hold in the knowledge base before service is invoked by passing the input I. To enable the interoperability of the functional properties, ontology reasoner is employed to reason about the properties of web services. To distinguish the changes between before and after service execution, these changes are modelled as property instances. For example, inputs and outputs assigned as variable names and further referenced in preconditions and postconditions, which can be distinguished as different instances from the knowledge base. In particular, preconditions is assigned to the description of requirements of inputs using logic formula. Herein the description of the formula considers the following cases that are demonstrated using Planning Domain Description Language (PDDL [31]):

- Conditions on the type of inputs, e.g., the payment of an online shopping website is made by Visa or Cash: $\phi : (Format(payment) = Visa \cup Format(payment) = Cash)$.
- Relationships among inputs, e.g., authoried users are required for an online shopping: $\phi : (Authoried?Useraccount)$.
- Conditions on the value of inputs, e.g., saving account balance has more than 100 dollars: $\phi : (\geq (amounts, savingaccount), 100)$

These preconditions must be hold in the state consistently while inputs are being passed to services. Similarly, the postcondition is restricted to the description of constraints on returned outputs, relationships between inputs and outputs, and changes caused by the service in the knowledge bases.

Nonfunctional Properties of Web services

Apart from the functional properties of web services discussed above, the non-functional properties of web services play an important part in composing services. For example, customers prefer lowest execution cost with highest response time and reliability. According to [113], 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.

Web Service Discovery

To generate service compositions, web service must provide mechanisms for discovery required services. Therefore, service discovery must be one fundamental technique to be considered in all service composition approaches. [1] discussed 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 the classes provided by service semantic annotation in WSMO-Lite language. Therefore, service requesters can use class names to express a goal offering a straightforward discovery from a set of classes. However, classes without clear meaning definition could lead to the issue of incomprehensibility of web service discovery. For example, several classes may declared in either different terms for the describing the same content or same terms for describing different content.

The second service discovery technique does not take classes into account, but consider functional properties of web service to include pre-conditions and post-conditions. In particular, a desired functionality description is defined. A discovery algorithm must be developed to handle a matching for different input, output, precondition and postcondition with associated concepts and relations in the provided domain knowledge base. The key idea of the matchmaking is to check whether services accept all the desired inputs provided bu users and whether the desired outputs is delivered by services. In addition, the matchmaking algorithm also checks for the satisfiability of implications that actual precondition and actual postcondition must imply the desired precondition and desired postcondition respectively. The strength of the second is that it potentially meet the demands of all the comprehensible discovery, while the weakness is a lack of efficiency and scalability.

The third service discovery technique is based on a hybridisation of classification and functionality-based discovery. Classification hierarchy is proposed to achieve automatic semantic reasoning in hierarchical functionality. For example, a functionality class is associated with super classes and sub classes for more general functionality class and more specific functionality class respectively. However, to make a consistency of classification hierarchy, the inputs, outputs, precondition and postcondition of a functionality class must satisfy the conditions that contains all the inputs, outputs, precondition and postcondition 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 classifications based and functionality based approaches. While the classification hierarchy needs to be kept 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 to build up a consistent ontology for classes and their functional attributes. That is not the focus of our research. *In the proposal, we use the second service discovery technique for meeting the comprehensible discovery. That is, different types of ontology reasoning are utilised to approach the matchmaking as a fundamental part of service composition algorithm.*

2.1.2 Web Service Composition

Since one atomic web service could not satisfy or fully satisfy users' complex requirements, web service composition is approached by composing web services together with more sophisticated functionalities to meet the demand. Due to fully human intervention in manual

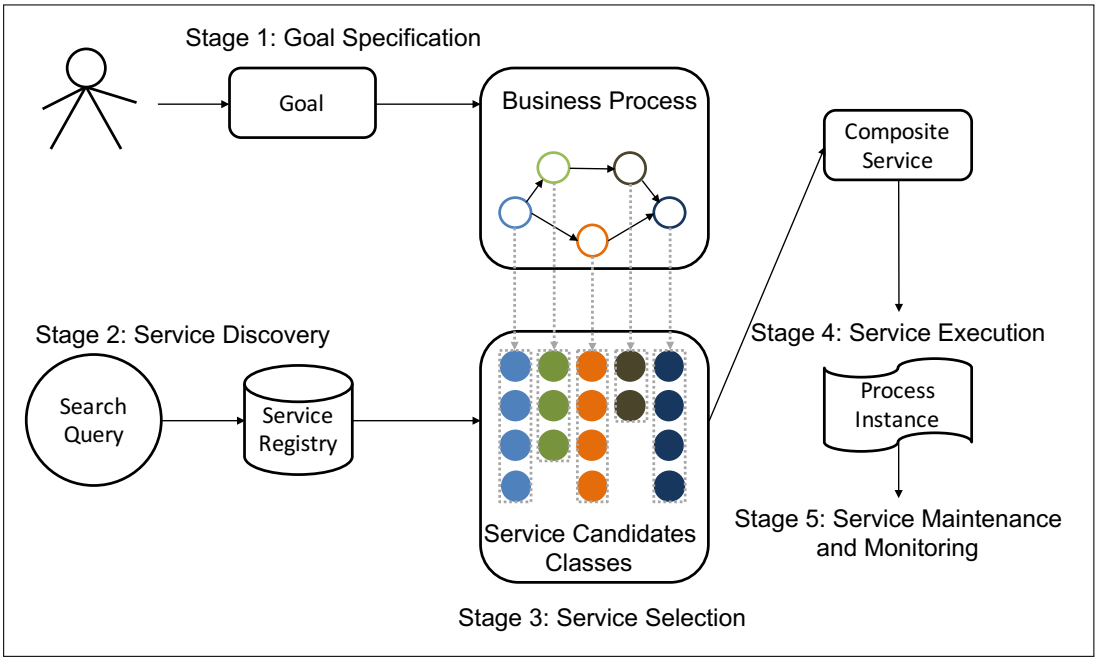


Figure 2.3: Web service composition lifecycle [64].

service composition, so it is very time-consuming and less productive. Therefore, many approaches have been developed to achieve semi-automated or fully automated service composition. The semi-automated service composition is inspired by the business process that required prior knowledge to build up the abstract workflow. This workflow can be decomposed into several functional services slots for proper services being fitted. These steps are further discussed in Subsection 2.1.2. On the other hand, when we are composing services, both semi-automated and fully automated web service composition holds the challenge in the interoperability of services discussed previously. In particular, several problems are simultaneously taken into account. That is I/O matchmaking (i.e., the mechanisms of services for ensuring the interoperability), discovery relevant services to the optimising the quality of service composition, e.g., overall QoS. Consequently, the following concerns are required to consider in generating composition solution.

Web Service Composition Lifecycle

Typical steps in a workflow-based automated Web service composition solution are shown in Fig. 2.3. The detail of the service lifecycle is discussed as follows:

1. *Goal specification:* 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 side (i.e., QoS). This step is achieved by building up an abstract workflow including a series of tasks with clearly defined functionality. Those tasks could be completed by selecting proper concrete services to reach desired QoS.

2. *Service discovery:* Once the goal is clearly specified, concrete web services are to be selected for each task regarding its functional requirement. Often, more than one concrete web service is likely to be found to match the one task. However, those matched web services are always different in QoS. Therefore, web services are classified by the functionality of each task, i.e., inputs and outputs.

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 of each task and overall business process. 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.

The web service lifecycle discussed above is a typical *semi-automated approach*. There is a distinction between semi-automated and fully automated approaches. On the one hand, during the goal specification stage of semi-automated approach, the abstract work is already provided. On the other hand, an abstract workflow is not provided at the stage of goal specification for *fully automated service composition*. Often, fully automated service composition rely on some algorithms (e.g., Graphplan algorithm [10]) to achieve service composition, during which service workflow is gradually built up along with the service discovery and service selection at the same time. In this proposal, I concentrate on developing methods for fully automated service composition since it has been shown the flexibility. Also, herein service discovery and service selection are considered as interrelated tasks that are interleaved with the composition algorithm.

Functional Properties of Web Service Composition

Substantial work [7, 49, 50, 52, 81] on semantic web service composition utilises Description logic (DL) reasoning between input and output parameters of web services for matchmaking. OWL and OWL-S are the most common semantic specifications used currently [76], and they enable automatic selection, composition, and interoperation of Web services to implement complicated composition tasks [61]. However, some matchmaking types (discussed below) may penalise the matchmaking quality and are less preferred by users. Therefore, exploring a effective mechanism for measuring the quality of semantic matchmaking in service composition is a very demanding research area.

Given two concepts a, b in ontology $\mathcal{O}$, four commonly used *Matchmaking types* are often used to describe the level of a match between outputs and inputs [71]:

- *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.

Often, the similarity of two instances of two knowledge representations encoded in the same ontology is also utilised to measure the quality of matchmaking regarding the four matchmaking types discussed above. The work [50] 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 link $sl_{i,j} \doteq \langle S_i, Sim_T(Out\_s_i, In\_s_j), S_j \rangle$ is created between two functional property instances for a input and a output. 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 utilised as the input of another web service. Thus the output requires Extra Description Equation 2.1 to enable proper service composition. As a result, Subsume and Intersection matching type is transferred to be Exact and PlugIn respectively to formulate a robust link.

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

Another method for measuring similarity is discussed in [86]. For concepts a, b in $\mathcal{O}$ the *semantic similarity* $sim(a, b)$ is calculated based on an edge counting method in a taxonomy like WorldNet or Ontology using Eq. (2.2) [86]. This method has the advantages of simple calculation and good performance . In Eq. (2.2), 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.

$$sim(a, b) = \frac{2N_c \cdot e^{-\lambda L/D}}{N_a + N_b} \tag{2.2}$$

For our purposes, λ can be set to 0 as we do not measure the similarities of neighbourhood concepts, the matching type not considered in this paper.

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 [49] *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 all the QoS of involved concrete web services in the solution. The aggregation value of QoS attributes for web services composition varies with respect to different constructs, which reflects how services associated with each other in a service composition [113].

- *Sequence construct*: the composite web service executes each atomic service associated with a sequence construct in a definite sequence order. The aggregation value for total time (T) and total cost (C) is as the sum of time and cost of web services involved respectively. The overall availability and reliability in a sequence construct are calculated by multiplying their corresponding availability and reliability of each web service in probability theory. This construct is shown in Fig. 2.4.

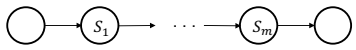
- *Parallel construct*: web services in a parallel construct are executed concurrently. The QoS aggregation value for total cost, availability and reliability are the same as these in sequence construct while the Total time (T) is determined by the most time-consuming path in the composite flow of the solution. This construct is presented in Fig. 2.5.

- *Choice construct*:

- *Loop construct*:

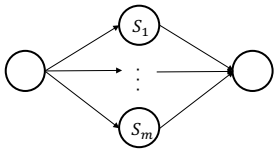
2.1.3 Evolutionary Computation Techniques Overview

Evolution Computing (EC) techniques are founded based on the principles of Darwin natural selection. The nature evolution and selection of individual in a population are automated simulated in EC. In particular, a population of individuals is initialised 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 near-optimal fitness. EC have been shown its promise in solving combinatorial optimisation problems [6]. This is due to its flexibility



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

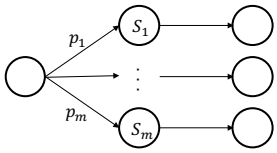
Figure 2.4: Sequence construct and calculation of its QoS properties [112].



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

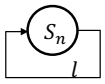
$$C = \sum_{n=1}^{m} c_n \qquad A = \prod_{n=1}^{m} a_n \qquad R = \prod_{n=1}^{m} r_n$$

Figure 2.5: Parallel construct and calculation of its QoS properties [112].



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

Figure 2.6: Parallel construct and calculation of its QoS properties [112].



$$T = \ell \cdot t_n \qquad C = \ell \cdot c_n \qquad A = a_n^{\ell} \qquad R = r_n^{\ell}$$

Figure 2.7: Parallel construct and calculation of its QoS properties [112].

in encoding the problems for the representation and its good performance in many scenarios. In particular, To manage the constraints in the problems, five main methods have been proposed deal with the constraints: coding, penalty functions, repair algorithm, indirect methods of representation and multi-objective optimisation [30]. In the context of service composition. Many EC techniques have been approached for handling optimisation problems for web service composition, such as Genetic Algorithms (GA) [101], Genetic Programming (GP) [45], Particle Swarm Optimisation (PSO) [40], and Clonal Section Algorithm [26]. 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. the tree structure has a terminal set and a function set, where variables, constants and functions are consisted of respectively. Also, 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 the elite individual 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 optimised solution found or a pre-defined number of generation reached.

PSO is one of swarm intelligence (SI) that based on the behaviour of decentralised, self organised system. PSO algorithm is initialised by a group of random particles, which direct or indirect present the solutions. Those particles explores for the optimisation 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 optimisation problems. Clonal Section Algorithm (CSA) is one of AIS for handling optimisation problems, and the principle of utilising Clonal Section Algorithm 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, and a proportion of antibody, rather than the best affinity antibody, are chosen to proliferation. Further more, speed and accuracy of the immune response grow higher and higher after each infection even confronting cross-reactive response. Apart from that, hypermutation and receptor editing contribute to avoiding local optimisation and selecting optimised 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 is to be discussed in two distinct groups: EC-based approaches, which mainly rely on the EC techniques to reach the optimal solutions, and Non-EC based methods, which do not utilise any bio-inspired methods. However, most of these approaches employ a single-objective fitness function for optimising a united QoS score as a simple Additive Weighting (SAW) technique [37].

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 behaviour of human, animals or even T-cells. To cope with different EC algorithms, proper representations are to be designed for direct or indirect 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 Optimisation (PSO), and Clonal Selection Algorithm (CSA).

Genetic Algorithm. GA is a very reliable and powerful technique for solving combinatorial optimisation problems [90]. It has been applied to handle optimisation problems for QoS-aware web service composition [97]. [14] developed a GA-based approach for semi-automated QoS-aware service composition, where an abstract workflow is given. In their work, GA methods are compared to linear integer programming. The experimental finding reveals GA method is preferred when the size of service candidates are increasing. [91] proposed a hybrid approach utilising GA and local search. In particular, a local optimiser is developed and only recalled in the initial population for improving QoS value. This local search contributes to a better overall performance compared with GA-methods without local search. In [49], a semi-automated service composition approach is developed in their paper for optimising the quality of semantic matchmaking and some quality criteria of QoS. In particular, the quality of matchmaking problem is transferred to measure the quality of

semantic links, which is proposed 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 utilised for handling tree-based representations. [83] relies on GP utilising a context-free grammar for population initialisation, and uses a fitness function to penalise 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 [83], [112] 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 adopt a higher crossover probability while trapped in local optimisation. 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. [59] proposes a hybrid approach combining GP and a greedy algorithm. In particular, a set of DAGs that represent valid solutions are initialised 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, [21] proposes a different transformation algorithm to present composition constructs as the functional nodes of trees. On the whole, all these GP-based approaches [59, 83, 21, 112] consistenly 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 [19] with graph-based genetic operators, which is utilised to evolve graph-based representation. 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). Another paper [22] investigated Directed Acyclic Graph with branches using GraphEvol approach [19] to find near-optimal QoS solution in web service composition comparing with GP approach in [20]. The experiment results reveal a significant improvement in execution time while slightly tradeoff in the fitness value. However, the service composition problem for handling branches is not generally formulated, i.e, only works for one choice construct. If more one nested choice constructs, their approach does not work any more.

Particle Swarm Optimisation. PSO is considered to a simple and effective approach for solving combinatorial optimisation problems with few parameters settings [57]. The paper [57] proposed an environmental-aware PSO approach for QoS-aware web service composition. In particular, an improved discrete PSO algorithm is developed for adapt the changes of composition environment (i.e., services) when a same service composition request is called more than one time. The paper [55] proposed a hybrid Genetic Particle Swarm Optimisation Algorithm (GPSA). In their work, GA is employed with only crossover operator to produce new individuals with n_1 iterations while PSO is only utilised 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 optimisation through a mechanism based on two thresholds, which determine the values of n_1 and n_2. However, [55, 57] handles semi-automated service composition problems. On the other hand, [24] proposes a PSO-based fully automated approach to generate a composition graph from a queue. The idea is to translate the particle location into a service queue as an indirect representa-

tion of composition graph, so finding the best fitness of the composite graph is to discover the optimised location of the particle in the search space. [24] proposes a PSO-based fully automated approach to generate a composition graph from an indirect representation, i.e., a service queue. This service queue is mapped to particles' locations. so finding the best fitness of the composite graph is to discover the optimised location of the particle in the search space. In particular, the dimension of the particle is set up as the same number as relevant web services, and the index of services is mapped to the location vectors in a particle and put services in a queue in ascending order, from which a graph is decoded using a forward GraphPlan Algorithm.

Clonal Selection Algorithm. The paper [105] introduces a novel web service composition approach using an immune algorithm for global optimisation considering optimum time under a constraint on cost. As a given abstract graph could be broken into several single pipelines, the optimisation problem is transferred into getting the optimum executing plan for the single pipelines. In pipelines, each involved tasked could be slotted with several alternative web services with QoS values labelled to their edges so that a weighted multistage graph is established for further longest path selection. In the immune algorithm, the service composition problem is encoded using a binary string as an antibody for evaluating the affinity value regarding the antigen (fitness function), and the antibody with low concentration will be selected in a high probability of crossover and mutation for new antibody generation. However, the efficiency of creating the weighted multistage graph would be considered to be less efficient. The paper [77] introduced an immune-inspired web service composition approach combining an enhancing planning graph (EPG) and a clonal selection algorithm to solve optimisation problem considering both semantic link quality and QoS. The EPG model is characterised with action and Layer involved in multiple stages, where each action represents clustered web service, and each layer represents input or output parameters 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 consisting of generating clones from selected antibody, affinity maturation process and replacing low-affinity antibody and re-select antibody to continue all the whole procedure. At last, the approach is proved to reach an optimal solution or a near-optimal solution in the experiments under trip and social event attendance planning domains.

AI Based Approaches

AI Planning techniques have been widely employed for service composition [60, 75]. The main idea of these techniques considers services as actions that are defined with functional properties (i.e., inputs, outputs, preconditions and postconditions) to generate validate service compositions using classic planning algorithms.

Various AI planning approaches [28, 36, 79, 95, 98] have been presented to solve semantic web service composition problems using the Graphplan [10] algorithm. [98] 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 particular, conditional branch structure is also correctly handled. The pitfalls of this approach are procuring only linear sequences of actions, and it is hard to deal with QoS optimisation. In paper [28], service-dependent QoS is modelled and considered 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, and 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 [50, 89] rely some frameworks supported by particular agent programming languages (e.g., Golog [89] and SHOP2 [88]) to composite web services. In [89], a service composition framework supported by Golog. Golog is used to present the generic procedure, and situation calculus and first order language (FOL) are used to describe the properties of services and users' preferences. Therefore, Golog can effectively perform a constant search to reach a terminating situation as a service composition solution.

As summarised here, given the desired solutions generated to meet users' complex requirements, AI planning techniques are considered to be less efficient, and not capable of dealing with optimisation solutions for service compostion (e.g., generate either optimal QoS or number of services) independently. In addition, they may suffer scalability issues when large repositories are given.

Other Approaches

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

Integer Linear Programming (ILP). ILP methods are utilised for achieving web service composition. Generally, an ILP model is created with three inputs provided: a set of decision variables, an objective function and a set of constraints. On the other hand, the outputs are maximised/minimise objective function and values of decision variables. Therefore, ILP is flexible in taking QoS into account, handling constraints for QoS and optimising the objective function for QoS-aware service composition problem. [32] define a zero-one IP model for web service composition based on an abstract service workflow, where services may be different in QoS, but classified into the same classes. [109] formulated web service composition problem based on the model introduced in [32]. Apart from that, compared with the previous work, they simultaneously take both QoS and constraints on QoS into account. However, on the one hand, due to the increase in the number of decision variables, ILP may lead to exponentially increase the complexity and cost in computation [54]. The resulted huge delay is not allowed in the real world scenarios. In addition, if non-linear function is utilised, the scalability is a big problem.

Dynamic Programming Approach. Dynamic programming is an effective method for solving problems, where many repetitions of their break-down subproblems and optimal substructures are presented. In [36], an efficient pruning approach is developed including a forward filtering algorithm for searching task related service candidates, a modified dynamic programming approach for dealing a subproblem of service composition (i.e., a problem on satisfactory of each concept pool of each graph layer), and a backward-search method for searching optimal composition results. This paper [103] forges a problem to solve large-scale service composition efficiently with QoS guarantee, where a dynamic programming algorithm named QDA is developed to for ensuring the optimisation of the composition problem. The problem is optimised by optimising every subproblem based consideration of web service composition with fewest services involved in. In particularly, best-known QoS are recorded and updated for all added web services, and service parameters by the maximum of the path using the traceback depth- first search to derive an execution plan. However, the global best QoS could be never reached since a trade-off in the efficiency of QoS guarantee in solutions.

ER Model-Based Approach (ILP). Most of the small and medium business rely on ER database to process information and data. [104] employs ER model to construct domain ontology and semantic web services. It potentially benefits large groups of organisations.

With regard to the ontology construction, it realises the transformation from ER to OWL-DL, which has maximum expressiveness while retaining computational completeness and decidability. Also, it constructs semantic web service described by OWL-S from ER. Therefore, the semantic web service composition problem is transferred to reason composite service based on a link path between entities in the ER model corresponding to the classes. However, other constructs such as loop and switch constructs can not be effectively expressed in their approach, which demands many further research.

2.2.2 Multi-objective, and Many-objective Composition Approaches

Maximising or minimising a single objective function is a most commonly used way to handle optimising problems in automated web service composition. That is a Simple Additive Weighting (SAW) [37] technique, which presents a utility function for all the individual quality criteria as a whole. This technique optimises 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 trade-offs. To overcome this limitation, a set of objectives corresponding to different independent quality criteria are optimised independently. Consequently, a set of promising solutions that present many quality criteria trade-offs are returned.

Multi-objective approaches

Many multi-objective techniques [56, 115, 110, 107, 102, 16] have been investigated to extensively study QoS-aware web service composition problems. A set of optimised 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. Especially, figuring out solutions are clearly dominating 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 the same execution time and 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, If those 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 employs GA [56], but other EC algorithms are also considered. For GA, [56] employs a service composition model, called MCOOP (i.e., muti-constraint and multi-objective optimal path) as web service composition solution for only a sequence service composition considered in the paper. In this 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. Apart from that, GPDSS is proposed to generate the outputs of Pareto optimal composition paths. In particular, two points crossover and mutation are applied to speed up the astringency of this algorithm. The work [93] investigates a semi-automated approach to SLA-aware web service composition problem. Each linear representation proposed here presents three service composition solutions designed for three group users' categories. The individuals are randomly initialised, evaluated and optimised with objectives from all the possible combinations of throughput, latency, cost and user category. In this work, two multi-objective genetic algorithms: E-MOGA and Extreme-E are developed. E-MOGA is proposed to search a set of solutions that equally distributed in the searching space by the

means of fitness function, where the production of domination value, Manhattan distance to the worst point and sparsity (i.e, Manhattan distance to the closest neighbour individual) is assigned to the feasible individual as fitness value, and SLA violation /domination value is assigned to the infeasible solutions. On the other hand, Extrem-E provide extreme solutions by employing fitness functional, 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 The work [107] combines genetic operators and particle swarm optimisation algorithm together to tackle the multi-objective SLA-aware web service composition problems. The method proposed in the paper is considered to be more effective in considering different scare of cases. It is called as HMDPSO, i.e., hybrid multi-objective discrete particle swarm optimisation. 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 is performed on the *gbest* particle if the proposed swarm diversity indicator is below some value. For the evaluation, the fitness values of individuals are assigned in the same way as the E-MOGA method in [93].

Multi-objective techniques with ACO Generally, ACO simulates foraging behaviours of a group of ants for optimising the traversed foraging path, where the strength of pheromones is taken account for. The work [115] turns the service composition problem into path selection problem for the given abstract workflow with different service candidate set. It 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 and have no overlapped abstract services. This decomposing strategy results in a much smaller length of the execution paths compared to those in the works [**?**]. 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. The paper [96] introduces non-functional attributes of web services to include trust degree according to the execution log. Also, a novel adaptive ant colony optimisation algorithm is proposed to overcome the slow convergence presented from the traditional ant colony optimisation algorithm. In particular, the pheromone strength coefficient is adjusted dynamically to control both the updating and evaporation of pheromone. The experiment results are analysed in an alternative way. That is, the total Pareto solutions are combined from different compared ACO algorithms, then the accurate rate of each algorithm is calculated based on the compared Pareto solutions identified in the total Pareto solutions. The results also show more Pareto solutions found compared to the traditional ACO methods. However, the experiment is only conducted for the evaluation of a small case study, where only a simple abstract workflow is studied.

Many-objective approaches

Herein, more than three objectives in Multi-objective problems (MOPs) are often considered as many-objective problems. Ishibuchi et al. [38] present an analysis of the multi-objective algorithm for handling optimisation problems with more than 3 objectives. However, they address that the searching ability is deteriorating while the number of objectives is increasing, since the non-dominated solution is very large, which make it harder to move solutions towards the Pareto Front.

The work [25] employs 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 examine the techniques to decrease the deterioration, two preference relations proposed by [9] 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 evaluated for studying the five different performance metrics (i.e., hypervolume [116], Generational Distance [92], Spread and Coverage [117], and pseudo Pareto front (i.e, a combination of all non-dominated solutions)), where An empirical evaluation is performed on. The experiment shows NSGA-II with AR outperforms others in both GD and Spread (i.e., more balanced solutions). However, a certain region of Pareto Front is generated by NSGA-II, rather than a wider distribution for the solutions. NSGA-II with MR performs intermediately compared to the other two algorithms. On the whole, this work, for the first time, takes two preference relations into account for solving many-objective service composition problem, and contribute to finding better solutions with many performance metrics.

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 that closed world assumption. Instead, a real world scenario is taken into account. For example, nonfunctional properties of web services may fluctuate over time or services are failed/ newly registered. A few researcher works on the second group to cope with the dynamic composition environment. To address this problem [67], a suitable mechanism for effectively and efficiently handle this problem raise a significant challenge for the practical value.

Dynamic Web Service Composition Approaches For 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 selection services based on the QoS advertised by service providers. However, QoS is fluctuating over the time. Herein the execution instances of a service for the run time indicate the dynamic and real QoS of a service. This dynamic QoS is formally modelled as uncertain QoS model. Based on this model, some studied [100] has been addressed recently. In [100], the QoS model describes the probabilities of different dominating relationships in the instances of all services within the same class. To efficient provide a relatively small set of services for selection that is 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 this paper for efficiently store and retrieve services for service selection.

Reinforcement learning (RL) is one technique of machine learning for solving sequential decision-making problems to maximise some long-term rewards. RL is utilised to deal with how actions are taken in an uncertain environment. In our context, this uncertain environment is related to QoS. In [66], two approaches are proposed based on multi-objective service composition in uncertain and dynamic environment (MORL) combining the advantages in multi-objective optimisation and reinforcement techniques. In particular, we service composition is modelled based on Partially Observable Markov Decision Process (POMDP), and the solutions to services composition are considered to be a set of decision policies, each of which is considered as a procedure of service selection (i.e., a single workflow). Two approaches are introduced in this paper to learn the optimal selection policy.

Dynamic Web Service Composition Approaches For Service Failure

Traditionally, re-selection of failed service is one of widely used approach to re-ensure the desired execution web service composition. The idea of this traditional approach restores many alternative service candidates for each component service involved in the composition solution. If component service confronts failure, the alternatives are used for the replacement. A framework, WS-Replication is introduced in [85], which mainly address service failure using the idea of the traditional method. [108] further introduce a more complicated model that address attributes of not only QoS but also transactionality for more reliable replacement. However, those approaches [85, 108] have a huge cost in computation due to the re-selection mechanisms [67]. To overcome the disadvantage of traditional resection approach, [67] proposes a QoS-aware performance prediction for 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 reselect desired services. To minimising 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) by accuracy in [65]. Those works mainly focus on services failure, and they do not recognise the importance of QoS changes of service in the repository. [94] proposes a method to cluster services that are back up for service failure. This method determines a set of backup services based on their functional properties for service repairing. By employing functional properties, services and their combinable services are identified. The fragmented clusters are formed based on the subsume relationship among their associated combinable services. Also, to merge these fragmented clusters, unified clusters are created by adding virtual services that subsume the fragmented clusters. Therefore, a set of backup services is provided in both the clusters that the failure services involved in and their sub-clusters, from which we could select suitable service.

2.2.4 Semantic Web Service Composition Approaches

Sophisticated real-world applications demand complex requirements in developing service composition. Apart from consideration of both the functional and nonfunctional properties of web services, more complex chaining strategies of semantic services are necessarily arranged to satisfy dependency constraints in the real world. To cope with these complicated requirements, preconditions and postconditions are semantically described using logic formula to enable the complex service composition. any standards have been established for supporting the chaining ability of semantic web services, such as OWL-S. OWL-S accommodates varieties of structural constructs for service composition, namely: Sequence, Unordered, Choice, If-Then-Else, Iterate, Repeat-While, Repeat-Until, Split and "Split + Join" [98]. For example, "Split+Join" calls for the process consisting of the parallel execution of a collection of process components with barrier synchronisation. That is, "Split+Join" doesn't completes until all of its components processes have been completed. Additionally, "Split" and "Split+Join" can be applied for partial synchronisation [98].

Most of the existing service composition approaches do not fully consider services registered in the repository are associated with preconditions and postconditions. Therefore, it is hard to generate service composition solutions with branch structure in a fully automated way. On the one hand, a large number of approaches [] only consider inputs and outputs while composing services, the importance of preconditions and postconditions are ignored since it specifies preconditions that need to be satisfied and postconditions that result from service execution, which causes the effect on the next services. On the other hand, some

approaches consider preconditions and postconditions using classic AI planning methods. The limitation of these approaches lies in the linear sequences of actions.

A generalised semantic web service composition is introduced in [7], preconditions and postconditions are effectively presented in a conditional directed acyclic graph where conditional nodes created with two outgoing edges representing the satisfied and the unsatisfied case at the run time. They filter the solutions based on the trust rate using Centrality Measure of Social Networks to find web service trusted by the customers. Therefore, a semantic web service composition engine is implemented for automated generated conditional composition and OWL-S description used for execution phrase, which minimises the query execution time by pre-processing and incremental updates to new service added or modified. The main limitation of this work is that the loops structure are not considered in their web service representation, and they leave optimisation problems aside.

This paper [99] addresses a critical problem by considering services with preconditions and postconditions. In a real world scenario, this problem can significantly affect service composition solutions. In their paper, an extensive Graphplan technique is proposed to support the representation of the problem with exection effects. In particular, a two-level directed graph, called planning graph (PG) is utilised and comprises of two kinds of nodes (proposition and action ones) and three kinds of edges (precondition-edges, add-edges, and delete-edges). The proposition and action level is alternated between each other in the PG. By utilising this presentation, the branch structure is supported in the service composition solution when some uncertain effects are produced by some services.

2.2.5 Summary and Limitations

An overview of recent related research on web service composition is presented in this chapter. The first related research discussed is **single-objective web service composition approaches**, which mainly works on generating composition solutions with optimal QoS or number of services based on the objective functions. EC-based approaches have been widely used for solving the above problems. 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 the increase in complexity of the problem or leave aside optimisation problem. The key limitation of single-objective web service composition approaches is that the importance of semantic matchmaking quality is ignored, which should be simultaneously taken into account with QoS. To cover this necessity, new representations and quality model must be proposed with effective and efficient service composition methods.

The second related research area discussed is **multi-objective, and many-objective composition approaches**, in which different independent quality criteria of QoS are separately optimised and a set of solutions is produced. These solutions present different tradeoffs among different quality criteria of QoS. Multi-objective composition approaches optimise two or three independently quality criteria, while many-objective composition approaches optimise more than three independently quality criteria. One limitation of multi-objective or many objectives approaches is one conflicting quality criteria (i.e., quality of semantic matchmaking) is not considered. Apart from that, constraints on SLA and quality of semantic matchmaking also raise the interest of researchers. Therefore, it is very interesting and challenging to simultaneously to take these interesting problems into account.

Dynamic web service composition approaches are discussed in the third related research area, where a closed world assumption is dropped. In particular, the composition environment is changing over time, which gives birth to two problems discussed previously: one is related to the changes in QoS in a repository, and another is to the changes in the avail-

ability of services in a repository. Existing dynamic web service composition mainly work on service selection using non-EC techniques. These techniques aim to adjust the changes in the solutions, such as re-selection of changed services. In these existing works, the cost of initial planning is consistently ignored and separated from the adaption of a dynamic environment. Apart from that, some interesting problem are not taken into account, such as the changes of ontology that is utilised to describe the functional properties of services, and new service may register in the services repository.

The last related research area focuses on **semantic Web service composition approaches**, which explore a more complex and realistic service composition. Most of existing semantic webs service composition is merely based on the inputs and outputs of services, Beside these functional properties, precondition and postconditions are occasionally considered when they compose services. However, The support for preconditions and postconditions is not fully investigated in existing works since they potentially bring various of composition constructs into consideration. Achieving semantic service composition consider all these composition constructs is a big challenge and motivates further research in this area.

Chapter 3

Preliminary Work

This chapter presents the highlights of the initial work conducted on the creation of a Web service composition approach that combines a planning algorithm, for candidate creation and modification, with EC techniques, for optimising candidates according to their overall Quality of Service. In particular, two types of direct solution representations are explored in alignment with Objective 1: a tree-based representation, which is compatible with the existing Genetic Programming algorithm but may lead to service duplication problems, and a graph-based representation, which prevents duplication issues but requires a more complex evolutionary algorithm. These two approaches are discussed and compared 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 [113].

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 [71]: 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 [49]. As argued in [49] $plugin$ 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 [51] 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 [86]. This method has the advantages of simple calculation and good performance [86]. 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', $type_e$ and sim_e 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 $\mathcal{SR}$.

We use formal expressions as in [58] 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 [59, 21].

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

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

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:

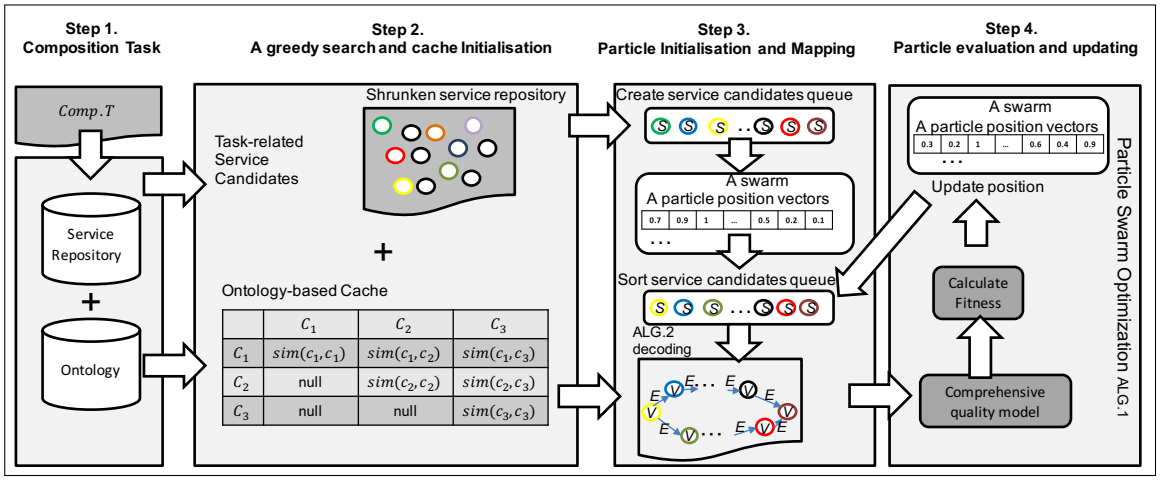


Figure 3.1: An overview of our PSO-based approach to comprehensive quality-aware automated semantic web service composition.

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

Step 2: The composition task is used to discover all task-related service candidates using a greedy search algorithm adopted from [59], 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 [87] 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 *WG* 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 *WG* 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 *WG*. In step 7, the fitness value of the created *WG* is the fitness value of the particle calculated by the comprehensive model discussed in Section **??**.

ALGORITHM 1. Steps of PSO-based service composition technique [24].

```
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 WG from the service queue;
7:          Calculate the WG 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 [10]. This algorithm takes one input, a sorted service queue from step 5 of Algorithm 1. Note that different service queues may lead to different *WGs*. In addition. I_T, O_T and *cache* are also taken as the inputs. Firstly, *Start* and *End* are added to V of *WG* 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, e is created with $type_e$ and sim_e 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 *WG*, these redundancies should be removed before *WG* is returned.

ALGORITHM 2. Create a WG from a sorted service queue.

Input : I_T, O_T, *queue*, *cache*
Output: WG
1: $WG = (V, E)$;
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: $e \leftarrow$ calculate $type_e$, sim_e using *cache*;
11: insert e into E;
12: remove *dangling nodes* and *edges* from WG;
13: **return** WG;

3.3 PSO-based Approach Experiment Study

In this section, we employ a quantitative evaluation approach with a benchmark dataset used in [59, 21], 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 3.6.1.

The parameters for the PSO are chosen from the settings from [87], 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 $type_e$ is set to 0.75 (*plugin* match) according to [49]. 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 [59] 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 [20].

The first column of Table 3.4 shows five tasks from WSC09. The second and third column of Table 3.4 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 reducing the original repository size significantly, which contributes to a reduced searching space. The fourth and fifth column of Table 3.4 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

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

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 [59, 24, 19]. 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, WG_1 and WG_2, which have been obtained as the best service compositions solutions based on the QoS model and on the comprehensive quality model, respectively. Both WGs 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_{e_n} associated to these different edges in WG_1 and WG_2. (Note: $sm_{e_n} = 0.5type_{e_n} + 0.5sim_{e_n}$), where ΔQ reveals the gain (positive ΔQ) or a loss (negative ΔQ) of the listed qualities for our comprehensive quality model. Therefore, we achieve 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 e_4 that is associated with the smallest ΔQ. The e_4 of WG_1 and WG_2 has two different source nodes, $Ser1640238160$

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

and *Ser*947554374, and two the same *End* nodes. *Ser*1640238160 and *Ser*947554374 are services with output parameters *Inst*582785907 and *Inst*795998200 corresponds to two concepts *Con*2037585750 and *Con*103314376 respectively in the given ontology shown in Fig. 3.2 (4). As *Inst*658772240 is a required parameter of *End*, and related to concept *Con*2113572083, *Inst*795998200 is closer to the required output *Inst*658772240 than *Inst*582785907. Therefore, *Ser*947554374 is selected with a better semantic matchmaking quality compared to *Ser*1640238160.

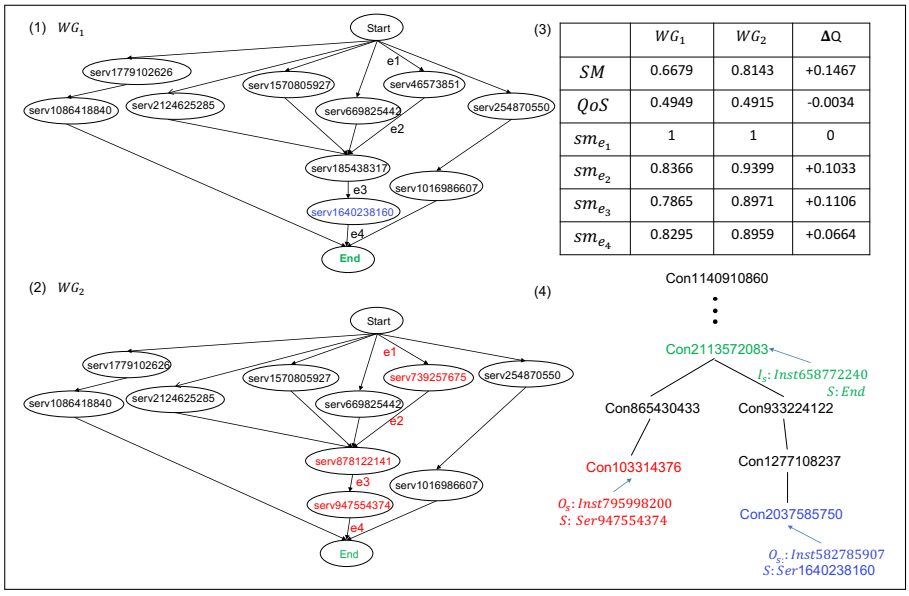


Figure 3.2: An example for the comparison of the best solutions obtained based on the QoS model and on the comprehensive quality model for Task 3.

3.4 Conclusion

In this work, 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. Future works can investigate multi-objective EC techniques to produce a set of composition solutions for the situations when the quality preference is not known.

3.5 GP-based Approach

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 [35, 21, 112]. Finally, we present our GP-based approach with newly designed genetic operation methods.

3.5.1 Tree-like Representation

Let $\mathcal{G} = (V, E)$ be a 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 $\mathcal{PI}_C = \mathcal{PI}_S$.

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*

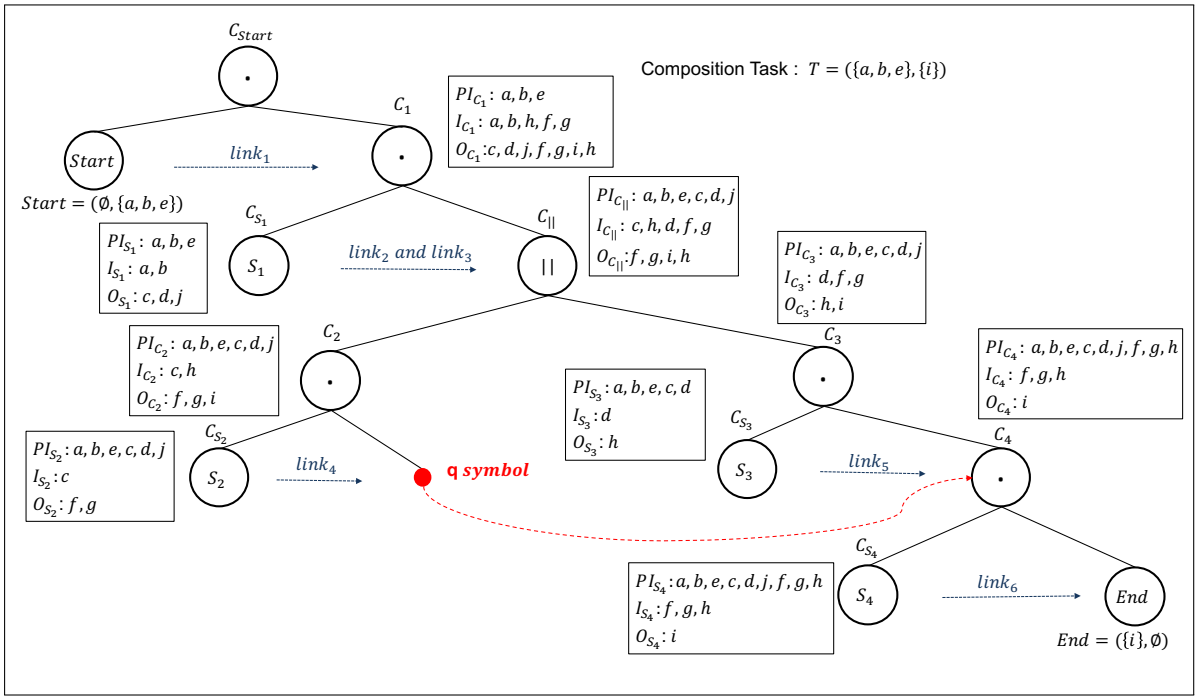


Figure 3.3: Example of a tree-like representation

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 [35, 21, 112]. 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 [45] 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 [59]. 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 [21] 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.

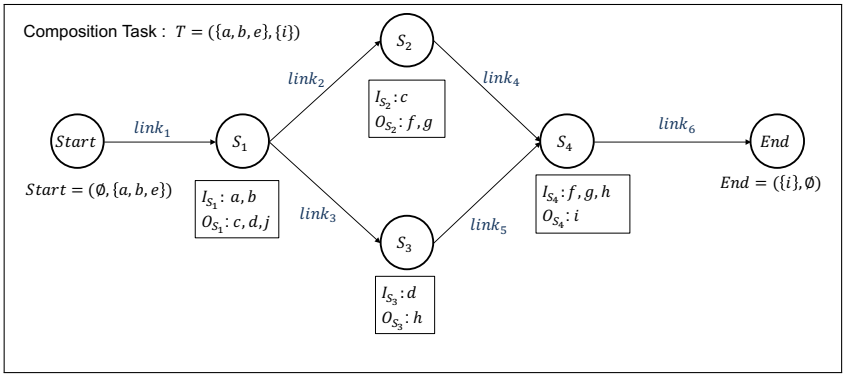


Figure 3.4: Example of a DAG used for transferring it into tree-like representation

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*).

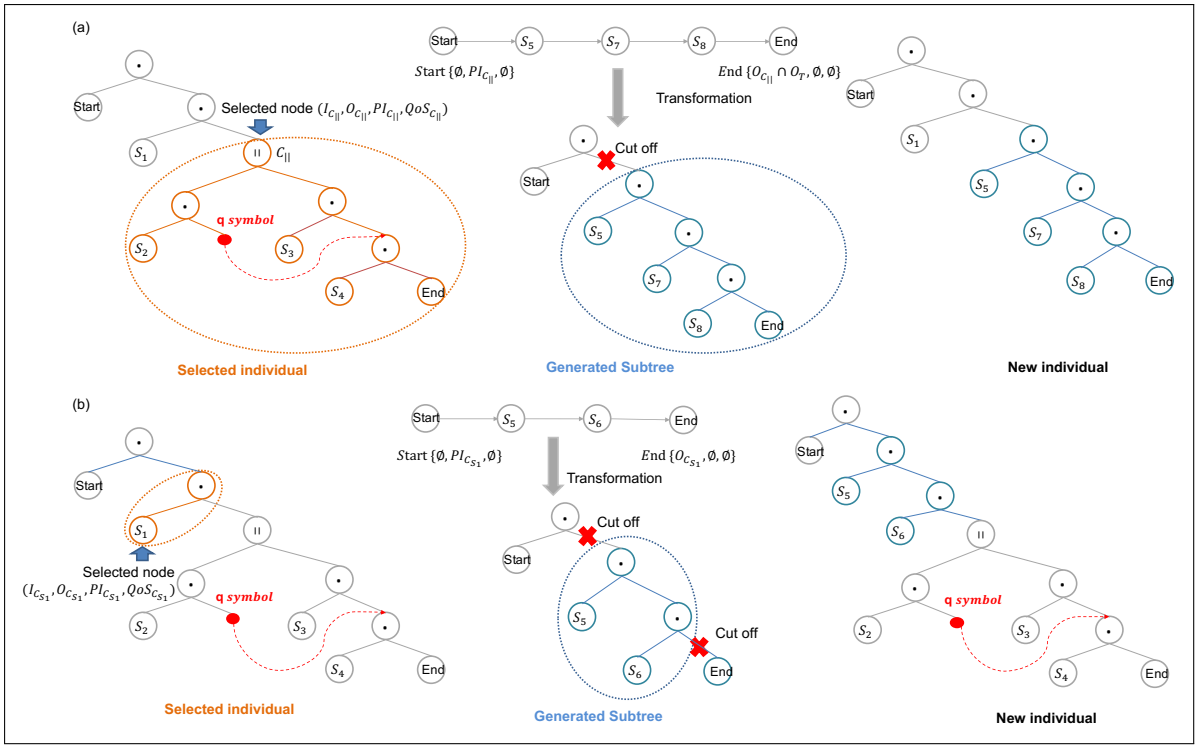


Figure 3.5: Examples of two mutations on terminal and functional nodes

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 individuals. 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 GP-based Approach Experiment Study

We have conducted experiments to evaluate our proposed approach. For our experiments we have used the benchmark datasets originating from OWLS-TC [46] , which have been extended with real-world QoS attributes and five composition tasks [59]. To explore the effectiveness and efficiency of our proposed GP-based approach, we compare it against one recent GP-based approach [59]. 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 [59]. 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 [49].

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 [59] 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 [59]
(Note: the higher the fitness the better)

Task	Our GP-based approach	Ma et al. approach [59]
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 [59] by no more than 1 second, which is acceptable for most real-word scenarios. Hence, in terms of efficiency our approach is comparable to [59].

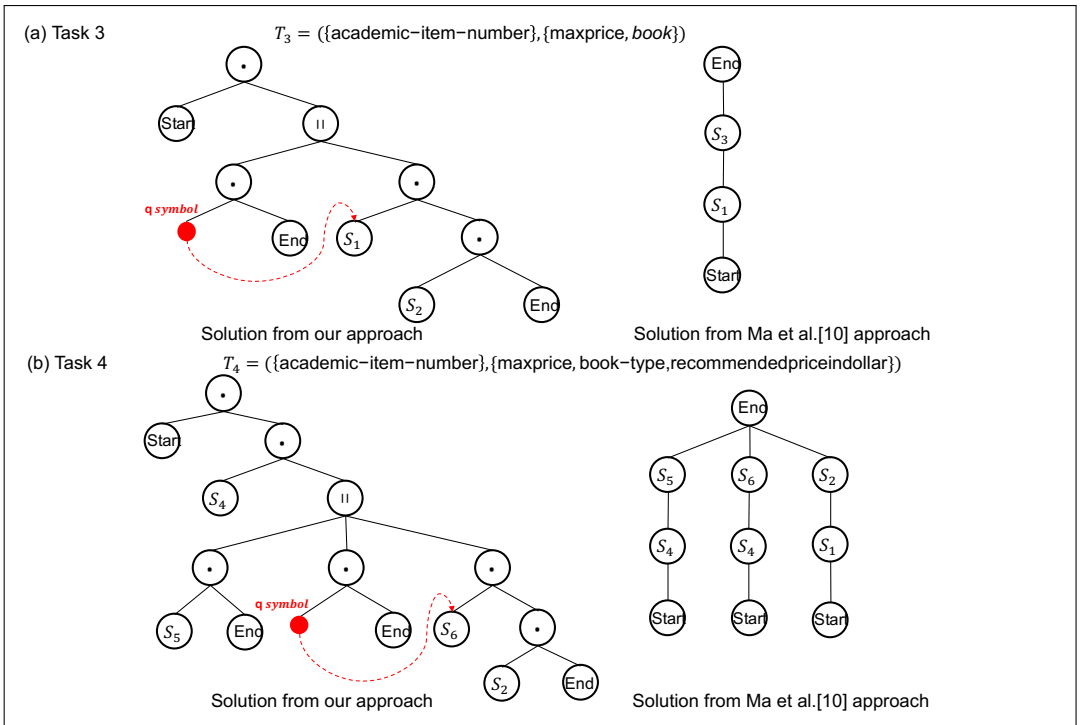
Table 3.5: Mean execution time (in ms) for our approach in comparison to [59] (Note: the shorter the time the better)

Task	Our GP-based approach	Ma et al. approach [59]
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 [59].

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).



(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

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 [59] 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 [59] 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 ouputs only.

3.7 Conclusion

This work introduces an 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 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. Future works can investigate multi-objective EC techniques to produce a set of composition solutions for the situations when the quality preference is not provided by users.

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., JUNGHANS, M., AND NORTON, B. D5. 3.2 second service discovery prototype. *Deliverable D5 3* (2009).

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

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

[11] 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).

[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] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

[19] 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.

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

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

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

[31] FOX, M., AND LONG, D. Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of artificial intelligence research* (2003).

[32] 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.

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

[34] 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.

[35] 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.

[36] 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.

[37] 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).

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

[39] 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.

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

[41] 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.

[42] 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.

[43] 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.

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

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

[46] 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.

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

[48] 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).

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

[50] 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.

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

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

[53] 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.

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

[55] 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.

[56] 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.

[57] 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.

[58] 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.

[59] 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.

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

[61] 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.

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

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

[64] 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.

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

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

[67] 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.

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

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

[70] 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.

[71] 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.

[72] 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.

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

[74] 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.

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

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

[77] 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.

[78] 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.

[79] 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.

[80] 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.

[81] 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.

[82] 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.

[83] 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.

[84] 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.

[85] 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.

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

[87] 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.

[88] 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.

[89] 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.

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

[91] 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.

[92] 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.

[93] 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.

[94] 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.

[95] 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.

[96] 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.

[97] 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.

[98] 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.

[99] 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.

[100] 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.

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

[102] 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.

[103] 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.

[104] 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.

[105] 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.

[106] 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.

[107] 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).

[108] 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.

[109] 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.

[110] 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.

[111] 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.

[112] 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.

[113] 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.

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

[115] 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.

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

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

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