

Comprehensive quality - aware Automated Semantic Web Service Composition

Chen Wang, Hui Ma, and Aaron Chen

School of Engineering and Computer Science,
Victoria University of Wellington, New Zealand
{Chen.Wang, Hui.Ma, and Aaron.Chen}@ecs.vuw.ac.nz

Abstract. Semantic web service composition has been a prevailing research area in recent years. There are two major governance challenges faced by researchers. One is semantic matchmaking for discovering interoperable web services, which demands a rich machine understanding language such as OWL-S (Web Ontology language for Web Services), WSML (Web Service Modeling language, SAWSDL (Semantic Annotations for WSDL and XML Schema). This interaction of web services enables us to pair service functionalities through understanding and reasoning those semantic descriptions. Optimisation problems are another challenge. Many scholars have looked into nonfunctional optimisation problems in QoS-aware web service composition applying AI planning and Evolution Computing techniques. However, it is not sufficient without considering good balance between functional and nonfunctional quality in most scenarios. Therefore, the consideration in these two quality dimensions is a must, so that service composition needs to deal with optimisation problems in both quality of semantic matchmaking and QoS. This paper develops a more applicable way for measuring a proposed comprehensive quality model in both semantic matchmaking and QoS for automated semantic web service composition.

1 Introduction

Service-oriented computing (SOC) is a novel computing paradigm that employs services as fundamental elements to achieve agile development of cost-efficient and integratable enterprise applications in heterogeneous environments [21]. Service Oriented Architecture (SOA) could abstractly realise service-oriented paradigm of computing. This accomplishment has been contributing to the reuse of software components, from functions to units, and from units to services during the development in SOA [4]. One of the most typical realisations of SOA is *web service*, which is designated as “modular, self-describing, self-contained applications that are available on the Internet” [5]. Several standards play a significant role in registering, enquiring and grounding web services including particularly UDDI, WSDL and SOAP [8].

Web service composition pertains to a combination of multiple web services to provide a value-added composite service that accommodates customers’ ar-

bitrarily complex requirements. This application is developed by integrating interoperable and collaborative functionalities over heterogeneous systems. Due to the increase in the number of large-scale enterprise applications, the number of Web services has increased dramatically and unprecedentedly, which cause an immense redundancy in functionality in a huge searching space. Therefore, manual and semi-automated web service composition are considered to be less efficient while automated web service composition is deemed to be less human intervention, less time consumption, and high productivity.

Two most notable challenges for web service composition are (1) ensuring interoperability of services and (2) achieving QoS optimisation [8]. *Interoperability* of web services presents challenge (1) in the dimensions of syntactic and semantics [8]. The syntactic dimension is covered the XML-based technologies (such as *WSDL*, *SOAP*). The semantics aspect, on the other hand, demands further research. Through semantics given by ontologies [19], web services understand and better collaborate with each other. Historically, there are many ontologies languages and formats for semantic service descriptions, such as OWL-S [16], WSML [9], and SAWSDL [12]. The logical characteristics in reasoning makes “machine understanding” possible through identifying and matching semantic similarity in input/output parameters of web services in heterogeneous environments. The second challenge (2) is related to finding *optimised solutions* to Quality of Service (*QoS*) over composite web services. This problems give birth to *QoS-aware service composition* that considers the composition of service-level agreements [22] manifesting a collection of SLA rules and policies for supporting QoS-based composition.

Existing works on service composition focus mainly addressing two challenges above. One group optimises the quality of compositions under a pre-defined abstract workflow, which is considered to be a *semi-automated Web service composition* approach. Another group attempts to generate a composite plan automatically in discovering and selecting suitable web services, which are deemed to NP-hard [18]. *Semantic web services composition* is distinguished from the syntactic service composition. Its benefits are presented in eliminating conflicts by the semantic level of web services. In the past few years, substantial works have been done on semantic web service composition [8,13]. However, few works have enabled truly automatic semantic web service composition, where both QoS and quality of semantic match making will be optimised simultaneously. However, customers often prefer a service composition that matches their requirement most with highest QoS.

The overall goal of this paper is to *develop an comprehensive quality-aware automated semantic web service composition that satisfactorily optimises both functional and non-functional requirements*. Particularly, this paper considers the quality of semantic matchmaking quality and QoS and two structural constructs (sequence and parallel) for building optimised composition solution using Particle Swarm Optimisation (PSO). We aim to provide a more applicable and adaptive way to measure semantic matchmaking in automated semantic web service composition. We will achieve three objectives in this work as follows:

1. To propose a comprehensive quality model that address QoS and semantic matchmaking quality in considering different matching types associated with corresponding semantic similarity. This approach is consider to be a more applicable and effective way of finding an optimised solution in semantic web service composition.
2. To propose a PSO-based service composition model that utilise the proposed quality model. In PSO, weighted graphs with edges and vertices associated with quality of semantic matchmaking and QoS respectively represent the solutions of semantic web service composition as particles in the searching space, which are decoded from an optimised service queue with the consideration of comprehensive quality.
3. To evaluate the performance of comprehensive quality awareness semantic web service composition by utilising benchmark datasets from Web Services Challenge 2009 (WSC09) [11].

The remains of this paper are organised as the following: Sect. ?? provides the background on the semantic web service composition, and related works; Sect. 4 presents an overview of comprehensive quality-aware semantic automated web service composition consisting of comprehensive quality evaluation model and PSO-based service composition algorithm; Sect. 5 describes the experiments conducted to test the effectiveness of proposed model; Sect. 6 presents and analyses the experiment; Sect. 7 concludes our work.

2 Related Work

Semantic web service matchmaking. Substantial work [2,17,26,27,29] on web service composition focused mainly on non-functional requirements consistently neglecting functional requirements. However few researchers addressed both addressed both of them at the same time in web service composition. To the best of our knowledge, [6,13] are the papers could be found so far. Semantic matchmaking utilises Description Logic(*DL*) [1] reasoning between input and output parameter concepts of web services to ensure a semantic matchmaking. It is considered be a process consisting of seeking similarity of parameters (i.e., input and output) of web services and mapping between two knowledge representations encoded utilising the ontology [14].

In [23], three approaches are surveyed three approaches regarding the similarity measures using taxonomies: (1) The first one is based on nodes, in which similarity is determined by the information content of the nodes; (2) the second is entirely based on edge, where concept distance in a hierarchy structure is evaluated, and (3) the hybrid approach features a combination of (1) and (2). A new similarity measure based on edge counting in taxonomy is introduced in [23], which extends similarity measure defined by Wu and Palmer [28]. Neighbourhood concepts are considered in their model when $\lambda = 1$ (default value as 0).

In [13], the quality of matchmaking problem is transferred to measure the quality of semantic links $sl_{i,j} \doteq \langle s_i, Sim_T(Out_{s_i}, In_{s_j}), s_j \rangle$, one possible measure is applied for the degree of similarity using Common Description rate of a semantic link, where Extra Description $In_{s_x} \setminus Out_{s_y}$ and Least common subsume $lcs(Out_{s_i}, In_{s_j})$ are required to be pre-calculated. They also chained the web service with five well-known matching types: Exact, Plug-in, Subsume, Intersection, and Disjoint. Therefore, the quality of the semantic link is estimated as quality criteria q_m associated with their corresponding quality of vector q_{cd} .

However, the weakness of semantic link quality is that calculating Extra description and Least common subsume requires well and completely defined ontology in class, class axioms and properties in OWL2. This makes it difficult to measure semantic matchmaking quality in most semantic web services applications, as it takes unpredictable cost and time for the domain experts to establish required ontology. Additionally, a semi-automated service composition approach is considered in their paper, rather than a fully automated method. In QoS, only cost and time are considered in the optimised problems. Therefore, we introduce a more applicable comprehensive quality model in a fully automated approach.

In [6], it covers the concerns in both functional and non-functional requirements in design time of semantic-based automatic service composition, where a GA-based approach with four unusual independent fitness functions are designed to solve the problems with these concerns. In details, a sequence of fitness functions are used in the binary selection of chromosomes, rather than a single objective consisting of both functional and non-functional quality. Apart from that, match types are ignored in their quality evaluation of semantic matching.

QoS-aware EC approaches. Evolution Computing techniques are widely used to solve optimisation problem. Genetic Programming (GP) [26,27] is a typical EC technique to solve automated web service composition. This GP-based approach utilises tree representations, on which service and constructs as represented as terminal nodes and functional nodes respectively. The crossover and mutation operations reproduce various individuals while ensuring the correctness of structure. In [29], the author optimises the overall quality of service composition by using a fitness function, which is also liable for the correctness in functionality through penalising infeasible solutions.

To simplify the checking of constraints for solutions, an indirect PSO-based approach was introduced in [26]. A general graph is used as representation as particles in their PSO algorithm considering QoS optimisation. However, this solution does not distinguish different matching types and parameter similarity, which could lead to the outputs returned by the selected service that we think may be too general. In fact, customers' perspectives, application domains and ontology granularity all could have significant impacted on the outputs requested by users. In some scenarios, the output returns too broad consequences that have no meaning for the customers, even though those web services selected leads to a very good overall QoS. Therefore, we fill the gap by considering different matching types and the similarity when evaluate an overall quality of web service

composition and utilising weighed graphs as different particle representations from [26].

3 Problem Description

The purpose of web service composition is to accomplish an arbitrarily complex task fulfilling customer's requirement, which could be denoted as a composite goal: $Comp.G(F(T_{Input}, T_{Output}), NF(T_{QoS}))$. This overall composite goal is demonstrated in two parts. The first functional part is a given task input or input set to get the desired Task output or output set. It typically refers to users' functional requirement. Another nonfunctional part specifies the acceptable level of composite quality of service. To accomplish the composite goal, two stages are involved: services discovery and service selection. Firstly, service discovery is to find matched web service: $S_n(F(S_{Input}, S_{Output}), NF(S_{QoS}))$ from a Service Repository: $S = \{S_1, S_2, \dots, S_n\}$ with the given T_{Input} . If no atomic web service could satisfy the composite goal, a combination of web services will be found concretely to meet $Comp.G$. To ensure the composite solution returns the desired $Comp.G$, we consider a comprehensive quality model for service discovery and selection, where challenges (1) and (2) mentioned in Sect. 1 are also addressed in Sect. 4.

3.1 Semantic Web Service matchmaking Type

The semantic service matchmaking aims to discover appropriate services from service repository relevant to a customer's functional requests. A semantic web service is defined by $S(F(S_{Input} \in C_1, S_{Output} \in C_2), NF(S_{QoS}))$ with both Input and Output are linked to concept C_1 and C_2 in an ontology (O) respectively, satisfying $O = \{C, Taxonomy\}$. A web service matching process is to match the output and input concepts of two services according to the Taxonomy within an Ontology (O). To measure the quality of semantic matchmaking, different matching levels are typically considered in the literature [20]. To understand these levels, let us define two web services associated with parameters in a particular domain. $S_1 (F(S_{Input} \in C_1, S_{Outputs} \in C_2), NF(S_{QoS}))$ and $S_2 (F(S_{Input} \in C_3, S_{Output} \in C_4), NF(S_{QoS}))$ and an Ontology(O) with C_1, C_2, C_3 , and C_4 . The matching levels to be considered are:

- *Exact* (\equiv): Output of Web service S_1 and Input of Web service S_2 are Exact match ($S_{output} \in S_1 \equiv S_{input} S_2$), if Concept C_2 and Concept C_3 are equivalent.
- *Plugin* (\sqsubseteq_n): Output of Web service S_1 and Input of Web service S_2 are Plugin match ($S_{output} \in S_1 \sqsubseteq_n S_{input} \in S_2$), if Concept C_2 is a sub-concept of Concept C_3 , and $n = \{1, 2, \dots, n\}$ presents the levels of children concepts ($n = 1$ stands for direct children).
- *Subsume* (\supseteq_n): Output of Web service S_1 and Input of Web service S_2 are Subsume matched ($S_{output} \in S_1 \supseteq_n S_{input} \in S_2$), if Concept C_2 is a sub-concept of Concept C_3 , and $n = \{1, 2, \dots, n\}$ presents the levels of parent concepts ($n = 1$ stands for direct parent).

- *Fail* (\perp). Output of Web service S_1 and Input of Web service S_2 are not matched (Fail) ($S_{output} \in S_1 \perp S_{input} \in S_2$), if none of the previous matches discovered.

Note that, to find relevant services for service compositions according to a given goal, we consider Exact, Plugin, Subsume and Fail match types in our paper. Therefore, these matches will be used to discover web service candidates for service composition in our proposed the comprehensive quality-aware model in Sect.4.1

3.2 Quality of Service and Composition Constructs

Currently, most of the optimisation problems [7,10,15,25] in web service composition are focusing on QoS, which covers aspects in non-functional requirements. This problem has been explored in both single objective and multi-objectives optimisation problems. Customers prefer lowest execution cost with highest response time and reliability so that it could be involved as a single objective optimisation problem while cost and reliability of web service are two conflicting objectives that could be optimised simultaneously in multi-objective problem. According to [30], four most often considered QoS parameters are as follows:

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

The aggregation value of QoS attributes for web services composition varies with respect to different constructs, which explains how services associated with each other in a service composition [30]. Here we consider two composite constructs: sequence and parallel constructs, in building the composite flow. The QoS calculation models are described as follows:

Sequence construct The composite web service executes each atomic service associated with a sequence construct in a definite sequence order. The aggregation value for total time (T) and total cost (C) is as the sum of time and cost of web services involved respectively. The overall availability and reliability in a sequence construct are calculated by multiplying their corresponding availability and reliability of each web service in probability theory. This construct is shown in Figure 1.

Parallel construct Web services in a parallel construct are executed in the meantime. The QoS aggregation value for total cost, availability and reliability are the same as these in sequence construct while the Total time (T) is delimited by the maximum time consumed path in the composite flow of the solution. This construct is presented in Figure 2.

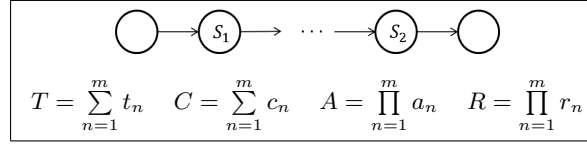


Fig. 1. Sequence construct and calculation of its QoS properties [29].

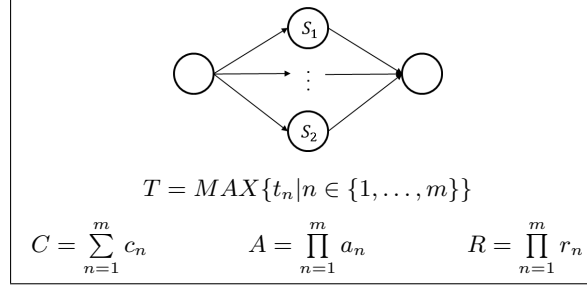


Fig. 2. Parallel construct and calculation of its QoS properties [29].

4 Comprehensive quality-aware Semantic Automated Web Service Composition

In this paper, we propose a comprehensive quality model for automated semantic service composition, and optimise both the quality of semantic matchmaking and QoS. PSO has shown its efficiency in solving combinatorial optimisation problems. Therefore, we will employ a PSO-based approach, which is considered to be simple and efficient without penalising or repairing comparing to GP [25]. Fig. 3 shows the overview of our approach with five steps. Step 1: The composite process is triggered by a composite goal defined in Subsection 3, which describes customers requirements both functional and non-functional. A functional requirement is defined as $F(T_{Input}, T_{Output})$, and a nonfunctional one is users' accepted level of QoS. Step 2: Those requirements are used to discover all relevant web services, which leads to a shrunken service repository that further used by PSO as a searching space. Step 3: A weighted graph representation is randomly built up interleaving with semantic matchmaking process, where web services are discovered as graph vertices while edges are assigned with semantic matchmaking quality as weights. Step 4,5: The selection of web services performs PSO algorithm in Sect. 4.5 to find best particle location. Overall, this approach is different from [26], as we use weighted graphs as particle presentation while considering a comprehensive quality.

4.1 Semantic Matchmaking

To perform semantic matchmaking, we transfer a function match between $S_1 : S_{output} \in C_1$ and $S_2 : S_{output} \in S_b$ to a pair of concept match in 3.1. The

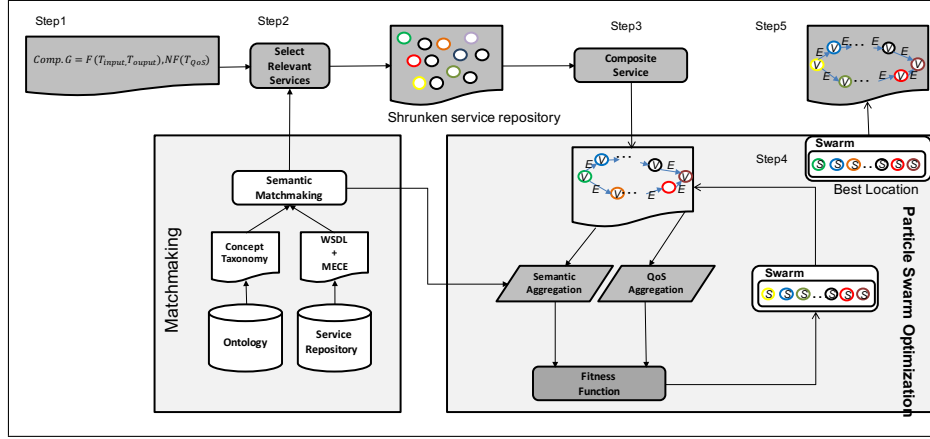


Fig. 3. Overview of the proposed approach.

matching process attempts to determine semantic matching between the source concepts of C_1 and the target concepts of C_2 . Meanwhile, the quality of matched concepts are calculated in the mode model in subSection 4.2. The purpose of semantic matchmaking is to find more component services that could also potentially satisfies the quality of QoS with considering its functional quality.

The semantic matchmaking is achieved by utilising OWL2 and OWL-S or other semantic markup languages for web services. In this paper, we use MECE (Mediation Contract Extension) [3] and OWL-DL, MECE is considered to be an alternative semantic annotation for WSDL. MECE defines the service-related inputs and outputs with parameter-related concepts. OWL-DL is a sublanguage of OWL from extension of RDF. which specifies semantic information of concepts involved in MECE.

4.2 Comprehensive Quality Model and Aggregation Matrix

In this paper, we propose a comprehensive quality model to evaluate the overall quality of semantic web service composition. This model overcome the disadvantages of current prevailing QoS-aware optimisation, in which quality of semantic matchmaking are not considered [2,17,26,27,29].

Semantic matchmaking model. Due to the discretisational characteristics of different match types driven by the cost of data type integration and manipulation [13], match types are considered to be one factor for the semantic matchmaking quality. Another factor in our proposed model is concept similarity, which could be evaluate based on edge counting method defined by [23], which is a taxonomy-based measurement for ontology matching. This formula 1 is used to estimate the similarity between parameter-related output concept and parameter-related input concept for selecting web services. Therefore, given the quality match type and the similarity of two parameters-related concepts,

the semantic matchmaking quality of matched parameters is defined by Formula 2, where the value of $q(p_{matchType})$ is set up as the same as the quality of the semantic link in [13] as 1 (Exact), 0.75 (Plugin), 0.5 (Subsume) or 0.25 (Intersection).

$$q(p_s) = \frac{2N \cdot e^{-\lambda L/D}}{N_1 + N_2} \quad (1)$$

$$q(p_{sm}) \doteq (q(p_{mt}), q(p_s)) \quad (2)$$

Further more, edges represent services connections in our weighted graph presentations, where assigned weight value $q(e_{sm})$ is considered to be semantic matching quality on edge level according to parameter aggregations. The weight value $q(e_{sm})$ is defined in 3, where $q(e_{matchType})$ and $q(e_{similarity})$ are the mean value of concept-related parameters quality in $q(p_{mt})$ and $q(p_s)$ respectively.

$$q(e_{sm}) \doteq (q(e_{mt}), q(e_s)) \quad (3)$$

Comprehensive quality model. Compared to QoS evaluation model, the comprehensive quality model is established to investigate both functional and non-functional requirements of a goal. The comprehensive quality of our service composition representation refers to QoS of service vertices and Semantic matching quality of Edges in weighted graphs. Consequently, the comprehensive quality model is defined in Formula 4, which could be further broken down into Formula 5.

$$q_{cq} \doteq (q(e_{sm}), q(v_{QoS})) \quad (4)$$

$$q_{cq} \doteq (q(e_{mt}), q(e_s), q(v_a), q(v_r), q(v_c), q(v_t)) \quad (5)$$

Quality aggregate matrix. The quality aggregation is defined based on the constructs of composite web services on the criteria of functional and non-functional properties. The quality on construct level is further calculated and by following the rules summarised in Table 1.

Table 1. Quality aggregate matrix for semantic web service composition

| Composition Construct | | Sequence | | Parallel |
|-----------------------|---------------|-------------|---------------------------|---------------------------|
| QualityFactors | Functional | $Q(e_{mt})$ | $\prod_{n=1}^m q(e_{mt})$ | $\prod_{n=1}^m q(e_{mt})$ |
| | | $Q(e_s)$ | $(\sum_{n=1}^m q(e_s))/m$ | $(\sum_{n=1}^m q(e_s))/m$ |
| | NonFunctional | $Q(v_a)$ | $\prod_{n=1}^m q(v_a)$ | $\prod_{n=1}^m q(v_a)$ |
| | | $Q(v_r)$ | $\prod_{n=1}^m q(v_r)$ | $\prod_{n=1}^m q(v_r)$ |
| | | $Q(v_c)$ | $\sum_{n=1}^m q(v_c)$ | $\sum_{n=1}^m q(v_c)$ |
| | | $Q(v_t)$ | $\sum_{n=1}^m q(v_t)$ | $max(q(v_t))$ |

4.3 Composition Weighted Graph

We defined our semantic web service composition solution as $WG = (V, E)Graph$, where V is a set of services as vertex: $V = [S1, S2...Sn]$ and E is a set of edges $E = e_1, e_2, ...e_n$. Each e is associated with $q(e_{sm})$ as weight value that mapped to a pair of quality values $q(e_{mt})$ and $q(e_s)$ on the edge level. where e_m is expressed as $(S_a, S_b) = q_{mt}, q_s$. Here we provide an example of the web service composition, which can be described in the Figure 4. The data of composite web service flows from the start to the end, where five web services involved and linked with each other using edge connections. Besides that, $q(s_{mt})$ and $q(s_s)$ are calculated for all edges $e_1, e_2, ...e_n$, and further aggregated in Q_{mt} and Q_s respectively on the construct level. In addition, Q_{QoS} is calculated by all the non-functional attributes of V using aggregation rules defined previously.

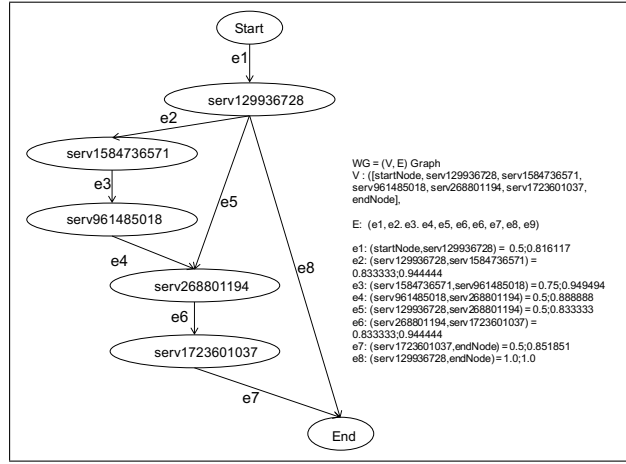


Fig. 4. Composition weighted graph solution

4.4 Fitness Calculation

The fitness value of a service composition solution is a weighted sum of all the quality components discussed previously shown in Formula 6, in which fitness value of 1 means the best comprehensive quality and 0 means the worst. As the sum of all the components' weights (w_1 to w_6) equals to 1, MT , S , A , R , T , and C must be normalised so that the fitness value falls within the range from 0 to 1. Also, the lowest T and C value represent the best component quality so that the objective function can be offset using $(1 - T)$ and $(1 - C)$ using normalisation formula 8, and normalisation formula 7 is for the rest components. Therefore, the whole problem is treated as a maximisation problem.

$$Fitness = w_1MT + w_2S + w_3A + w_4R + w_5(1 - T) + w_6(1 - C) \quad (6)$$

where $\sum_{i=1}^6 w_i = 1$

$$normalise(value) = \begin{cases} \frac{value-min}{max-min} & \text{if } max - min \neq 0. \\ 1 & \text{otherwise.} \end{cases} \quad (7)$$

$$normalise(value) = \begin{cases} \frac{max-value}{max-min} & \text{if } max - min \neq 0. \\ 1 & \text{otherwise.} \end{cases} \quad (8)$$

4.5 QoS-aware Semantic Web Service Composition Algorithm

ALGORITHM 1. Steps of the PSO-based Web service composition technique.

1. Map each relevant service to an index in the particle's position vector.
 2. Randomly initialise each particle in the swarm.
 - while** *max. iterations not met* **do**
 - forall** *particles in the swarm* **do**
 - 3. Create queue of services using the particle's position vector.
 - 4. Build a weighted graph using the queue.
 - 5. Calculate the fitness of the weighted graph.
 - if** *fitness value better than pBest* **then**
 - 6a. Assign current fitness as new pBest.
 - else**
 - 6b. Keep previous pBest.
 - 7. Assign best particle's pBest value to gBest, if better than gBest.
 - 8. Calculate the velocity of each particle according to the equation:
 - $v_{id} = v_{id} + c_1 * rand() * (p_{id} - x_{id}) + c_2 * rand() * (p_{gd} - x_{id})$
 - 9. Update the position of each particle according to the equation:
 - $x_{id} = x_{id} + v_{id}$
-

The overall algorithm investigated here is a PSO-based web service composition algorithm 1 from [26] with a different decoded algorithm 2 to generate a composition weighted graph from a queue. In the first algorithm, the idea is to translate the particle location into a service queue as an indirect representation of composition weighted graph, so finding the best fitness of the composite weighted graph is to discover the optimised location of the particle in the search space. In PSO, the dimension of the particle is set up as the same number as relevant web services, and the index of services is mapped to the location vectors in a particle, and we put services in a queue in ascending order, from which we

ALGORITHM 2. Create a composition weighted graph from a queue

Input : $I, O, queue$
Output: composition weighted graph WG

1. Create *start* vertex with outputs O and *end* vertex with inputs I .
2. Create graph WG containing the *start* vertex.
3. Create available outputs set containing *start* outputs.
- while** *available outputs do not satisfy end inputs* **do**
 4. Get next candidate from queue.
 - if** *candidate inputs are satisfied by available outputs* **then**
 5. Create edge assigned with weight.
 6. Connect vertex to graph with the edge.
 7. Remove it from queue and go back to the queue's beginning.
8. Connect end vertex.
9. Remove dangling vertices from graph WG .
10. Remove dangling edges from graph WG .

return WG .

decode a weighted graph using Algorithm 2. We select and connect service vertex to the start vertex from the service queue if the web service can satisfy the output of start vertex. Meanwhile, an Output Set is initialised and kept being updated using outputs from the new-added service vertex in the weighted graph, later on, more services vertex are connected with edges if services' inputs could be satisfied by any output in the Output Set. At last, The end vertex is connected if the updated Output Set contains all end inputs. In addition, dangling service vertex and edges should be removed.

5 Experiment Design

In this section, a quantitative evaluation approach is adopted in our experiment design. The objectives of the evaluation are (1) to measure the effectiveness of the comprehensive quality model in automated semantic web service composition approach, (2) to explore the impacts of the semantic matchmaking that contributes to overall composition quality, and (3) to compare solutions generated by QoS-aware approach with our method.

We utilise benchmark dataset web service challenge 2009 (WSC09) [11] to perform a evaluation. WSC09 provides problems where five tasks consisted in with variable size in both services, and ontologies. Therefore, it is a proper dataset for the scalability measures in our defined quality evaluation model. Table 2 presents the features of the WSC09 dataset. The number of concepts, individuals in the ontology and services in each data set is shown in the second, third, fourth column respectively. Also, we extend all the datasets with QoS attributes from service providers to enable our evaluation.

We run the experiment under computing grid comprising of almost 170 NetBSD (Unix operating system) workstations operated by the Sun Grid Engine. Experimentation is done using the approach explained in Section 4, and

Table 2. Features of the WSC09 datasets

| Dataset | No.Concept | No.Individual | No.Service |
|----------|------------|---------------|------------|
| WSC09 01 | 1578 | 3102 | 572 |
| WSC09 02 | 12388 | 24815 | 4129 |
| WSC09 03 | 18573 | 37316 | 8138 |
| WSC09 04 | 18673 | 37324 | 8301 |
| WSC09 05 | 31044 | 62132 | 15211 |

evaluated in the comprehensive quality model introduced in Subsection 4.2. The parameters were chosen based on general settings from [24] for our PSO-based approach, 30 particles evolved in 100 generations in the searching space, in which problem dimension's size equals to the number of relevant services in service repository. We run 30 times independently for each dataset. In addition, weight settings are flexibly defined by users' preferences. As part of their requirement, we configure weight of fitness function to properly balance functional side and nonfunctional side. Therefore, w_1 and w_2 are equals to 0.1 and 0.4, and w_3, w_4, w_5, w_6 are all set to 0.125 accordingly.

6 Results and Analysis

6.1 Comparison Test

In this section, we analyse the composition solution generated by using our approach comparing with QoS-aware approach. In particularly, we show that our proposed approach can produce solutions that results better matchmaking, which meets user's goal better. In particularly, we look at mean value of Q_{mt} , Q_s and Q_{QoS} in the comprehensive quality awareness approach and QoS awareness approach are compared to reveal how good our evaluation mode in Table 3. In each task, we compare Q_{mt} , Q_s and Q_{QoS} , and Q_{QoS} is normalised to make it comparable to another method in our comprehensive awareness approach

In Table 3, Task 1 represents the same solution using different methods where the values of Q_{mt}, Q_s and Q_{QoS} are all the same. It is due to the size of dataset 01 is very small in both available services and ontologies. However, As the size of dataset increase, we can recognise a trade-offs in task 2 using our evaluation approach, where a 0.021668 increase in similarity quality and a 0.00098 decrease in matchType quality with weights 0.4 and 0.1 respectively so that solution to task 2 is still considered to be an improvement in the functional quality part while a decrease value of 0.007249 in QoS. In task 3, the benefit of functional part quality is clearer with an increase value of 0.056648 and 0.078862 in similarity quality and matchType quality respectively while a slight decrease in QoS. later on, the rest tasks perform the same behaviour in Q_{mt}, Q_s and Q_{QoS} as task 3. In conclusion, we can perceive that our evaluation could find out better functional quality with a reasonable trade off in QoS.

To visually compare the results generated from two evaluation approaches, we demonstrate the differences of composition weighted graphs under two dif-

Table 3. Mean Quality for comprehensive quality-aware methods and QoS-aware approach

| | | QoS-aware Evaluation | Comprehensive Quality Evaluation |
|-------|-----------|-------------------------|-------------------------------------|
| Task1 | Q_{mt} | .232635 | .232635 |
| | Q_s | .903572 | .903572 |
| | Q_{QoS} | .578577 | .578577 |
| Task2 | Q_{mt} | .001828 | .000851↓ |
| | Q_s | .917172 | .938840↑ |
| | Q_{QoS} | .480251 | .473002↓ |
| Task3 | Q_{mt} | .191188 | .247836↑ |
| | Q_s | .951509 | .973723↑ |
| | Q_{QoS} | .493927 | .491538↓ |
| Task4 | Q_{mt} | .919678 | .943406↑ |
| | Q_s | .000000 | .000009↑ |
| | Q_{QoS} | .480818 | .472485↓ |
| Task5 | Q_{mt} | .000094 | .000109↑ |
| | Q_s | .927476 | .930975↑ |
| | Q_{QoS} | .476236 | .475128↓ |

ferent methods. The Figure 5 shows two composition weighted graphs, which are the solutions to Task 3 for comprehensive quality-aware and QoS-aware web service composition respectively. The two weighted graphs have exactly the same workflow structure, in which the different service vertex and edges are denoted in red for making a comparison.

Table 4. My caption

| | | Comprehensive-aware | QoS-aware | ΔQ |
|-----|-------------|---------------------|-----------|------------|
| e1 | $q(e_{mt})$ | 1 | 1 | 0 |
| | $q(e_s)$ | 1 | 1 | 0 |
| e2 | $q(e_{mt})$ | 0.9167 | 0.75 | |
| | $q(e_s)$ | 0.9630 | 0.9232 | |
| e3 | $q(e_{mt})$ | 0.8333 | 0.75 | |
| | $q(e_s)$ | 0.9608 | 0.8230 | |
| e4 | $q(e_{mt})$ | 0.875 | 0.75 | |
| | $q(e_s)$ | 0.9167 | 0.9090 | |
| SM | | | | |
| QoS | | 0.4915 | 0.4949 | |

6.2 Convergence Test

To analysis the effectiveness of our approach, we explore the convergency rate of the proposed method in this section to investigate the convergency of five tasks

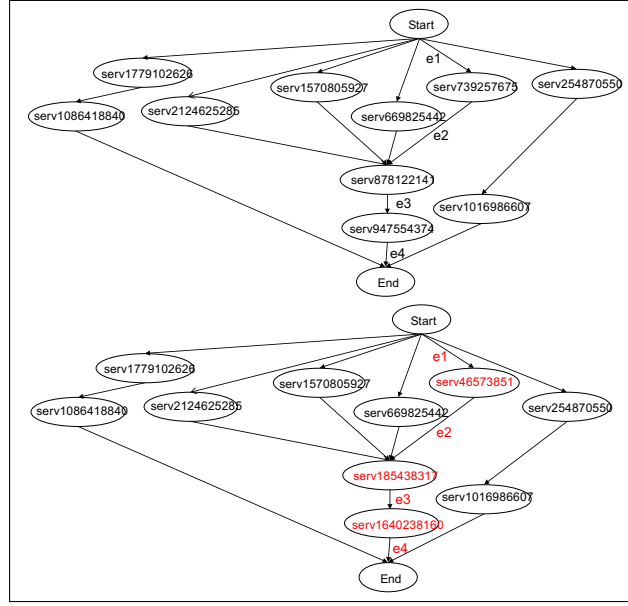


Fig. 5. Example Comparison of solutions to Task 3 under different approaches.

in WSC09. To further study an overall effectiveness of evaluation model and impacts of all involved components in the fitness function. We analysis all the performances of these components in the whole evolutionary process in Figure 6, in which the five tasks' experiment results are arranged in four groups consisting of average fitness, average matchType quality, average similarity quality and average QoS for generation 0-99 with optimum.

Firstly, the average fitness value with optimum is calculated by the average value of best fitness found in each generation over 30 independent runs. The value plotted in the first column group ranges from 0 to 1 in figure 6, and it is considered to be more optimised if its value is closer to value 1. We can see that there is a significant increase in the fitness values with optimum between generation 0 and generation 15-25, the remained generation continues to produce a steady but moderate improvement in the fitness value and eventually reach a plateau with no further changes. The same behaviour is observed over the rest tasks. Also, there is a fast convergence of the fitness value that be observed. That is simply because of many repetitive solutions in the indirect web service composition representations in the search space.

We also investigate the variation of two factors consisted in the functional quality part — average matchType quality with optimum and average similarity quality with optimum from generation 0 to 99 in second and third column groups of Figure 6. In these groups, both the average matchType quality with optimum and average similarity quality with optimum are the mean value corresponding to the best fitness value found in each generation over 30 independent runs. The

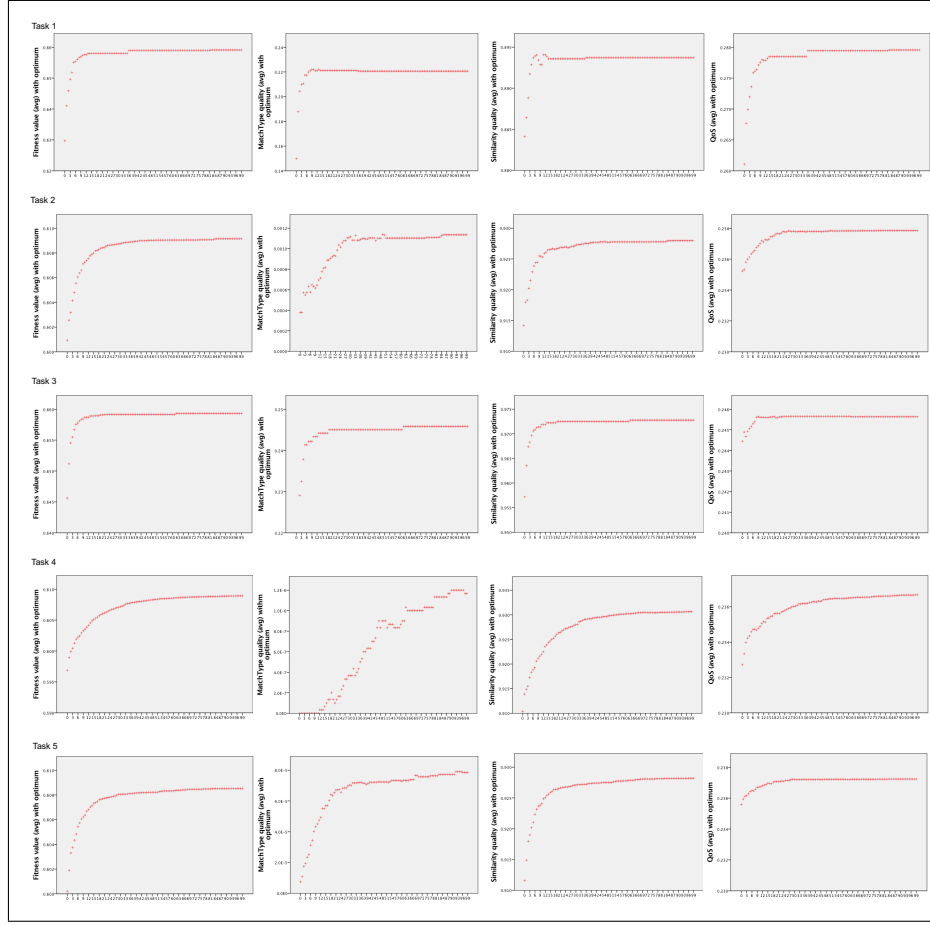


Fig. 6. Average Fitness, Average MatchType Quality, Average Similarity quality and Average QoS per generation with comprehensive quality optimum

dominant tendency of the marked value representing a similar characteristic compared to the average fitness value group trend in five tasks. However, there are some slight fluctuations over the generations, due to the single objective fitness function that prevents selecting the worst component quality without granting for the best.

Lastly, it is interesting to look at the average QoS per generation for the 30 independent runs when the behaviour of functional quality moves upward. Here average QoS with optimum is also the value corresponding to the fitness value with optimum in each generation, and we display the results in the fourth column group in Figure 6. It is obvious that the overall trend of QoS moves upward to a high constant level in five independent tasks. Additionally, we don't see too much trade-off from the QoS.

7 Conclusion

This work introduced a evaluation model for QoS-aware automated semantic automated web service composition that is based on a semantic matchmaking quality extends existing QoS. The results shows that our comprehensive quality model is proved to obtain better functional quality with a reasonable trade-off in QoS. Future works in this area could explore the comprehensive quality with multi-objectives to maximising quality of matchmaking and to optimise QoS, and improve efficiency in calculating semantic quality.

References

1. Baader, F.: The description logic handbook: Theory, implementation and applications. Cambridge university press (2003)
2. Bansal, S., Bansal, A., Gupta, G., Blake, M.B.: Generalized semantic web service composition. *Service Oriented Computing and Applications* 10(2), 111–133 (2016)
3. Bleul, S., Comes, D., Kirchhoff, M., Zapf, M.: Self-integration of web services in bpm processes. In: *Proceedings of the Workshop Selbstorganisierende, Adaptive, Kontextsensitive verteilte Systeme (SAKS)* (2008)
4. Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., Orchard, D.: Web services architecture. w3c working note. *W3C Working Notes* (2004)
5. Curbera, F., Nagy, W., Weerawarana, S.: Web services: Why and how. In: *Workshop on Object-Oriented Web Services-OOPSLA*. vol. 2001 (2001)
6. FanJiang, Y.Y., Syu, Y.: Semantic-based automatic service composition with functional and non-functional requirements in design time: A genetic algorithm approach. *Information and Software Technology* 56(3), 352–373 (2014)
7. Feng, Y., Ngan, L.D., Kanagasabai, R.: Dynamic service composition with service-dependent qos attributes. In: *Web Services (ICWS), 2013 IEEE 20th International Conference on*. pp. 10–17. IEEE (2013)
8. Fensel, D., Facca, F.M., Simperl, E., Toma, I.: *Semantic web services*. Springer Science & Business Media (2011)
9. Fensel, D., Lausen, H., Polleres, A., De Bruijn, J., Stollberg, M., Roman, D., Domingue, J.: *Enabling semantic web services: the web service modeling ontology*. Springer Science & Business Media (2006)
10. Huang, Z., Jiang, W., Hu, S., Liu, Z.: Effective pruning algorithm for qos-aware service composition. In: *2009 IEEE Conference on Commerce and Enterprise Computing*. pp. 519–522. IEEE (2009)
11. Kona, S., Bansal, A., Blake, M.B., Bleul, S., Weise, T.: Wsc-2009: a quality of service-oriented web services challenge. In: *2009 IEEE Conference on Commerce and Enterprise Computing*. pp. 487–490. IEEE (2009)
12. Lausen, H., Farrell, J.: Semantic annotations for wsdl and xml schema. *W3C recommendation, W3C* pp. 749–758 (2007)
13. Lécué, F.: Optimizing qos-aware semantic web service composition. In: *International Semantic Web Conference*. pp. 375–391. Springer (2009)
14. Lécué, F., Léger, A.: A formal model for semantic web service composition. In: *International Semantic Web Conference*. pp. 385–398. Springer (2006)
15. Ma, H., Wang, A., 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*, pp. 180–205. Springer (2015)

16. 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, 2007–04 (2004)
17. Mier, P.R., Pedrinaci, C., Lama, M., Mucientes, M.: An integrated semantic web service discovery and composition framework (2015)
18. Moghaddam, M., Davis, J.G.: Service selection in web service composition: A comparative review of existing approaches. In: *Web Services Foundations*, pp. 321–346. Springer (2014)
19. O’Leary, D.: Review: Ontologies: A silver bullet for knowledge management and electronic commerce. *The Computer Journal* 48(4), 498–498 (2005)
20. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.: Semantic matching of web services capabilities. In: *International Semantic Web Conference*. pp. 333–347. Springer (2002)
21. 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*. pp. 3–12. IEEE (2003)
22. Sahai, A., Machiraju, V., Sayal, M., Van Moorsel, A., Casati, F.: Automated sla monitoring for web services. In: *International Workshop on Distributed Systems: Operations and Management*. pp. 28–41. Springer (2002)
23. Shet, K., Acharya, U.D., et al.: A new similarity measure for taxonomy based on edge counting. *arXiv preprint arXiv:1211.4709* (2012)
24. Shi, Y., et al.: Particle swarm optimization: developments, applications and resources. In: *evolutionary computation, 2001. Proceedings of the 2001 Congress on*. vol. 1, pp. 81–86. IEEE (2001)
25. da Silva, A.S., Ma, H., 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)*. pp. 3127–3134. IEEE (2014)
26. da Silva, A.S., Mei, Y., Ma, H., Zhang, M.: Particle swarm optimisation with sequence-like indirect representation for web service composition. In: *European Conference on Evolutionary Computation in Combinatorial Optimization*. pp. 202–218. Springer (2016)
27. da Silva, A., Ma, H., Zhang, M.: Graphevol: A graph evolution technique for web service composition. In: Chen, Q., Hameurlain, A., Toumani, F., Wagner, R., Decker, H. (eds.) *Database and Expert Systems Applications, Lecture Notes in Computer Science*, vol. 9262, pp. 134–142. Springer International Publishing (2015)
28. Wu, Z., Palmer, M.: Verbs semantics and lexical selection. In: *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*. pp. 133–138. Association for Computational Linguistics (1994)
29. Yu, Y., Ma, H., Zhang, M.: An adaptive genetic programming approach to qos-aware web services composition. In: *2013 IEEE Congress on Evolutionary Computation*. pp. 1740–1747. IEEE (2013)
30. Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q.Z.: Quality driven web services composition. In: *Proceedings of the 12th international conference on World Wide Web*. pp. 411–421. ACM (2003)