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 Semantic Web Service Composition

Chen Wang

Supervisors: Dr. Hui Ma, Dr. Aaron Chen

July 6, 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 [78, 79]. 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 [9, 76]. One of the most typical implementation of SOA is *web service*, which designated as "modular, self-describing, self-contained applications that are available on the Internet" [21]. Several standards play a significant role in registering, enquiring and grounding web services across the web, such as UDDI [20], WSDL [52] and SOAP [31]. *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 [31]. *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 [113], such as the previously discussed *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 [75], in which many standards have been established. E.g., OWL-S [68], Web Service Modeling Ontology (WSMO) [53], SAWSDL [50], Semantic Web Services Ontology (SWSO) [83]. This dimension bring around some other underlying functionality of services (i.e., precondition and postcondition) that could effect the execution of web services and their composition. The challenge of the discussed interoperability of services gives birth to *Semantic web services composition*, which is distinguished from the traditional service composition (i.e., only syntactic dimension is presented in web services). Due to the resources of semantic web services are described semantically, the quality of semantic matchmaking is always used to measure the goodness of interoperability for 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. 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 extended with *Service Level Agreement (SLA)* [93], i.e., binding constraints on QoS. This extended problem is called *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, service chosen at the planning stage may not available to be invoked at the runtime, or new services are registered in the service repository. Most of importance is *static web service composition* supports the environment change badly because of outdated composition solutions. Therefore, *Dynamic web service composition* become a very demanding research field with a growing interest for providing solutions that adapt to the changing environment. Particularly, some mechanisms are required to be developed to automatically detect the changes or recover from the faults [14]. 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 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. Therefore, fully automation of composition process is introduced in web service composition for less human intervention, less time consumption, and high productivity. The differences in fully automated approach is that an abstract workflow is not 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 [73], which means the composition solution is not likely to be found with reasonable computation times in a large searching space. *Artificial Intelligence (AI) planning-based approaches*, *Evolutionary Computation (EC) techniques* and hybrid techniques are 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, where 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 proposed to solve *QoS-aware web service composition* problems with different designed data structures for representation. That is, different solution representation utilised in different EC-based method have been investigated in QoS-aware web service composition problems, since they could significantly impact on the performance while performing fully automated service composition. In the third approach, a hybrid of AI planning-based approaches and EC-based approaches [24, 63] are proposed to fulfil the correctness in constructing workflows with users' constraints, while the quality of composition solutions (e.g., QoS) are also optimised according to users' requirement. From the literature, hybrid approaches generally outperform single independent 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 summaried here, a few previously addressed challenges for automated web service compositions that are presented by quality of semantic matchmaking, QoS optimisation and dynamic service composition, have not been fully explored in the past.

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.

1.2 Motivations

The motivations of automated semantic web service composition lies in the requirements from five key aspects that simultaneously account for. 1. *Various techniques of hybridisation*. 2. *Quality of service composition*. 3. *Single-objective and 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 are utilised to solve service composition problems, such as AI planning, local searching and EC-based techniques. AI planning is a prominent technique for handling web service composition problems while ensuring the correctness of the solution (i.e., all the inputs of involved services are satisfied) at every step [102]. Local search is a exhaustive search technique for solving optimisation problems. In local search, solutions keep moving to the neighbour solutions with one locally maximised criterion until near-optimal solution found [80]. However, this technique has the shortage in handling global optimisation. On the other hand, EC-based techniques are outstanding at solving combinatorial optimisation problems for finding the globally optimised solutions in large searching space. To take the benefits from various techniques, various techniques of hybridisation allows escaping local optimal easily and improving the rate of convergence rate [90].

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) [67, 82]. 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, leaving aside optimisation problems [80]. On the other hand, some researchers combine AI planning and local search to handle optimisation problems, e.g., a combination of Graphplan [8] and Dijkstras algorithm is proposed by [30] to achieve a correct solution with optimised QoS. On the other hand, 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 [80]. From the techniques discussed above, these techniques are problem-specific for optimising QoS or number of services. To cope the composition problems featured in all the motivations discussed below, new hybrid methods must be proposed to adapt in the problems specified in the thesis, e.g., a hybrid method for automatically optimising both semantic matchmaking quality and QoS.

Hybridisation of Quality of Semantic Matchmaking and Quality of Service

Web service compositions are optimised and ranked on 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 the semantic descriptions are abided with by these information [55]. The quality of the matches (i.e., quality of semantic matchmaking) are one part of the goal for achieving service compositions [54]. 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: Often, 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 the discovery of an atomic service, i.e., one-to-one matching of user requirements to a single service. Some works [5, 10, 70] 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 [22, 26, 40, 63, 88, 114]. In these works, some approaches to QoS-aware web service composition do consider the semantic matchmaking while composing solutions, but do not recognise the importance of semantic matchmaking quality for optimising and ranking 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, new 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 an optimisation for 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 features of 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.

Existing research on the automated web service composition mainly concentrates on single objective problems for QoS-aware web service compositions. I.e., there is only one solution promoted by a unified QoS ranking score to the users. However, in multi-objective context, some works [61, 98, 109, 110] 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. Meanwhile, 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 [36]. 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 impact 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 monitor and update composition solutions when they are outdated [56].

The major techniques endeavoured to update outdated or incorrect compositions allow for dynamic adaptation of the solutions based on implementing variability constructs at the language level. This approach is difficult to manage, and error-prone. Based on and by extending the previous approaches, variability model [2] is proposed to support the adaption. However, most of the EC-based approaches to web service composition have been studied in static scenarios, rather than dynamic ones. Although a lack of research in this field, they have been showing its confidence in its behaviour for handling dynamic web service composition for two reasons as below: a proper amount of population stored could be used for retrieving an alternative composition solution in the case of failure. Also, The stored population could be further evolved while taking changes of QoS into account. This ”self-heal“ process supports the adaption of a dynamic environment. Presented those benefits above, 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 execution 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. E.g., In the scenario of an online book shopping system adapted from [102], services are composed to provide an operation for book shopping. Users expect purchasing outcome (e.g. receipt) returned If book 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 are on their current account balance needs to be handled during the execution of the service composition.

Most of the approaches to automated web service composition are approached through services represented by only inputs and outputs, which are simply utilised to achieve service composition. However, the underlying functional knowledge of services (i.e., prerequisites for execution, and result in some changes, often know as precondition and effects) is not included [77]. On the one hand, many promising approaches [11] 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 optimisation problem in web service composition based on preconditions and effects. These methods always present less efficiency, poor scalability, and intensive computation. 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 fully explored for automated web service composition based on precondition and effects.

1.3 Research Goals

The overall goal of this thesis is to develop new and effective hybrid approaches to comprehensive quality-aware automated semantic web service composition. More specifically, the focus will be on developing hybrid approaches that could explicitly support a proposed comprehensive quality for jointly optimising 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 to dynamic service composition for handling 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. On the other hand, 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)[49] and OWLS-TC [51].

    (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 presenting 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 benefits from both graph-based and tree-based representations, we aim to propose a tree-like representation representation. The *indirect representations* do not present the final service composition solutions, they must be decode to executable service composition. Previous studies have shown their better performances in searching optimal solution for QoS-aware web service composition [25, 26]. 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. 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 optimising the comprehensive quality of service composition**. In practice, many quality criteria proposed in our comprehensive quality are often simultaneously desired and normally conflicting. Existing works [16, 108, 110, 61, 112, 116] mainly concentrate on semi-automated QoS-aware web service composition. Therefore, a study needs to be carried out for not only a better understanding the trade-offs between different objectives (e.g., quality of semantic matchmaking and QoS are naturally considered as two conflicting objectives), but also a good investigation on fully automated approaches utilising those algorithms (e.g, NSGA-II [27], SPEA2 [118] and MOEA/D [115]). These algorithms are needed for finding a Pareto front of evolved solutions that satisfies users' interests. Meanwhile, different representations may not perform equally effective, 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, as well as customised matchmaking levels that needs to be proposed to bring the flexibility in meeting different requirements of segmented users. The development of this approach can be divided into the following three sub-objectives:

   (a) *Develop a fundamental EC-based approaches to multi-objective fully automated comprehensive quality semantic web service composition*.

   Here we develop a fundamental muti-objective optimisation approach using some multi-objective EC-based algorithm. (e.g, NSGA-II [27], SPEA2 [118] and MOEA/D [115]), where different representations and modified multi-objective EC algorithms are simultaneously taken into account for studying both their effectiveness and efficiency. This sub-objective is also established for mainly studying each independent quality criteria from 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 discussed above.

(b) *Develop hybrid approaches to multi-objective fully automated comprehensive quality semantic web service composition*. Since we have achieved the sub-objective 2a, the effectiveness and efficiency are the next focus. The fundamental 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 to multi-objective fully automated comprehensive quality semantic web service composition abiding by constraints on SLA and customised matchmaking level*. In the real world scenario, satisfactory on given SLA constraints are required to be meet other than only optimising QoS. Therefore, this sub-objective should be further extended to considering some additional constraints on QoS (i.e., multilevel constraints with lower and upper bound for different individual QoS criterion [110]). 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 copy with the constraints on the different accepted matchmaking level.

3. **Develop hybrid techniques to handle dynamic semantic web service composition effectively**. The objectives 1 and 2 are proposed assuming the environment of service composition is static. In our context, environments refer to the registered services in the service repository and their non-functional attributes remain constant, as well as the ontology utilised for describing the resources of web services. Since existing approaches are only executed once to generate a composition plan from a given composition task, some factors could significantly impact the generated solution. E.g., 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. To effectively handle these changes, three studies are performed as three sub-objectives as follows:

(a) *Develop fundamental 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 excluding the best service candidate found. Those discarded solutions 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, it would be very motivated to keep the discard solution candidates where both diversity and elitism are preserved. We aim to propose an effective approach to re-optimise these maintained candidates for further use.

(b) *Develop hybrid techniques to re-optimise solution candidates for changes in QoS and Ontology.* Once the fundamental 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 better handling 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.

(c) *Develop hybrid techniques for handling service failure and newly registration using updated candidates in the population.* Apart from the changes in the QoS and Ontology. A dynamic web service composition also tackle the issue of occasionally service fail or new service registration. For the case of service fail, some methods must be proposed to either handle those un-invokable services for replacement. 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 service in the service repository. For the case of new service registration, giving a priority for newly registered services should be properly considered for selecting services. We could discard parts (e.g 50 percetage) of the current population, and then top up the current population from updated services repository.

4. **Develop hybrid approaches to semantic web service compositions based on preconditions and effects. (Optional)** Particularly, we extend most service composition approaches (i.e., satisfactory on inputs and outputs) to include preconditions and effects. These conditional constrains also raise the study of various of composition constructs in the field of automated semantic web service composition, e.g., loop and choice. Therefore, two sub-objectives are proposed to target as follow.

(a) *Develop fundamental techniques to 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. Remodelling service composition problem by additionally considering the preconditions and effects are the initial tasks required to be completed. In particular, we need to establish a fundamental mechanism of satisfactory on preconditions and effects involved in the consideration of only sequence and parallel composition constructs for automated semantic web service composition problem. We aim to develop fundamental approaches to effectively this problem. In particular, representations are needed to be proposed for coping with the newly modelled problem.

(b) *Develop hybrid techniques to semantic web service composition based on preconditions and effects*. Once the fundamental techniques to semantic web service composition based on preconditions and effects are proposed. It followed that more effective and efficient works. In particular, we aim to utilise the motivated hybridisation of various techniques for improving the performances of existing proposed methods.

(c) *Develop hybrid techniques to semantic web service composition based on preconditions and effects for supporting loops and choice*. We initially extend the fundamental mechanism of satisfactory on preconditions and effects to include loops and choice. To extensively cope with these two constructs, new effective representations are required to be studied. Apart from that, hybrid approaches are aimed to developed to efficiently solving fully automated comprehensive quality-aware semantic web service composition 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 the first objective of this research, but the earlier works focus on static web service composition using single-objective optimisation technique, the results show indirect representation shows its promise in solving our service composition problem. The following are the publications made from the preliminary studies:

- Wang, C., Ma, H., Chen, A., Hartmann, S.: Comprehensive quality-aware automated semantic web service composition. In: Australasian Joint Conference on Artificial Intelligence (To appear)

1.5 Organisation of Proposal

The remainder of the proposal is organised as follows: Chapter **??** provides a fundamental definition of the web service composition problem and performs a literature review covering a range of works in this field; Chapter **??** discusses the preliminary work explores direct and indirect representations for comprehensive quality-aware semantic web service composition using the 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 of web service composition in Section **??**. Followed that Section **??** discuss the single-objective service composition for both EC and non-EC based approaches. Section **??** reviews existing works in muti-objective approaches and many-objective approaches. Dynamic web service composition is covered in Section **??**. Section **??** discussed most AI planning-based approaches to semantic service composition. Lastly, Section **??** summarises some critical points discussed in this chapter as well as some inadequacies or limitations in the literature review.

2.1 An Overview of Web Service Composition

At its basic level, a Web service composition is the connection of several atomic services in different configurations in order to reach a result, given that there are multiple services offering the same functionality. They key aspect of compositions is that, in order to achieve the desired result, atomic services must all be executed in a particular order, forming a workflow of tasks. A workflow-based Web service composition approach can usually be decomposed into a series of steps, reflecting the process required to produce a solution [73]. These steps are shown in Figure 2.1 and discussed below:

1. *Goal specification:* The initial step in a Web service composition is to gather the user's goal for the solution to be produced. This is typically done through the generation of an abstract workflow that records the desired data flow and functionality details. This workflow is generated by the user and is referred to as *abstract*, since it contains a series of tasks that can be accomplished by employing a number of different existing Web service implementations. In addition to this workflow, the QoS requirements are determined based on the user's preferences.

2. *Service discovery:* Once an abstract workflow and a set of preferences have been provided, the next step is to discover candidate concrete services that are functionally and non-functionally suitable to fill the workflow slots in a service repository. The focus at this stage is to find candidates that provide the functionality required to fulfil the tasks, regardless of their quality levels.

3. *Service selection:* After pools of candidate services have been identified for each workflow slot, a technique is employed to determine which discovered services best fulfil each slot. The result of this process is the creation of a concrete Web service composition.

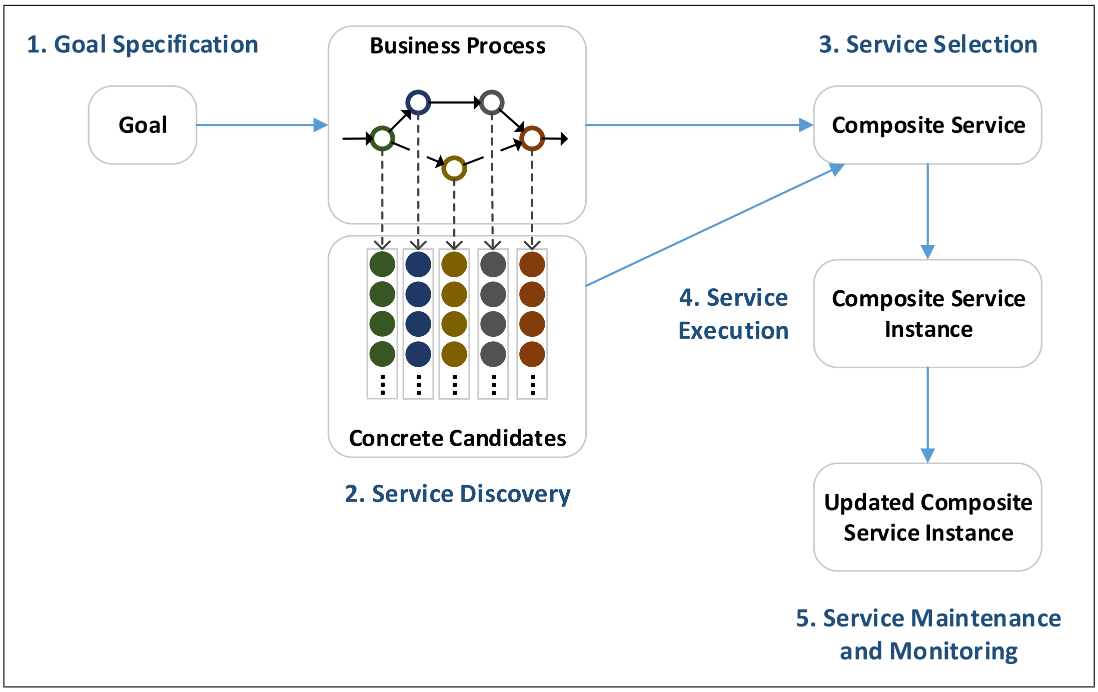

Figure 2.1: Typical steps in a workflow-based automated Web service composition solution [73].

4. *Service execution:* The creation of the composition is followed by the execution of an instance of this composed Web service.

5. *Service maintenance and monitoring:* During execution, the created instance is constantly monitored for failures and/or changes to the composing atomic services, and corrective actions are dynamically carried out as necessary.

It is important to draw a distinction between semi-automated and fully automated composition approaches [89]. The steps discussed above are typical of a **semi-automated approach**, where an astract workflow is provided in the goal specification stage and the composition algorithm is only required to complete the abstract slots of this workflow. In a **fully automated approach**, on the other hand, an abstract workflow is not provided during the goal specification stage, and instead is calculated at the same time that services are selected based on user preferences such as the desired overall composition inputs and outputs. Consequently, service discovery may at times be also executed in tandem with the selection stage. Fully automated approaches have been shown to be more flexible than approaches with fixed abstract workflows (i.e. semi-automated approaches) with regards to solution optimisation [23], thus they are the focus of this project. It must also be noted that this work is focused on exploring new techniques to performing service selection, but not service discovery, maintenance, and monitoring.

Besides identifying the atomic Web services most suitable to fulfilling key composition tasks, the selection process often takes additional user constraints into account. A survey of the literature in the area shows that two types of contraints are commonly taken into account. The first group comprises the creation of service compositions that are optimised according to the Quality of Service (QoS) constraints on its constituent atomic services, where QoS attributes may be thought of as features that indicate the quality of a given Web service, such as the time it requires to return a response and its financial cost of execution [23]. The majority of works in this area use maximising and minimising functions for the different

QoS attributes, meaning that they attempt to obtain solutions with the best possible qualities without any thresholds [104, 80, 111, 35, 116, 101, 73, 46, 57]. However, there also exist approaches focused on what may be referred to as a Service Level Agreement (SLA)-aware optimisation, which is solutions must meet certain predetermined QoS thresholds in order to be considered valid (e.g. the financial cost of each service used in a composition must not exceed 50 dollars) [111, 110, 16, 6].

The second group comprises the creation of service compositions that have multiple execution branches, indicating user preferences at runtime [102, 12, 45, 66, 7]. For example, the output of a service *A* determines which service to execute next; if the output is greater than a certain threshold, then service *B* should be executed; otherwise, service *C* should be executed instead. These preferences are expressed logically as conditions, and affect the workflow used to establish the connections between services rather than the individual services themselves. These two types of user constraints are discussed in more detail in subsequent Sections in this chapter.

2.1.1 Web Service Composition in Practice

The research material published on Web service composition is highly theoretical and frequently employs layers of abstractions and simplifications intended to make the problem at hand manageable. However, it is also important to investigate how these ideas fit into practice, which is the objective of this Subsection. As mentioned earlier, while significant amounts of work are being performed in the field of automated Web service composition, these approaches ultimately remain on the theoretical level due to difficulties that have not yet been fully addressed. On the one hand, a study [62] has shown that the composition of services is fraught with issues. A central problem is that there are discrepancies between the concepts used by different services: the schemas used differ from each other, even if the services handle exactly the same domain. Another problem is the existence of services that produce too much data, low-quality data, or that incur too much latency. These characteristics may slow down the execution of the composition and even require human intervention, defeating automation efforts.

On the other hand, an automated framework for Web service compositions which can be applied to real services has already been proposed [65]. The functionality of this framework was demonstrated by creating a composition that uses the Amazon Virtual Cart service, the Amazon Book Search service and a bank's Point of Sale Web service. The authors of the framework point out that it is very challenging to find service documentation that is comprehensive enough, and that functionality details had to be identified from natural language explanations intended for developers who are working manually. Additionally, the specification of control flow requirements must be performed manually using a logic programming language. Despite these difficulties, the evaluation of this framework shows that a composition can be found within seconds when using it as opposed to an estimated 20 hours of manual development, showing that research on Web service composition provides some immediate benefits.

These two works provide opposing views on the viability of automated Web service composition, however a more recent approach [72] presents the interesting middle ground of user-centered design. This systems combines manual and automated techniques, providing a browsing tool that allows users to explore the repository and gain some understanding of the offered QoS value ranges of services before having to write any QoS requirements, as users may often be unaware of what a reasonable QoS value would be for a given dataset. Users are also supported through the selection process by utilising a UI that diminishes their cognitive overload, and helps them express their requirements using a standardised language. In addition to these tools, this approach also proposes a clustering algorithm that groups service candidates according to their range of QoS values, with the objective of providing selection options when users have fuzzy (i.e. soft) requirements. The strength of this approach is that it does not undervalue human intervention in the composition process, instead providing tools that empower system users. As Web service composition always requires some degree of user involvement, at the very least when setting composition requirements, this type of approach may prove to become an increasingly popular composition solution.

2.2 Single-Objective QoS-Aware Composition Approaches

This section discusses composition approaches that optimise solutions according to their overall QoS. These approaches can be divided into biologically inspired methods, which use Evolutionary Computation to reach their goal, ang general optimisation approaches that do not turn to evolutionary techniques. These two groups are quite distinct, but they have the commonality of using an objective function as the guide with which to measure the quality of the candidate solutions.

2.2.1 Biologically-Inspired Approaches

Biologically-inspired Web service composition approaches rely on evolutionary computation algorithms, which implement their search strategies by drawing inspiration from nature, namely the behaviour of social animals such as bees, fish, and ants. An important distinction between these different bio-inspired approaches is in their representations of the composition problem. These varying representations are discussed throughout this Subsection, and visually summarised in figure 2.2. **Genetic Algorithms (GA)** are a popular choice for tackling combinatorial optimisation problems, and thus have been widely applied to the problem of Web service composition [101]. The encoding scheme for a composition is commonly done as a vector of integers, where each integer corresponds to a candidate Web service, even though some authors have attempted to use matrix representations that also include semantic information about services. A population of candidates is evolved for several generations using generic operators, typically crossover and mutation: in crossover, equivalent sections of the vectors in two distinct candidates are swapped; in mutation, a section of one candidate's crossover is modified at random in order to introduce some genetic diversity. An observed problem with the GA technique is that it tends to prematurely converge to solutions, thus preventing the exploration of further possibilities. **Particle Swarm Optimisation (PSO)** bears similarities with GA, also relying a on vector representation for candidates. However, instead of employing genetic operators to carry out the search process, PSO uses the concept of position updates to move candidate particles across the search space. As PSO may also present the problem of not fully optimising solutions (i.e. converging prematurely), hybrid approaches have been attempted to improve its efficiency and optimisation power [101]. A key limitation of both GA and PSO is in the underlying vector representation used by candidates, since it makes it very challenging to encode workflow information and thus perform any type of fully automated Web service composition.

Ant Colony Optimisation (ACO) has also been proposed as a solution to QoS-aware Web service composition [116]. ACO is particularly suitable to a directed acyclic graph (DAG) representation, in other words, the workflow composition representation commonly used in the field. In ACO, the Web service workflow is built to be traversed by a group of ants (agents). At each fork in the graph, the ants choose which path to follow based on probabilities that take into account the strength of the pheromones left by other ants,

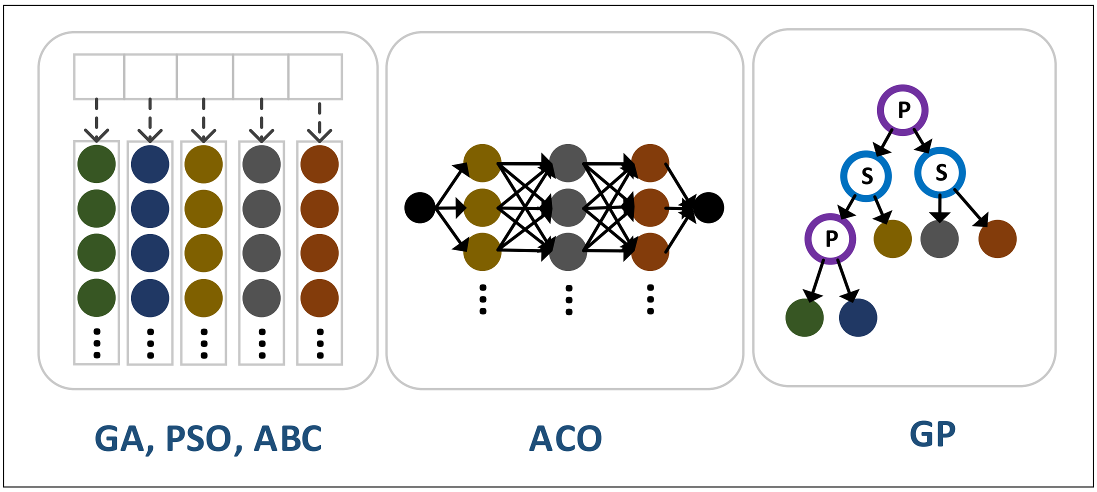

Figure 2.2: The different problem representations employed by biologically-inspired Web service composition approaches.

and also a heuristic function for that particular graph. The pheromones left by the ants are updated after all ants have toured through the graph once, with paths of higher fitness resulting in a larger pheromone increment for the edges of those paths. Meanwhile, the pheromone level of all edges gradually decreases (i.e. evaporates) after each tour of the ants. The graph representation in this technique follows the abstract workflow idea, with a pool of concrete Web services associated to each abstract Web service. Each pool of candidates is a layer that is fully connected to the layers of any following abstract services, so that an optimal path can be chosen from the edges laid out. For each concrete Web service, the heuristic factor is calculated based on its QoS values, and the fitness function also measures QoS attributes. As with GA and PSO, this representation also amounts to the idea of semi-automated Web service composition, even though in this case the encoding of workflow information leading to a fully-automated approach would seem to be trivial.

The works in [116] and [104] apply the **Artificial Bee Colony (ABC)** algorithm to Web service composition. The ABC algorithm simulates the behaviour of bees search for food sources. The position of food corresponds to candidate solutions in the search space, encoded as a service vector, and there are three types of bees dedicated to searching: *Employed bees* exploit the neighbourhood of a single food source already found; *Onlooker bees* exploit the neighbourhood of different food sources depending on the dance behaviour displayed by employed bees; *Scout bees* are the bees sent to random food sources after the neighbourhood they were previously exploiting does not present any food sources that are better than the original. The roles of the bees change according to the colony's needs, which is a feature unique to this algorithm. One of the issues with this approach, as pointed out by the authors, is the search space it explores. The problem with the typical search space for Web service composition is that it is organised based on the proximity of the components included into the composition. For example, given two adjacent solutions a and b in the search space, a will only have one service that is different from all the services included in b. Despite being neighbours, however, the fitness values of a and b may be radically different, and as the optimisation occurs according to a fitness function, this means that the search space is not entirely continuous. These works modify the traditional ABC algorithm to take this problem into account, filtering the neighbourhood of each solution during the search and excluding radically different neighbours from consideration.

Genetic Programming (GP) is, from this project's perspective, possibly the most interesting technique for the problem of Web service composition. That is because GP, unlike the previously discussed techniques, is capable of supporting fully automated Web service composition. In GP approaches [3, 91], workflow constructs are typically represented as the GP tree's non-terminal tree nodes while atomic Web service are represented as the terminal nodes. In this context, workflow constructs represent the output-input connections between two services. For example, if two services are sequentially connected, so that output of service A is used as the input of service B, this would be represented by a sequence workflow construct having A as the left child and B as the right one. The initial population may be created randomly, in which case the initial compositions represented in that generation are very unlikely to be executable due to their mismatched inputs and outputs, or it may be created using an algorithm that restricts the tree structure configurations allowed in the tree to feasible solutions only. A fitness function is employed for the QoS optimisation of candidates, and the genetic operators employed for this evolutionary process are crossover, where two subtrees from two individuals are randomly selected and swapped, and mutation, where a subtree for an individual is replaced with a randomly generated substitute. One of the difficulties of tackling the problem of Web service composition using GP is that it does not intrinsically support the use of constraints [19], meaning that even if all candidates in a population meet the feasibility constraints, there is no guarantee that subsequent generations will maintain conformance to them. The approaches discussed above handle this problem in one of two ways: by *indirect constraint handling*, where feasibility constraints are incorporated into the fitness function so that the optimal function value reflects the satisfaction of all constraints, or by *direct constraint handling*, where the basic GP algorithm is adapted at the initialisation and genetic operation stages to ensure that the feasibility constraints are met. Indeed, the tree representation of an underlying workflow composition may pose difficulties whenever constraint verification is necessary.

Graph Variations of GP

Genetic programming using candidates constructed as graphs (instead of trees) would be ideal for the problem of Web service composition, since dependencies between services could be intuitively encoded. Even though variations of GP with graph candidates do exist, they have not been employed in this domain, therefore the focus of this section is on the techniques and not on the composition problem. Galvan-Lopez [33] proposes a general-purpose EC technique that is modification of the usual GP tree reprentation, allowing the creation of graphs. The key extension proposed here is the addition of pointer functions to certain non-terminal nodes, meaning that they can have connections to nodes in independent subtrees. Program inputs are provided as terminal nodes (similarly to the output-as-root representation discussed earlier), but instead of having a single output location at the root node, they may have other output locations as well, inserted as necessary amongst the non-terminal nodes of the tree. Since the overall tree structure is maintained, it becomes possible to perform the crossover and mutation operations similarly to their implementation in the case of a simple tree. The main difference is that any pointers present in the original tree may not be modified when these operations are applied. In order to ensure that the number of outputs in a tree remains correct throughout these genetic operations, each node in a tree is classified beforehand to establish which ones may be selected for the crossover operation. This representation makes it easier to evolve graphs, however it does not support strongly-typed GP, which is of interest to our research. In addition, each tree is still required to have a single root, meaning that there are connections between the different output nodes. In our problem domain, this is a hindrance because the output nodes must be completely independent from each other.

Miller and Thomson [71] present Cartesian Genetic Programming (CGP), a popular technique for evolving graph structures. The simplicity of SGP lies in the fact that it can represent the genotype of candidates as a string of fixed length, meaning that crossover and mutation operations are trivial to implement provided that they obey some simple constraints. The core idea of CGP is to create a two-dimensional array of programmable nodes of a predefined size. Each node has a predefined number of outputs, and the overall array has a predefined number of inputs and outputs. Then, as this structure is evolved, the functions inside of each node can be reprogrammed, and so can the inputs those nodes require. From that point onward, the structure can be optimised according to the algorithm's fitness function. One important observation is that CGP may have unexpressed genes, meaning that not all the nodes in the two-dimensional array are necessarily components of the final answer. One limitation of this approach is that CGP does not easily handle strongly-typed GP, thus restricting the range of problems it can be applied to. Another problem is that it requires predefined numbers of nodes and node inputs/outputs, making it difficult to represent compositions with varying numbers of services and service inputs/outputs.

Mabu, Hirasawa and Hu [64] introduce a technique for the evolution of graph-based candidates, called Genetic Network Programming (GNP). GNP has a fixed number of nodes within its structure, categorised either as processing nodes (responsible for processing data) or as judgment nodes (perform conditional branching decisions), and works by evolving the connections between these fixed nodes. Connections are represented in a linear gene structure, and the number of outgoing connections from a node is dependent on the type of that node: processing nodes have a single outgoing connection, while judging nodes have more than one (depending on the number of branches desired). Because of the linear representation of connections between nodes, the genetic operators employed for the evolutionary process are quite simple. The mutation operator randomly chooses the destination of a node's outgoing connection; the crossover operator swaps two nodes with the same label from two different solutions, taking their outgoing edges with them. Li, Yang and Hirasawa [59] extend the basic GNP idea by using the Artificial Bee Colony (ABC) approach to evolve candidates. While these approaches present the advantage of simple genetic operations, the number of nodes and outgoing edges per node must be fixed throughout the evolutionary process, meaning that it suffers from the same limitations discussed above.

Poli [84] presents an EC algorithm where solutions are represented directly as graphs. This graph is mapped to a grid, where each row represents a layer of the graph, and columns can be used freely to accommodate the nodes in a layer. By using this representation, it is possible to perform relatively simple mutation and crossover operations to a graph without compromising its syntactic correctness. For the crossover operation, a crossover point (node) is selected in each candidate, and all the ancestors of that node are identified. These two subgraphs of ancestors are then swapped, a process that may overwrite nodes from the other graph but that maintains any original edges. Whenever this swap causes parts of the subgraphs to be placed outside of the grid, these nodes are "wrapped around" their respective rows, meaning that they are moved to the other extreme of the row. For the mutation operator, an existing subgraph is selected and replaced with a newly created one, using the same general mechanism as the crossover. Once again, while this approach makes the implementation of genetic operations simple, it does not allow for correctness constraints (i.e. input-ouput connections) to be maintained. Additionally, this representation assumes that the duplication of nodes is acceptable, which may pose some problems in the domain of Web service composition.

Kantschik and Banzhaf [44] propose linear-graph GP, a structure that allows programs with multiple execution paths to be optmimised using evolutionary computing. This approach is an extension of the simple linear program representation where each element of a vector contains one program instruction. In this representation, a program graph contains several nodes, and each of these nodes contains a portion of a linear program that may be executed according to preceding branching conditions. Within each of these nodes, there exists also a branching node that is responsible for determining which outgoing edge to follow (i.e. which execution path to choose after executing the current node). The branching functions contained in each of these nodes may read the values of variables manipulated in the linear program portion that is located in the same graph node. The mutation operator in this approach may alter an individual entry within a linear program vector, the branching function within a branching node, or the number of outgoing edges of a given graph node; the crossover operator may exchange individual entries within linear vectors of two candidates, or exchange a group of contiguous graph nodes between two graphs. Despite allowing branching constructs, this approach is not useful to our research because non-sequential relationships between different Web services cannot be encoded into linear representations.

The works in [37, 13, 74] present a graph-based genetic algorithm that is used to evolve representations of molecules. Atoms are represented as nodes, and their bonds as edges. Two types of genetic operations are supported: mutation, which can be the appending or removing of a node and its connecting bonds, and crossover, where edges are removed from each candidate until each graph is divided into two disconnected subgraphs that are then reconnected to create new child candidates. The reason why these genetic operators can be used without compromising the strucuture of the molecule is that the only restriction when creating a new connection is the valence of a given atom (i.e. the number of bonds it can make), but bonds do not need to be directed edges and cyclic structures are allowed. In the Web service composition domain, however, the need for additional restrictions means that these genetic operators are no longer suitable.

2.2.2 Other Optimised Composition Approaches

Other methods exist for producing optimised Web service composition results in addition to biologically inspired approaches, and a subset of them is presented in this section. **Tabu search** [38] is a combinatorial optimisation strategy for identifying an optimised solution amongst a group of candidates, typically in problems where exhaustive search is prohibitively expensive. In Tabu search, an objective function (either linear or nonlinear) is used to measure the goodness of solutions, encouraging solutions with the least penalty (i.e. optimal solutions). Then, a range of moves that lead from one candidate solution to another is defined. For a particular candidate solution, there is a set of moves that can be applied to it, and this is known as the neighbourhood function. One of the biggest advantages of Tabu search is that, unlike the hill climbing technique, it can avoid being stuck at local optima when searching. The technique keeps track of a set of tabu moves, which are moves that violate a given collection of tabu conditions. The objective of having a tabu set is to prevent the search from reaching solutions whose best next move has already been visited (i.e. prevent cyclic search moves). Due to its relative simplicity and its flexibility of implementation, tabu search has a wide range of practical applications for combinatorial optimisation problems. For example, a technique that combines Tabu search and a hybrid Genetic Algorithm (GA) implementation has been proposed recently [80]. For the Tabu search component of the technique, a move is defined as a change in one of the concrete services used as the solution, and the Tabu set is a fixed-size set of the latest n solutions visited. For the hybrid GA, the same basic idea of the traditional GA is applied (set candidate sizes, two-point crossover, mutation that randomly chooses another concrete service), however a local improver is also employed. This improvement process randomly explores a percentage of a solution's neighbourhood. A drawback of using Tabu search for Web service composition is that the structure used for the representation of candidates is typically linear [80, 4], which restricts the problem model to address semi-automated compositions only.

Integer Linear Programming (ILP) has also been applied to composing Web services [111]. ILP is flexible in the way it represents problems, therefore a fully automated Web service composition approach that also takes non-functional attributes into account when constructing the best solution can be solved using it. An objective function is defined for the achieving the optimal QoS values, and several other functions are used to restrict the functionality of the solutions (i.e. restrict the search space). For linear programming, we first determine the "corners" of the restricted search space (i.e. where two constraint lines meet), and then apply the objective function to each of these solutions. One of these "corner" solutions is the optimal one, provided that all boundary functions are linear, so the best objective function score indicates the final solution. While ILP applied to Web service compositions is guaranteed to find an optimal solution, it has been shown to be very inefficient in comparison to EC-based approach as the complexity of the problem increases [3]. Another problem with the ILP approach is that it is likely to pose difficulties when used for creating compositions with multiple execution paths, as branching constraints would have to be represented using a linear function.

Structural Equation Modelling (SEM), a statistical analysis method for forecasting values according to current measurements, has been used in a service selection strategy that considers the future trends of the QoS values of the candidate services [57] as opposed to relying on static QoS snapshots. By being able to predict the QoS trends of services, it is possible to create a composition that is optimal at execution time. One key advantage of SEM is that it is capable of accommodating changes in the user preferences (weights) associated with each QoS attribute, and can use the errors in QoS measurement histories to create a forecast model. The proposed approach was tested using different composition algorithms (e.g. ILP) enhanced with SEM, with a dataset that simulates the changes of QoS parameters over time. Results show that the prediction model for QoS values is initially quite inaccurate, but the accuracy increases for both algorithms as days go by and a history of values is built. While this method is very effective at predicting QoS values to aid in the optimisation of solutions, it cannot be used alone to compose Web services. Thus, SEM is not considered further in this work.

Finally, **Algebraic Expressions (AE)** have been employed to the problem of Web service composition [32, 46]. A formal representation of a Web service composition is used in this approach, relying on algebraic constructs to describe the behaviour of atomic Web services and to constraint the characteristics of a correct composition solution. One of the main advantages of AE is that this technique is expressive enough to emulate the behaviour of Web service composition languages such as BPEL4WS, thus it is possible to design and verify composition solutions entirely through AE. A more flexible composition option, explored in [32], involves constructing a mapping between algebraic expression and BPEL, to allow for an automated translation between these two representations. The work in [46] goes further, proposing a composition algorithm that also performs QoS optimisation based on algebraic expressions. Despite promising, the disadvantage of this technique is that it requires the composition task to be described in formal terms that are challenges to system users without the necessary background.

2.3 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) [42] 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.

2.3.1 Multi-objective approaches

Many multi-objective techniques [61, 116, 112, 110, 108, 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 [61], but other EC algorithms are also considered. For GA, [61] 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 [98] 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 [110] 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 [98].

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 [116] 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 criteron. The paper [100] introduces nonfunctional 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 algorithmd. 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.

2.3.2 Many-objective approaches

2.4 AI Planning-based Composition Approaches

AI planning approaches to Web service composition ensure feasibility by building a composition solution step by step. This solution is represented as a graph, a may either be built to enforce a set of user constraints, or it may be used to find an optimal solution in terms of QoS. A number of works in this area [30, 99, 106, 102] have been based on a fast planning algorithm named Graphplan [8]. In this algorithm, a solution is constructed gradually, at each time adding a new atomic service to the composition. A service may only be added to the solution if all of its conditions are met, that is, all of its inputs are fulfilled. Finally, the execution of Graphplan is stopped once an atomic service has been added that leads to meeting the overall composition objectives (i.e. the composition now produces all of the required outputs). Figure **??** shows a basic example run of Graphplan applied to Web service composition. In step 1, a start node is added to the graph structure. This node produces the overall input values provided by the composition requestor, *ZipCode* and *Date*. In step 2, the service *LocationByZip* is connected, since its input of *ZipCode* can be fulfilled by the existing structure. However, the algorithm continues to execute, since the overall output has only been partially fulfilled (i.e. only *City* can be produced). Finally, in step 3, the service *Weather* is connected, since its inputs are fulfilled by both the start nodes and the *LocationByZip* service. As the overall output has been fulfilled, the graph's end node is also connected to the structure.

In [15], authors combine a planning algorithm and a graph search algorithm to achieve both QoS optimisation and feasibility in Web service compositions. The generic Graphplan algorithm first builds a representation of the search space as a planning graph, then finds a solution within this graph by traversing it backwards. This standard planning approach is modified to use Dijkstra's algorithm [95] when performing the backwards traversal, thus finding an optimised solution. The planning graph is extended to include labels associated with each proposition (i.e. each intermediate action between two vertices), where each label contains a layer number and associated execution costs. Dijkstra's algorithm is used to calculate the upcoming costs of each node in the graph. Then, a backtracking algorithm uses this information to determine the optimal solution.

The work in [28] proposes a planning-graph approach to creating a Web service composition technique that is capable of identifying the top K solutions with regards to overall QoS values. According to authors, AI planning is highly suitable to the domain of service composition, however many planning approaches are not efficiently executed. A notable exception to this trend is the planning-graph method, where the search space is greatly reduced through an initial search, thus allowing the remainder of the algorithm to focus on the relevant areas of the search space. In this work, the planning-graph approach is employed in a three-part algorithm. In the first part, a forward search is executed from the input node aiming to find the output node, gradually filtering the services that could be used in the composition. In the second part, the optimal local QoS of each service in the remaining graph is calculated using functions that take into consideration the QoS of the services that could possibly feed its input, also executed from the input node towards the output. Finally, a backward search algorithm is executed to generate the top K solutions according to local values calculated in the previous step (a threshold is provided when running this algorithm to prune out the substandard composition options).

An automated Web service composition approach that uses a filtering algorithm to reduce the number of services considered for the composition, organising the remaining services as a graph according to the ways in which their inputs and outputs match, is proposed in [41]. Once the graph has been determined, a modified dynamic programming approach is applied to it in order to calculate the composition with the optimal QoS. Dynamic programming is a method that breaks problems into smaller subproblems that are then solved, ultimately leading to the solution of the parent problem. In this case, the optimal QoS of each atomic service in the graph is calculated, taking into account its input dependencies. At the end of this process, the overall optimal QoS values are known and the subgraph containing the solution can be extracted by searching the graph backwards. Experiments with the WSC2009 dataset show that the algorithm has good execution times for various dataset sizes, demonstrating its scalability. This work was extended in [43], with the presentation of a composition tool called QSynth, and the performance of formal comparisons with other state-of-the-art approaches, also producing superior results.

A number of other works in the area employ formal AI planning techniques and frameworks to create compositions [7]. The work in [96] presents an approach to include user preferences into the process of Web service composition. This is accomplished by relying on a framework written in Golog, a language created for agent programming. Golog is used to specify the particular attributes of generic workflows that represent commonly requested composition procedures (an example of a generic workflow would be one that is dedicated to booking inter-city transportation). The syntax of a logic-based language used to specify user preferences is described, allowing for branching according to conditions, and for expressing preferences over alternative services. Despite supporting branching, only one set of final outputs is allowed, meaning that the branches must be merged before reaching the end node of the composition workflow.

An approach for modelling the data flow between Web services through the use of *domain objects* is presented in [47]. For example, the travel domain contains objects such as flight ticket, hotel reservation, etc. The key idea is to use these objects to connect the composition needs to the services that can address them. In order to do so, authors annotate how each Web service operation relates to a given domain object. By creating this abstraction layer of objects, it is possible to reduce the composition's dependency on implementation details for correct execution. Now services can be thought of as having an object port proxy that leads to specific service ports. Compositions can then be achieved by identifying the necessary domain objects for the required task. The paper goes on to show how this technique can be integrated into existing service composition techniques, in this case AI-planning based, through the creation of a formal framework. This framework was implemented and run with a virtual travel agency scenario. According to the authors, the implementation of such framework was not trivial, however it successfully demonstrates that service implementations can be modified without impacting the overall composition.

A Web service composition approach that allows users to specify constraints on the data flow of the solutions (i.e. which routes a message is allowed to take and which manipulations it can undergo) is presented in [66]. For example, consider a Web service composition that is supposed to book a holiday for a customer using a flights, accommodation, and a map service. If it is possible to book a suitable flight but it is not possible to book a hotel, the customer should not accept the offer. This is the type of requirement addressed by this approach using a data flow modelling language. This is a visual language that supports the definition of inputs/outputs, forking messages, merging messages, operations on messages, etc. By connecting these elements we obtain *data nets* whose satisfiability can be clearly verified. The composition of Web services is performed using a planning framework that is capable of interpreting and respecting the constraints of a data net. At the time this paper was written, this approach had not yet been implemented or tested.

2.4.1 Hybrid Approaches

Hybrid approaches combine elements of AI planning and optimisation techniques for solving the composition problem with functionally correct, optimised solutions [18, 87, 107, 17]. These hybrid approaches are quite similar to each other, relying on a directed acyclic graph as the base representation for a candidate solution, and then applying the optimisation techniques to this structure. However, despite incorporating the use of planning techniques, they do not include any discussion on the issue of producing solutions that satisfy conditional constraints or preferences. Another commonality between these works is that they require the use of SAWSDL-annotated datasets for testing, but these are not widely available to the research community. Therefore, authors developed their own datasets, and utilised each dataset's optimal task solution as the benchmark with which to evaluate the success of their implementation. More specifically, authors calculated the percentage of runs that culminated in the identification of the global optimum as the recommended solution.

In [85], an approach that combines AI planning and an immune-inspired algorithm is used to perform fully automated QoS-aware Web service composition, also considering semantic properties. One significant contribution of this work is the proposal of an Enhanced Planning Graph (EPG), which extends the traditional planning graph structure by incorporating semantic information such as ontology concepts. Given this data structure, the composition algorithm selects the best solution configuration from a set of candidates. A fitness function considering QoS values and semantic quality is used to judge the best solution, and a clonal selection approach is employed to perform the optimisation. Candidates cells (solutions) are cloned, matured (mutated by replacing services with others from the same cluster in the EPG) and the cell most suited to combating the invading organism (i.e. the best solution) is discovered.

The work in [86] proposes the employment of the Firefly meta-heuristic technique for performing QoS-aware Web service composition, in conjunction with an AI planning strategy that uses an EPG as the basis for solutions. The firefly meta-heuristic is based on the behaviour of mating fireflies, which emit a flashing light to attract potential mates. Each artificial firefly investigates the search space, with each position representing a composition solution. The brightness of the firefly is represented by the fitness of the current solution (location) associated with it. Fireflies are attracted to others according to their brightness, which varies with distance. Finally, fireflies move towards the individuals they are attracted to, meaning that small modifications occur in the current solution. The fitness function takes into account the QoS attributes of the composition.

2.5 Semantic Selection Approaches

One important issue when creating compositions is that of selecting services that are compatible to each other. The simplest way of achieving this is by ensuring that the conceptual output and input types of any two services we wish to connect are perfectly matched, as illustrated by the Graphplan algorithm example in Section 2.4. This involves accessing each service's WSDL, which is a formal description of a Web service's interface [20], and determining whether the two services offer compatible operations. However, in a more realistic scenario it may be very difficult to identify two services with such a precise fit. Because of this, an area of focus in the field of composition is that of semantic Web service selection. The fundamental idea of this approach is to annotate each service with additional semantic information so that the matching of services can be accomplished at a conceptual level. A well-known semantic standard for Web services is OWL-S (Web Ontology Language for Services) [69] standard. OWL-S provides a formal specification of the workings of a service, allows for the association of conceptual classes with each Web service, and supports the use of ontologies that record the relationships between members of these different classes, thus establishing a common vocabulary for inter-service interactions. These features are conducive to automating the handling of Web services, and facilitating the discovery of those that are relevant for a specific task. Another more recent standard for semantic Web service annotation is SAWSDL (Semantic Annotations for WSDL) [50], a language that builds on WSDL by embedding pointers in the WSDL description that refer to semantic concepts. These pointers are called annotations, and are linked to concepts in a higher-level ontology layer.

A number of different selection techniques that use semantic descriptions have been proposed recently [97, 103, 35, 60, 45, 12, 92, 58, 117]. The work in [11], for example, presents a selection strategy that considers more than just WSDL-level descriptions for Web services. In this approach, objects with additional information that is useful to the selection process are associated with each service. These objects have an independent ontology that describes how they interrelate (e.g. a service for making medical appointments has associated objects such as doctor, patient, hospital/clinic, etc), and at composition time the compatibility of these objects is verified. To accomplish this, a framework with service providers, ontology providers, information agents and a composer is proposed. This framework takes selection constraints set by the user into account.

An automated semantic composition method is proposed in [94]. In this method, services are classified according to a functionality tag consisting of an {action, object} pair (e.g. a service for calculating the distance between two cities has the tag {Calculate, Distance}). A service relation graph is created to illustrate the dependencies between concepts, and it is divided in three parts: a graph showing relationships between actions, another graph showing relationships between objects (input/output relationships), and a mapping between items in these graphs. The relationships between these items are determined using domain ontology trees, with the assumption that these trees have already been provided. Given these dependencies, an algorithm is used to find a composition path. The path is found through the action graph and service connections are made based on the object graph, not only the object names. This approach was shown to require substantially less time to execute for larger datasets than previously proposed methods.

The work in [39] proposes a semantic Web service selection method that performs matching based on four distinct levels. At the *Application Domain Matching* level the domain that best matches the user request is identified through the use of category ontologies, and a list of potential service description candidates is retrieved. This is performed by calculating a similarity degree between the user request and the semantic information associated with each domain. Subsequently, at the *Service Description Matching* level, vectors are created for each potential service candidate description based on a given domain ontology, and another vector is created to represent the user requirements. A Vector Space Model is created, employing cosine similarity and TF-IDF to select the best service description. At the *Service Function Matching* level, information from service providers is compared to the service description using a similarity measure, and a set of all services whose functionality fulfils the requirements is returned. Finally, at the *QoS Matching* level, a matching array of quality values showing how closely a Web service matches the user's requirements is calculated, and the optimal candidate is returned.

A framework for performing semantic Web service composition that also allows for user constraints to be specified is proposed in [34]. Initially, services are grouped into distributed "service communities" according to their OWL-S semantic descriptions, where each community has services that cater for similar domains and consequently present similar functionalities. Then, users formulate their composition needs, including the necessary constraints, using terms from the semantic service community descriptions. Effectively, they create an abstract workflow for semi-automated composition by using the community descriptions and specifying their own preferences for the services to be selected. These constraints may be restrictions in input value ranges, in the output value ranges, or other service parameters. A language called KIF was chosen to express the constraints according to the corresponding OWL-S service descriptions. Interestingly, this approach can also handle world-altering services and non-deterministic functionality because it makes use of statecharts to model the behaviour of services.

2.6 Dynamic Web Service Composition

The approaches discussed thus far can be classified as static Web service composition, since they maintain a closed world assumption, that is, they assume that that the quality and the state of the services in the composition remains constant over time. However, in reality the state of the services available in a repository changes as time goes by, causing fluctuations in quality and even occasional failures. The area of dynamic Web service composition removes the closed world assumption, exploring solutions that can cope with quality fluctuations and service failures [1, 6, 105]. The work in [48] proposes an algorithm that quickly creates a solution to satisfy a runtime service request, modelling the service composition problem as a graph in which a path is to be found. Forward and backward chaining approaches based on input/output matches are used for path exploration, relying on heuristics to encourage the exploration of the most promising paths and to reduce the number of services considered. In the approach proposed by this paper, both forward chaining and backward chaining are employed simultaneously, with the intention of having their paths meet in the middle. By doing so, the number of branches to be considered is greatly reduced. Experiments showed that the bidirectional search requires the exploration of a consistently smaller number of services when compared to the exclusively forward and exclusively backward approaches.

A multi-objective Web service selection approach that optimises solutions according to two functions, a static one that calculates the overall quality of service (QoS) of a solution, and a dynamic one which calculates the *trust degree* of a solution at a given time, is presented in [100]. The trust degree is defined as the number of successful executions of a service over its total number of executions, a piece of information that can be obtained by analysing execution logs. The optimisation algorithm used in this approach is an adaptive form of ant colony optimisation (ACO), where pheromones are adjusted according to the trust degree (calculated anew at every iteration) and the QoS is used by each ant as the heuristic for choosing the next node to visit. Each node in the graph explored by the ants represents an abstract service, with multiple concrete service candidates which can provide that functionality. The algorithm works by generating a set of of solutions and then identifying its Pareto subset by comparing all solutions. The Pareto subset is used in the next iteration, for updating the path pheromones. A case study is presented, comparing the performance of adaptive ACO to that of of the standard ACO algorithm, with results showing that adaptive

ACO has a higher accuracy percentage than the standard ACO.

The notion of transactions can be associated with service compositions, where different transactional properties guarantee certain runtime behaviours in a dynamic environment. The work in [29] proposes a semi-automated Web service composition approach that not only takes the functionality and quality of the services into account, but also their transactional properties. In this work, transaction properties are defined as the behaviour of Web services when interacting with one another. Knowing the behaviour of Web services is important for estimating how reliable their execution is and which ones might require recovery strategies at runtime. A service is considered *retriable* if it can terminate successfully after multiple invocations, *compensatable* if there is another service that can semantically undo its effects, and *pivot* if its effects cannot be undone once it is executed but if it fails there are no effects. The system takes an abstract workflow and a set of user preferences as its inputs, where the user preferences contain QoS weights and risk levels corresponding to the transitional requirements for the composition. Then, a planner engine assigns one concrete Web service to each abstract slot of the provided workflow. Whenever a service is assigned to a part of the workflow, its transactional properties influence any subsequent services, thus the risk must be recalculated along with the overall QoS. Experiments were run for various risk scenarios, and computation time was found to remain under 2 seconds even for the largest dataset (comprising 3602 atomic Web services), at the same time meeting user preferences.

An approach to QoS-aware Web service composition that is focused on *rebinding*, that is, deciding which concrete services to bind to each abstract task at runtime in order to take the current state of the environment into consideration, is presented in [81]. This approach considers global QoS constraints (e.g. the overall composition price must be lower than 5), local QoS constraints (e.g. the individual price for a service must be no higher than 1), and service dependency constraints (e.g. as many services as possible should be used from the same provider). Two techniques are employed during the composition process: GRASP, which is used to construct initial solutions, and Path Relinking, which is used to perform improvement on these solutions. *GRASP* (Greedy Randomized Adaptive Search Procedure) iteratively builds a *valid* solution vector, adding one candidate to fulfil each solution slot at a time and ensuring that this candidate respects the pre-established user constraints. The order of candidate addition matters, so it is performed randomly each time. Once a valid solution has been built, GRASP identifies a list of replacement candidates for each task slot, including the most promising candidates while also respecting the constraints observed by the valid solution. *Path Relinking* explores the neighbouring solutions of the initial valid solution, seeking to further improve it. To do so, it slowly modifies solutions by changing services from one task slot at a time. The objective function used in this paper encourages the improvement of QoS values at the same time it enforces the relevant user constraints.

2.7 Summary and Limitations

This chapter presented an overview of the recent research conducted on different aspects of the automated Web service composition problem. The first area explored was that of **single-objective composition**, which aims to create composite solutions with the best possible overall Quality of Service (QoS) by conducting optimisations according to an objective function. Different approaches have been attempted for this, including both traditional approaches such as Integer Linear Programming (ILP) and Evolutionary Computation (EC) approaches such as Genetic Programming (GP), though in certain cases traditional approaches may not scale well as the composition problems grow in complexity. One key limitation of single-objective composition approaches is that they neglect the issue of branching, that is, they do not allow the creation of solutions with multiple alternative outputs that may be produced depending on a condition. To overcome this problem, a new candidate representation that also encodes this form of conditional branching needs to be proposed.

Another area discussed is that of **multi-objective, top-K, and many-objective composition**, where each QoS attribute is optimised separately, and a set of solutions presenting trade-offs between these different quality attributes is generated. Multi-objective approaches work best for optimisation problems that involve two or three independent dimensions, while many-objective approaches are capable of handling more than three of them; top-K aims to produce a predetermined number of solution options based on a ranking strategy. The optimisation of multiple independent objectives is complex and provides many possibilities for further exploration. One interesting problem is that of many-objective optimisation for SLA-aware Web service composition, which refers to the constrained optimisation of several independent objectives to reflect the minimum quality requirements of the composition requestor. Accomplishing this is particularly challenging when creating solutions with multiple output possibilities.

A number of **AI planning-based composition** approaches are also presented in this chapter. Their fundamental idea is to build solutions step by step, adding one atomic service to the composition at a time and subsequently checking whether the overall desired output has now been produced. These approaches are conducive to ensuring that the generated solutions are feasible and also included conditional branches into the solution's workflow whenever necessary. Despite these advantages, it is difficult to globally optimise the quality of a solution produced through planning, since its components are not modified or improved once they have been selected. A promising strategy to overcome this problem would be to combine planning algorithms with a population-based approach for optimisation, though this avenue has not yet been significantly explored by researchers.

The **semantic Web service selection** approaches examined in this chapter propose a more realistic way of selecting the atomic components of a composition. Instead of expecting the return values and parameter types of service operations to match perfectly, the closest possible output-input matches are calculated using semantic distance measurements. The limitation of most works in this area is that they restrict their selection technique to semi-automated composition, where it is assumed that a framework with abstract service slots that are to be filled by concrete atomic services has already been provided. In the case of composition approaches based on AI planning techniques, where the workflow is built as services are connected to the solution, more flexible semantic selection methods are necessary. This motivates further research in this area.

Finally, this chapter focuses on the issue of **dynamic Web service composition**. In this type of composition a closed world is abandoned, meaning that the state of services is expected to change throughout time. This brings two main problems: firstly, when the quality of the services in a repository changes, a solution that was once optimised according to the previous quality measures may suddenly transform into a low-standard alternative; secondly, certain services may become unavailable, meaning that solutions which incorporate them are henceforth unusable. The dynamic composition approaches discussed in this chapter use a variety of self-healing and recovery techniques to adjust solutions according to these changes, however they do not rely on EC techniques for doing so. These techniques can quickly re-optimise the QoS of existing solutions and provide composition backups (i.e. other candidates in the population) in case of failure, thus making EC an interesting dynamic alternative.

Bibliography

[1] Alférez, G. H., and Pelechano, V. Facing uncertainty in web service compositions. In *Web Services (ICWS), 2013 IEEE 20th International Conference on* (2013), IEEE, pp. 219–226.

[2] 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.

[3] Aversano, L., Di Penta, M., and Taneja, K. A genetic programming approach to support the design of service compositions. *International Journal of Computer Systems Science & Engineering 21*, 4 (2006), 247–254.

[4] Bahadori, S., Kafi, S., Far, K., and Khayyambashi, M. Optimal web service composition using hybrid ga-tabu search. *Journal of Theoretical and Applied Information Technology 9*, 1 (2009), 10–15.

[5] 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.

[6] Berg, H. v. d., et al. Revenue optimization of service compositions using conditional request retries. In *Web Services (ICWS), 2013 IEEE 20th International Conference on* (2013), IEEE, pp. 1–9.

[7] Bertoli, P., Kazhamiakin, R., Paolucci, M., Pistore, M., Raik, H., and Wagner, M. Control flow requirements for automated service composition. In *Web Services, 2009. ICWS 2009. IEEE International Conference on* (2009), IEEE, pp. 17–24.

[8] Blum, A. L., and Furst, M. L. Fast planning through planning graph analysis. *Artificial intelligence 90*, 1 (1997), 281–300.

[9] 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).

[10] 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.

[11] Boustil, A., Maamri, R., and Sahnoun, Z. A semantic selection approach for composite web services using OWL-DL and rules. *Service Oriented Computing and Applications 8*, 3 (2014), 221–238.

[12] Boustil, A., Sabouret, N., and Maamri, R. Web services composition handling user constraints: towards a semantic approach. In *Proceedings of the 12th International*

Conference on Information Integration and Web-based Applications & Services (2010), ACM, pp. 913–916.

[13] BROWN, N., MCKAY, B., GILARDONI, F., AND GASTEIGER, J. A graph-based genetic algorithm and its application to the multiobjective evolution of median molecules. *Journal of chemical information and computer sciences 44*, 3 (2004), 1079–1087.

[14] 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.

[15] CHEN, M., AND YAN, Y. Qos-aware service composition over graphplan through graph reachability. In *Services Computing (SCC), 2014 IEEE International Conference on* (2014), IEEE, pp. 544–551.

[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] CHIFU, V. R., POP, C. B., SALOMIE, I., SUIA, D. S., AND NICULICI, A. N. Optimizing the semantic web service composition process using cuckoo search. In *Intelligent distributed computing V*. Springer, 2012, pp. 93–102.

[18] COTTA, C., AND FERNÁNDEZ, A. J. Memetic algorithms in planning, scheduling, and timetabling. In *Evolutionary Scheduling*. Springer, 2007, pp. 1–30.

[19] CRAENEN, B., EIBEN, A., AND MARCHIORI, E. How to handle constraints with evolutionary algorithms. *Practical Handbook Of Genetic Algorithms: Applications* (2001), 341–361.

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

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

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

[23] DA SILVA, A. S., MA, H., AND ZHANG, M. A graph-based particle swarm optimisation approach to qos-aware web service composition and selection. In *2014 IEEE Congress on Evolutionary Computation (CEC)* (2014), IEEE, pp. 3127–3134.

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

[25] DA SILVA, A. S., 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.

[26] 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.

[27] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. A fast and elitist multi-objective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation 6*, 2 (2002), 182–197.

[28] Deng, S., Wu, B., Yin, J., and Wu, Z. Efficient planning for top-k web service composition. *Knowledge and information systems 36*, 3 (2013), 579–605.

[29] El Hadad, J., Manouvrier, M., and Rukoz, M. Tqos: Transactional and qos-aware selection algorithm for automatic web service composition. *Services Computing, IEEE Transactions on 3*, 1 (2010), 73–85.

[30] 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.

[31] Fensel, D., Facca, F. M., Simperl, E., and Toma, I. *Semantic web services*. Springer Science & Business Media, 2011.

[32] Ferrara, A. Web services: a process algebra approach. In *Proceedings of the 2nd international conference on Service oriented computing* (2004), ACM, pp. 242–251.

[33] Galvan-Lopez, E. Efficient graph-based genetic programming representation with multiple outputs. *International Journal of Automation and Computing 5*, 1 (2008), 81–89.

[34] Gamha, Y., Bennacer, N., Naquet, G., Ayeb, B., and Romdhane, L. B. A framework for the semantic composition of web services handling user constraints. In *Web Services, 2008. ICWS'08. IEEE International Conference on* (2008), IEEE, pp. 228–237.

[35] García, J. M., Ruiz, D., Ruiz-Cortés, A., and Parejo, J. A. Qos-aware semantic service selection: An optimization problem. In *Services-Part I, 2008. IEEE Congress on* (2008), IEEE, pp. 384–388.

[36] Garey, M. R., and Johnson, D. S. A guide to the theory of np-completeness. *WH Freemann, New York 70* (1979).

[37] Globus, A., Lawton, J., and Wipke, T. Automatic molecular design using evolutionary techniques. *Nanotechnology 10*, 3 (1999), 290.

[38] Glover, F. Tabu search-part i. *ORSA Journal on computing 1*, 3 (1989), 190–206.

[39] Guo, G., Yu, F., Chen, Z., and Xie, D. A four-level matching model for semantic web service selection based on qos ontology. In *Information Science and Engineering (ISISE), 2010 International Symposium on* (2010), IEEE, pp. 630–634.

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

[41] Huang, Z., Jiang, W., Hu, S., and Liu, Z. Effective pruning algorithm for qos-aware service composition. In *2009 IEEE Conference on Commerce and Enterprise Computing* (2009), IEEE, pp. 519–522.

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

[43] JIANG, W., ZHANG, C., HUANG, Z., CHEN, M., HU, S., AND LIU, Z. Qsynth: A tool for qos-aware automatic service composition. In *Web Services (ICWS), 2010 IEEE International Conference on* (2010), IEEE, pp. 42–49.

[44] KANTSCHIK, W., AND BANZHAF, W. Linear-graph gp-a new gp structure. In *Genetic Programming*. Springer, 2002, pp. 83–92.

[45] KARAKOC, E., AND SENKUL, P. Composing semantic web services under constraints. *Expert Systems with Applications 36*, 8 (2009), 11021–11029.

[46] KATTEPUR, A., GEORGANTAS, N., AND ISSARNY, V. Qos composition and analysis in reconfigurable web services choreographies. In *Web Services (ICWS), 2013 IEEE 20th International Conference on* (2013), IEEE, pp. 235–242.

[47] KAZHAMIAKIN, R., MARCONI, A., PISTORE, M., AND RAIK, H. Data-flow requirements for dynamic service composition. In *Web Services (ICWS), 2013 IEEE 20th International Conference on* (2013), IEEE, pp. 243–250.

[48] KHAKHKHAR, S., KUMAR, V., AND CHAUDHARY, S. Dynamic service composition. *International Journal of Computer Science and Artificial Intelligence* (2012).

[49] 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.

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

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

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

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

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

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

[56] 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.

[57] LI, M., ZHU, D., DENG, T., SUN, H., GUO, H., AND LIU, X. GOS: a global optimal selection strategies for qos-aware web services composition. *Service Oriented Computing and Applications 7*, 3 (2013), 181–197.

[58] LI, W., BADR, Y., AND BIENNIER, F. Towards a capability model for web service composition. In *Web Services (ICWS), 2013 IEEE 20th International Conference on* (2013), IEEE, pp. 609–610.

[59] LI, X., YANG, G., AND HIRASAWA, K. Evolving directed graphs with artificial bee colony algorithm. In *Intelligent Systems Design and Applications (ISDA), 2014 14th International Conference on* (2014), IEEE, pp. 89–94.

[60] LIN, N., KUTER, U., AND SIRIN, E. *Web service composition with user preferences*. Springer, 2008.

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

[62] LU, J., YU, Y., ROY, D., AND SAHA, D. Web service composition: A reality check. In *Web Information Systems Engineering–WISE 2007*. Springer, 2007, pp. 523–532.

[63] 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.

[64] MABU, S., HIRASAWA, K., AND HU, J. A graph-based evolutionary algorithm: genetic network programming (gnp) and its extension using reinforcement learning. *Evolutionary Computation 15*, 3 (2007), 369–398.

[65] MARCONI, A., PISTORE, M., AND POCCIANTI, P. Automatedweb service composition at work: the amazon/mps case study. In *Web Services, 2007. ICWS 2007. IEEE International Conference on* (2007), IEEE, pp. 767–774.

[66] MARCONI, A., PISTORE, M., AND TRAVERSO, P. Specifying data-flow requirements for the automated composition of web services. In *Software Engineering and Formal Methods, 2006. SEFM 2006. Fourth IEEE International Conference on* (2006), IEEE, pp. 147–156.

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

[68] 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.

[69] MARTIN, D., BURSTEIN, M., MCDERMOTT, D., MCILRAITH, S., PAOLUCCI, M., SYCARA, K., MCGUINNESS, D. L., SIRIN, E., AND SRINIVASAN, N. Bringing semantics to web services with owl-s. *World Wide Web 10*, 3 (2007), 243–277.

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

[71] MILLER, J. F., AND THOMSON, P. Cartesian genetic programming. In *Genetic Programming*. Springer, 2000, pp. 121–132.

[72] MOBEDPOUR, D., AND DING, C. User-centered design of a qos-based web service selection system. *Service Oriented Computing and Applications 7*, 2 (2013), 117–127.

[73] 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.

[74] NICOLAOU, C. A., APOSTOLAKIS, J., AND PATTICHIS, C. S. De novo drug design using multiobjective evolutionary graphs. *Journal of Chemical Information and Modeling 49*, 2 (2009), 295–307.

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

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

[77] 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.

[78] 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.

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

[80] 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.

[81] PAREJO, J. A., SEGURA, S., FERNANDEZ, P., AND RUIZ-CORTÉS, A. Qos-aware web services composition using grasp with path relinking. *Expert Systems with Applications 41*, 9 (2014), 4211–4223.

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

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

[84] POLI, R. *Parallel distributed genetic programming*. Citeseer, 1996.

[85] POP, C. B., CHIFU, V. R., SALOMIE, I., AND DINSOREANU, M. Immune-inspired method for selecting the optimal solution in web service composition. In *Resource Discovery*. Springer, 2010, pp. 1–17.

[86] POP, C. B., ROZINA CHIFU, V., SALOMIE, I., BAICO, R. B., DINSOREANU, M., AND COPIL, G. A hybrid firefly-inspired approach for optimal semantic web service composition. *Scalable Computing: Practice and Experience 12*, 3 (2011).

[87] POP, C. B., VLAD, M., CHIFU, V. R., SALOMIE, I., AND DINSOREANU, M. A tabu search optimization approach for semantic web service composition. In *Parallel and Distributed Computing (ISPDC), 2011 10th International Symposium on* (2011), IEEE, pp. 274–277.

[88] 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.

[89] RAO, J., AND SU, X. A survey of automated web service composition methods. In *Semantic Web Services and Web Process Composition*. Springer, 2005, pp. 43–54.

[90] 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.

[91] 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.

[92] SABOOHI, H., AND KAREEM, S. A. World-altering semantic web services discovery and composition techniques-a survey. In *The 7th International Conference on Semantic Web and Web Services (SWWS)* (2011), pp. 91–95.

[93] 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.

[94] SHIN, D.-H., AND LEE, K.-H. An automated composition of information web services based on functional semantics. In *Services, 2007 IEEE Congress on* (2007), IEEE, pp. 300–307.

[95] SKIENA, S. Dijkstra's algorithm. *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading, MA: Addison-Wesley* (1990), 225–227.

[96] 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.

[97] SOYDAN BILGIN, A., AND SINGH, M. P. A daml-based repository for qos-aware semantic web service selection. In *Web Services, 2004. Proceedings. IEEE International Conference on* (2004), IEEE, pp. 368–375.

[98] 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.

[99] 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.

[100] 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.

[101] 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.

[102] 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.

[103] WANG, X., VITVAR, T., KERRIGAN, M., AND TOMA, I. A qos-aware selection model for semantic web services. In *Service-Oriented Computing–ICSOC 2006*. Springer, 2006, pp. 390–401.

[104] WANG, X., WANG, Z., AND XU, X. An improved artificial bee colony approach to qos-aware service selection. In *Web Services (ICWS), 2013 IEEE 20th International Conference on* (2013), IEEE, pp. 395–402.

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

[106] XIA, Y.-M., AND YANG, Y.-B. Web service composition integrating qos optimization and redundancy removal. In *Web Services (ICWS), 2013 IEEE 20th International Conference on* (2013), IEEE, pp. 203–210.

[107] XIANG, C., ZHAO, W., TIAN, C., NIE, J., AND ZHANG, J. Qos-aware, optimal and automated service composition with users' constraints. In *e-Business Engineering (ICEBE), 2011 IEEE 8th International Conference on* (2011), IEEE, pp. 223–228.

[108] 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.

[109] 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.

[110] 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).

[111] 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.

[112] 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.

[113] 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.

[114] 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.

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

[116] 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.

[117] ZHANG, Z., ZHENG, S., LI, W., TAN, Y., WU, Z., AND TAN, W. Genetic algorithm for context-aware service composition based on context space model. In *Web Services (ICWS), 2013 IEEE 20th International Conference on* (2013), IEEE, pp. 605–606.

[118] Zitzler, E., Laumanns, M., Thiele, L., et al. Spea2: Improving the strength pareto evolutionary algorithm, 2001.