Seeking Quality of Web Service Composition in a Semantic Dimension

Article *in* IEEE Transactions on Knowledge and Data Engineering · July 2011

DOI: 10.1109/TKDE.2010.237 · Source: IEEE Xplore

2 authors:

Freddy Lecue
The University of Manchester
76 PUBLICATIONS **810** CITATIONS
SEE PROFILE

Nikolay Mehandjiev
The University of Manchester
152 PUBLICATIONS **1,356** CITATIONS
SEE PROFILE

Seeking Quality of Web Service Composition in a Semantic Dimension

Freddy Lécué and Nikolay Mehandjiev

Abstract—Ranking and optimization of web service compositions represent challenging areas of research with significant implications for the realization of the "Web of Services" vision. "Semantic web services" use formal semantic descriptions of web service functionality and interface to enable automated reasoning over web service compositions. To judge the quality of the overall composition, for example, we can start by calculating the semantic similarities between outputs and inputs of connected constituent services, and aggregate these values into a measure of *semantic quality* for the composition. This paper takes a specific interest in combining semantic and nonfunctional criteria such as *quality of service (QoS)* to evaluate quality in web services composition. It proposes a novel and extensible model balancing the new dimension of semantic quality (as a functional quality metric) with a QoS metric, and using them together as ranking and optimization criteria. It also demonstrates the utility of Genetic Algorithms to allow optimization within the context of a large number of services foreseen by the "Web of Services" vision. We test the performance of the overall approach using a set of simulation experiments, and discuss its advantages and weaknesses.

Index Terms—Web service, semantic web, ontology, description logics, service composition, quality of service/composition.

1 Introduction

The *Semantic Web* [7], where the semantics of information is indicated using machine-processable languages such as the *Web Ontology Language* (OWL) [47], is considered to provide many advantages over the current version of the World Wide Web, which focuses on how information is represented. OWL, for example, is underpinned by Description Logics (DL) [6] and ontologies [23] (i.e., a formal conceptualization of a particular domain). This allows automatic processing of information assets tagged with OWL, focusing on their semantics rather than on the way they are shown on the web. Information about web services can also be semantically tagged to describe their functionalities in terms of input, output parameters, preconditions, effects, and invariants [48]. These *semantic web services* can then be automatically discovered, composed into more complex services, and executed.

Automating web service composition through the use of semantic technologies is currently a focus of a large number of research projects in the area of Service-Oriented Computing, yet work on optimizing such compositions is comparatively rare. Contrary to [2], [4], and [10] which address runtime web service selection to achieve composition optimization, this work focuses on a design-time perspective, aiming at preparing optimal compositions ready to be executed. Our approach of optimization uses a combination of functional and nonfunctional considerations so that we can cover both perspectives. The functional perspective comprises a set of metrics related to how well the functionalities of the constituent services fit together. *Semantic quality* is such a core metric, measuring the degree of semantic similarity between the outputs produced by constituent services and the inputs required by their peers. Such a quality is one of the measures of the overall *functional quality* for the composition, indicating the "goodness of fit" between the functionalities of the constituent services. Other metrics include the degree to which the composition satisfies the overall goal to be achieved, the degree to which pre- and postconditions are satisfied, etc. In this paper, we focus on *semantic quality* as the main indicator of *functional quality*.

To measure the degree of semantic similarity, we use the concept of *semantic link* [32], which is defined as the semantic connection between the corresponding pairs of web service parameters, analyzed using DL-based matchmaking. This concept is used to evaluate, compare, and classify the quality of connections and their compositions. This is important to ensure semantic cohesion of data exchanged (or shared) between services closest and avoiding services "misfiring" and ignoring incompatible data. Indeed some services can only interpret some semantic descriptions, rejecting any others. By selecting services for a composition which are better aligned and hence have better semantic quality of their connections, we reduce the time-consuming task of manual integration using mediators to align exchanged data, aiming at automating composition. Web service compositions could thus be optimized and ranked using not only nonfunctional parameters such as the well-known *Quality of Service* (QoS) [11], [21], [35], [54], [56] but also semantic quality as a core indicator of functional quality [33]. In this work, we propose to unify both types of criteria in an innovative and extensible model, allowing us to estimate and optimize the quality of service compositions. We demonstrate the utility of this model and its advantage over single-perspective optimization models using a number of simulation experiments.

• *The authors are with the Centre for Service Research, The University of Manchester, Booth Street West, Manchester M15 6PB, United Kingdom. E-mail: {freddy.lecue, nikolay.mehandjiev}@manchester.ac.uk.*

Manuscript received 3 Apr. 2009; revised 23 Nov. 2009; accepted 9 Aug. 2010; published online 18 Nov. 2010.
Recommended for acceptance by Y. Chen.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2009-04-0323.
Digital Object Identifier no. 10.1109/TKDE.2010.237.

Optimizing the quality of service composition using this model is essentially a multiobjective optimization problem [4] with constraints and preferences on the quality of services and their semantic links. Such a problem is known to be NP-hard [43], thus putting into question the practical applicability of such optimization for compositions of realistic scale as envisioned in the "Web of Services." Proposals to address this issue center on stochastic approaches [12], Constraint Programming [25], heuristics-based approaches [35], and Integer linear Programming (IP) [2], [4], [33], [56], with the latter considered to show most promise.

To speed up the application of the proposed optimization model in a context of realistic scale, we propose an approach using *Genetic Algorithms* (GAs) which adapts the methods of [10], [11], and [38]. Using a number of simulation experiments, we validate the approach and compare its effectiveness against an approach based on Linear Integer Programming.

The remainder of this paper is organized as follows: the next section comments on related work and aligns our contributions with key results in the field. Section 3 briefly reviews semantic web service composition by focusing on 1) semantic web service, 2) their semantic links, and 3) a way to model composition. Section 4 introduces the quality criteria for QoS-aware semantic web service composition. Section 5 details the GA-based evolutionary approach, including the strategies of the crossover, mutation, and fitness function. Section 6 reports and discusses the results from our experiments. Finally, Section 7 draws some conclusions and outlines directions for further work.

2 Related Work

Starting from an initial set of available services, we can define web service composition as follows:

Definition 1 (Web Service Composition). *Web service composition aims at selecting and interconnecting web services provided by different partners in order to achieve a particular goal.*

Automating web service composition aims to overcome the problem where no single service can satisfy the goal specified by the service consumer. A number of different approaches have been proposed, including *Logic-based* [45], *Matchmaking-based* [30], *Graph-Theory-based* [55], *Golog-based* [39], and *AI-Planning-based* [53]. However, very few approaches have addressed the *optimization* (Definition 2) of the resulting compositions.

Definition 2 (Web Service Composition Optimization). *Optimization of web service composition aims at selecting appropriate service components to optimize the overall quality of the composition according to a set of predefined metrics.*

In this section, we appraise the existing approaches to optimizing service composition and classify them according to the following three dimensions:

- the extent to which an approach considers the *nonfunctional quality* of compositions;
- the extent to which an approach considers the *semantic quality* of compositions; and
- the *scalability* of the approach.

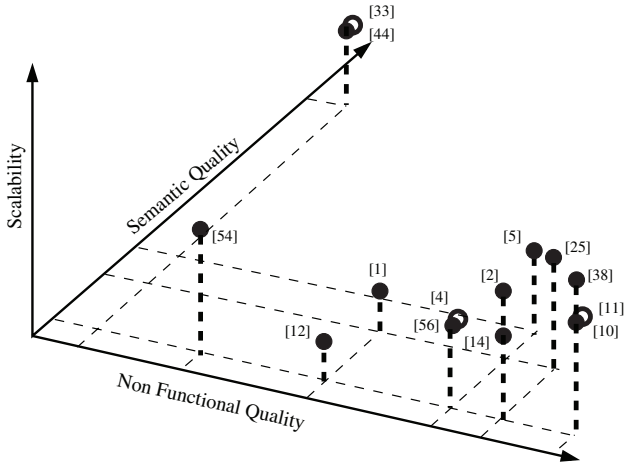

Fig. 1. Classification of approaches to optimization.

The chart in Fig. 1 positions the reviewed approaches in relation to these three dimensions. These dimensions will be used to structure the remainder of this section.

2.1 Nonfunctional Quality Dimension

The *nonfunctional quality* dimension classifies approaches based on their ability to consider nonfunctional (QoS) properties of compositions. In this dimension, the work of Lécué et al. [33] is classified with a low value since they fail to address nonfunctional quality in their composition evaluation model. Using semantic descriptions of services, they only consider optimization in terms of their compatibilities in a composition, no matter the quality the services expose. Therefore, such an approach is not able to rank compositions according to some business requirements. In contrast, the work of Canfora et al. [11] is high since it allows quality of compositions to be evaluated using several nonfunctional criteria such as the *response time*, *reliability*, *availability*, *execution price* as well as *domain-dependent attributes*. In [11], runtime-based optimization is considered in the same direction as [12], but only suboptimal solutions are identified, and the global constraints are satisfied only statistically. Others [1], [2], [4], [12], [14], [56] consider a smaller set of nonfunctional criteria. Most of these approaches achieve composition optimization through runtime selection of services. While Alrifai and Risse [2] combine global optimization with local selection of services, Ardagna and Pernici [4] extend [56] by presenting a negotiation approach to identify a feasible solution of the optimization problem and reduce the number of compositions invocation failures. Claro et al. [14] present a multiobjective evolutionary approach wherein objectives refer to different nonfunctional dimensions. They identify a set of Pareto optimal solutions for optimization but do not introduce a ranking among different quality dimensions. In their approach, the improvement of any quality dimension of identified solutions will automatically affect the quality of other dimensions.

The approaches classified with a high value in this dimension and a low value in the *Semantic Quality*

dimension will be referenced by *nonfunctional quality-based approaches* (such as the latter cited).

2.2 Semantic Quality Dimension

The second dimension ranks approaches according to their ability to optimize compositions using *semantic quality*. By maximizing this quality of compositions and their connections, they aim at reducing the number of ontology-based mediators (manually generated) which are required in case of semantic heterogeneity between data exchanged/shared in a composition.

Most of nonfunctional quality-based approaches are ranked very low in this dimension since they only take into account precise semantic matches along output-input connections (semantic links) of web services. Others [2], [11], [14], [54] do not address semantic evaluation of compositions.

At the same time, approaches with low rank on the nonfunctional quality dimension score very well here. For example, Lécué et al. [33] introduce a formal and extensible model to evaluate the semantic quality of a composition. From this, they formulate an optimization problem which is solved by adapting an IP-based approach, where all quality criteria are linearized and used for specifying both constraints and an objective function. The approach of Arpinar et al. [5] partially addressed this issue by, respectively, valuing simple levels of matching between output and input parameters of service in their composition approach. Ragone et al. [44] introduced an approach that computes approximate compositions and provides an explanation of which part of the request is not covered by the composite service. To this end, they use nonstandard DL reasoning methods such as *Concept covering*. The approaches classified with a high value in this dimension and a low value in the *Nonfunctional Quality* dimension will be referenced by *Semantic quality-based approaches* (such as the latter cited in this section).

From the view of this dimension, most of nonfunctional quality-based approaches limit the scope of their solutions since they do not consider heterogeneity of service description, hence no real focus on *data integration* between services. In the case of large web-based applications, they assume a manual and pure syntactical-based approach to compute data flow (i.e., relationships between services) in composition. The costs of such a data integration could very high regarding the overall process of composition.

2.3 Scalability Dimension

The ability of an approach to support compositions with a large number of services is ranked using the *Scalability* dimension. In this dimension, the GA-based approaches such as [10], [11], and [38] rank higher than IP-based approaches [1], [4], [33], [56]. Indeed, GA-based approaches provide better scaling up to a large number of services even if suboptimal solutions are reached in some cases. The main reasons will be explained and demonstrated using simulation experiments in Section 6. The approach of Alrifai and Risse [2] obtains better results than classic IP-based methods, and even better than some classic GA-based approaches [14] by combining global optimization with local selection.

Regarding the pure GA-based methods, the approach of Ma and Zhang [38] improves on the scalability of [10] and [11] by experimenting with different GA parameters such as population diversity, evolution policy, enhancement of initial population policy, aiming at improving the algorithm's convergence rate.

The optimization problem can also be modeled as a *knapsack problem* [54], wherein Arpinar et al. [5] and Cardoso et al. [12], respectively, propose stochastic-based search and dynamic programming to solve the problem.

2.4 The Need for a Holistic Approach

Review of existing approaches to optimizing web service compositions reveals that no approach has specifically addressed optimization of service composition using both *QoS* and *semantic similarities* dimensions in a context of *significant scale*. Indeed, main approaches focus on either nonfunctional criteria such as QoS or on functional criteria such as semantic similarities between output and input parameters of web services for optimizing web service composition. This motivates our innovative model that addresses both types of quality criteria as a trade-off between semantic quality of data connections in data flow and nonfunctional quality for optimizing web service composition.

Regarding this issue, we follow [10] and [11] and suggest the use of GAs to achieve scalable optimization in web service composition, yet we also extend their model by

- using semantic links to consider data flow;
- considering not only QoS to satisfy end users constraints such as price but also semantic quality of compositions to estimate the effort required to ensure seamless compositions; and
- revisiting the fitness function in order to avoid local optimal solution (i.e., compositions disobeying constraints are considered).

3 Semantic Web Service Composition

In this section, we review semantic web service composition by focusing on its main components as follows: 1) Semantic Web Services (we will assume without loss of generality that each service refers to a single operation), 2) their Semantic Links (also known as *Causal Links* in [32]) as a formal way of representing their semantic connections, and 3) a way to model a composition through its constituent semantic links.

3.1 Semantic Web Services

The formal model required to represent semantics of web services and their functional parameters (e.g., inputs and outputs) is provided by an ontology. The particular ontology $\mathcal{T}$, which is based on the DL $\mathcal{EL}$ [6], is part from a larger pair $\langle \mathcal{T}, \mathcal{A} \rangle$. $\mathcal{T}$ and $\mathcal{A}$ refer, respectively, to a Terminological Box (or TBox, i.e., intentional knowledge) and an Assertional Box (or ABox, i.e., extensional knowledge) in DL systems. In the following, we will focus on the TBox $\mathcal{T}$ that supports inference on service parameters by means of DL reasoning. Fig. 2 shows a fragment of an example TBox $\mathcal{T}$.

In addition, the previous TBox is completed with the following structural pieces of information: $Address$, $Decoder$, $Email$, $FTLocal$, IP, Max, $NoNilSpeed$,

$$NetworkConnection \equiv \exists netPro.Provider \sqcap \exists netSpeed.Speed$$
$$SlowNetworkConnection \equiv NetworkConnection \sqcap \exists netSpeed.Adsl1M$$
$$FastNetworkConnection \equiv NetworkConnection \sqcap \exists netSpeed.AdslMax$$
$$Speed \equiv \exists mBytes.NoNilSpeed$$
$$Adsl1M \equiv Speed \sqcap \exists mBytes.1M$$
$$AdslMax \equiv Speed \sqcap \exists mBytes.Max$$
$$Max \sqsubseteq 1M \sqsubseteq NoNilSpeed$$
$$IPAddress \equiv Address \sqcap \exists protocol.IP$$
$$VoIPId \equiv Address \sqcap \exists network.FTLocal$$
$$VideoDecoder \equiv Decoder \sqcap \exists decrypt.Video$$

Fig. 2. Part of an $\mathcal{EL}$ TBox.

$PhoneNum$, $Provider$, $Video$, $ZipCode$, and $1M$. Contrary to concepts in the TBox, they are not associated with a specific description and a textual match is more efficient than a DL reasoning task to structure them. Therefore, a prefiltering criterion is adopted to match such structural information before starting DL reasoning.

According to this model, the OWL-S profile [3], WSMO capability [20], or SA-WSDL [27] (formerly WSDL-S) can be used to describe services. In other words, syntactic web services have been enhanced with semantic annotation of their functional parameters. Therefore, in a semantic context, web services require input parameters (described according to $\mathcal{T}$) to be processed and return some output parameters (also described according to $\mathcal{T}$).

Example 1 (A Semantic Web Service). Suppose a web service `AdslEligibility` , which, starting from a `Phone Number`, a `ZipCode`, and an `EMail` address, returns the `Network Connection` of the desired zone. Its input, output parameters are semantically annotated using $\mathcal{T}$.

3.2 Semantic Links

Semantic connections between the DL-annotated parameters of web services can be defined, using the semantic similarity between an output parameter Out_s_i of s_i and an input parameter In_s_j of s_j, where both are DL concept descriptions (e.g., see Fig. 2). Similarities are judged using a matchmaking function between two knowledge representations encoded using the same TBox $\mathcal{T}$.

Semantic connections are modeled and valued using semantic links [34], defined as follows:

Definition 3 (Semantic Link). *A semantic link noted $sl_{i,j}$ (Fig. 3), i.e.,*

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

is defined as a relation between an output parameter Out_s_i of a service s_i and an input parameter In_s_j of another service s_j, using a matchmaking function $Sim_{\mathcal{T}}$. Both Out_s_i and In_s_j are described according to the definition in $\mathcal{T}$.

The existence of $sl_{i,j}$ implies that 1) s_i precedes s_j since an output of s_i is exploited by an input of s_j; and 2) no web service is interleaved between In_s_j and Out_s_i.

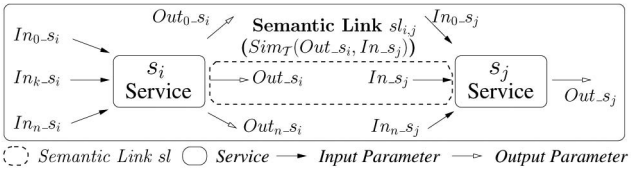

Fig. 3. A semantic link $sl_{i,j}$ between services s_i and s_j.

Example 2 (A Semantic Link). Suppose `AdslEligibility` s_1 and `VoiceOverIP` s_2 are two web services such that s_1 precedes s_2. A semantic connection between an output parameter of s_1 and an input parameter of s_2 is possible since the intersection of these parameters is satisfiable. The satisfiability is estimated by $Sim_{\mathcal{T}}$ according to Definition 3 applied to semantic link between s_1 and s_2, i.e., $sl_{1,2}$ as $\langle s_1, Sim_{\mathcal{T}}(NetworkConnection, SlowNetworkConnection), s_2 \rangle$ (Fig. 4).

In the following, we focus on two main properties related to semantic links: 1) their valuation and 2) the definition of their *Missing* and *Common Descriptions*.

3.2.1 Valuation

The matchmaking function $Sim_{\mathcal{T}}$ goes beyond the commonly used *Exact* matching type and covers the four well-known matching types [42] plus the extra matching type Intersection [18], [36]:

1. **Exact ($\equiv$).** If the output parameter Out_s_i of s_i and the input parameter In_s_j of s_j are equivalent concepts; formally, $\mathcal{T} \models Out\_s_i \equiv In\_s_j$.
2. **PlugIn ($\sqsubseteq$).** If Out_s_i is subconcept of In_s_j; formally, $\mathcal{T} \models Out\_s_i \sqsubseteq In\_s_j$.
3. **Subsume ($\sqsupseteq$).** If Out_s_i is superconcept of In_s_j; formally, $\mathcal{T} \models Out\_s_i \sqsupseteq In\_s_j$.
4. **Intersection ($\sqcap$).** If the intersection of Out_s_i and In_s_j is satisfiable; $\mathcal{T} \not\models Out\_s_i \sqcap In\_s_j \sqsubseteq \bot$.
5. **Disjoint ($\bot$).** Otherwise, Out_s_i and In_s_j are incompatible, i.e., $\mathcal{T} \models Out\_s_i \sqcap In\_s_j \sqsubseteq \bot$.

For instance, the PlugIn matching type means that an output parameter of a service s_i is subsumed by an input parameter of the succeeding service s_j. These matching types are not all mutually exclusive. Since our focus is on measuring the best achievable quality, it is important to order them. In this case, we assign the first one which is satisfied (e.g., Exact rather than Plugin, and Subsume rather than Intersection). From the perspective of goal satisfaction, using of Exact or PlugIn is comparable, and then the match type based on equivalence would not be strictly required. However, the distinction is important regarding the quality

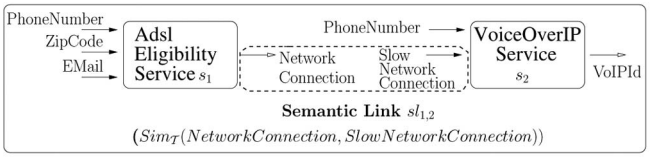

Fig. 4. A semantic link $sl_{1,2}$ between s_1 and s_2.

of matching types (e.g., their semantic proximity) which is the focus of this work.

Example 3 (Matching Type). Consider the semantic link $sl_{1,2}$ between s_1 and s_2 in Example 2 and Fig. 4. Such a link is valued by a Subsume matching type since $\mathcal{T} \models NetworkConnection \sqsupseteq SlowNetworkConnection$ with respect to $\mathcal{T}$ in Fig. 2.

The function $Sim_{\mathcal{T}}$ enables, at design time, finding some levels of semantic compatibilities (i.e., *Exact*, *PlugIn*, *Subsume*, and *Intersection*) and incompatibilities (i.e., *Disjoint*) among independently defined service descriptions.

Remark 1 (Semantic Links Computation). Semantic links and their values are estimated during a preprocessing phase [34] bounded by the product of number of input and output parameters of services. This phase can be improved, without impacting the rest of our approach, by modeling both the input and the output parameters of the same service as two different concepts defined as conjunction of the inputs and the outputs, respectively [44].

3.2.2 Missing and Common Description

Computing the matching type of a semantic link can be completed with a more detailed information, i.e., the DL concept descriptions: *Missing* and *Common Descriptions* (first defined as the *Extra* and *Common Descriptions* in [31]) between Out_s_i and In_s_j of a link $sl_{i,j}$.

On the one hand, the computation of *Missing Descriptions* is done by exploiting a nonstandard inference matching type: the *difference* or subtraction operation [8] for comparing $\mathcal{EL}$ DL-based descriptions, thus obtaining a compact representation of the metric:

1. the *Missing Description* $In\_s_j \backslash Out\_s_i$

$$In\_s_j \backslash Out\_s_i \doteq \min_{\preceq_d} \{E | E \sqcap Out\_s_i \equiv In\_s_j \sqcap Out\_s_i\}, \quad (1)$$

which refers to information required by In_s_j but not provided by Out_s_i in order to ensure a correct data flow between s_i and s_j (i.e., all information which is a part of the description In_s_j but not a part of the description Out_s_i). The computation of (1) is elaborated with respect to the subdescription ordering $\preceq_d$ [29], proposed to deal with syntactical redundancies. According to this ordering, the more removed primitive concept names and exixtential restrictions in the description, the more is the reduction. In the same way as In_s_j, Out_s_i, their conjunction, and their difference, E refers to a DL concept description. The *Missing Description* returned by (1) is not only necessary to explain where a semantic link-based composition may fail, but also why a semantic link failed and how to improve it.

On the other hand, the *Common Description* of Out_s_i and In_s_j is defined as

2. their Least Common Subsumer [15] lcs as a DL concept description, i.e.,

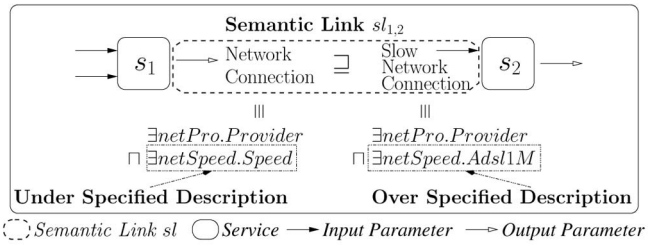

Fig. 5. A semantic link and its subsume matching type.

$$lcs(Out\_s_i, In\_s_j) \doteq \{F | Out\_s_i \sqsubseteq F \wedge In\_s_j \sqsubseteq F \wedge \forall F' : Out\_s_i \sqsubseteq F' \wedge In\_s_j \sqsubseteq F' \Rightarrow F \sqsubseteq F'\}, \quad (2)$$

which refers to information required by In_s_j and actually provided by Out_s_i.

Example 4 (Missing and Common Description). Suppose the semantic link $sl_{1,2}$ between s_1 and s_2 (Example 2 and Fig. 4), valued by a Subsume matching type (Example 3). According to (1) and Fig. 5, such a link requires a semantic refinement to be applied in a composition of web services.

The description missing in NetworkConnection to be plugged in the input parameter SlowNetwork-Connection is referred by the Missing Description, i.e., $SlowNetworkConnection \backslash NetworkConnection$, i.e., $\exists netSpeed.Adsl1M$.

In addition, the Common Description defined by the Least Common Subsumer of the output parameter of s_1 and the input parameter of s_2 is referred by the information required by $SlowNetworkConnection$ and effectively provided by $NetworkConnection$, i.e., $lcs(SlowNetworkConnection, NetworkConnection)$, i.e., $NetworkConnection$.

Finally, we remark that the intersection between the output parameter *NetworkConnection* and the Missing Description *SlowNetworkConnection**NetworkConnection*, i.e., $\exists netSpeed.Adsl1M$ is an Exact match with `Slow-NetworkConnection`.

Example 4 illustrates a link that does not make the composition seamless (in terms of semantics of data exchanged). Indeed, some descriptions are missing in the link $sl_{1,2}$, in terms of what is expected by its "sink" service s_2. This simply means that s_2 cannot accept or proceed data from the "feeder" service s_1. Indeed, data provided by s_1 (as output) are not specific enough compared to data accepted by s_2 (Subsume matching type—Example 3). Thus, data from s_1 should be filtered first, so a step of mediation is required to make the connection seamless. In the case of a PlugIn matching type, there are no missing descriptions since s_2 could proceed any data more specific than its input parameter type.

Remark 2 (Alternatives for Missing Description). Other approaches such as the 1) difference operator [49] or 2) Concept Abduction [17] can be used to remove from a given description all the information contained in another description. On the one hand, (1) is a refinement of [49]'s difference that considers the syntactic minimum ($\preceq_d$)

between incomparable $\mathcal{ALE}$ descriptions instead of a semantic maximum (ordering according to the subsumption operator). The result of the former does not contain redundancies and its result is more readable by a human user. On the other hand, concept abduction considers $\mathcal{ALN}$ DLs and their authors also define several minimality criteria. Teege [49] and (1) perform an equivalence between two concept descriptions ($\mathcal{T} \models E \sqcap Out\_s_i \equiv In\_s_j$ or $E \sqcap Out\_s_i \equiv In\_s_j \sqcap Out\_s_i$), whereas the concept abduction computes a subsumption of concept descriptions ($\mathcal{T} \models E \sqcap Out\_s_i \sqsubseteq In\_s_j$).

3.3 Semantic-Link-Based Composition Model

We introduce the concept of a goal that can be attached to any service or composition and then adapt the Definition 1 by presenting *Semantic-Link-based Composition*.

Definition 4 (Goal). *Given a TBox $\mathcal{T}$, a goal $\mathcal{G}(e)$ of element e is defined by the 4-tuple $\langle In\_e, Out\_e, \mathcal{P}_e, \mathcal{E}_e \rangle$ where In_e, Out_e are, respectively, input and output parameters as DL descriptions in $\mathcal{T}$. The preconditions $\mathcal{P}_e$ and effects $\mathcal{E}_e$ are, respectively, Horn-like rules [37] expressed in terms of DL concepts In_e and Out_e.*

The proposed rules are required to model and reason on facts related to conditions, i.e., preconditions and effects (e.g., through implication).

Definition 5 (Semantic-Link-Based Composition). *A Semantic-Link-based Composition c is defined as a partial ordering of services achieving a goal $\mathcal{G}(c)$ wherein services are connected with semantic links.*

In the following, we will focus on compositions defined by Definition 5, where all input parameters of services, except the initiator services, are connected to another output parameter by means of a semantic link. The process model of such compositions and their semantic links could be specified by means of any existing behavior modeling languages suitable for expressing typical control-flow dependencies such as statechart [24]. A *state* in our composition refers to a web service being activated. *Transitions* between states signify the passing of control from one service to another one, and can, therefore, be labeled with semantic links. In addition, some basic composition flow constructs such as sequence (i.e., linear compositions), conditional branching (i.e., OR-Branching), and concurrent threads (i.e., AND-Branching) can be found. To simplify the presentation and align our work with composition constructs supported by many approaches [9], [13], [45], [46], [55], we initially assume that all considered statecharts are acyclic.

In case a composition's statechart model contains cycles, a technique for *unfolding* it into its acyclic form can be applied first, e.g., [56].

3.3.1 Template-Based Composition Model

In this work, we assume a higher level of compositions generated by *template-based* and *parametric-design-based* (or task-based) approaches [40], [51]. They consist in compositions of tasks (achieving a specific goal).

Definition 6 (Tasks). *Tasks are defined in the same way as semantic web services (i.e., input, output parameters and preconditions, effects), but no binding to specific services is attached.*

We conceptualize it as tasks T_i and T_j connected by abstract semantic links $sl^A_{i,j}$. These links are used to express branching or data flow in task-based compositions.

Example 5 (A Task-Based Composition). Suppose $T_{i,1 \leq i \leq 8}$ are eight tasks involved in a task-based composition. This task-based composition, illustrated in Fig. 7, comprises a sequence of tasks, an OR-Branching, an AND-Branching, and nine abstract semantic links. The conditional output parameter of task T_1 can be bound either to the input parameter of T_2 or to the input parameter of T_4. Therefore, two potential conditional compositions are possible at design time, i.e., one that contains T_2 and another one that contains T_4, depending on the result of T_1. The output parameter of T_5 is semantically linked to the inputs of tasks T_6 and T_7, i.e., T_6 and T_7 are done concurrently.

We now focus on concretization of task-based compositions: how to assign well-defined web services to tasks? This will usually be chosen from a set of candidates. In more detail, we assume that each task T_i can be concretized by any service s_i if and only if both task and service have *close* goal (Definition 7).

Definition 7 (Close Goal of Service and Task). *The goals $\mathcal{G}(T_i)$ of a task T_i and $\mathcal{G}(s_i)$ a service s_i are close if they meet the following requirements:*

1. *all the functional input (and output) parameters of T_i have semantic similarities (see matching types in Section 3.2.1) with all s_i's input (and output) parameters and*
2. *the preconditions of s_i (logically) imply the preconditions of T_i, and the effects of T_i are (logically) implied by the effects of s_i.*

Note that all goals refer to DL descriptions in an ontology of goals. This ensures to estimate their semantic proximity and make difference between tasks and services having same inputs and outputs but achieving different things with them. In the following, we will focus on DL descriptions of input and output parameters, i.e., item 1) of Definition 7.

Example 6 (Task and a Candidate Web Service). According to Definition 7, it is obvious that service s_2 illustrated in Fig. 6 is a candidate to achieve task $\texttt{VoiceOverIP}^T$. They have the same overall goal, i.e., computing a valid address given a valid network connection and a valid phone number. Both service and task have same preconditions and effects. Moreover, there are semantic relationships between parameters of the service and task (see Fig. 6):

- They have the same semantic description for the input parameter `Phone Number`, i.e., they are linked with the Exact matching type.
- The input parameter `SlowNetworkConnection` of `VoiceOverIP` (s_2) is subsumed by `Network-Connection`, i.e., the input parameter of task $\texttt{VoiceOverIP}^T$.

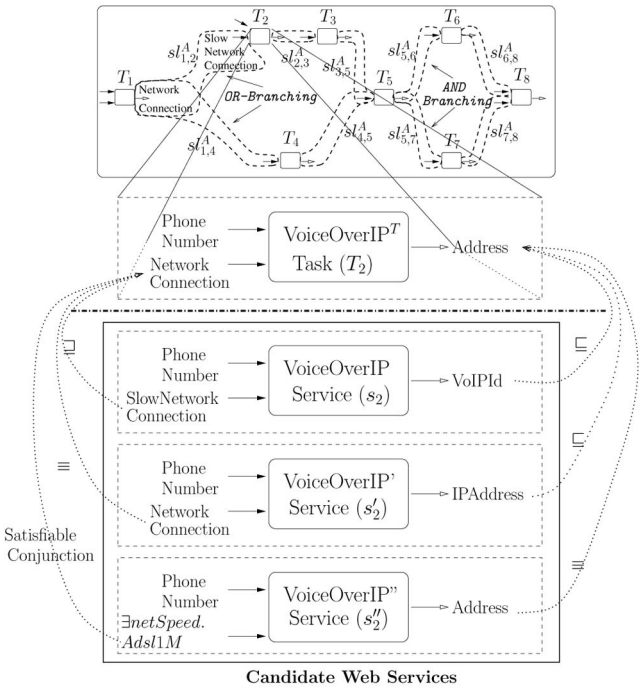

Fig. 6. A task and its collection of candidate services.

- The output `Address` of `VoiceOverIP`^{T} subsumes the output parameter `VoIPId` of `VoiceOverIP`.

We can also imagine an extension of the latter example wherein a number of candidate services with a *close* goal and different nonfunctional properties (e.g., QoS) could achieve the same task T_i in the task-based composition.

Example 7 (A Collection of Candidate Services). Fig. 6 illustrates a collection of services that are candidates to achieve task `VoiceOverIP`^{T}, i.e., $\{s_2, s'_2, s''_2\}$.

Example 8 (A Semantic-Link-Based Composition). Suppose Example 5 and its task-based composition. Fig. 7 illustrates one of its concretization: this refers to a particular case of the composition described in Example 7 and illustrated in Fig. 7; s_1 is `AdslEligibility` and s_2 is `VoiceOverIP`. Services $s_{i,1\leq i\leq 8}$ are selected candidate services to achieve, respectively, tasks $T_{i,1\leq i\leq 8}$ in the task-based composition. The semantic links $sl^1_{i,j}$ have been selected to concretize the abstract semantic link $sl^A_{i,j}$ between tasks T_i, T_j.

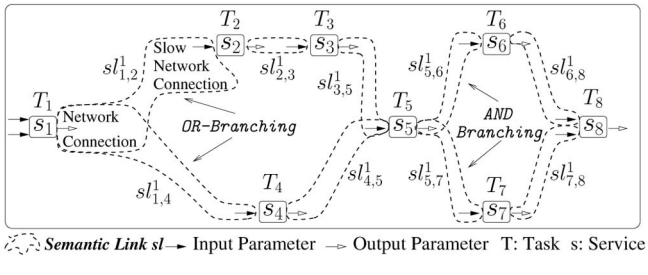

Fig. 7. Concretization of a composition.

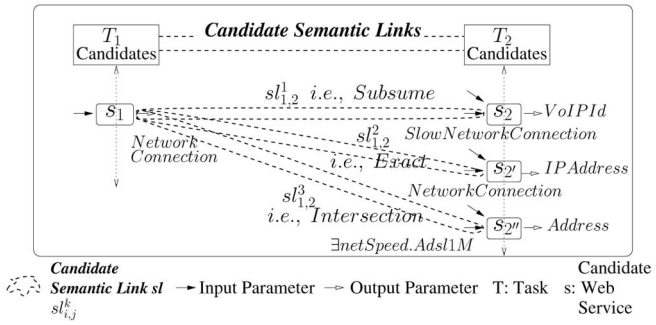

Fig. 8. Candidate semantic links between tasks T_1, T_2.

Selecting a service for a task impacts the semantic links between web services, e.g., their matching type (Section 3.2.1) or their common description (2).

Example 9 (Concretization of a Composition). Given the abstract semantic link $sl^A_{1,2}$ between tasks T_1, T_2 (Example 5), suppose we have one candidate s_1 that achieves task T_1, and three candidate services s_2, s'_2, and s''_2 that could achieve task T_2 (Fig. 6). Thus, tasks T_1 and T_2 can be related by three candidate semantic links (Fig. 8): 1) $sl^1_{1,2}$ between s_1, s_2, valued by Subsume; 2) $sl^2_{1,2}$ between s_1, s'_2, valued by Exact; and 3) $sl^3_{1,2}$ between s_1, s''_2, valued by Intersection.

Even if our approach is illustrated on direct links (e.g., between s_2 and s_3 in Fig. 7), it is also suitable for nondirect links. For instance, an extension of Fig. 7 with links between s_3 and s_8 fits to our model.

3.4 Focus of Investigation

Examples 8 and 9 illustrate how a task-based composition can be concretized by choosing a service s_i for T_i and thus semantic links $sl^1_{i,j}$ for the abstract ones $sl^A_{i,j}$. Since a task can be achieved by more than one web service, a large number of potential concretization of compositions can achieve the same or a *close* goal. Here, we address the issue of selecting and composing a large and changing collection of semantic web services to achieve a set of tasks, while among to optimize both:

- the nonfunctional quality of the composition; and
- the overall quality of links within the composition.

4 A Two-Dimensional Quality Model

Different services and semantic links can be, respectively, used to concretize tasks and abstract links. A way to differentiate their services (e.g., s_2, s'_2, and s''_2 in Example 7) and semantic links (e.g., $sl^1_{1,2}$, $sl^2_{1,2}$, and $sl^3_{1,2}$ in Example 9) consists in considering their respective nonfunctional and functional properties.

For this purpose, we first present a model to value the quality of semantic links. Then, we suggest to extend it with a measure of nonfunctional properties (here QoS) to estimate both quality levels of any compositions. Finally, we turn our attention to web service composition through aggregation functions for sequential, AND-Branching, and OR-Branching compositions. For each criterion, we provide

a definition, provide rules to compute its value for a given service and semantic link, and indicate motivations. Finally, we draw some directions for adapting and extending the quality model.

4.1 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 1) *Common Description* rate (Section 3.2.2), and 2) *Matching Quality* (Section 3.2.1).

4.1.1 Common Description Rate

Definition 8 (Common Description Rate of a 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*

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

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

The expressions in between $|$ refer to the size of $\mathcal{EL}$ concept descriptions ([29] p.17), i.e., $|\top|$, $|A|$, and $|\exists r|$ are 1; $|C \sqcap D| \doteq |C| + |D|$; $|\exists r.C|$ is $1 + |C|$. For instance, $|Adsl1M|$ is 4 with respect to Fig. 2.

Example 10 (Common Description Rate of a Link). Suppose the Example 9 and its Fig. 8. According to (3), the common description rate of $sl^1_{1,2}$, i.e., $q_{cd}(sl^1_{1,2})$ is

$$\frac{|lcs(NC, SlowNC)|}{|SlowNC \backslash NC| \ + \ |lcs(NC, SlowNC)|} \quad i.e., \ \frac{|NC|}{|\exists netSpeed.Adsl1M| \ + \ |NC|} \ i.e., \frac{1}{2}, \quad (4)$$

where NC stands for NetworkConnection. By applying (3) on $sl^2_{1,2}$, we obtain

$$q_{cd}(sl^2_{1,2}) = 1. \quad (5)$$

Regarding the common description rate of $sl^3_{1,2}$, i.e., $q_{cd}(sl^3_{1,2})$, we obtain with C_1 and C_2, respectively, defined by $\exists netSpeed.Adsl1M$ and $\exists netSpeed.Speed$:

$$\frac{|lcs(NC, C_1)|}{|C_1 \backslash NC| \ + \ |lcs(NC, C_1)|} \ i.e., \ \frac{|C_2|}{|C_1| \ + \ |C_2|} = \frac{3}{8}. \quad (6)$$

The common description rate is precomputed and provided through DL reasoning by Lécué and Delteil [31].

4.1.2 Matching Quality

Definition 9 (Matching Quality of a 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 [34] to compare semantic links and their values. Such an ordering is based on the binary and logical implication relation of Intersection from 1) PlugIn and Exact and also 2) Subsume and Exact. For instance, regarding 1), if $\mathcal{T} \models Out\_s_i \equiv In\_s_j$ (Exact), then $\mathcal{T} \models Out\_s_i \sqsubseteq In\_s_j$ (PlugIn), and then $\mathcal{T} \not\models Out\_s_i \sqcap In\_s_j \sqsubseteq \bot$ (Intersection). From a technical view, the assignment of values to matching types is driven by data integration. Behind each matching type, tasks of Extensible Markup Language (XML) 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 terms of computation costs) for the *Exact* matching type than for *PlugIn* matching type.

Example 11 (Matching Quality of a Link). According to the Example 9, Fig. 8, and the previous definition of matching quality, we have

$$q_m(sl^1_{1,2}) = \frac{1}{2}, \quad (7)$$

$$q_m(sl^2_{1,2}) = 1, \quad (8)$$

$$q_m(sl^3_{1,2}) = \frac{1}{4}. \quad (9)$$

Contrary to q_{cd}, q_m does not estimate similarity between the parameters of links but gives a general overview (discretized values) of their semantic relationships. We focus on a more abstract view of semantic valuation by introducing this criterion. In the same way as the common description rate, our system advertises the matching quality of links by precomputing them.

4.1.3 Combining the Qualities

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

$$q(sl_{i,j}) \doteq \big(q_{cd}(sl_{i,j}), q_m(sl_{i,j})\big). \quad (10)$$

By considering this quality model, we aim at evaluating the level of heterogeneity between data (as output, input parameters) exchanged by services. Obviously, a link with a quality of $(1, 1)$ refers to a composition of two services exchanging data with the same semantic description, i.e., the same data model. Alternatively, a lower quality could refer from slight to high mismatches between exchanged data. In most of these cases, data mediators are required to ensure that incoming services can proceed data from outgoing services. Therefore, in case of "imperfect" semantic links (not $(1, 1)$), composition designers are expected to inspect the difference between input and output parameters and then to compensate with data mediators [19] to create seamless compositions. We can also imagine requesting external companies which provide such mediators. Another potential technique should be to discover new relevant services that can be plugged to an "imperfect" semantic links to compensate the missing parts [31].

Example 12 (Quality of Semantic Links). Suppose the task T_3 be achievable by a service s_3 that checks the availability of an $IPAddress$ (as input parameter) and provides an acknowledgment status. Assume a subset of the task-based composition in Fig. 7 wherein T_1, T_2, and

TABLE 1
Quality Values of Candidate Semantic Links

Abstract Semantic Link $sl^A_{i,j}$		$sl^A_{1,2}$			$sl^A_{2,3}$		
Candidates		$sl^1_{1,2}$	$sl^2_{1,2}$	$sl^3_{1,2}$	$sl^1_{2,3}$	$sl^2_{2,3}$	$sl^3_{2,3}$
Vector $q(sl^k_{i,j})$	q_{cd}	$\frac{1}{2}$	1	$\frac{3}{8}$	$\frac{1}{3}$	1	$\frac{1}{3}$
	q_m	$\frac{1}{2}$	1	$\frac{1}{4}$	$\frac{1}{4}$	1	$\frac{1}{2}$

T_3 could be, respectively, achievable by $\{s_1\}$, $\{s_2, s'_2, s''_2\}$, and $\{s_3\}$. Table 1 illustrates the quality values of their different candidate semantic links, i.e.,

1. $sl^1_{1,2}$ between s_1 and s_2,
2. $sl^2_{1,2}$ between s_1 and s'_2,
3. $sl^3_{1,2}$ between s_1 and s''_2,
4. $sl^1_{2,3}$ between s_2 and s_3,
5. $sl^2_{2,3}$ between s'_2 and s_3, and
6. $sl^3_{2,3}$ between s''_2 and s_3.

The selection of service s'_2 is the most appropriate choice for T_2 in order to maximize semantic links $sl^A_{1,2}$ (with $sl^2_{1,2}$) and $sl^A_{2,3}$ (with $sl^2_{2,3}$). Indeed, the latter choice ensures a seamless composition of s_1, s'_2, and s_3 since all links are valued by an Exact matching type. In other words, the services of this composition use exactly the same semantics to describe data they exchange or share. Alternatively, the selection of s_2 would require data mediators to ensure data provided by s_1 are correctly filtered to the format of the input parameter of s_1.

In case s_i and s_j are related by more than one semantic link, the value of each criterion is retrieved by computing their average. This average is computed by means of the Sequential/And-Branching row of Table 2, independently along each dimension of the quality model.

The quality of semantic links in Table 1 can be compared by analyzing 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, in case of conflicts, e.g., 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, we compare a weighted average (with a weight of $\frac{1}{2}$) of their normalized components.

Remark 3 (A Quality Model for $\mathcal{ALN}$ TBoxes). In (3) as well as in the matching quality metric, $Out\_s_i \sqcap In\_s_j$ is supposed to be satisfiable since only relevant links between two services are considered in our model.

However, our quality model can be extended for $\mathcal{ALN}$ TBoxes by first considering *abduction* rather than *difference*. When $Out\_s_i \sqcap In\_s_j$ is not satisfiable, we can adapt our model by computing contraction [16] between Out_s_i and In_s_j, and thus valuing the Disjoint matching type. Indeed, such a matching type will have undesired impacts on the composition and some service inputs may remain unsatisfied. In such cases, the approach of Lécué and Delteil [31] could discover new services or interact with the end users in order to satisfy these inputs.

4.2 QoS-Extended Quality of Semantic Link

Here, we extend the quality model (10) by exploiting the nonfunctional properties of services (also known as QoS attributes [41]) involved in each semantic link. We simplify the presentation by considering only:

- **Execution price.** The 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.** The response time $q_t(s_i) \in \Re^+$ of service s_i, i.e., the expected delay between the request and result moments.

The latter values of execution price and response time are given by service providers or third parties, e.g., by means of some logs of previous executions. A quality vector of a service s_i is then defined as follows:

$$q(s_i) \doteq (q_{pr}(s_i), q_t(s_i)). \tag{11}$$

Actually, depending on the attribute of q, i.e., either a semantic link $sl_{i,j}$ or a service s_i, we interpret q either as a quality of semantic link (10) or quality of service (11).

Given an abstract link between tasks T_i and T_j, one may select the link with the best functional quality (matching quality, common description rate), and nonfunctional (the cheapest and fastest services) quality values, or may be a compromise (depending on the user preferences) between the four by coupling (10) and (11). The selection could be influenced by predefining some constraints, e.g., a service response time lower than a given value.

On one hand, selecting links with the best functional quality will ensure easy end-to-end integration between services by minimizing semantic and syntactic mediators, and providing seamless deployment and execution of computed compositions. The choice of such criteria should be dominant to reduce cost of data mediation. The latter is rarely considered in state-of-the-art approaches, yet it could be really taxing in terms of either price (with outsourced mediators) or execution time (with internal mediators). On

TABLE 2
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	$\frac{1}{\lvert sl_{i,j}\rvert}\sum_{sl_{i,j}} q_{cd}(sl_{i,j})$	$\prod_{sl_{i,j}} q_m(sl_{i,j})$	$\frac{\sum_{s_i} q_t(s_i)}{\max_s q_t(s)}$	$\sum_{s_i} q_{pr}(s_i)$
OR-Branching	$\sum_{sl_{i,j}} q_{cd}(sl_{i,j}).p_{sl_{i,j}}$	$\sum_{sl_{i,j}} q_m(sl_{i,j}).p_{sl_{i,j}}$	$\sum_{s_i} q_t(s_i).p_{s_i}$	$\sum_{s_i} q_{pr}(s_i).p_{s_i}$

the other hand, selecting links including services with the best nonfunctional quality values will ensure quality of compositions in aspects understandable by most of users: price, response time. Such criteria should be dominant in case data shared by service are not as heterogeneous as in the previous case (e.g., services aligning their data at description time where most of their exchanged data match perfectly), or in case all data mediators are known at design time. Thus, applications as Intranet-based are appropriate, whereas web-based applications dealing with different services' providers and larger scale are not.

Example 13 (QoS-Extended Quality of Semantic Link). Suppose T_2 and two of the three candidate services s_2, s'_2 wherein $q(s'_2) < q(s_2)$. According to Example 12, s'_2 should be preferred regarding the quality of its links with s_1 and s_3, whereas s_2 should be preferred for its QoS. What about the best candidates for $sl^A_{1,2}$ and $sl^A_{2,3}$ regarding both criteria?

In the next section, we turn our attention on a quality model to value web service composition, and then address the issue related to Example 13 in Section 5.

4.3 Quality of Composition

Here, we present Definitions 10 and 11, which are required to compare and rank different compositions along the common description rate and matching quality dimension. The rules for aggregating quality values for any composition are provided in Table 2. The approach for computing semantic quality of such a composition c is adapted from the application-driven heuristics of Lécué et al. [33], while the computation of its nonfunctional QoS is similar to [12].

4.3.1 Common Description Rate

Definition 10 (Common Description Rate of a Composition). *The Common Description rate of a composition typifies a set of degree of similarity between all corresponding parameters of services linked by a semantic link, of which it is a function.*

The Common Description rate of a composition measures the average degree of similarity between all corresponding parameters of services linked by a semantic link. Definition 10 motivates how to compute such a rate (as a general view of the semantic link quality) for sequential, AND-Branching, and OR-Branching compositions. According to this definition, the common description rate Q_{cd} of both latter compositions c is defined as the arithmetic mean of the common description rate $q_{cd}(sl_{i,j})$ for its links $sl_{i,j}$. Thus, the overall common description rate of any compositions is a linear function of links' common description rate. The rates for different links are all considered with the same importance. The arithmetic mean has been considered for its simplicity but any other linear average could be applied.

The rate Q_{cd} of an OR-Branching composition

$$\sum_{sl_{i,j}} q_{cd}(sl_{i,j}).p_{sl_{i,j}}, \tag{12}$$

is defined as 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 runtime. Such probabilities (also required by Q_m, Q_t, and Q_{pr}) are initially specified by the composition designer, and then eventually updated considering the information obtained by monitoring the workflow executions.

4.3.2 Matching Quality

Definition 11 (Matching Quality of a Composition). *The matching quality of a composition estimates the overall matching quality of semantic links involved in the composition. Contrary to the common description rate, this criterion aims at easily distinguishing and identifying between very good and very bad matching quality.*

Definition 11 computes matching quality for sequential, AND-Branching, and OR-Branching compositions.

The matching quality Q_m of a sequential and AND-Branching composition c is defined as a product of $q_m(sl_{i,j})$. All different (nonempty) matching qualities involved in such compositions require to be considered together in a (nonlinear) aggregation function to make sure that compositions that contain 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 c is defined as (12) by changing $q_{cd}(sl_{i,j})$ by $q_m(sl_{i,j})$.

4.3.3 Execution Price

The execution price Q_{pr} of a sequential and AND-Branching composition c is a sum of the execution prices $q_{pr}(s_i)$ of services s_i that participate in c.

The execution price of an OR-Branching composition c is defined in as (12), by changing $q_{cd}(sl_{i,j})$ by $q_{pr}(s_i)$.

4.3.4 Response Time

The response time Q_t of a sequential composition c is a sum of the response time $q_t(s_i)$ of services s_i that participate in c, whereas the Q_t of an AND-Branching composition c is bound by the highest response time of its AND-Branches (due to the design-time composition).

The response time of an OR-Branching composition c is defined in as (12), by changing $q_{cd}(sl_{i,j})$ by $q_t(s_i)$.

4.3.5 Combination of Functional Quality and QoS

Using the aggregation rules described in Table 2, the quality vector of any composition can be defined by

$$Q(c) \doteq (Q_{cd}(c), Q_m(c), Q_t(c), Q_{pr}(c)). \tag{13}$$

In practice, the elements of the vector $Q(c)$ will, at this stage, be normalized and fed into an appropriate Multiple-Criteria Decision Analysis tool [28]. Details of this are outside the scope of this paper, which 1) treats the trade-offs using a weighted function presented later in function (15) and 2) focuses on choosing compositions where different qualities of components satisfy user-specified constraints.

4.4 Adapting and Extending Quality of Composition

Even if the introduced quality model focuses on specific function aggregations such as the heuristics-based Q_{cd}, Q_m, Q_t, and Q_{pr}, the method for computing the value of the quality criteria is not unique (e.g., valuation of the Matching type, execution price) and then can be adapted, depending

on the formalization requirements (e.g., other heuristics). Other computation methods can be designed to fit the needs of specific applications (see [12] and [56] for nonfunctional criteria). For instance, the *Common Description rate* q_{cd} can be changed by the *Missing Description* rate q_{md} in (14), i.e., the rate of description missing in the semantic links.

$$q_{md}(sl^k_{i,j}) = \frac{|In\_s_j \backslash Out\_s_i|}{|In\_s_j \backslash Out\_s_i| \ + \ |Out\_s_i \sqcap In\_s_j|}. \qquad (14)$$

Alternatively, the quality Q_{cd} of sequential, AND-Branching, and OR-Branching compositions can be computed by valuing the upper bound of common description of their semantic links rather than their average.

Although the adopted quality model has a limited number of criteria (for the sake of illustration), Definitions 10, 11 as well as 13 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 [31] in semantic links can be considered. Contrary to [33], 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. In addition, other nonfunctional criteria such as reputation, availability, reliability, successful execution rate, etc., can also be considered in such an extension.

5 Optimizing Service Composition

The second main feature of our approach is using Genetic Algorithms to compute the optimal composition among a set of potential solutions. First of all, we formalize the problem as a *Constraint Satisfaction Optimization Problem* (CSOP) and then apply Genetic Algorithms to compute an optimal solution that meets constraints on 1) the quality of their services and 2) the quality of their semantic links. To this end, the quality models (10) and (11) are used to, respectively, model local constraints on semantic links and services, whereas (13) is considered to model global constraints. We use GA for the optimization because of its scalability.

Example 14 (Compositions and Constraints). Given a composition of tasks (e.g., in Fig. 7) 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 she cannot accept a matching quality Q_m below a given limit. We can also imagine local constraints on specific tasks and semantic links. From these constraints, the end user could expect the optimal one regarding its quality.

5.1 Constraint Satisfaction Oriented Optimization

Here, we formalize web service composition as a *Constraint Satisfaction Optimization Problem*. We use the term CSOP to signify the addition of the requirement for finding an optimal solution to the standard *Constraint Satisfaction Problem* (CSP) as defined in [50].

CSOP is a key formalism for many optimization-driven combinatorial problems such as ours. The success of this paradigm is due to its simplicity, its natural mapping to several real-world applications, and especially the efficiency of existing underlying solvers. In addition, such a formalism allows a generic representation of any optimization-based web service composition problem with local and global constraints (Definition 12).

Definition 12 (Composition Driven CSOP). *A composition driven CSOP is defined as* (T, D, C, f)*:*

- T *is the set of tasks (*variables*)* $\{T_1, T_2, \ldots, T_n\}$ *defined in the composition;*
- D *is the set of* domains $\{D_1, D_2, \ldots, D_n\}$*, each* D_i *represents a set of services that fulfill the task* T_i*;*
- C *is the set of* constraints*, i.e., local* C_L *and global* C_G*; and*
- f *is an evaluation function that maps every solution tuple* $s \in S$ *of the CSP* (T, D, C) *to a numerical value. Given a solution tuple* s*,* $f(s)$ *is called the* f*-value of* s*.*

C_L and C_G are related to users constraints regarding both the semantic and nonfunctional quality of composition, services, and their semantic links. Unlike constraints C_L, which need to be satisfied for any given assignment (i.e., services, links) to specific tasks T, constraints C_G need to be met by the overall composition.

Solving a composition driven CSOP consists in finding the solution tuple (i.e., an assignment of services $s_{i,1\leq i\leq n} \in D_1 \times \cdots \times D_n$ to tasks $T_{i,1\leq i\leq n}$ that satisfy all the constraints C) with the optimal f-value with regard to the application-dependent optimization function f.

5.2 A Genetic Algorithm-Based Optimization

Determining the best set of services for a composition to optimize a set of quality constraints is an NP-hard optimization problem. The naive approach which finds all the solutions first, and then compares their f-values conceptualizes as an exhaustive search for the optimal composition among the exponential number of possible concretizations of compositions.

Since such an approach is impractical for large-scale composition, we address this issue by presenting a GA-based approach [22] which 1) supports constraints not only on QoS but also on quality of semantic links and 2) requires the set of selected services as a solution to maximize a given objective f.

5.2.1 GA Parameters for Optimizing Composition

Similar to other GA-based approaches, the optimal solution (represented by its *genotype*) is determined by simulating the evolution of an *initial population* along a number of generations leading to the survival of the *fittest* individuals (here compositions) satisfying some *constraints*. Each generation is obtained by *crossover*, *mutation*, and *selection* of compositions from the previous one. In more detail, we parameterized GA as following:

- **Genotype.** It is defined by an array of integers. The number of elements in the array is equal to the number of distinct tasks involved in the composition. Each element, in turn, contains an index to an array of candidate services for that task, indicating a specific chosen service. Thus, each composition, as a

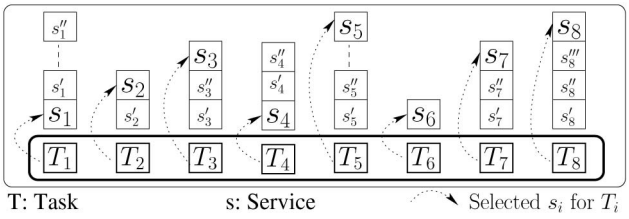

Fig. 9. Genotype encoding for service composition.

potential solution, can be encoded using this genotype. Fig. 9 illustrates the composition depicted in Fig. 7 using the genotype encoding.

- **Initial population.** It consists of an initial set of compositions (characterized by their genotypes) wherein services are randomly selected.
- **Global and local constraints.** They have to be met by compositions c, e.g., $Q_{cd}(c) > 0.8$.
- **Fitness function.** This function f is required to quantify the "quality" of any composition c by considering its genotype. f needs to maximize semantic quality attributes (higher values are better), while minimizing the QoS attributes (smaller values are better) of c:

$$f(c) = \frac{\omega_{cd}\hat{Q}_{cd}(c) + \omega_m\hat{Q}_m(c)}{\omega_{pr}\hat{Q}_{pr}(c) + \omega_t\hat{Q}_t(c)}, \quad (15)$$

where $\hat{Q}_{l\in\{pr,t,cd,m\}}$ refers to $Q_{l\in\{pr,t,cd,m\}}$ (using Table 2) normalized in the interval $[0,1]$. $\omega_l \in [0,1]$ is the weight assigned to the lth quality criterion and $\sum_{l\in\{pr,t,cd,m\}} \omega_l = 1$. In this way, preferences and constraints 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 toward constraints satisfaction. Thus, compositions that do not meet the constraints are penalized by extending (15) using an augmented Lagrangian:

$$f'(c) = f(c) - \omega_{pe} \sum_{l\in\{pr,t,cd,m\}} \left(\frac{\Delta\hat{Q}_l}{\hat{Q}_l^{\max}(c) - \hat{Q}_l^{\min}(c)} \right)^2, \quad (16)$$

where $\hat{Q}_l^{\max}$ and $\hat{Q}_l^{\min}$ are, respectively, the maximum and minimal value of the lth quality constraint among the values of other available candidate compositions c. For instance, $\hat{Q}_{cd}^{\max}$ refers to the maximum description rate among the description rates of all other compositions c. ω_{pe} weights the penalty factor. $\Delta\hat{Q}_{l\in\{pr,t,cd,m\}}$ is

$$\Delta\hat{Q}_l = \begin{cases} \hat{Q}_l - \hat{Q}_l^{\max} & \text{if } \hat{Q}_l > \hat{Q}_l^{\max} \\ 0 & \text{if } \hat{Q}_l^{\min} \le \hat{Q}_l \le \hat{Q}_l^{\max} \\ \hat{Q}_l^{\min} - \hat{Q}_l & \text{if } \hat{Q}_l < \hat{Q}_l^{\min} \end{cases}. \quad (17)$$

Contrary to [11], compositions that violate constraints do not receive the same penalty. Indeed, the factor ω_{pe} is further penalized in (16). This function avoids local optimal by considering also compositions that disobey constraints.

Unfortunately, the fitness function defined in (16) contains a penalty for 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 a 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 may allow, 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 $f''(c, gen)$ (to be maximized) is defined as follows:

$$f(c) - \omega_{pe} \cdot \frac{gen}{maxgen} \cdot \sum_{l\in\{pr,t,cd,m\}} \left(\frac{\Delta\hat{Q}_l}{\hat{Q}_l^{\max}(c) - \hat{Q}_l^{\min}(c)} \right)^2, \quad (18)$$

where *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:
  - *Composition mutation*, i.e., random selection of a task and replacing its service with another one among those available. Mutation is performed on each composition of the population with probability p_{mut}.
  - The standard two-points *crossover*, i.e., random combination of two compositions. In the same way as mutation, crossover is performed on each composition of the population, but with probability p_{cross} such that $p_{mut} < p_{cross}$. Further details on GA can be found, for example, in [22].
  - *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. We iterate until the constraints are met (i.e., $\Delta Q_l = 0\ \forall l \in \{pr, t, cd, m\}$). However, if this does not happen within a maximum number of generations, then no feasible solution has been found. Otherwise, once constraints are satisfied and feasibility is checked at the end of the search, we iterate until the best fitness composition remains unchanged (and its solution feasible) for a given number of generations. Therefore, solutions are do feasible at the end of the GA process.

5.2.2 Executing GA for Optimizing Composition

The execution of the GA consists in the following steps:

1. defining the initial population (as a set of compositions), and computing the fitness function (evaluation criterion) for each composition;
2. evolving the population by applying mutation and crossover of compositions;
3. selecting compositions;
4. evaluating compositions of the population; and
5. back to step 2 if the stopping criterion is not satisfied in the GA evolution.

Tasks for which only one candidate service is available are taken out from the GA evolution.

In case no solution exists, users are requested to relax constraints of the optimization problem in order to compute alternative solutions still providing a reasonable quality of composition (at both QoS and semantic levels).

6 Experimental Results

We analyze the performances of our approach by

- discussing, in Section 6.2, the benefits of combining QoS and functional criteria;
- comparing, in Section 6.3, the evolution of the composition quality's factors (i.e., Q_{cd}, Q_m, Q_{pr}, and Q_t) over the GA generations by considering both static (16) and dynamic constraint penalties (18);
- observing, in Section 6.4, the evolution of the composition quality f'' in (18) (we focus on equal weights assigned to the different quality criteria) over the GA generations by varying the number of tasks and candidate services;
- studying, in Section 6.5, the behavior of our approach regarding large-scale compositions;
- evaluating performance, in Section 6.6, after decoupling the GA and the (online) DL reasoning processes which are both required in our approach;
- comparing, in Section 6.7, GA with IP-based approaches; and
- focusing, in Section 6.8, on the performance of the GA process by comparing the convergence of our approach (18) with the Canfora et al. [11].

6.1 Context of Experimentation

6.1.1 Services, Semantic Links, and Their Qualities

The services are defined by their semantic descriptions using an $\mathcal{EL}$ TBox (formally defined by 1,100 concepts and 390 properties) in the Telecommunication domain, provided by a commercial partner. The use 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).

On the one hand, we have incorporated estimated values for QoS parameters (i.e., price and response time). The QoS values of services were varying according to some Gaussian distribution [10], [11], [56] function, and better duration time offers corresponded to higher prices. On the other hand, the functional quality of semantic links (i.e., common description rate and matching quality) has been automatically computed, given semantic descriptions of services and a DL reasoning process. These descriptions have been randomly selected from the TBox but satisfying Definition 7 and using the semantic descriptions of tasks they are supposed to be assigned.

Compositions with up to 30 tasks and 35 candidate services per task (i.e., a potential number of 35^2 candidate semantic links between each pair of tasks) have been considered in Sections 6.2, 6.3, 6.4, 6.6, and 6.8, especially for obtaining convincing results toward their applicability in real (industrial) scenarios.

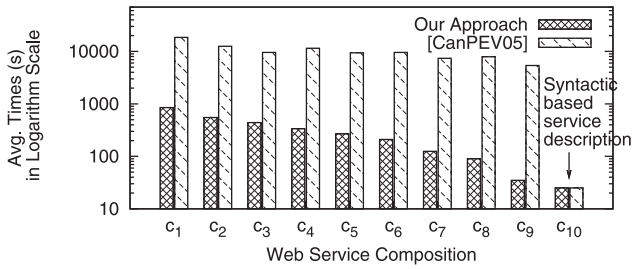

Fig. 10. Costs of data integration.

The quality of the composition is evaluated by means of the percentage gap (described as Max. Fitness (percent) in Sections 6.4, 6.5, 6.8, and percent of Global optimum in Section 6.7) of the GA solution with respect to the global optimum. The latter is obtained by running the IP approach with no time limit.

6.1.2 Implementation Details

The common description rate (3) is calculated by computing the *Missing Description* (1), the *Common Description* (2), and the size ([29], p.17) of DL concepts. These DL inferences and the matching types (Section 3.2.1) have been achieved by means of a DL reasoning process (adaptation of Fact++ [26] with DL difference).

The aggregation rules from Table 2 are then used for computing each quality dimension of any composition. Finally, the combination of QoS with semantic calculation is computed by means of (16) and (18), thus obtaining the final quality score for the composition.

6.1.3 Genetic Algorithm Process

Our GA is extending the GPL library JGAP (http://jgap.sourceforge.net/). The optimal compositions are computed using an elitist GA where the best two compositions are kept alive across generations. A crossover probability of 0.7, a mutation probability of 0.1, and a population of 200 compositions are used. The roulette wheel selection has been adopted as selection mechanism. Since the GA performance varies with the value assigned to global constraints, we use a simple stopping criterion of 400 generations. Indeed, under severe global constraints, our GA may not find a feasible solution. Beyond these values for parameters, there is a little deterioration in terms of performance.

For each experiment, GA is executed 50 times and average values are reported. Standard deviation has been observed as always below the 5 percent of the mean values. Experiments were also replicated on compositions of different sizes and complexity, confirming the results reported below. The experiments have been conducted on Intel(R) Core(TM)2 CPU, 2.4 GHz, and 2 GB RAM.

6.2 Benefits of Combining Quality Criteria

In this first experiment (Fig. 10), we focus on the benefits of combining QoS and functional criteria. To this end, we study the impact of higher functional quality on the costs of data integration enabling the composition of services. The data integration process aligns the data flow of a composition by manipulating and transforming the semantic

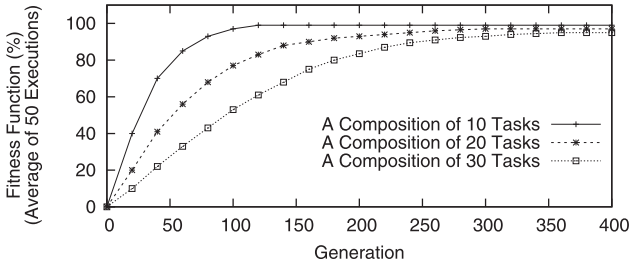

Fig. 11. Evolution of the composition quality.

descriptions of contents of outgoing message and incoming messages of annotated web services.

Our approach and the method of Canfora et al. [11] are compared on 10 compositions $c_{i,1\leq i\leq 10}$ with $Q_{cd}(c_i) = 0.1 \times i$ and $Q_m(c_i) = 10^{i-10}$ as functional quality to reflect gradually better quality.

On one hand, as expected, the costs of data integration (through data flow) are trivial for both approaches when compositions with the best quality are considered, e.g., c_{10} and the composition is without mismatch in data flow. On the other hand, these costs decrease with the increase of functional quality of compositions in our approach, whereas they are steady but very high for compositions computed by Canfora et al. [11]. This is due to 1) the lack of specification of functional quality (hence, a hard task to build semantic data flow from scratch), and 2) the manual approach used to link data in the composition. Therefore, appropriate qualities of semantic links are very useful for discovering data flow in composition, then limiting their costs of data integration.

6.3 Evolution of Satisfaction Constraints

To compare the different evolution of QoS and nonfunctional constraints during optimization, we present results obtained optimizing a composition containing 35 candidate services for each of 30 distinct tasks.

As shown in Fig. 12, the optimization problem is constrained on common description rate, matching quality, execution price, and response time. The evolution shows how the GA is able to find a solution that meets the constraint and, at the same time, optimizes the different parameters of the composition quality function of f'' in (18) (i.e., maximizing the common description and matching quality while minimizing execution price and response time). For our optimization problem, the dynamic fitness does not outperform the static fitness. Even different calibrations of the fitness weights did not help. Different tests with significance level $\alpha = 5$ percent showed that differences were not significant, except for the common description rate where the dynamic fitness increased faster, although at the end of the evolution, the result achieved was not significantly different.

TABLE 3
Overview of Computation Costs

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

6.4 Evolution of the Composition Quality

Fig. 11 reports the evolution of the composition quality over a number of GA generations, for different number of tasks, with 35 candidate services per task. In such a configuration, the potential number of semantic links between two tasks T_i and T_j is $|s_i| \times |s_j|$. This figure illustrates different levels of convergence to an optimized composition that meets the constraints.

Table 3 presents the computation costs, the number of generations required to obtain the maximal fitness value.

The convergence results of Table 3 can be improved by further investigating population diversity, evolution and selection policies (cf. Section 2 and [38]), or enhancement of initial population policy.

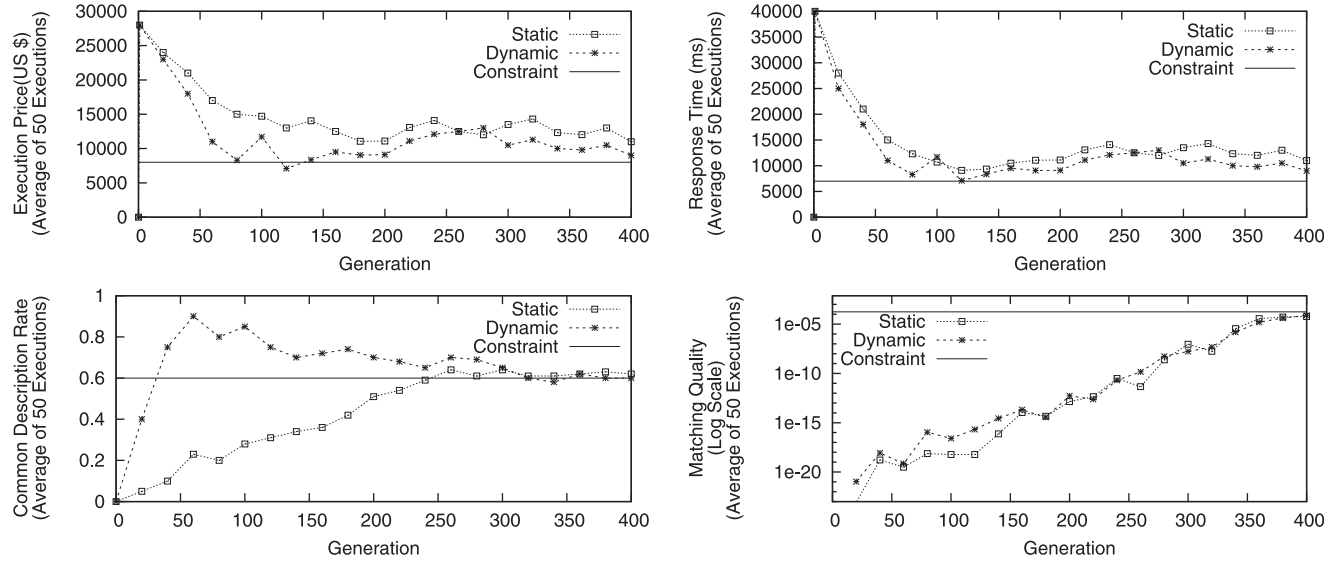

Fig. 12. Evolution of satisfaction constraints—static versus dynamic fitness.

TABLE 4
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

6.5 Toward Large-Scale-Based Compositions

We study our approach with a large number of tasks (up to 500 tasks) and candidate services (500). We focus on its scalability and the impact of the number of generations as well as the population size on the GA success.

As illustrated in Table 4, 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.

Regarding the optimization 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 percent of the maximum, whereas considering a number of generations of 3,000 and a population size of 1,000 achieves 95 percent 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 percent. This shows that each optimization problem converges to a limit. In our GAs-based approach, this limit is often identified as a local optimum (approximately 95 percent of the global optimum in our experiments).

6.6 Decoupling GA Process and DL Reasoning

Contrary to QoS given, in general, by providers, the quality of semantic links is estimated according to DL reasoning through Subsumption for q_m, Difference and Least Common Subsumer lcs for q_{cd}. Since most of the computational costs of our approach are mainly dependent on both reasoning mechanisms and the GA-based optimization process (i.e., the five steps of Section 5.2.2), we decouple them and report the breakdown of the computation costs presented in Table 3 in Fig. 13. For each individual (or composition) of each generation, we evaluate the quality of their semantic links. The results of all computations are stored in order to avoid redundant computation of quality of similar links.

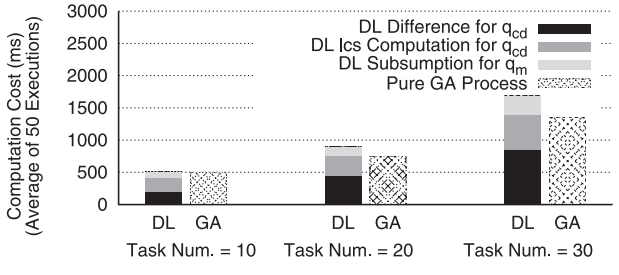

Fig. 13. DL and GA processes in our approach.

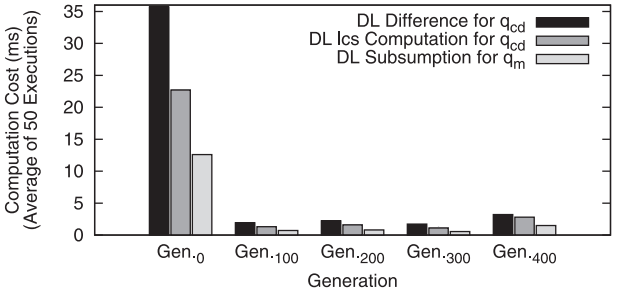

Fig. 14. DL computation costs along GA generations.

DL reasoning is the most time-consuming process in optimization where the number of tasks and candidate services are greater than 10 and 35. This is caused by the complexity of q_{cd} computation through combination of DL Difference, LCS, and Subsumption even if the subsumption problem is polynomial in $\mathcal{EL}$ DL.

Fig. 14 provides a more detailed view of DL computation costs along the GA generations for a composition of 30 tasks with 35 potential services each (42 semantic links are present). As expected, the DL computation costs are the most important in the initialization step (see Gen.$_0$) since a quality estimation of 42 semantic links is required to evaluate the quality of each of the 200 compositions. These costs are estimated to 4.2 percent of the overall DL computation costs along the 400 generations, whereas these costs are estimated to only 0.24 percent for each next generation (mainly because of the low probability of mutation, i.e., 0.1). In case of a higher probability of mutation, the proportion of latter costs increases for each generation. The more the generations, the less the impact of the DL computation costs during the GA initialization step. However, in case of reduced problems (e.g., in terms of tasks, services, and number of generations), with a low probability of mutation, it should be better to precompute some semantic qualities, improving the performance of the whole optimization process.

6.7 Comparing IP- and GA-Based Approaches

As mentioned before, the IP-based approach is one of the most adopted method to solve optimization in web service composition for both semantic and nonfunctional optimization. An analytic description of this approach is out of the scope of this paper, for details, see [56], [12], [33], or [4].

However, here, we compare our approach with the IP-based approach of Zeng et al. [56] and Lécué et al. [33] by, respectively, extending their quality criteria to 1) semantic links and 2) QoS. To this end, we focus on the computation (or convergence) time of both approaches to optimize a revisited fitness function of (16) with close solutions (<1 percent for each quality criteria) meeting the same constraints. The quality criteria $\hat{Q}_{cd}$ and $\hat{Q}_{pr}$ of (16) are unchanged, whereas $\hat{Q}_m$ and $\hat{Q}_t$ are linearized (e.g., by taking the logarithm for the aggregation function $\hat{Q}_m$) 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 [52] (LINDO API version 5.0, Lindo Systems, Inc., http://www.lindo.com/).

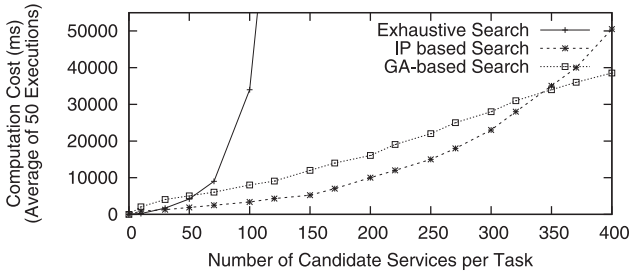

Fig. 15. Costs for reaching 95 percent of the global optimum.

Here, we studied a composition of 500 tasks wherein the number of candidate services varies from 1 to 400.

IP- and GAs-based approaches are compared along two set of experiments. First, we focus on computation costs to obtain a suboptimal solution that reaches 95 percent of the global optimum. The results reported in Fig. 15 support the adoption of GAs for local optimum search with a large number of candidate services per task (>340). Such results are, in parts, explained by the exponential number of IP variables required to represent the search tree as the set of potential compositions. On the contrary, the size of the GA problem is bound by its population and its genotype, which is bound by the number of tasks in the composition.

The second set of experiments focuses on computation costs to obtain the global optimum rather than local optimum. The results shown in Fig. 16 support the adoption of IP-based composition optimization, especially in case of a global optimum search. Indeed, CPLEX (also used by Ardagna and Pernici [4]) is very efficient in finding a feasible solution with these large instances which is very close to the global optimum. On the contrary, reaching the global optimum with GAs is more problematic.

Contrary to linear IP approaches, GAs do not impose linearity constraints on the quality aggregation rules (and thus on the objective function and constraints) and thus allow the handling of generic and customized quality criteria such as q_m and q_t.

According to these results, a valuable direction of further work would be to define a mechanism for selecting the best approach to be adopted for a given composition task and domain, to ensure acceptable computation costs of optimal composition. The latter constitutes an important requirement for many scenarios, such as interactive (or soft-real-time) service-oriented systems.

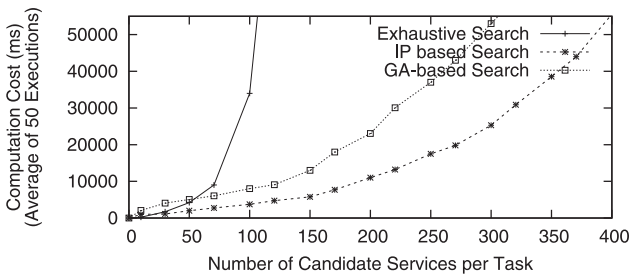

Fig. 16. Costs for reaching 99 percent of the global optimum.

TABLE 5
GA-Based Approaches (Population Size of 200)

Tasks Num.	Approach	Max. Fitness (%)	Generation Num.	Time (ms)
10	Our Model (18)	99	120	1012
	[CanPEV05]	97	156	1356
20	Our Model (18)	97	280	1650
	[CanPEV05]	94	425	2896
30	Our Model (18)	95	360	3142
	[CanPEV05]	85	596	6590

6.8 Convergence of GA-Based Approaches

In this experiment, we compare the convergence of our approach in (18) with the main alternative at present [11]. To this end, the functional criteria of our approach are disregarded in order to focus only on the GA-driven aspects of the optimization process.

According to Table 5, the advantage of our approach is twofold. First, we obtain better fitness values for the optimal composition than the approach of Canfora et al. [11] (actually an average of 97 percent of the maximum). Second, our approach converges faster than the approach of Canfora et al. [11]. Our function also avoids getting trapped by local optimums by 1) further penalizing compositions that disobey constraints (the factor ω_{pe} in (18)) and 2) suggesting a dynamic penalty, i.e., a penalty having a weight that increases with the number of generations.

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

7 Conclusion

To address *QoS*-aware *semantic* web service composition in a context of *significant scale*, we propose a GA-based approach to optimizing web service compositions, which considers both the nonfunctional qualities of the composition, and the quality of semantic fit. Combining both allows us to consider both the user perspective to desired qualities, and the composition perspective of costs involved in interservice alignment within the composition.

The first feature of our approach is an innovative and extensible model to evaluate the quality of 1) web services (i.e., QoS), 2) their semantic links, and 3) their compositions. The semantic extension aims at evaluating the level of heterogeneity between the output and input parameters of services. In cases of low quality of semantic links, reflecting mismatches between exchanged data of services, data mediators are required to ensure seamless compositions. Composition designers are then expected to compensate the difference between services parameters by means of data mediators, the cost of this is provided by our semantic quality model. The high costs of the data integration in the overall process of service composition can also be reduced by combining QoS and functional quality of semantic links proposed here.

The second feature of our approach concerns the use of GA-based approach for discovering near-optimal compositions under a large number of candidate services. This relies on formalizing our extended quality model as an optimization problem with multiple constraints. The GA-based

approach was shown to be faster than IP under some circumstances, especially when looking for "close enough" solutions with more than 350 candidate services for each task. The experimental results have also shown acceptable computation costs of our approach despite the time-consuming process of the DL reasoning.

In terms of further work, one direction is to consider compositions based on the contextual availability of web services. Indeed, our approach relies on a precomputed tasks model, this forces the goal to be satisfied according to the designed subtasks and constraints the choice among available services. Such precomputations may lead to unsatisfied goals depending on the context. In real world instead, more flexibility is required.

Another direction for future work is to consider how goals (and more specially preconditions and effects—Definition 7) of tasks and their candidate web services match together (e.g., from a semantic dimension point of view). For instance, a task can be fulfilled by services achieving a goal with more general or specific preconditions and effects, but which still fit the goal of the task. In the same way, we can reason with cases where both the preconditions and effects of tasks and their candidate services do not match in a semantically precise fashion.

It would be interesting to consider further research in a more precise difference operator, which is also easy to compute in expressive DLs. According to the results in Section 6.6, it seems important to optimize DL reasoning to scale up the overall process of optimization.

Determining the most appropriate parameters for the GA phase requires further experimentation if this is to be made truly usable for its target end users. Section 6 presents some GA parameterization, but guidelines to set up these parameters need to be investigated. In addition, we will investigate how to improve the GA performance especially in the last iterations, when we want to preserve feasible solutions.

Another area of investigation is the selection of initial compositions of tasks as a powerful tool to optimize the overall quality [40], [51], at present we take the task composition template as given. Two final considerations for further work would be the dynamic distribution of CSOP to improve convergence time of our approach in larger domains, and security aspects of service composition quality.

Acknowledgments

This work is conducted within the European Commission VII Framework IP Project SOA4All (Service Oriented Architectures for All) (http://www.soa4all.eu/), Contract No. IST-215219.

References

[1] R. Aggarwal, K. Verma, J.A. Miller, and W. Milnor, "Constraint Driven Web Service Composition in METEOR-S," *Proc. Int'l Conf. Services Computing*, pp. 23-30, 2004.

[2] M. Alrifai and T. Risse, "Combining Global Optimization with Local Selection for Efficient QoS-Aware Service Composition," *Proc. Int'l Conf. World Wide Web*, pp. 881-890, 2009.

[3] A. Ankolenkar, M. Paolucci, N. Srinivasan, and K. Sycara, "The Owl-S Coalition, Owl-S 1.1," technical report, 2004.

[4] D. Ardagna and B. Pernici, "Adaptive Service Composition in Flexible Processes," *IEEE Trans. Software Eng.*, vol. 33, no. 6, pp. 369-384, June 2007.

[5] I.B. Arpinar, R. Zhang, B. Aleman-Meza, and A. Maduko, "Ontology-Driven Web Services Composition Platform," *Information Systems and E-Business Management*, vol. 3, no. 2, pp. 175-199, 2005.

[6] F. Baader and W. Nutt, *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge Univ. Press, 2003.

[7] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific Am.*, vol. 284, no. 5, pp. 34-43, 2001.

[8] S. Brandt, R. Kusters, and A. Turhan, "Approximation and Difference in Description Logics," *Proc. Knowledge Representation*, pp. 203-214, 2002.

[9] A. Brogi, S. Corfini, and R. Popescu, "Composition-Oriented Service Discovery," *Proc. Fourth Int'l Workshop Software Composition*, pp. 15-30, 2005.

[10] G. Canfora, M. Di Penta, R. Esposito, and M.L. Villani, "A Framework for QoS-Aware Binding and Re-Binding of Composite Web Services," *J. Systems and Software*, vol. 81, no. 10, pp. 1754-1769, 2008.

[11] G. Canfora, M. Di Penta, R. Esposito, and M.L. Villani, "An Approach for QoS-Aware Service Composition Based on Genetic Algorithms," *Proc. Genetic and Evolutionary Computation Conf.*, pp. 1069-1075, 2005.

[12] J. Cardoso, A.P. Sheth, J.A. Miller, J. Arnold, and K. Kochut, "Quality of Service for Workflows and Web Service Processes," *J. Web Semantics*, vol. 1, no. 3, pp. 281-308, 2004.

[13] J. Cardoso and A. Sheth, "Semantic E-Workflow Composition," *J. Intelligent Information Systems*, vol. 21, pp. 191-225, 2003.

[14] D.B. Claro, P. Albers, and J.K. Hao, "Selecting Web Services for Optimal Composition," *Proc. Second Int'l Workshop Semantic and Dynamic Web Processes*, pp. 32-45, 2005.

[15] W.W. Cohen, A. Borgida, and H. Hirsh, "Computing Least Common Subsumers in Description Logics," *Proc. 10th Nat'l Conf. Artificial Intelligence (AAAI '92)*, pp. 754-760, 1992.

[16] S. Colucci, T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello, "Concept Abduction and Contraction in Description Logics," *Proc. Description Logics*, 2003.

[17] S. Colucci, T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello, "Concept Abduction and Contraction for Semantic-Based Discovery of Matches and Negotiation Spaces in an E-Marketplace," *Electronic Commerce Research and Applications*, vol. 4, no. 4, pp. 345-361, 2005.

[18] T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello, "A System for Principled Matchmaking in an Electronic Marketplace," *Proc. Int'l Conf. World Wide Web*, pp. 321-330, 2003.

[19] J. Euzenat and P. Shvaiko, *Ontology Matching*. Springer-Verlag, 2007.

[20] D. Fensel, M. Kifer, J. de Bruijn, and J. Domingue, "Web Service Modeling Ontology (WSMO) Submission, w3c Member Submission," http://www.w3.org/Submission/WSMO/, June 2005.

[21] M. Gillmann, G. Weikum, and W. Wonner, "Workflow Management with Service Quality Guarantees," *Proc. SIGMOD Int'l Conf. Management Of Data*, pp. 228-239, 2002.

[22] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[23] T.R. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199-220, 1993.

[24] D. Harel and A. Naamad, "The Statemate Semantics of Statecharts," *ACM Trans. Software Eng. and Methodology*, vol. 5, no. 4, pp. 293-333, 1996.

[25] A. Ben Hassine, S. Matsubara, and T. Ishida, "A Constraint-Based Approach to Web Service Composition," *Proc. Int'l Semantic Web Conf.*, pp. 130-143, 2006.

[26] I. Horrocks, "Using an Expressive Description Logic: FaCT or Fiction?," *Proc. Knowledge Representation*, pp. 636-649, 1998.

[27] J. Kopecký, T. Vitvar, C. Bournez, and J. Farrell, "SAWSDL: Semantic Annotations for WSDL and XML Schema," *IEEE Internet Computing*, vol. 11, no. 6, pp. 60-67, Nov./Dec. 2007.

[28] C.-L. Hwang and K. Yoon, "Multiple Criteria Decision Making," *Lecture Notes in Economics and Mathematical Systems*, Springer Verlag, 1981.

[29] R. Küsters, *Non-Standard Inferences in Description Logics*. Springer, 2001.

[30] O. Lassila and S. Dixit, "Interleaving Discovery and Composition for Simple Workflows," *Proc. Semantic Web Services, AAAI Spring Symp. Series*, pp. 22-26, Mar. 2004.
[31] F. Lécué and A. Delteil, "Making the Difference in Semantic Web Service Composition," *Proc. 22nd Nat'l Conf. Artificial Intelligence (AAAI '07)*, pp. 1383-1388, 2007.
[32] F. Lécué and A. Léger, "A Formal Model for Semantic Web Service Composition," *Proc. Int'l Semantic Web Conf.*, pp. 385-398, 2006.
[33] F. Lécué, A. Delteil, and A. Léger, "Optimizing Causal Link Based Web Service Composition," *Proc. European Conf. Artificial Intelligence*, pp. 45-49, 2008.
[34] F. Lécué, O. Boissier, A. Delteil, and A. Léger, "Web Service Composition as a Composition of Valid and Robust Semantic Links," *Int'l J. Cooperative Information Systems*, vol. 18, no. 1, pp. 1-62, Mar. 2009.
[35] Q. Liang, X. Wu, and H. Chuin Lau, "Optimizing Service Systems Based on Application-Level QoS," *IEEE Trans. Services Computing*, vol. 2, no. 2, pp. 108-121, Apr. 2009.
[36] L. Li and I. Horrocks, "A Software Framework for Matchmaking Based on Semantic Web Technology," *Proc. Int'l Conf. World Wide Web*, vol. 52, no. 5, pp. 331-339, 2003.
[37] J.W. Lloyd, *Foundations of Logic Programming*, second ed. Springer, 1987.
[38] Y. Ma and C. Zhang, "Quick Convergence of Genetic Algorithm for QoS-Driven Web Service Selection," *Proc. Computer Networks*, pp. 1093-1104, 2008.
[39] S.A. McIlraith, T.C. Son, and H. Zeng, "Semantic Web Services," *IEEE Intelligent Systems*, vol. 16, no. 2, pp. 46-53, Mar./Apr. 2001.
[40] E. Motta, "Reusable Components for Knowledge Modelling Case Studies," *Parametric Design Problem Solving*, IOS Press, 1999.
[41] J. O'sullivan, D. Edmond, and A.H.M. ter Hofstede, "What's in a Service?," *Distributed and Parallel Databases*, vol. 12, nos. 2/3, pp. 117-133, 2002.
[42] M. Paolucci, T. Kawamura, T.R. Payne, and K. Sycara, "Semantic Matching of Web Services Capabilities," *Proc. Int'l Semantic Web Conf.*, pp. 333-347, 2002.
[43] C.H. Papadimtriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982.
[44] A. Ragone, T. Di Noia, E. Di Sciascio, F.M. Donini, S. Colucci, and F. Colasuonno, "Fully Automated Web Services Discovery and Composition through Concept Covering and Concept Abduction," *Int'l J. Web Services Research*, vol. 4, no. 3, pp. 85-112, 2007.
[45] J. Rao, P. Küngas, and M. Matskin, "Composition of Semantic Web Services Using Linear Logic Theorem Proving," *Information Systems*, vol. 31, nos. 4/5, pp. 340-360, 2006.
[46] E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. Nau, "HTN Planning for Web Service Composition Using SHOP2," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 1, pp. 377-396, 2004.
[47] M.K. Smith, C. Welty, and D.L. McGuinness, "Owl Web Ontology Language Guide," *W3c Recommendation*, W3C, 2004.
[48] K.P. Sycara, M. Paolucci, A. Ankolekar, and N. Srinivasan, "Automated Discovery, Interaction and Composition of Semantic Web Services," *J. Web Semantics*, vol. 1, no. 1, pp. 27-46, 2003.
[49] G. Teege, "Making the Difference: A Subtraction Operation for Description Logics," *Proc. Knowledge Representation*, pp. 540-550, 1994.
[50] E. Tsang, *Foundations of Constraint Satisfaction*. Academic Press, 1993.
[51] B. Wielinga and G. Schreiber, "Configuration-Design Problem Solving," *IEEE Expert*, vol. 12, no. 2, pp. 49-56, Mar./Apr. 1997.
[52] L. Wolsey, *Integer Programming*. John Wiley and Sons, 1998.
[53] D. Wu, B. Parsia, E. Sirin, J.A. Hendler, and D.S. Nau, "Automating DAML-S Web Services Composition Using SHOP2," *Proc. Int'l Semantic Web Conf.*, 195-210, 2003.
[54] T. Yu and K.J. Lin, "Service Selection Algorithms for Composing Complex Services with Multiple QoS Constraints," *Proc. Int'l Conf. Service Oriented Computing*, pp. 130-143, 2005.
[55] R. Zhang, I.B. Arpinar, and B. Aleman-Meza, "Automatic Composition of Semantic Web Services," *Proc. Int'l Conf. Web Services*, pp. 38-41, 2003.
[56] L. Zeng, B. Benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-Aware Middleware for Web Services Composition," *IEEE Trans. Software Eng.*, vol. 30, no. 5, pp. 311-327, May 2004.

Freddy Lécué received the PhD degree from the École des Mines de Saint-Etienne, France, in 2008. He is a research fellow with the Centre for Service Research at the University of Manchester, United Kingdom. His research interests include knowledge representation and reasoning, and service-oriented computing.

Nikolay Mehandjiev received the PhD degree from the University of Hull, United Kingdom, in 1997. He is a senior lecturer with the Centre for Service Research at the University of Manchester, United Kingdom. He researches the design of flexible service systems, using software agents and collaborative service composition models. He has coauthored two books and has guest edited three special issues of international journals, including the *Communications of ACM*.