Optimizing QoS-Aware Semantic Web Service Composition[⋆]

Freddy Lécué

The University of Manchester
Booth Street East, Manchester, UK
firstname.lastname@manchester.ac.uk

Abstract. Ranking and optimization of web service compositions are some of the most interesting challenges at present. Since web services can be enhanced with formal semantic descriptions, forming the "semantic web services", it becomes conceivable to exploit the quality of semantic links between services (of any composition) as one of the optimization criteria. For this we propose to use the semantic similarities between output and input parameters of web services. Coupling this with other criteria such as quality of service (QoS) allow us to rank and optimize compositions achieving the same goal. Here we suggest an innovative and extensible optimization model designed to balance semantic fit (or functional quality) with non-functional QoS metrics. To allow the use of this model in the context of a large number of services as foreseen by the strategic EC-funded project SOA4All we propose and test the use of Genetic Algorithms.

Keywords: Semantic Web, Web service, Service composition, Quality of service and composition, Automated reasoning.

1 Introduction

The *Semantic Web* [1], where the semantic content of the information is tagged using machine-processable languages such as the *Web Ontology Language* (OWL) [2], 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, e.g., their functionality in terms of input and 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 such as selection, discovery and composition.

Here we focus on web service composition and more specifically on its functional level, 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 [4], defined as a semantic connection (i.e., part of data flow) between an output and an input parameter of two services. Web service compositions could thus be estimated

⋆ Foundation Project: Supported by European Commission VII Framework IP Project Soa4All.

and ranked not only along well known non functional parameters such as *Quality of Services* (QoS) [5] but also along the dimension of semantic similarity as indicator of functional fit [6]. Considering semantics on connections of services is useful in case the information required and provided by services does not match perfectly in every data flow. This is the case of semantic-based description of services. In this work we propose to unify both types of criteria in an innovative and extensible model allowing us to estimate and optimise the quality of service compositions.

Maximizing the quality of service composition using this model is essentially a multi-objective optimization problem with constraints on quality of services and semantic links, which is known to be NP-hard [7]. Most approaches in the literature addressing optimization in web service composition are based on *Integer linear Programming* (IP) e.g., [8]. However, IP approaches have been shown to have poor scalability [5] 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 a large number of semantic services will be accessible globally. This is the vision of SOA4All, a strategic EC-funded project. Rapid computation of optimal compositions is especially important for interactive systems providing service composition facilities for end users, where long delays may be unacceptable. Here we demonstrate that the optimisation problem can be automated in a more scalable manner using *Genetic Algorithms* (GAs), and propose an approach to tackle QoS-aware *semantic* web service composition.

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 QoS-aware semantic web service composition. Section 4 details the GA-based evolutionary approach, including the strategies of the crossover, mutation and fitness function. Section 5 reports and discusses results from the experimentations. Section 6 briefly comments on related work. Finally section 7 draws some conclusions and talks about possible future directions.

2 Background

In this section we describe how semantic links can be used to model web service composition. In addition we remind the definition of *Common Description* in semantic 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 $\mathcal{T}$ (e.g., Fig.2), where the OWL-S profile [9] or SA-WSDL [10] can be used to describe them (through semantic annotations). At functional level web service composition consists in retrieving some semantic links [4] noted $sl_{i,j}$ (Fig.1) i.e.,

$$sl_{i,j} \doteq \langle s_i, Sim_{\mathcal{T}}(Out\_s_i, In\_s_j), s_j \rangle \quad (1)$$

[1] Distributed ontologies are not considered here but are largely independent of the problem addressed in this work.

between output parameters $Out\_s_i \in \mathcal{T}$ of services s_i and input parameters $In\_s_j \in \mathcal{T}$ of other services s_j. Thereby s_i and s_j are partially linked according to a matching function $Sim_{\mathcal{T}}$. Given a terminology $\mathcal{T}$, [11] and [12] value the range of the latter function along five matching types: i) *Exact* i.e., $Out\_s_i \equiv In\_s_j$, ii) *PlugIn* i.e., $Out\_s_i \sqsubseteq In\_s_j$, iii) *Subsume* i.e., $In\_s_j \sqsubseteq Out\_s_i$, iv) *Intersection* i.e., $\neg(Out\_s_i \sqcap In\_s_j \sqsubseteq \perp)$ and v) *Disjoint* i.e., $Out\_s_i \sqcap In\_s_j \sqsubseteq \perp$.

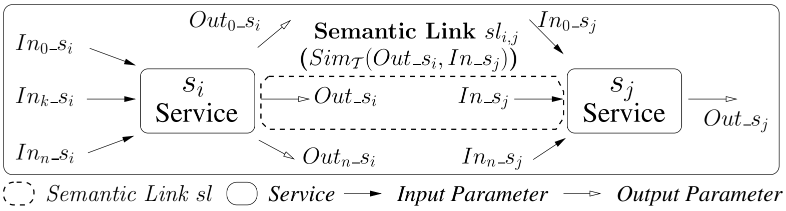

Fig. 1. A Semantic Link $sl_{i,j}$ between Services s_i and s_j

Example 1 ***(Matching Type)***
Suppose $sl_{1,2}$ (Fig.3) be a semantic link between two services s_1 and s_2 such that the output parameter `NetworkConnection` of s_1 is (semantic) linked to the input parameter `SlowNetworkConnection` of s_2. According to Fig.2 this link is valued by a Subsume matching type since $NetworkConnection \sqsupset SlowNetworkConnection$.

The matching function $Sim_{\mathcal{T}}$ enables, at design time, finding some types of semantic compatibilities (i.e., Exact, PlugIn, Subsume, Intersection) and incompatibilities (i.e., Disjoint) among independently defined service descriptions. In this direction the latter function can be used to define the quality of data flow in web service composition at semantic level.

2.2 Common Description of a Semantic Link

Besides computing the matching type of a semantic link, authors of [13] suggest computing a finer level of information i.e., the Extra and Common Descriptions between Out_s_i and In_s_j of a semantic link $sl_{i,j}$. They adapt the definition of syntactic difference [14] for comparing $\mathcal{ALE}$ DL descriptions and then obtaining a compact representation. The Extra Description $In\_s_j \backslash Out\_s_i$:

$$In\_s_j \backslash Out\_s_i \doteq \min_{\preceq_d}\{E | E \sqcap Out\_s_i \equiv In\_s_j \sqcap Out\_s_i\} \tag{2}$$

refers[2] to information required by In_s_j but not provided by Out_s_i to ensure a correct data flow between s_i and s_j. The Common Description of Out_s_i and In_s_j, defining as their Least Common Subsumer [15] lcs, refers to information required by In_s_j and provided by Out_s_i[3].

[2] With respect to the subdescription ordering $\preceq_d$.
[3] In case $Out\_s_i \sqcap \neg In\_s_j \sqsubseteq \perp$, $In\_s_j \backslash Out\_s_i$ is replaced by its more general form i.e., $In\_s_j \backslash lcs(In\_s_j, Out\_s_i)$.

$$
\begin{aligned}
&NetworkConnection \equiv \forall netPro.Provider \sqcap \forall netSpeed.Speed \\
&SlowNetworkConnection \equiv NetworkConnection \sqcap \\
&\qquad\qquad\qquad\qquad\qquad \forall netSpeed.Adsl1M \\
&Adsl1M \equiv Speed \sqcap \forall mBytes.1M
\end{aligned}
$$

Fig. 2. Sample of an $\mathcal{ALE}$ Domain Ontology $\mathcal{T}$

Example 2 ***(Extra & Common Description)***
Suppose $sl_{1,2}$ in Example 1. On the one hand the Extra Description *missing in* `Network Connection` *to be used by the input parameter* `SlowNetwork Connection` *is defined by $SlowNetworkConnection \backslash NetworkConnection$ i.e., $\forall netSpeed.Adsl1M$. On the other hand the* Common Description *is defined by $lcs(SlowNetworkConnection, NetworkConnection)$ i.e., $NetworkConnection$.*

2.3 Modelling Web Service Composition

In this work, the process model of web service composition and its semantic links is specified by a statechart [16]. Its states refer to services whereas its transitions are labelled 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 Web Service Composition)***
Suppose a composition extending Example 1 with six more services $s_{i,1 \le i \le 8}$, eight more semantic links $sl_{i,j}$. Its process model is depicted in Fig.3.

The example 3 illustrates a composition wherein tasks T_i and abstract semantic link $sl^A_{i,j}$ have been respectively concretized by one of their n candidate services (e.g., s_i) and n^2 candidate links (e.g., $sl_{i,j}$). Indeed some services with common functionality,

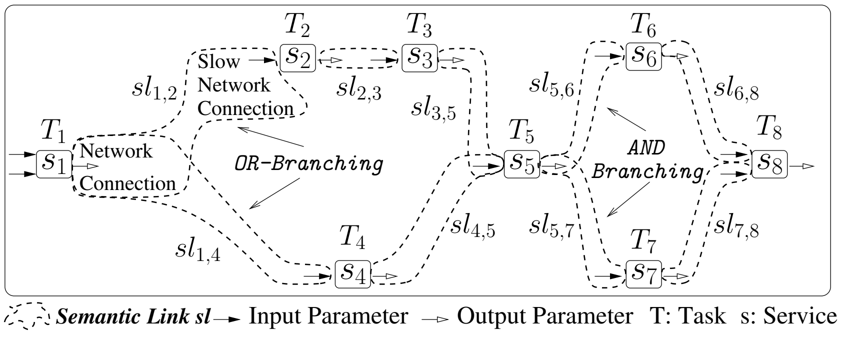

Fig. 3. A (Concrete) Web Service Composition

preconditions and effects although different input, output parameters and quality can be selected to perform a target task T_i and obtaining a concrete composition. Such a selection will have a direct impact on semantic links involved in the concrete composition.

In the following we assume that compositions of tasks (achieving a goal) have been pre-computed. This computation can be performed by *template-based* and *parametric-design-based* composition approaches [17]. So the control flow of the pre-computed compositions is fixed since it is pre-defined. The choice of the service (that fulfills a given task) will be done at composition time, based on both quality of i) services and ii) their semantic links (i.e., quality of data flow).

3 Quality Model

Here we present a quality criterion to value semantic links. Then we suggest to extend it with the non functional QoS 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 $\langle s_i, Sim_{\mathcal{T}} (Out\_s_i, In\_s_j), s_j\rangle$: its i) *Common Description* rate, and ii) *Matching Quality*.

Definition 1 ***(Common Description rate of a Semantic Link)***
Given a semantic link $sl_{i,j}$ between s_i and s_j, the Common Description rate $q_{cd} \in (0, 1]$ provides one possible measure for the degree of similarity between an output parameter of s_i and an input parameter of s_j. This rate is computed using the following expression:

$$q_{cd}(sl_{i,j}) = \frac{|lcs(Out\_s_i, In\_s_j)|}{|In\_s_j \backslash Out\_s_i| \; + \; |lcs(Out\_s_i, In\_s_j)|} \tag{3}$$

This criterion estimates the proportion of descriptions which is well specified for ensuring a correct data flow between s_i and s_j.

The expressions in between | refer to the size of $\mathcal{ALE}$ concept descriptions ([18] p.17) i.e., $|\top|$, $|\bot|$, $|A|$, $|\neg A|$ and $|\exists r|$ is 1; $|C \sqcap D| \doteq |C| + |D|$; $|\forall r.C|$ and $|\exists r.C|$ is $1 + |C|$. For instance $|Adsl1M|$ is 3 in the ontology illustrated in Fig. 2.

Definition 2 ***(Matching Quality of a Semantic Link)***
The Matching Quality q_m of a semantic link $sl_{i,j}$ is a value in $(0, 1]$ defined by $Sim_{\mathcal{T}} (Out\_s_i, In\_s_j)$ i.e., either 1 (Exact), $\frac{3}{4}$ (PlugIn), $\frac{1}{2}$ (Subsume) or $\frac{1}{4}$ (Intersection).

The discretization of the matching types follows a partial ordering [19] where the assignment of values to matching types is driven by the data integration costs. Behind each matching type, tasks of XML (Extensible Markup Language) data type integration and manipulation are required. The PlugIn matching type is more penalized than the *Exact* matching type in this model. Indeed the data integration process is lower (in term of computation costs) for the *Exact* matching type than for *PlugIn* matching type.

Contrary to q_{cd}, q_m does not estimate similarity between the parameters of semantic links but gives a general overview (discretized values) of their semantic relationships. Given these quality criteria, the quality vector of a semantic link $sl_{i,j}$ is defined by:

$$q(sl_{i,j}) \doteq \big(q_{cd}(sl_{i,j}), q_m(sl_{i,j})\big) \tag{4}$$

The quality of semantic links can be compared by analysing their q_{cd} and q_m elements. For instance $q(sl_{i,j}) > q(sl'_{i,j})$ if $q_{cd}(sl_{i,j}) > q_{cd}(sl'_{i,j})$ and $q_m(sl_{i,j}) > q_m(sl'_{i,j})$. Alternatively we can compare a weighted average of their normalised components in case the value of the first element of $sl_{i,j}$ is better than the first element of $sl'_{i,j}$ but worse for the second element [20].

Example 4 ***(Quality of Semantic Links)***
Let s'_2 be another candidate service for T_2 in Fig.3 with `NetworkConnection` *as an input. The link $sl'_{1,2}$ between s_1 and s'_2 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. This average is computing by means of the And-Branching row of Table 1, independently along each dimension of the quality model.

3.2 QoS-Extended Quality of Semantic Link

We extend the latter quality model by exploiting the non functional properties of services (also known as QoS attributes [21] - given by service providers or third parties) involved in each semantic link. We simplify the presentation by considering only:

- **Execution Price** $q_{pr}(s_i) \in \Re^+$ of service s_i i.e., the fee requested by the service provider for invoking it.
- **Response Time** $q_t(s_i) \in \Re^+$ of service s_i i.e., the expected delay between the request and result moments.

A quality vector of a service s_i is then defined as follows:

$$q(s_i) \doteq (q_{pr}(s_i), q_t(s_i)) \tag{5}$$

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

$$\overset{*}{q}\ (sl_{i,j}) \doteq (q(s_i), q(sl_{i,j}), q(s_j)) \tag{6}$$

Given an abstract link between tasks T_i, T_j, one may select the link with the best functional quality (matching quality, common description rate), and non-functional (the cheapest and fastest services) quality values, or may be a compromise (depending on the enduser preferences) between the four by coupling (4) and (6) in (6). 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 $sl^A_{1,2}$ regarding both criteria: $\overset{*}{q}$? Before addressing this question in Section 4 through equation (9)*, we first focus in quality of composition in Section 3.3.*

3.3 Quality of Composition

We present definitions for comparing and ranking different compositions along the common description rate and matching quality dimension. The rules for aggregating quality values (Table 1) for any concrete composition c are driven by them. In more details the approach for computing semantic quality of c is adapted from the application-driven heuristics of [6], while the computation of its non functional QoS is similar to [22].

Definition 3 ***(Common Description rate of a Composition)***
The Common Description rate of a composition measures the average degree of similarity between all corresponding parameters of services linked by a semantic link.

The Common Description rate Q_{cd} of both a sequential and AND-Branching composition is defined as the average of its semantic links' common description rate $q_{cd}(sl_{i,j})$. The common description rate of an OR-Branching composition is a sum of $q_{cd}(sl_{i,j})$ weighted by $p_{sl_{i,j}}$ i.e., the probability that semantic link $sl_{i,j}$ 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.

Definition 4 ***(Matching Quality of a Composition)***
The matching quality of a composition estimates the overall matching quality of its semantic links. Contrary to the common description rate, this criteron aims at easily distinguishing and identifying between very good and very bad matching quality.

The matching quality Q_m of a sequential and AND-Branching composition is defined as a product of $q_m(sl_{i,j})$. All different (non empty) matching qualities involved in such compositions require to be considered together in such a (non-linear) aggregation function to make sure that compositions that contains semantic links with low or high matching quality will be more easily identified, and then pruned for the set of potential solutions. The matching quality of an OR-Branching composition is defined as its common description rate by changing $q_{cd}(sl_{i,j})$ by $q_m(sl_{i,j})$.

Details for computing **Execution Price** Q_{pr} and **Response Time** Q_t can be found in Table 1, and further explained in [22].

Table 1. Quality Aggregation Rules for Semantic Web Service Composition

Composition Construct	Quality Criterion			
	Functional		Non Functional	
	Q_{cd}	Q_m	Q_t	Q_{pr}
Sequential/ AND- Branching	$\frac{1}{\lvert sl_{i,j}\rvert}\sum_{sl_{i,j}} q_{cd}(sl_{i,j})$	$\prod_{sl_{i,j}} q_m(sl_{i,j})$	$\sum_{s_i} q_t(s_i)$ / $\max_s q_t(s)$	$\sum_{s_i} q_{pr}(s_i)$
OR-Branching	$\sum_{sl_{i,j}} q_{cd}(sl_{i,j}).p_{sl_{i,j}}$	$\sum_{sl_{i,j}} q_m(sl_{i,j}).p_{sl_{i,j}}$	$\sum_{s_i} q_t(s_i).p_{s_i}$	$\sum_{s_i} q_{pr}(s_i).p_{s_i}$

Using Table 1, the quality vector of any concrete composition can be defined by:

$$Q(c) \doteq (Q_{cd}(c), Q_m(c), Q_t(c), Q_{pr}(c)) \tag{7}$$

Although the adopted quality model has a limited number of criteria (for the sake of illustration), (4), (5), (6) as well as (7) are extensible: new functional criteria can be added without fundamentally altering the service selection techniques built on top of the model. In this direction the binary criterion of robustness [13] in semantic links can be considered[4]. In addition, other non-functional criteria such as reputation, availability, reliability, successful execution rate, etc., can also be considered in such an extension.

4 A Genetic Algorithm Based Optimization

The optimization problem (i.e., determining the best set of services of a composition with respect to some quality constraints) which can be formalized as a *Constraints Satisfaction Optimization Problem* (T, D, C, f) where T is the set of tasks (variables) in the composition, D is the set of services' domains for T (each D_i representing a set of possible concrete services that fulfil the task T_i), C is the set of constraints and f is an evaluation function that maps every solution tuple to a numerical value, is NP-hard. In case the number of tasks and candidate services are respectively n and m, the naive approach considers an exhaustive search of the optimal composition among all the m^n concrete compositions. Since such an approach is impractical for large-scale composition, we address this issue by presenting a GA-based approach [23] which i) supports constraints on QoS and also on quality of semantic links and ii) requires the set of selected services as a solution to maximize a given objective. Here compositions refer to their concrete form.

4.1 GA Parameters for Optimizing Composition

By applying a GA-based approach the optimal solution (represented by its *genotype*) is determined by simulating the evolution of an *initial population* (through generation) until survival of best *fitted* individuals (here compositions) satisfying some *constraints*. The survivors are obtained by *crossover*, *mutation*, *selection* of compositions from previous generations. Details of GA parameterization follow:

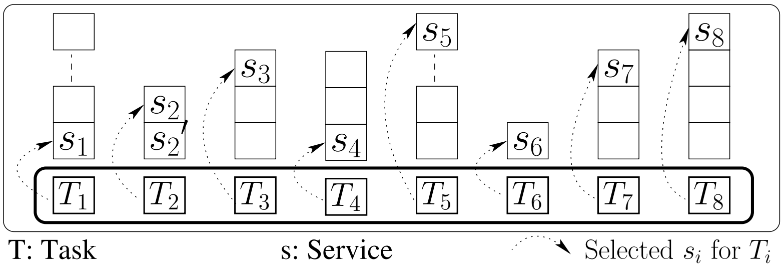

Fig. 4. Genotype Encoding for Service Composition

[4] Contrary to [6], we did not consider robustness because of its strong dependency with the matching quality criterion. Indeed they are not independent criteria since the robustness is 1 if the matching type is either Exact or PlugIn, and 0 otherwise.

• **Genotype:** It is defined by an array of integer. The number of items is equal to the number tasks involved in the composition. Each item, in turn, contains an index to an array of candidate services matching that task. Each composition, as a potential solution of the optimization problem, can be encoded using this genotype (e.g., Fig.4 is encoding the genotype of composition in Fig.3).

• **Initial Population:** It consists of an initial set of compositions (characterized by their genotypes) wherein services are randomly selected.

• **Global, Local Constraints** have to be met by compositions c e.g., $Q_{cd}(c) > 0.8$.

• **Fitness Function:** This function is required to quantify the "quality" of any composition c. Such a function f needs to maximize semantic quality attributes, while minimizing the QoS attributes of c:

$$f(c) = \frac{\omega_{cd}\hat{Q}_{cd}(c) + \omega_m \hat{Q}_m(c)}{\omega_{pr}\hat{Q}_{pr}(c) + \omega_t \hat{Q}_t(c)} \tag{8}$$

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

In addition f must drive the evolution towards constraint satisfaction. To this end compositions that do not meet the constraints are penalized by extending (8) wrt. (9).

$$f'(c) = f(c) - \omega_{pe} \sum_{l\in \substack{\{pr,t,\\ cd,m\}}} \left(\frac{\Delta\hat{Q}_l}{\hat{Q}_l^{\max}(c) - \hat{Q}_l^{\min}(c)}\right)^2 \tag{9}$$

where $\hat{Q}_l^{\max}$, $\hat{Q}_l^{\min}$ are respectively the maximum and minimal value of the l^{th} quality constraint, ω_{pe} weights the penalty factor and $\Delta\hat{Q}_{l\in\{pr,t,cd,m\}}$ is defined by:

$$\Delta\hat{Q}_l = \begin{cases} \hat{Q}_l - \hat{Q}_l^{\max} & \text{if } \hat{Q}_l > \hat{Q}_l^{\max} \\ 0 & \text{if } \hat{Q}_l^{\min} \le \hat{Q}_l \le \hat{Q}_l^{\max} \\ \hat{Q}_l^{\min} - \hat{Q}_l & \text{if } \hat{Q}_l < \hat{Q}_l^{\min} \end{cases} \tag{10}$$

Contrary to [5], compositions that violate constraints do not receive the same penalty. Indeed the factor ω_{pe} is further penalized in (9). This function avoids local optimal by considering also compositions that disobey constraints. Unfortunately, (9) contains a penalty for concrete compositions, which is the same at each generation. If, as usual, the weight ω_{pe} for this penalty factor is high, there is a risk that also concrete composition violating the constraints but "close" to a good solution could be discarded.

The alternative is to adopt a dynamic penalty, i.e., a penalty having a weight that increases with the number of generations. This allows, for the early generations, to also consider some individuals violating the constraints. After a number of generations, the population should be able to meet the constraints, and the evolution will try to improve only the rest of the fitness function. The dynamic fitness function (to be maximized) is:

$$f''(c, gen) = f(c) - \omega_{pe}.\frac{gen}{maxgen}.\sum_{l \in \substack{\{pr,t,\\ cd,m\}}} \left(\frac{\Delta\hat{Q}_l}{\hat{Q}_l^{\max}(c) - \hat{Q}_l^{\min}(c)}\right)^2 \tag{11}$$

gen is the current generation, while *maxgen* is the maximum number of generations.

• **Operators on Genotypes:** They define authorized alterations on genotypes not only to ensure evolution of compositions' population along generations but also to prevent convergence to local optimum. We use: i) *composition mutation* i.e., random selection of a task (i.e., a position in the genotype) in a concrete composition and replacing its service with another one among those available, ii) the standard two-points *crossover* i.e., randomly combination of two compositions and iii) *selection of compositions* which is fitness-based i.e., compositions disobeying the constraints are selected proportionally from previous generations.

• **Stopping Criterion:** It enables to stop the evolution of a population. First of all we iterate until the constraints are met (i.e., $\Delta Q_l = 0 \ \forall l \in \{pr, t, cd, m\}$) within a maximum number of generations. Once the latter constraints are satisfied we iterate until the best fitness composition remains unchanged for a given number of generations.

4.2 GA for Optimizing Composition in a Nutshell

The execution of the GA consists in i) defining the initial population (as a set of compositions), and computing the fitness function (evaluation criterion) of each composition, ii) evolving the population by applying mutation and crossover of compositions (Tasks with only one candidate service are disregarded), iii) selecting compositions, iv) evaluating compositions of the population, and v) back to step (ii) if the stopping criterion is not satisfied. Section 5 further details the parameters.

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

5 Experimental Results

We analyze the performances of our approach by i) discussing the benefits of combining QoS and functional criteria, ii) observing the evolution of the composition quality f'' in (11) (equal weights are assigned to the different quality criteria) over the GA generations by varying the number of tasks, iii) studying the behaviour of our approach regarding the optimisation of large scale compositions, iv) evaluating performance after decoupling the GA and the (on-line) DL reasoning processes, and v) comparing the convergence of our approach (11) with [5]. Before turning our attention to the latter five sets of experiments, we first draw the context of experimentation.

5.1 Context of Experimentation

Services, Semantic Links and their Qualities. Services[5] are defined by their semantic descriptions using an $\mathcal{ALE}$ ontology (formally defined by 1100 concepts and 390 properties, 1753 individuals, without data property), provided by a commercial partner. We have incorporated estimated values for Qos parameters (price and response time). Common description rate and matching quality of semantic links are computed according to an on-line DL reasoning process.

Implementation Details. The common description rate (3) is calculated by computing the *Extra Description* (2), the Least Common Subsumer [15], and the size ([18] p.17) of DL-based concepts. These DL inferences and the matching types have been achieved by a DL reasoning process i.e., an adaptation of Fact++ [24] for considering DL difference.

The aggregation rules of Table 1 are then used for computing each quality dimension of any composition. Finally the combination of QoS with semantic calculation is computed by means of (11), thus obtaining the final quality score for the composition.

Our GA is implemented in Java, extending a GPL library[6]. The optimal compositions are computed by using an elitist GA where the best 2 compositions were kept alive across generations, with a crossover probability of 0.7, a mutation probability of 0.1, a population of 200 compositions. The roulette wheel selection has been adopted as selection mechanism. We consider a simple stopping criterion i.e., up to 400 generations. We conducted experiments on Intel(R) Core(TM)2 CPU, 2.4GHz with 2GB RAM.

Compositions with up to 30 tasks and 35 candidates per task (35^2 candidate semantic links between 2 tasks) have been considered in Sections 5.2, 5.3, 5.5, 5.6, especially for obtaining convincing results towards their applicability in real (industrial) scenarios. Experiment results reported in Section 5.2 provide some benefits of combining QoS and functional criteria for the overall quality of composition, whereas those in Sections 5.3, 5.4, 5.5 and 5.6 are related to scalability.

5.2 Benefits of Combining QoS and Functional Criteria

Fig.5 reports the benefits of combining QoS and functional criteria. In more details, we studied the impact of functional quality on the costs of data integration, which enabling the end-to-end composition of services. The data integration process aligns the data flow specification of a composition by manipulating and transforming the semantic descriptions of contents of outgoing and incoming messages of annotated services.

Our approach and [5] are compared on ten compositions $c_{i,1 \leq i \leq 10}$ with $Q_{cd}(c_i) = 10^{-1} \times i$ and $Q_m(c_i) = 10^{i-10}$ as functional quality to reflect gradually better quality.

[5] The choice of proprietary services has been motivated by the poor quality of existing benchmark services in terms of number of services or expressivity (limited functional specification, no binding, restricted RDF-based description). We plan further experimentations with OWL-S TC 3.0 (`http://www.semwebcentral.org/frs/?group_id=89`) and SA-WSDL TC1 (`http://projects.semwebcentral.org/projects/sawsdl-tc/`).

[6] http://jgap.sourceforge.net/

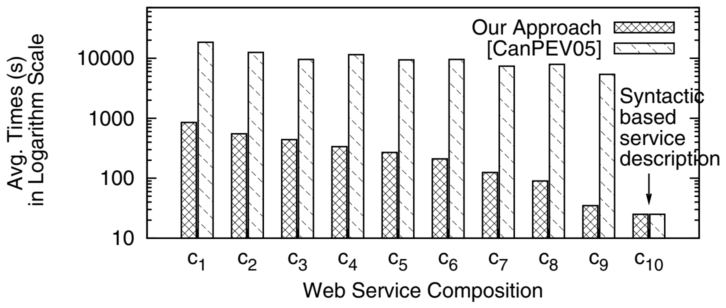

Fig. 5. Costs of Data Integration (through Data Flow Specification)

On the one hand, as expected, the costs of data integration is low (actually trivial) for both approaches, regarding the composition with the best quality (i.e., c_{10}). Indeed the parameters of services match exactly, hence no further specification is needed..

On the other hand, these costs decrease with the functional quality of compositions in our approach, whereas they are steady but very high for compositions computed by [5] (purely based on non functional quality of composition). This is due to i) the lack of specification of functional quality (hence a hard task to build semantic data flow from scratch), and ii) the manual approach used to link data in compositions.

Appropriate qualities of semantic links are very useful i) to discover data flow in composition, ii) to ease the specification (semi-automated process with Assign/Copy elements + XPath/XQuery processes a la BPEL4WS) of syntactic (and heterogeneous) data connections (for the persons who develop these mappings), so limiting the costs of data integration. Indeed the better the quality of semantic links the better the semantic mapping between the outgoing and incoming (SA-WSDL for instance) messages.

5.3 Evolution of the Composition Quality

Fig.6 reports the evolution of the composition quality over the GA generations, by varying the number of tasks. This illustrates different levels of convergence to a composition

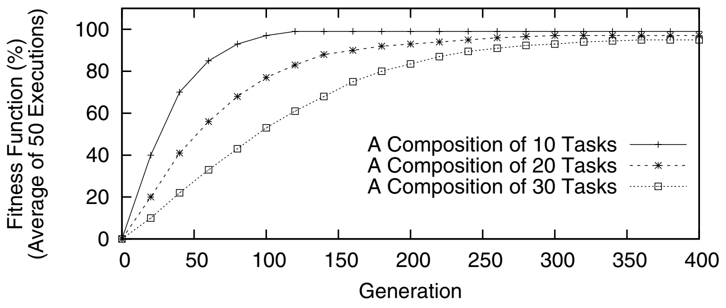

Fig. 6. Evolution of the Composition Quality

Table 2. Overview of Computation Costs

Tasks Num.	Max. Fitness (%)	Generation Num.	Time (ms)
10	99	120	1012
20	97	280	1650
30	95	360	3142

that meets some constraints and optimizes its different criteria by maximizing the common description and matching quality while minimizing price and response time.

Table 2 and Fig.6 present the computation costs and the number of generations required to obtain the maximal fitness value. The more tasks (and services) the more time consuming to converge to the optimum. Obviously, the population size and the number of generations should be extended to reach the optimum of more complex compositions.

5.4 Towards Large Scale Based Compositions

In this experiment we suggest to study the behaviour of our approach regarding the optimisation of compositions with a large number of tasks (up to 500 tasks) and candidate services (500). To this end we focus on its scalability and the impact of the number of generations as well as the population size on the GA success.

Table 3. Large Scale Compositions

Tasks Num.	Max. Fitness (%)	Generation Num./ Population Size	Time (ms)
100	85	400/200	4212
	96	700/400	9182
300	47	400/200	5520
	95	1500/500	19120
500	24	400/200	7023
	95	3000/1000	51540

As illustrated in Table 3, increasing both the number of generations and the population size does actually result in better fitness values for problems with a larger number of tasks and candidate services. For example, regarding the optimisation of a composition of 500 tasks with 500 candidate services, a number of generations of 400 and a population size of 200 do result in a low fitness value of 24% of the maximum, whereas considering a number of generations of 3000 and a population size of 1000 achieve 95% of the maximum. Note that better fitness values can be reached by further increasing the sizes of generations and populations. However doubling these sizes only improves the fitness value by 2%. This shows that each optimisation problem converges to a limit.

5.5 Decoupling GA Process and DL Reasoning

Since our approach is mainly depending on DL reasoning (i.e., Subsumption for q_m, Difference and lcs for q_{cd}) and the GA-based optimization process, we suggest to decouple and detail the computation costs of Table 2 in Fig.7.

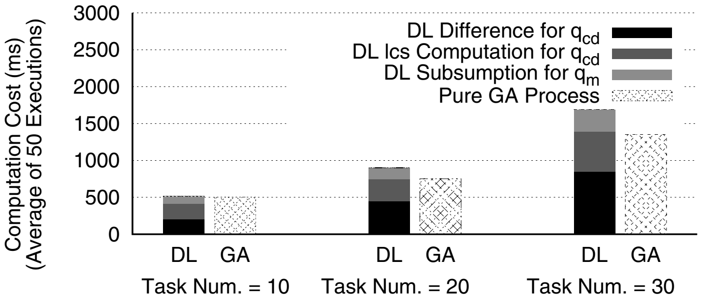

Fig. 7. DL and GA Processes in our Approach

DL reasoning is the most time consuming process in optimisation of QoS-aware semantic web service composition wherein the number of tasks and candidate services are greater than 10 and 35. This is caused by the critical complexity of q_{cd} computation through DL Difference (even in $\mathcal{ALE}$ DL).

5.6 Convergence of GA-Based Approaches

In this experiment, we compare the convergence of our approach (11) with the main alternative at present [5]. To this end the functional criteria of our approach are disregarded in order to focus only on the GA-driven aspects of the optimisation process.

According to Table 4, the advantage of our approach is twofold. Firstly we obtain better fitness values for the optimal composition than the approach of [5]. Secondly, our approach converges faster than the approach of [5]. In addition our function avoids getting trapped by local optimums by i) further penalizing compositions that disobey constraints (the factor of ω_{pe} in (9) and (11)) and ii) suggesting a dynamic penalty, i.e., a penalty having a weight that increases with the number of generations.

Table 4. Comparing GA-based Approaches (Population size of 200)

Tasks Num.	Approach	Max. Fitness (%)	Generation Num.	Time (ms)
10	Our Model (11)	99	120	1012
	[5]	97	156	1356
20	Our Model (11)	97	280	1650
	[5]	94	425	2896
30	Our Model (11)	95	360	3142
	[5]	85	596	6590

These results support the adoption of our model in the cases where a large number of tasks and services are considered.

6 Related Work

Review of existing approaches to optimising web service compositions reveals that no approach has specifically addressed optimisation of service composition using both *QoS* and *semantic similarities* dimensions in a context of *significant scale*.

Indeed main approaches focus on either QoS [5,8] or on functional criteria such as semantic similarities [6] between output and input parameters of web services for optimising web service composition. In contrast, we present an innovative model that addresses both types of quality criteria as a trade-off between data flow and non functional quality for optimizing web service composition.

Solving such a multi-criteria optimization problem can be approached using IP [8,6], GA [5], or Constraint Programming [25]. The results of [5] demonstrate that GAs are better at handling non-linearity of aggregation rules, and provide better scaling up to a large number of services per task. In addition they show that dynamic programming (such as IP-based approaches) is preferable for smaller compositions. We follow [5] and suggest the use of GAs to achieve optimization in web service composition, yet 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, iii) revisiting the fitness function to avoid local optimal solution (i.e., compositions disobeying constraints are considered).

The optimization problem can be also modelled as a knapsack problem [26], wherein [27] performed dynamic programming to solve it. Unfortunately the previous QoS-aware service composition approaches consider only links valued by Exact matching types, hence no semantic quality of compositions. Towards the latter issue [6] introduces 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 [8]. All quality criteria are used for specifying both constraints and objective function.

7 Conclusion

We studied *QoS*-aware *semantic* web service composition in a context of significant scale i.e., how to effectively compute optimal compositions of QoS-aware web services by considering their semantic links. On the one hand the benefits of a significant domain such as the Web is clear e.g., supporting a large number of services providers, considering large number of services that have same goals. On the other hand, the benefits of combining semantic links between services and QoS are as following:

> *The computation of web services composition whilst optimising both the non functional qualities and the quality of semantic fit along non-trivial data flow, where the information required and provided by services does not match perfectly in every dataflow, using semantic-based description of services.*

By addressing non trivial data flow in composition , we aimed at limiting the costs of (semantic heterogeneity) data integration between services by considering appropriate quality of semantic links. To this end 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 an optimization problem with multiple constraints. Since one of our main concerns is about optimization of large-scale web service compositions (i.e., many services can achieve a same functionality), we suggested to follow a GA-based approach, faster than applying IP. The experimental results have shown an acceptable computation costs of our GA-based approach despite the time consuming process of the on-line DL reasoning. In case of "semantic link"-less models, the benefits are mainly based on penalizing constraint violation (of the fitness function) which makes the approach faster than [5].

In future work we will consider a finer difference operator, which is also easy-to-compute in expressive DLs. Determining the most appropriate parameters for the GA phase requires further experimentations.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American 284(5), 34–43 (2001)
2. Smith, M.K., Welty, C., McGuinness, D.L.: Owl web ontology language guide. W3c recommendation, W3C (2004)
3. Baader, F., Nutt, W.: The Description Logic Handbook: Theory, Implementation, and Applications (2003)
4. Lécué, F., Léger, A.: A formal model for semantic web service composition. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 385–398. Springer, Heidelberg (2006)
5. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: An approach for qos-aware service composition based on genetic algorithms. In: GECCO, pp. 1069–1075 (2005)
6. Lécué, F., Delteil, A., Léger, A.: Optimizing causal link based web service composition. In: ECAI, pp. 45–49 (2008)
7. Papadimtriou, C.H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall, Englewood Cliffs (1982)
8. Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q.Z.: Quality driven web services composition. In: WWW, pp. 411–421 (2003)
9. Ankolenkar, A., Paolucci, M., Srinivasan, N., Sycara, K.: The owl-s coalition, owl-s 1.1. Technical report (2004)
10. Kopecký, J., Vitvar, T., Bournez, C., Farrell, J.: Sawsdl: Semantic annotations for wsdl and xml schema. IEEE Internet Computing 11(6), 60–67 (2007)
11. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.: Semantic matching of web services capabilities. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 333–347. Springer, Heidelberg (2002)
12. Li, L., Horrocks, I.: A software framework for matchmaking based on semantic web technology. In: WWW, pp. 331–339 (2003)
13. Lécué, F., Delteil, A.: Making the difference in semantic web service composition. In: AAAI, pp. 1383–1388 (2007)
14. Brandt, S., Kusters, R., Turhan, A.: Approximation and difference in description logics. In: KR, pp. 203–214 (2002)
15. Baader, F., Sertkaya, B., Turhan, A.Y.: Computing the least common subsumer w.r.t. a background terminology. In: DL (2004)

16. Harel, D., Naamad, A.: The statemate semantics of statecharts. ACM Trans. Softw. Eng. Methodol. 5(4), 293–333 (1996)
17. Motta, E.: Reusable Components For Knowledge Modelling Case Studies. In: Parametric Design Problem Solving. IOS Press, Netherlands (1999)
18. Küsters, R.: Non-Standard Inferences in Description Logics. LNCS (LNAI), vol. 2100. Springer, Heidelberg (2001)
19. Lécué, F., Boissier, O., Delteil, A., Léger, A.: Web service composition as a composition of valid and robust semantic links. IJCIS 18(1) (March 2009)
20. Hwang, C.-L., Yoon., K.: Multiple criteria decision making. Lecture Notes in Economics and Mathematical Systems (1981)
21. O'Sullivan, J., Edmond, D., ter Hofstede, A.H.M.: What's in a service? Distributed and Parallel Databases 12(2/3), 117–133 (2002)
22. Cardoso, J., Sheth, A.P., Miller, J.A., Arnold, J., Kochut, K.: Quality of service for workflows and web service processes. J. Web Sem. 1(3), 281–308 (2004)
23. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing Company, Inc., Reading (1989)
24. Horrocks, I.: Using an expressive description logic: FaCT or fiction? In: KR, pp. 636–649 (1998)
25. Ben Hassine, A., Matsubara, S., Ishida, T.: A constraint-based approach to web service composition. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 130–143. Springer, Heidelberg (2006)
26. Yu, T., Lin, K.-J.: Service selection algorithms for composing complex services with multiple qos constraints. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 130–143. Springer, Heidelberg (2005)
27. Arpinar, I.B., Zhang, R., Aleman-Meza, B., Maduko, A.: Ontology-driven web services composition platform. Inf. Syst. E-Business Management 3(2), 175–199 (2005)