Comprehensive Quality-Aware Automated Semantic Web Service Composition

Chen Wang[1] Hui Ma[1] Aaron Chen[1] Sven Hartmann[2]

[1] School of Engineering and Computer Science,
Victoria University of Wellington, New Zealand
{chen.wang, hui.ma, aaron.chen}@ecs.vuw.ac.nz

[2] Department of Informatics,
Clausthal University of Technology, Germany
sven.hartmann@tu-clausthal.de

Abstract. Web service composition has been a prevailing research direction in recent years. There are two major challenges faced by researchers, semantic matchmaking and Quality of Service (QoS) optimisation. Semantic matchmaking aims to discover interoperable web services that can interact with each other by their resources described semantically. QoS optimisation aims to optimise the non-functional requirements of service users, such as minimum cost and maximum reliability. To meet the requirements of service users, both semantic matchmaking quality and QoS should be considered simultaneously. Most existing works on web service composition, however, focus only on one of these two aspects. Therefore, we propose a comprehensive quality model that takes both semantic matchmaking quality and QoS into account with the aim of achieving a more desirable balance of both sides. Further, we develop a PSO-based service composition approach with explicit support for the proposed comprehensive quality model. We also conduct experiments to explore the effectiveness of our PSO-based approach and the desirable balance achieved by using our comprehensive quality model.

1 Introduction

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 [5]. *Interoperability* of web services presents challenge in syntactic and semantic dimensions. The syntactic dimension is covered by the XML-based technologies, such as $WSDL$, $SOAP$. The semantic dimension enables a better collaboration through ontology-based semantics, such as OWL-S, WSML, and SAWSDL [12]. *Semantic web services composition* is distinguished from the syntactic service composition, the resources of semantic web services are described semantically to enable a better interoperability for chaining web services. Another challenge is related to QoS optimisation. This problem gives birth to *QoS-aware service composition* that aims to find composition solutions with optimised QoS.

Existing works on service composition focus mainly on addressing only one challenge above. In these works, huge efforts have been devoted to QoS-aware web service compositions assuming a pre-defined abstract workflow is given. This is generally considered as a *semi-automated web service composition* approach. Generating composition plans automatically in discovering and selecting suitable web services is a NP-hard problem [10]. In the past few years, many approaches [6,8,14,19,20,21] to QoS-aware web service composition employ Evolutionary Computation (EC) techniques to automatically generate composition solutions. Genetic Programming (GP) based approaches produce promising results, but these approaches often require repairing or penalising the solutions [8,21]. However, Particle Swarm Optimisation (PSO) is considered to be an easy way to maintain the correctness of solutions in solving combinatorial optimisation problems [19]. All these works have enabled an *automatic web service composition*, but do not optimise QoS and quality of semantic matchmaking simultaneously to achieve a desirable balance on both sides.

The overall goal of this paper is to *develop a PSO-based approach to comprehensive quality-aware automated semantic web service composition that satisfactorily optimises both QoS and semantic matchmaking quality*. Particularly, this paper extends existing works on QoS-aware service composition by considering jointly optimising QoS and semantic matchmaking quality, which is proposed as a comprehensive quality model. We will achieve three objectives in this work:

1. To propose a comprehensive quality model that addresses QoS and semantic matchmaking quality simultaneously with a desirable balance on both sides.
2. To propose a PSO-based service composition approach using the proposed comprehensive quality model. To do that, we aim to find a service candidate queue that can be decoded into a service composition with near-optimal comprehensive quality.
3. To address the effectiveness of our PSO-based approach and a desirable balance achieved using our comprehensive quality model, we first compare our PSO-based approach with one recent GP-based approach [8] using our proposed quality model, and then compare our proposed quality model with one widely used QoS model using our proposed PSO-based approach.

2 Related Work

While web service composition has attracted much research over the last decade, most efforts focus on either semantic web service composition [1,3,9] or QoS-aware web service composition [6,8,14,19,20,21], Only very few works address both semantic matchmaking quality and QoS for web service composition problems. To the best of our knowledge, [4,7,13] reported about first attempts that consider both aspects together.

Semantic web service composition [1,3,9] captures semantic descriptions of the parameters of web services using some kind of logic (i.e., description logic) to ensure the interoperability of web services. In these approaches, the goal is often to minimise the number of services or the size of a graph representation for

a web service composition to obtain optimised composition solutions. However, these approaches do not guarantee an optimised QoS of service compositions.

QoS-aware web service composition, on the other side, has been studied using classical optimisation techniques or EC techniques for finding near-optimised solutions. [14] proposes a local optimisation and enumeration method, where a small number of promising candidates related to each task are considered by local selection, and composition solutions are enumerated to reach the near optimal QoS. EC techniques are widely used to automatically generate solutions with optimal QoS. [6] employs a modified Genetic Algorithm (GA) using a binary string as an individual, which demands to be decoded into composition solutions. [21] uses GP for finding near-optimal solutions that are reached by penalising infeasible solutions using a fitness function. A hybrid approach employing a greedy search and GP is introduced in [8] to generate functionally correct tree-based representations, which are transformed from graph-based representations. [20] introduces a promising graph-evolutionary approach to eliminate the transformation process. [19] proposes an indirect PSO-based approach, where a service queue is used as an indirect representation that is decoded into a directed acyclic graph. However, these QoS-aware approaches [6,14,8,19,20,21] do not consider the semantic matchmaking quality of service compositions.

Only a few works [4,7,13] consider both semantic matchmaking quality and QoS simultaneously. [7] proposes a semi-automated web service composition using GA to encode a given abstract service workflow, where the evaluation of semantic matchmaking quality requires a complete and formal definition of ontology using description logic, which is associated to the resources of web services. Another GA-based approach [4] utilise process description language to encode pre-stored cases-based workflows, where workable services are composed to complete this workflow. An automated immune-inspired web service composition approach [13] employs a clonal selection algorithm to proliferate decoded planning graphs, but this approach is only evaluated with some simple cases.

In summary, despite a large number of approaches for semantic web service composition and QoS-aware service composition approaches, there is a lack of a fully automated semantic service composition approach to optimise semantic matchmaking quality and QoS simultaneously.

3 Problem Description and Comprehensive Quality Model

Our goal is to develop a PSO-based approach for automatically generating good service compositions. 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. We therefore propose a *comprehensive quality model* in considering semantic matchmaking quality and QoS simultaneously.

We consider a *semantic web service* (*service*, for short) as a tuple $S = (I_S, O_S, QoS_S)$ where I_S is a set of service inputs that are consumed by S, O_S is a set of service outputs that are produced by S, and $QoS_S = \{t_S, c_S, r_S, a_S\}$ is a set of non-functional attributes of S. The inputs in I_S and outputs in O_S are parameters modeled through concepts in a domain-specific ontology $\mathcal{O}$. The attributes t_S, c_S, r_S, a_S refer to the response time, cost, reliability, and availability of service S, respectively. These four QoS attributes are most commonly used [22].

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

A service composition is commonly represented as a *directed acyclic graph* (DAG). Its nodes correspond to the services in the composition. Two services S and S' are connected by an edge e if some outputs of S service serve as inputs for S'. Apparently, such outputs and inputs must semantically match to ensure the correct execution of the service composition. The mechanism to compose services relies on the semantic descriptions of inputs and outputs, which enables inputs of services to be matched by outputs of other services. The following *matchmaking types* are often used to describe the level of a match [11]: For concepts a, b in $\mathcal{O}$ the *matchmaking* returns *exact* if a and b are equivalent ($a \equiv b$), *plugin* if a is a sub-concept of b ($a \sqsubseteq b$), *subsume* if a is a super-concept of b ($a \sqsupseteq b$), and $fail$ if none of previous matchmaking types is returned.

In this paper we are only interested in robust compositions where only *exact* and *plugin* matches are considered, see [7]. As argued in [7] *plugin* matches are less preferable than *exact* matches due to the overheads associated with data processing. We suggest to consider the semantic similarity of concepts when comparing different *plugin* matches. For concepts a, b in $\mathcal{O}$ the *semantic similarity* $sim(a, b)$ is calculated based on the edge counting method defined in Eq. (1) from [15], where N_a, N_b and N_c measure the distances from concept a, concept b, and the closest common ancestor c of a and b to the top concept of the ontology $\mathcal{O}$, respectively.

$$sim(a, b) = \frac{2N_c \cdot e^{-\lambda L/D}}{N_a + N_b} \tag{1}$$

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

Given a service request $T = (I_T, O_T)$, we represent a service composition solution for T with services $S_1, \ldots, S_n$ by a weighted DAG, $WG = (V, E)$ with node set $V = \{Start, S_1, S_2, \ldots, S_n, End\}$ and edge set $E = \{e_1, e_2, ...e_m\}$. $Start$ and

End are two special services defined as $Start = (\emptyset, I_T, \emptyset)$ and $End = (O_T, \emptyset, \emptyset)$ that account for the input and output requirements given by the request. Each edge e from a service S to a service S' means that service S produces an output $a \in O_S$ that is matched (*exact* or *plugin*) to an input $b \in I_{S'}$ to be consumed by service S' in the composition. Based on the matchmaking type, the *semantic matchmaking quality* of edge e can be defined as follows:

$$type_e = \begin{cases} 1 & \text{if } a \equiv b \ (exact \text{ match}), \\ p & \text{if } a \sqsubseteq b \ (plugin \text{ match}) \end{cases} \tag{2}$$

$$sim_e = sim(a, b) = \frac{2N_c}{N_a + N_b} \tag{3}$$

with a suitable parameter $p, 0 < p < 1$, see Section **??**. However, if more than one pair of matched output and input exist from service S to service S', $type_e$ and sim_e will take on their average values.

The *semantic matchmaking quality* of the service composition can be obtained by aggregating over all edges in E as follows:

$$MT = \prod_{j=1}^{m} type_{e_j} \tag{4}$$

$$SIM = \frac{1}{m} \sum_{j=1}^{m} sim_{e_j} \tag{5}$$

The QoS of the service composition can be obtained by aggregating the QoS values of the participating services [8]. For a service composition with services $S_1, S_2, ...S_n$ we obtain the reliability $R = \prod_{k=1}^{n} r_{S_k}$, the availability $A = \prod_{k=1}^{n} a_{S_k}$, the cost $C = \sum_{k=1}^{n} c_{S_k}$, and the response time T is the time of most time-consuming path in the composition, i.e.,

$$T = MAX\{\sum_{k=1}^{\ell_j} t_{S_k} |\ j \in \{1, \ldots, m\} \text{ and } \ell_j \text{ is the length of a path } P_j\}.$$

When multiple quality criteria are involved into decision making, then the overall fitness of a solution can be defined as a weighted sum of the individual criteria:

$$Fitness = w_1 \hat{MT} + w_2 \hat{SIM} + w_3 \hat{A} + w_4 \hat{R} + w_5(1 - \hat{T}) + w_6(1 - \hat{C}) \tag{6}$$

with $\sum_{k=1}^{6} w_k = 1$. We call this objective function the *comprehensive quality model* for service composition. The weights can be adjusted according to users' preferences. We normalise MT, SIM, A, R, T, and C so that the function value falls within the range from 0 to 1 using Eq. (7). To simplify the presentation we also use the notation $(Q_1, Q_2, Q_3, Q_4, Q_5, Q_6) = (MT, SIM, A, R, T, C)$. MT and SIM have minimum value 0 and maximum value 1. The minimum and maximum value of A, R, T, and C are calculated across all task-related candidates in the service repository $\mathcal{SR}$ using the greedy search in [8,18].

$$\hat{Q}_k = \begin{cases} \frac{Q_k - Q_{k,min}}{Q_{k,max} - Q_{k,min}} & \text{if } k = 1, \ldots, 4 \text{ and } Q_{k,max} - Q_{k,min} \neq 0, \\ \frac{Q_{k,max} - Q_k}{Q_{k,max} - Q_{k,min}} & \text{if } k = 5, 6 \text{ and } Q_{k,max} - Q_{k,min} \neq 0, \\ 1 & \text{otherwise.} \end{cases} \quad (7)$$

To solve the composition task satisfactorily our goal is to maximize the objective function in Eq. (6).

4 PSO-based Approach to Comprehensive Quality-Aware Automated Semantic Web Service Composition

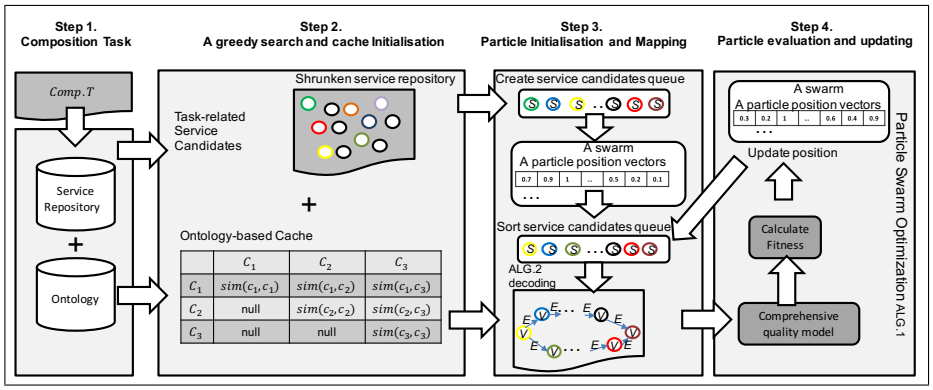


Fig. 1: An overview of our PSO-based approach to comprehensive quality-aware automated semantic web service composition.

4.1 An Overview of our PSO-based Approach

As PSO has shown promise in solving combinatorial optimisation problems, we propose a PSO-based approach to comprehensive quality-aware automated semantic web service composition. Fig. 1 shows an overview of our approach consisting of four steps:

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

Step 2: The composition task is used to discover all task-related service candidates using a greedy search algorithm adopted from [8], which contributes to a shrunken service repository. This greedy search algorithm keeps adding outputs of the invoked services as available outputs (initialised with I_T) , and these available outputs are used to discover task-related services from a service repository and updated with the outputs of these discovered services. This operation is repeated until no service is satisfied by the available outputs. During the greedy search, an ontology-based cache (*cache*) is initialised, which stores the concept similarities of matched inputs and outputs of task-related candidates. This *cache*

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

Step 3 and Step 4: These two steps follow the standard PSO steps [16] except for some differences in particles mapping and decoding processes. In particular, these two differences are related to sorting a created service queue using service-to-index mapping for a particle' position vectors and evaluating the fitness of a particle after decoding this service queue into a WG respectively. Those differences are further addressed in Algorithms 1 and 2 in Section 4.2.

4.2 The Algorithms for our PSO-based Approach

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

ALGORITHM 1. Steps of PSO-based service composition technique [19].

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

Algorithm 2 is a forward graph building algorithm extended from [2]. This algorithm takes one input, a sorted service queue from step 5 of Algorithm 1. Note that different service queues may lead to different WGs. In addition. I_T, O_T and *cache* are also taken as the inputs. Firstly, *Start* and *End* are added to V

of WG as an initialisation, and $OutputSet$ is also created with I_T. The following steps are repeated until O_T can be satisfied by $Outputset$ or the service queue is $null$. If all the inputs I_S of the first popped S from $queue$ can be satisfied by provided outputs from $OutputSet$, this S is added to V and its outputs are added to $OutputSet$, and S is removed from $queue$. Otherwise, the second popped S from $queue$ is considered for these operations. Meanwhile, e is created with $type_e$ and sim_e if S is added, and calculated using information provided from $cache$. This forward graph building technique could lead to more services and edges connected to the WG, these redundancies should be removed before WG is returned.

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

```
   Input  : I_T, O_T, queue, cache
   Output: WG
 1: WG = (V, E);
 2: V ← {Start, End } ;
 3: OutputSet ← {I_T};
 4: while O_T not satisfied by OutputSet do
 5:     foreach S in queue do
 6:         if I_S satisfied by OutputSet then
 7:             insert S into V ;
 8:             adjoin O_S to OutputSet;
 9:             queue.remove S;
10:             e ← calculate type_e, sim_e using cache;
11:             insert e into E;
12: remove dangling nodes and edges from WG;
13: return WG;
```

5 Experiment Study

In this section, we employ a quantitative evaluation approach with a benchmark dataset used in [8,18], which is an augmented version of Web Service Challenge 2009 (WSC09) including QoS attributes. Two objectives of this evaluation are to: (1) evaluate the effectiveness of our PSO-based approach, see comparison test in Section 5.1. (2) evaluate the effectiveness of our proposed comprehensive quality model to achieve a desirable balance on semantic matchmaking quality and QoS, see comparison test in Section 5.2.

The parameters for the PSO are chosen from the settings from [16], In particular, PSO population size is 30 with 100 generations. We run 30 times independently for each dataset. We configure the weights of fitness function to properly balance semantic matchmaking quality and QoS. Therefore, w_1 and w_2 are set equally to 0.25, and w_3, w_4, w_5, w_6 are all set to 0.125. The p of $type_e$ is set to 0.75 (*plugin* match) according to [7]. In general, weight settings and parameter p are decided by users' preferences.

5.1 Comparison Test for GP-based vs. PSO-based approach

To evaluate the effectiveness of our proposed PSO-based approach, we compare our PSO-based method with one recent GP-based approach [8] using our proposed comprehensive quality model. We extend this GP-based approach by measuring the semantic matchmaking quality between parent nodes and children nodes. To make a fair comparison, we use the same number of evaluations (3000 times) for these two approach. We set the parameters of that GP-based approach as 30 individuals and 100 generations, which is considered to be proper settings referring to [17].

The first column of Table 1 shows five tasks from WSC09. The second and third column of Table 1 show the original service repository size and the shrunk service repository size after the greedy search respectively regarding the five tasks. This greedy search helps reducing the original repository size significantly, which contributes to a reduced searching space. The fourth and fifth column of Table 1 show the mean fitness values of 30 independent runs accomplished by two methods. We employ independent-samples T tests to test the significant differences in mean fitness value. The results show that the PSO-based approach outperforms the existing GP-based approach in most cases except Task 3. Note that all p-values are consistently smaller than 0.01. Using our PSO-based approach, small changes to sorted queues (particles in PSO) could lead to big changes to the composition solutions. This enables the PSO-based approach to escape from local optima more easily than the GP-based approach.

Table 1: Mean fitness values for comparing GP-based approach

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

5.2 Comparison Test for Comprehensive Quality Model vs. QoS Model

Recently, a QoS Model, $Fitness = w_1\hat{A} + w_2\hat{R} + w_3(1-\hat{T}) + w_4(1-\hat{C})$, where $\sum_{i=1}^{4} w_i = 1$, is widely used for QoS-aware web service composition [8,19,20]. To show the effectiveness of our proposed comprehensive quality model, we compare the best solutions found by this QoS model and our comprehensive model using our PSO-based approach. We record and compare the mean values of both SM ($SM = 0.5\hat{MT} + 0.5\hat{SIM}$) and QoS($QoS = 0.25\hat{A} + 0.25\hat{R} + 0.25(1-\hat{T}) + 0.25(1-\hat{C})$) of best solutions over 30 independent runs. To make the comparison informative, all these recorded values have been normalised from 0 to 1, and compared using independent-samples t tests, see Table 2.

We observe an interesting pattern from Table 2. The mean values of QoS using QoS model are significantly higher than those using comprehensive quality model for Tasks 2, 3, 4 and 5. However, the mean value of SM using the

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

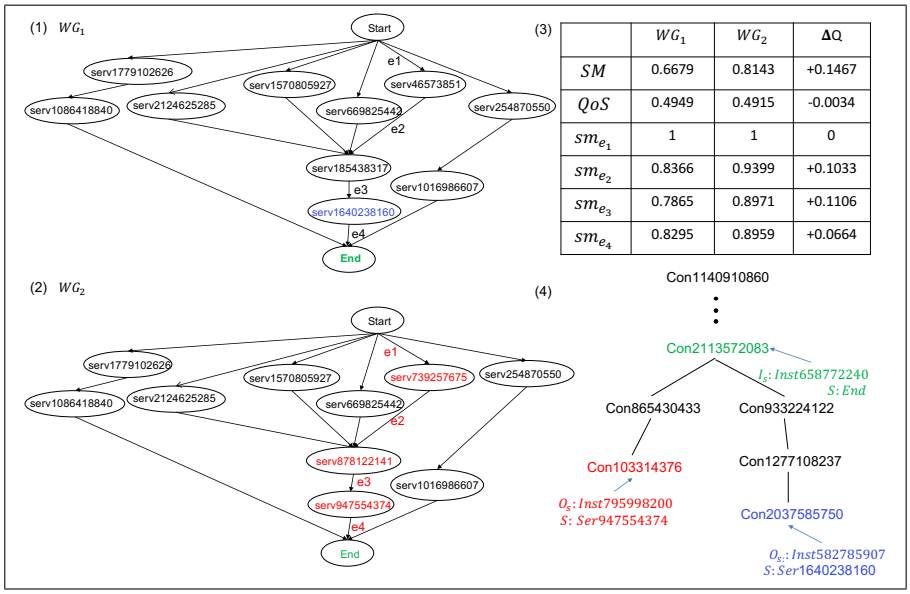
Table 2: Mean values of SM, QoS and sum of SM and QoS for QoS model and comprehensive quality model using PSO-based approach

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

5.3 Further Discussion

To analyse the effectiveness of achieving a good comprehensive quality at the expense of slightly reduced QoS, we demonstrate two best solutions produced using Task 3 as an example. Fig. 2 (1) and (2) show two weighted DAGs WG_1 and WG_2 that have been obtained as the best service compositions solutions based on the QoS model and on the comprehensive quality model, respectively. Both WGs have exactly the same service workflow structure, but some service vertices and edges denoted in red are different. To better understand these differences, we list the overall semantic matchmaking quality SM, overall QoS and semantic matchmaking quality associated to these different edges in WG_1 and WG_2. (Note: $sm_{e_n} = 0.5type_{e_n} + 0.5sim_{e_n}$), where ΔQ reveals the gain (positive ΔQ) or a loss (negative ΔQ) of the listed qualities for our comprehensive quality model. Therefore, we achieve a comprehensive quality gain of 0.1433, which is the result of a gain in semantic matchmaking quality (+0.1467) and a loss in QoS (-0.0034). To understand the improvement of semantic matchmaking quality from these numbers, we pick up e_4 that is associated with the smallest ΔQ. The e_4 of WG_1 and WG_2 has two different source nodes, $Ser1640238160$ and $Ser947554374$, and two the same end nodes. $Ser1640238160$ and $Ser947554374$ are services with output parameters $Inst582785907$ and $Inst795998200$ corresponds to two concepts $Con2037585750$ and $Con103314376$ respectively in the given ontology

shown in Fig. 2 (4). As $Inst658772240$ is a required parameter of End, and related to concept $Con2113572083$, $Inst795998200$ is closer to the required output $Inst658772240$ than $Inst582785907$. Therefore, $Ser947554374$ is selected with a better semantic matchmaking quality compared to $Ser1640238160$.



	WG_1	WG_2	ΔQ
SM	0.6679	0.8143	+0.1467
QoS	0.4949	0.4915	-0.0034
sm_{e_1}	1	1	0
sm_{e_2}	0.8366	0.9399	+0.1033
sm_{e_3}	0.7865	0.8971	+0.1106
sm_{e_4}	0.8295	0.8959	+0.0664

Fig. 2: An example for the comparison of the best solutions obtained based on the QoS model and on the comprehensive quality model for Task 3.

6 Conclusion

In this work, we propose an effective PSO-based approach to comprehensive quality-aware semantic web service composition, which also has shown promise in achieving a better comprehensive quality in terms of a combination of semantic matchmaking quality and QoS compared to existing works. Future works can investigate multi-objective EC techniques to produce a set of composition solutions for the situations when the quality preference is not known.

References

1. Bansal, S., Bansal, A., Gupta, G., Blake, M.B.: Generalized semantic web service composition. Service Oriented Computing and Applications 10(2), 111–133 (2016)
2. Blum, A.L., Furst, M.L.: Fast planning through planning graph analysis. Artificial intelligence 90(1), 281–300 (1997)
3. Boustil, A., Maamri, R., Sahnoun, Z.: A semantic selection approach for composite web services using owl-dl and rules. Service Oriented Computing and Applications 8(3), 221–238 (2014)
4. 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)

5. Fensel, D., Facca, F.M., Simperl, E., Toma, I.: Semantic web services. Springer Science & Business Media (2011)
6. Gupta, I.K., Kumar, J., 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. pp. 1–6. IEEE (2015)
7. Lécué, F.: Optimizing qos-aware semantic web service composition. In: International Semantic Web Conference. pp. 375–391. Springer (2009)
8. 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)
9. Mier, P.R., Pedrinaci, C., Lama, M., Mucientes, M.: An integrated semantic web service discovery and composition framework (2015)
10. 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)
11. 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)
12. Petrie, C.J.: Web Service Composition. Springer (2016)
13. Pop, C.B., Chifu, V.R., Salomie, I., Dinsoreanu, M.: Immune-inspired method for selecting the optimal solution in web service composition. In: International Workshop on Resource Discovery. pp. 1–17. Springer (2009)
14. Qi, L., Tang, Y., Dou, W., Chen, J.: Combining local optimization and enumeration for qos-aware web service composition. In: Web Services (ICWS), 2010 IEEE International Conference on. pp. 34–41. IEEE (2010)
15. Shet, K., Acharya, U.D., et al.: A new similarity measure for taxonomy based on edge counting. arXiv preprint arXiv:1211.4709 (2012)
16. 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)
17. da Silva, A.S., Ma, H., Zhang, M.: A gp approach to qos-aware web service composition including conditional constraints. In: Evolutionary Computation (CEC), 2015 IEEE Congress on. pp. 2113–2120. IEEE (2015)
18. da Silva, A.S., Ma, H., Zhang, M.: Genetic programming for qos-aware web service composition and selection. Soft Computing pp. 1–17 (2016)
19. 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)
20. da Silva, A., Ma, H., Zhang, M.: Graphevol: A graph evolution technique for web service composition. In: Database and Expert Systems Applications, vol. 9262, pp. 134–142. Springer International Publishing (2015)
21. 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)
22. 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)