Towards Scalability of Quality Driven Semantic Web Service Composition

Freddy Lécué, Nikolay Mehandjiev
The University of Manchester
Booth Street East, Manchester, UK
{(firstname.lastname)@manchester.ac.uk}

Abstract

Optimizing semantic web service compositions is known to be NP-hard, so most approaches restrict the number of services and offer poor scalability. We address the scalability issue by selecting compositions which satisfy a set of constraints rather than attempting to produce an optimal composition. Firstly, we define constraints within an innovative and extensible quality model designed to balance semantic fit (or functional quality) with quality of service (QoS) metrics. The semantic fit criterion evaluates the quality of semantic links between the semantic description of Web services parameters, whilst QoS focuses on non-functional criteria of services. Coupling these criteria allows us to further constrain and select valid compositions. To allow the use of this model in the context of millions of services as foreseen by the strategic EC-funded project SOA4All, we i) formulate the selection problem as a Constraint Satisfaction Problem and ii) test the use of a stochastic search method. Finally we compare the latter with state-of-the-art approaches.

1 Introduction

The Semantic Web [5], where the semantic content of the information is tagged using machine-processable languages such as the Web Ontology Language (OWL) [25], is considered to provide many advantages over the current "formatting only" version of the World-Wide-Web. OWL is based on concepts from Description Logics [3] and ontologies, formal conceptualization of a particular domain. This allows us to describe the semantics of services [26], e.g., their functionality in terms of input, output parameters, preconditions, effects and invariants. Such descriptions can then be used for automatic reasoning about services and automating their use to accomplish “intelligent” tasks including selection, discovery and composition.

We focus on web service composition and more specifically on its functional aspects, where a set of services is composed to achieve a goal on the basis of the semantic similarities between input and output parameters as indicators of service functionality. To measure semantic similarity, we use the concept of (functional) semantic link [18], defined as a semantic connection between an output and an input parameter of two services. Web service compositions could thus be estimated and ranked not only along well known non functional parameters such as Quality of Services (QoS) [6] but also along the dimension of semantic similarity as indicator of functional fit [17]. In this paper we propose to unify both types of criteria in an innovative and extensible model that could be used to estimate, select and optimize the quality of any semantic Web service composition.

Maximizing the quality of service composition using this model is essentially a multi-objective optimization problem with constraints on quality of services and their semantic links. Such a problem is known to be NP-hard [22] and untractable [17, 30] in a context of a large number of services. Indeed most approaches have been shown to have poor scalability in terms of time taken to compute optimal compositions when the size of the initial set of services grows. Such a case can arise in the future semantic web, where millions of semantic services will be accessible globally. This is the vision of SOA4All, a strategic EC-funded project aiming to create the tools, languages and conceptual techniques where scalability is one of its main objective.

Rapid computation of (non necessarily optimal) compositions is especially important for interactive systems providing service composition facilities for end users, where long delays may be unacceptable. A typical scenario would involve an end-user retrieving a template for a desired composition, and specifying some constraints in addition to the ones already in the template. The software should then propose a combination of services which satisfies these constraints. Such a case can occur in real-time even for a very complex composition template with a large number of alternative matching services. In this work we address this scalability issue by selecting a composition (among a large number achieving the same goal) satisfying some constraints (defined along the innovative quality model) rather than
computing the optimal composition. To this end we i) formulate quality-driven semantic web service composition as a Constraint Satisfaction Problem (CSP) and ii) test the use of a stochastic search method. Finally we compare the latter with state-of-the-art approaches.

The remainder of this paper is organised as follows. In the next section we briefly review i) semantic links, ii) their common descriptions and iii) the web service composition model. Section 3 introduces the quality criteria for quality-driven semantic web service composition. Section 4 details its CSP formalization and tests the use of a stochastic search method. Section 5 reports and discusses results from the experimentations. Section 6 briefly comments on related work. Finally section 7 draws some conclusions.

2 Preliminaries

We describe how semantic links introduced in [18] can be used to model web service composition and how the concept of Common Description can be used to measure the semantic fit of the links.

2.1 Semantic Links between Web Services

In the semantic web, input and output parameters of services referred to concepts in a common ontology\(^1\) or Terminology \(T\), where the OWL-S profile [1], WSMO capability [10] or SA-WSDL [14] can be used to describe them (through semantic annotations). At functional level, web service composition consists in retrieving some services referred to concepts in a common ontology or Ter-

\[ \text{Example 1. (Semantic Link and Matching Type)} \]

Suppose \(s_{l,2}\) (Figure 3) is a semantic link between two services \(s_1\) and \(s_2\) such that the output parameter NetworkConnection of \(s_1\) is (semantically) linked to the input parameter SlowNetworkConnection of \(s_2\). According to Figure 2 and the definition of matching types, this link is valued by a Subsume matching type since NetworkConnection \(\sqsubseteq\) SlowNetworkConnection.

The matching function \(\text{Sim}_T\) enables, at design time, the discovery of certain levels of semantic compatibilities (i.e., Exact, PlugIn, Subsume, Intersection) and incompatibilities (i.e., Disjoint) among service descriptions.

2.2 Common Description

Besides computing the matching type of a semantic link, [16] suggest to compute a finer level of information i.e., the Extra and Common Descriptions between Out\(_i\) and In\(_j\) of a semantic link \(s_{i,j}\). They adapt the definition of Concept Abduction [8] (Definition 1) in the context of web service composition. Then, they compare the service parameters defined in A\(\mathcal{LN}\) Description Logic (DL).

\[ \text{Definition 1. (Concept Abduction)} \]

Let \(\mathcal{L}\) be a DL, \(C, D\) be two concepts in \(\mathcal{L}\), and \(T\) be a set of axioms in \(\mathcal{L}\). A Concept Abduction Problem (CAP), denoted as \(\langle \mathcal{L}, C, D, T \rangle\), is finding a concept \(H \in \mathcal{L}\) such that \(T \models C \sqcap H \sqsubseteq D\).

This produces a compact representation of the “difference” \(H\) between descriptions Out\(_i\) and In\(_j\) of a semantic link \(s_{i,j}\). The Extra Description \(H\) is defined by \(T \models \text{Out\(_i\)} \sqcap H \sqsubseteq \text{In\(_j\)}\), hence it provides a solution of the Concept Abduction problem \(\langle \mathcal{L}, \text{Out\(_i\)}, \text{In\(_j\)}, T \rangle\). In other words \(H\) refers to information required by \(\text{In\(_j\)}\) but not provided by \(\text{Out\(_i\)}\) to ensure a correct data flow between services \(s_i, s_j\). The Common Description of \(\text{Out\(_i\)}\) and \(\text{In\(_j\)}\), defining as their Least Common Subsumer [4] lcs, refers to information required by \(\text{In\(_j\)}\) and effectively provided by \(\text{Out\(_i\)}\).

\[ \text{NetworkConnection} \equiv \text{\texttt{\textbackslash nNetPro.Provider}} \sqcap \text{\texttt{\textbackslash nNetSpeed.Speed}} \]
\[ \text{SlowNetworkConnection} \equiv \text{NetworkConnection} \sqcap \text{\texttt{\textbackslash nNetSpeed.Speed}} \equiv \text{Adsl1M} \]
\[ \text{Adsl1M} \equiv \text{\textbackslash nSpeed.Speed} \sqcap \text{\textbackslash n1mBytes} \]

\[ \text{Figure 2. Sample of an A\(\mathcal{LN}\) Terminology } T. \]

\[ \text{Example 2. (Extra & Common Description)} \]

Suppose \(s_{l,2}\) in Example 1. On the one hand the description missing in NetworkConnection to be used by the input parameter SlowNetworkConnection is referred by the Extra Description \(H\) of the Concept Abduction Problem...
\( \langle L, \text{NetworkConnection, SlowNetworkConnection, T} \rangle \)
i.e., \( \forall \text{netSpeed.Adsl1M} \). On the other hand the Common Description is defined by \( \text{lcs(NetworkConnection, NetworkConnection)} \) i.e., \( \text{NetworkConnection} \).

2.3 Modelling Web Service Composition

The process model of web service composition and its semantic links is specified by a statechart [11]. Each state refers to a service being activated, whereas its transitions coincide with semantic links. In addition some basic composition constructs such as sequence, conditional branching (i.e., OR-Branching), structured loops, concurrent threads (i.e., AND-Branching), and inter-thread synchronization can be found. To simplify the presentation, we initially assume that all considered statecharts are acyclic and consists of only sequences, OR-Branching and AND-Branching.

Example 3. (Process Model of a Composition)

Suppose a composition (Figure 3) extending Example 1 with six more services \( s_{1,3 \leq i \leq 8} \), and additional semantic links \( sl_{i,j} \). Its process model consists of sequences, OR-Branching and AND-Branching.

![Figure 3. A (Concrete) Composition.](image)

The Example 3 illustrates a composition wherein tasks \( T_i \) and abstract semantic link \( sl_{i,j} \) have been respectively concretized by one of their \( n \) candidate services (e.g., \( s_1 \) and \( n^2 \) candidate links (e.g., \( sl_{i,j} \)). Indeed some services with common functionality, preconditions and effects although different input, output parameters and quality (e.g., QoS) can be selected to perform a target task \( T_i \) and obtaining a concrete composition. Such a selection will have a direct impact on their semantic links.

In the following we address the issue of selecting and composing a large and changing collection of services. The choice of services will be done at composition time, based on both quality of services and of their semantic links.

3 Quality Model

Firstly we present a quality criterion to value semantic links. Then we proposed how it can be extended to also cover non-functional QoS, thus allowing us to estimate both quality levels of any compositions.

3.1 Quality of Semantic Link

We consider two generic quality criteria for semantic links \( sl_{i,j} \) defined by \( \langle s_i, \text{Sim}_T(Out_{s_i}, In_{s_j}), s_j \rangle \):

- **Common Description rate** \( q_{cd} \in [0, 1] \) is defined by

\[
q_{cd}(sl_{i,j}) = \frac{|\text{lcs}(Out_{s_i}, In_{s_j})|}{|H| + |\text{lcs}(Out_{s_i}, In_{s_j})|}
\]

wherein the Extra Description \( H \) is a solution of the Concept Abduction problem \( \langle L, Out_{s_i}, In_{s_j}, T \rangle \) (Definition 1): the difference between service parameters \( Out_{s_i} \) and \( In_{s_j} \). In other words \( q_{cd} \) estimates the rate of descriptions which is well specified for ensuring a correct data flow between \( s_i \) and \( s_j \).

- **Matching Quality** \( q_{m} \) of a link \( sl_{i,j} \) is a value in \([0, 1]\) defined by \( \text{Sim}_T(Out_{s_i}, In_{s_j}) \) i.e., either \( 1 \) (Exact), \( \frac{3}{3} \) (PlugIn), \( \frac{1}{2} \) (Subsume) and \( \frac{1}{3} \) (Intersection). Contrary to \( q_{cd} \), \( q_{m} \) does not estimate similarity between the parameters of semantic links but gives an overview (discretized values) of their semantic relationships.

In case we consider \( Out_{s_i} \cap In_{s_j} \) to be not satisfiable, it is straightforward to extend and adapt our quality model by i) computing contraction [9] between \( Out_{s_i} \) and \( In_{s_j} \), and ii) valuing the Disjoint matching type. Thus, the two quality criteria can be updated in consequence.

Given the above quality criteria, the quality vector of a semantic link \( sl_{i,j} \) is defined as follows:

\[
q(sl_{i,j}) = (q_{cd}(sl_{i,j}), q_{m}(sl_{i,j}))
\]

Example 4. (Quality of Semantic Links)

Let \( s_3 \) be another candidate service for \( T_2 \) (Figure 3) with \( \text{NetworkConnection as input} \). The link \( sl_{1,2} \) between \( s_1 \) and \( s_3 \) is better than \( sl_{1,2} \) since \( q(sl_{1,2}) > q(sl_{1,2}) \).

In case \( s_i, s_j \) are related by more than one link, the value of each criterion is retrieved by computing their average.

3.2 Quality of Semantic Link and QoS

Here we extend the latter quality model by also exploiting the non functional properties of services (also known as QoS attributes [20]) involved in each semantic link. We simplify the presentation by considering only:

- **Execution Price** \( q_{pr}(s_i) \in \mathbb{R}^+ \) of service \( s_i \) i.e., the fee requested by the service provider for invoking it.

- **Response Time** \( q_{rt}(s_i) \in \mathbb{R}^+ \) of service \( s_i \) i.e., the expected delay between the request and result moments.

\( |\cdot| \) refers to the size of ALCN concept descriptions ([15] p.17) i.e., \( |\cdot|, |\cdot|, |\cdot| \) are 1; \( |C \cap D| = |C| + |D| \); \( |\forall r.C| \) is 1 + |C|; \( |\geq nr| \) and \( |\leq nr| \) are \( 2 + |\log(n + 1)| \) (binary encoding of n). E.g., \( |\text{Adsl1M}| = 1 + 2 + \log(2) \) i.e., 4 in Figure 2.
The latter values of execution price and response time are given by service providers or third parties. A quality vector of a service \( s_i \) is then defined as follows:

\[
q(s_i) = (q_{pr}(s_i), q_{t}(s_i))
\]

Thus a QoS-extended quality vector of a semantic link \( sl_{i,j} \):

\[
\hat{q}(sl_{i,j}) = (q(s_i), q(sl_{i,j}), q(s_j))
\]

Given an abstract link between tasks \( T_i, T_j \), one may select the link with the best matching quality, common description rate, the cheapest and fastest services, or maybe a compromise between the four within (5). Moreover the selection could be influenced by predefining some constraints e.g., a service response time lower than a given value.

**Example 5. (QoS-Extended Quality of Semantic Link)**

Suppose \( T_2 \) and its two candidate services \( s_2, s_2' \) wherein \( q(s_2') < q(s_2) \). According to Example 4, \( s_2' \) should be preferred regarding the quality of its semantic link with \( s_1 \), whereas \( s_2 \) should be preferred regarding its QoS. So what about the best candidate for \( sl_{1,2} \) regarding both: \( \hat{q} \)?

3.3 Quality of Composition

Here we describe how to compute the quality of any concrete composition (Table 1), given the quality of its underlying services and semantic links (here \( s \) and \( sl \) stand respectively for \( s_i \) and \( sl_{i,j} \)). The approach for computing semantic quality of such a composition \( c \) is adapted from the application-driven heuristics of [17], while the computation of its non functional QoS follows similar logic to [7].

- **Common Description rate** \( Q_{cd} \) of both a sequential and AND-Branching composition \( c \) is defined as the average of its semantic links’ common description rate \( q_{cd}(sl) \). The common description rate of an OR-Branching composition is a sum of \( q_{cd}(sl) \) weighted by \( p_{al} \) i.e., the probability that semantic link \( sl \) be chosen at run time. Such probabilities are initialized by the composition designer, and then eventually updated considering the information obtained by monitoring the workflow executions.

- **Execution Price** \( Q_{pr} \) of a sequential and AND-Branching composition \( c \) is a sum of every service’s execution price \( q_{pr}(s) \). The execution price of an OR-Branching composition \( c \) is defined in the same way as \( Q_{cd}(c) \), by changing \( q_{cd}(sl) \) by \( q_{pr}(s) \).

- Details for computing **Matching Quality** \( Q_m(c) \) and **Response Time** \( Q_t(c) \) can be found in Table 1.

Using the above aggregation rules, the quality vector of any concrete composition can be defined by (6). Contrary to QoS criteria \( Q_{i \in \{t, pr\}} \), the higher \( Q_{i \in \{cd, m\}}(c) \) the higher its \( i^{th} \) quality for semantic criterion.

\[
Q(c) = (Q_{cd}(c), Q_m(c), Q_t(c), Q_{pr}(c))
\]

Definitions (3), (4), (5) as well as (6) can be extended by considering further quality of semantic links (e.g., their robustness [16]) and services (e.g., reliability [20]).

4 A Scalable Approach for Quality Driven Semantic Web Service Composition

Since scalability is one of our main concerns we suggest to compute a single concrete composition (by assigning a services to each task; Figure 3) among a set of potential solutions rather than computing the optimal composition (impractical in large domains). The selection process of such a concrete composition will be driven by constraints on i) the quality of component services and ii) the quality of their semantic links. To this end the quality model (5) is used to model local constraints on both semantic links and services whereas (6) is considered to model global constraints.

**Example 6. (Compositions and Constraints)**

Given a composition of tasks (e.g., in Figure 3) achieving a specific goal, the end-user is requested to provide some constraints on the composition she expects. For example, the end-user may have a limited budget and thus the execution price \( Q_{pr} \) is constrained, or he/she cannot accept a matching quality \( Q_m \) below a given limit. We can also imagine local constraints on specific tasks and semantic links.

Towards this issue we formalize the problem as a **Constraint Satisfaction Problem** (CSP) and apply a stochastic search method to compute a solution that meets constraints. In this Section compositions refer to their concrete form.

4.1 CSP Formalization

CSP [27] is a key formalism for many combinatorial problems such as ours. The success of this paradigm is due to its simplicity, its natural fit to a number of several real-world applications and especially the efficiency of existing underlying solvers. In addition such a formalism allows a generic representation of any Web service composition problem with constraints. Hence, we formalize Web service composition as a CSP (Definition 2) with local and global constraints.

**Definition 2. (Composition-Driven CSP)**

A Composition-Driven CSP is defined as a triple \((T, D, C)\):

- \(T\) is the set of tasks (variables) \(\{T_1, T_2, ..., T_n\}\) defined in the composition;
**Quality Criterion**

\[ \sum_{Q} \]

**Semantic**

\[ \Pi \]

**Non Functional**

\[ \sum_{Q} \]

\[ \sum_{Q} \]

Constraints can be applied on execution price (e.g., less than a constraint can be defined by \( q_v \) should be higher than a given value \( v \) is required to be higher than a given value.

**4.2 A Stochastic Search Method**

The composition-driven CSP is solved by adapting a stochastic search method [27]. Such a method sacrifices completeness (i.e., all solutions) for speed by computing “a single” solution. More precisely we adapt the Hill Climbing algorithm [24, 23], which is the most appropriate for a large number of services in domains \( D_i \leq i \leq n \). The Hill Climbing algorithm requires two functions:

- an evaluation function \( f \) which maps every composition \( c \) in the search space to a value according to (9).
- \( f \) assigns a better value to compositions \( c \) with high semantic quality attributes and low QoS attributes.

\[
f(c) = \frac{\omega_{sl} \tilde{Q}_{sl}(c) + \omega_{m} \tilde{Q}_{m}(c)}{\omega_{pr} \tilde{Q}_{pr}(c) + \omega_{l} \tilde{Q}_{l}(c)}
\]

(9)

where \( \tilde{Q}_{l} \in \{ pr, t, cd, m \} \) refer to \( Q_{l} \) normalized in the interval \([0, 1] \). \( \omega \in [0, 1] \) is the weight assigned to the \( t \)th quality criterion and \( \sum_{l \in \{ pr, t, cd, m \}} \omega = 1 \). In this way preferences on quality of the desired compositions can be done by simply adjusting \( \omega \) e.g., the Common Description rate could be weighted higher.

- an adjacency function which maps every composition to some other compositions. Here two compositions are considered to be adjacent to each other if they differ in exactly one assignment \( (s, T) \) between them.

From this formalization the Hill Climbing algorithm starts with a random composition (let’s say \( c_{final} \)) in the search space. All the compositions which are adjacent to \( c_{final} \) are evaluated using the evaluation function. In case some of its adjacent compositions have higher values than \( c_{final} \), then one is picked non-deterministically to become the new final composition. The algorithm iterates until all the constraints are satisfied by \( c_{final} \), even if the algorithm can climb to a higher composition (in the sense of \( f \)). A solution to the composition-driven CSP is the first set of assignments which does not violate constraints.

In case no solution exists, users may revise and relax constraints. Instead, fuzzy logic could be used to address the imprecision in specifying quality constraints, estimating quality values and expressing composition quality.

All the CSP based search methods have worst case complexity which is exponential to the number of variables. However the stochastic search method such as the Hill Climbing algorithm scales better (in general cases) since it focuses on the first composition that met constraints.
5 Experimental Results

To test the applicability of the approach to problems of realistic complexity, we analyze its performance by:

i) observing the evolution of constraints satisfaction over the iterations of the stochastic search method (here Hill Climbing algorithm),

ii) observing the evolution of the composition quality (9) over the iterations of the stochastic search method,

iii) decoupling the stochastic search method and the (off-line) DL reasoning process required in our approach,

iv) comparing our approach with state-of-the-art approaches such as Genetic Algorithm (GA) [6] and Integer Programming (IP) [30] based approaches.

For the experiments (Sections 5.1, 5.2 and 5.3) we have generated the descriptions of 300 tasks, and 350 alternative semantic service descriptions for each task (3502 candidate semantic links between 2 tasks). We use an actual industrial (telecoms) ontology described in ALCN with 1880 concepts and 560 properties, and estimate values for price and response time. Common description rate and matching quality of semantic links are computed according to a DL reasoning process.

We conducted experiments on Intel(R) Core(TM)2 CPU, 2.4GHz with 2GB RAM. Our approach is implemented in Java, extending an open source library. Standard DL reasoning inference such as subsumption are achieved by means of a DL reasoner Fact++ [13] whereas Abduction (required for computing difference between output and input parameters of services) is performed by MAMAS-tng.

5.1 Evolution of Constraints Satisfaction

Figure 4 reports the evolution of satisfaction constraints over the iterations (also known as the number of nodes expanded in the search tree) of the stochastic search method, by varying the number of tasks i.e., 100, 200, 300.

This figure illustrates different levels of convergence to a composition that meets the constraints by iteratively maximizing the common description and matching quality while minimizing price and response time (i.e., maximizing f).

Table 2 presents the computation costs and the number of iterations required to obtain a composition that met initial constraints. According to this and Figure 4 the more tasks, the more time consuming to converge to a solution.

5.2 Evolution of Composition Quality

Figure 5 reports the evolution of the composition quality (9) (i.e., f with equal weights assigned to the different quality criteria) over the iterations of the stochastic search method, by varying the number of tasks i.e., 100, 200, 300.

According to Figure 5 and Table 3, retrieving an (local) optimal composition is very time consuming, and then cannot be considered in large scale domains. However computing compositions that simply satisfied initial constraints is more scalable. Note that compositions with an average quality of 53% met the initial constraints in our experiments.
5.3 Search Process and DL Reasoning

Contrary to QoS given (in general) by providers, the quality of semantic links are estimated according to DL reasoning (i.e., Subsumption for \( q_{cd} \); Abduction and \( lcs \) for \( q_{cd} \)). Since our approach is depending of the latter reasoning and the stochastic search method, we suggest to decouple and detail their computation costs in Figure 6.

![Figure 6. DL Reasoning and Search Method.](image)

DL reasoning is the most time consuming process in large-scale problem of quality-driven semantic web service composition (i.e., number of tasks and candidate services greater than 100 and 350). This is caused by i) the large number of potential semantic links between tasks and ii) the critical complexity of \( q_{cd} \) computation through DL Abduction (even in \( \mathcal{ALN} \) DL).

5.4 Comparison With Other Approaches

We compare our approach with alternatives based on IP [30] and GA [6] by upgrading their quality criteria with quality of semantic links. To this end we focus on the computation (or convergence) time of these approaches to satisfy a given set of quality constraints of a composition. Here 90\% of constraints should be satisfied.

In more details, the evaluation function (9) requires to be maximized with the IP and GA based approaches. (9) is restricted to quality criteria \( \hat{Q}_{cd}, \hat{Q}_{pr} \) in order to satisfy the linearity constraint attached to the IP approach.

The IP-based optimization problem is solved by running CPLEX, a state of the art IP solver based on the branch and cut technique \(^8\) [28] whereas the GA approach is implemented in Java, extending a GPL library\(^9\). As a case study we considered a composition of 300 tasks where the number of candidate services varies from 1 to 500.

Even if our stochastic search method does not compute optimal composition, the results of this experimentation (Figure 7) confirm its adoption for large domains such as the web since it outperforms both GA and IP for a large number of candidate services per task (from 280).

Such results of IP and GA based approaches are, in parts, explained by the exponential search required to compute the optimal composition.

6 Related Work

Despite considerable work in area of service composition, no effort has specifically addressed selection of service composition constrained by both QoS and semantic similarities in a context of large domains. Indeed main approaches focus on either QoS [6, 30] such as price, reliability and execution time (hence independence between web services), or functional criteria such as semantic links [17]. In contrast, we present an innovative model that addresses both types of quality criteria for selecting web service composition in large scale domains.

Solving such a problem falls in the category of search problems, which are NP-hard. To the best of our knowledge most approaches focus on optimizing the composition by mapping it to a multi-criteria optimization problem. Despite some scalability issues, the latter can be approached using IP [30, 17], GA [6], or Constraint Programming [12].

Contrary to IP [30], GAs are better at handling non-linearity of aggregation functions, and provide better scaling up to a large number of candidate services per task (actually 16 tasks and 25 candidate services per task). The selection problem can be also modelled as a knapsack problem [29], wherein [2] performed dynamic programming to solve it. Unfortunately the previous QoS-driven service composition approaches consider only links valued by Exact matching types, hence they do not take into account the semantic quality of compositions. This is addressed by [17], who introduce a general and formal model to evaluate such a quality. From this they formulate an optimization problem which is solved by adapting the IP-based approach of [30].

All quality criteria are used for specifying both constraints and objective function.

Here we follow [12] and suggest the use of CSP to model quality-driven semantic web service composition. However, our approach presents a stochastic search method to compute a single composition that meets intial constraints

\(^8\) LINDO API v5.0, Lindo Systems Inc. http://www.lindo.com/

\(^9\) http://jgap.sourceforge.net/
rather than computing the optimal composition. Moreover we also extend their model by i) using semantic links to consider data flow in composition, ii) considering not only QoS but also semantic quality (and contraints) of composition.

7 Conclusion

We studied quality-driven semantic web service composition. Our approach has been directed to meet the main challenges facing this problem i.e., how to effectively compute compositions of QoS-driven web services by considering their semantic links in large domains such as the Web. First of all we have presented an innovative and extensible model to evaluate quality of i) web services (QoS), ii) their semantic links, and iii) their compositions. In regards to the latter criteria, the problem is formalized as a CSP problem with multiple constraints. Since one of our main concerns is about selection of large-scale web service compositions (i.e., many services can achieve a same functionality), we suggested to follow a stochastic search method which is faster than optimizing them. The experimental results have confirmed the latter and have shown acceptable computations cost of our approach despite the time consuming process of the off-line DL reasoning.

The main direction for future work is to consider a finer abduction operator, which is easy-to-compute in expressive DLs. We also plan to extend the quality model by considering domain-specific/dependent QoS criteria (e.g., a temperature service could have QoS attributes such as precision or refresh frequency) for which the aggregation function has to be user-specific. We will also focus on an extension of the semantic link definition, by considering the type assigned to input and output variable at run time. Finally the dynamic distribution of the CSP on different peers needs also to be studied to improve the convergence time of our approach.

References