QoS-aware optimal and automated semantic web service composition with user's constraints

Amina Bekkouche, Sidi Mohamed Benslimane, Marianne Huchard, Chouki Tibermacine, Fethallah Hadjila, Mohammed Mohammed Merzoug

Noname manuscript No.
(will be inserted by the editor)

QoS-Aware Optimal and Automated Semantic Web Service Composition With User's Constraints

Amina BEKKOUCHE · Sidi Mohammed BENSLIMANE · Marianne HUCHARD · Chouki TIBERMACINE · Fethallah HADJILA · Mohammed MERZOUG

Received: date / Accepted: date

Abstract Automated semantic web service composition is one of the critical research challenges of service-oriented computing, since it allows users to create an application simply by specifying the inputs that the application requires, the outputs it should produce, and any constraints it should respect. The composition problem has been handled using a variety of techniques, from Artificial Intelligence (AI) planning to optimization algorithms. However no approach so far has focused on handling three composition dimensions simultaneously, producing solutions that are: (1) fully functional (i.e. fully executable) by using a mechanism of semantic matching between the services involved in the solutions, (2) are optimised according to non-functional Quality of Service (QoS) measurements, and (3) respect global QoS constraints. This paper presents a novel approach based on a Harmony Search (HS) algorithm that addresses these three dimensions simultaneously through a fitness function, to select the optimal or near optimal solution in semantic web service composition. In our approach, the search space is modeled as a Planning Graph structure which encodes all the possible composition solutions for a given user request. To improve the selection process we have compared the original Harmony Search algorithm with its recently developed variants Improved Harmony Search (IHS) algorithm and Global Best Harmony Search (GHS) algorithm. An experimentation of the approach conducted with an extended version of the Web Service Challenge 2009 dataset showed that: 1) our approach is efficient and effective to extract the optimal or near optimal composition in diverse scenarios; and 2) both variants IHS and GHS algorithms have brought improvements in terms of fitness and execution time.

Keywords Semantic Web Service Composition · Semantic Matching · Planning Graph · Harmony Search Algorithm · Quality of Service (QoS)

Amina BEKKOUCHE
Computer Science Departement, Abou Bekr Belkaid University of Tlemcen, B.P 119 Faculty of Sciences, Tlemcen, Algeria
E-mail: ami_ bekkouche@mail.univ-tlemcen.dz

Sidi Mohammed BENSLIMANE
LabRi Laboratory, École Supérieure en Informatique, BP 73, El Wiam City, Sidi Bel Abbes, Algeria
E-mail: s.benslimane@esi-sba.dz

Marianne HUCHARD
LIRMM, CNRS and Montpellier University, 161 rue Ada, 34095 Montpellier, France
E-mail: marianne.huchard@lirmm.fr

Chouki TIBERMACINE
LIRMM, CNRS and Montpellier University, 161 rue Ada, 34095 Montpellier, France
E-mail: chouki.tibermacine@lirmm.fr

Fethallah HADJILA
Computer Reasearch Laboratory, Abou Bekr Belkaid University of Tlemcen, B.P. 119 Faculty of Sciences, Tlemcen, Algeria
E-mail: f_hadjila@mail.univ-tlemcen.dz

Mohammed MERZOUG
Computer Reasearch Laboratory, Abou Bekr Belkaid University of Tlemcen, B.P. 119 Faculty of Sciences, Tlemcen, Algeria
E-mail: mohamed.merzoug@mail.univ-tlemcen.dz

1 Introduction

Web services provide a standardized way to achieve interoperability between heterogeneous software systems independently from underlying implementa-

tion technologies and platforms. However, the limited functionality offered by an atomic web service cannot always satisfy complex user needs and appropriately reflect intricate business processes [2]. Web service composition techniques attempt to solve such issues by combining and integrating suitable atomic web services into a composite one.

As the number of web services increases quickly, it is already beyond the human ability to generate the composite service manually. Thus the automated web service composition aiming at finding a composite service to satisfy the user request becomes an important technique to reuse existing resources and accelerate the development of web applications. However, composite services should go beyond achieving a concrete functionality and take into account other requirements such as Quality of Service (QoS) to generate compositions that fit the needs of users. QoS is a broad concept that encompasses a group of non functional properties, such as execution price, execution duration, availability, execution success rate, and reputation [65]. Given a set of multiple global QoS constraints, the challenge is how to efficiently construct a composite service such that its overall QoS is optimal, while all the QoS constraints are satisfied.

Most conventional approaches consider the search of correct work plans (automated web service composition problem) and the ones that give the optimal QoS (web service selection problem) as two separate sub-problems of the general QoS aware automated service composition problem.

Research in [65, 39, 19, 4, 66, 58, 18] solves the QoS aware service selection problem. They assume the work plan is pre-defined and each task in the plan is not a concrete service but representing a service class with multiple candidates of different QoS measures.

On the other hand, many automated composition approaches do not take the QoS attributes into account. For example, AI planning techniques have been traditionally used in service composition to generate valid composition plans by mapping services to actions in the planning domain [43, 8, 52, 51, 27, 1]. Some approaches [68] search all the solutions, while others [16] give priority to the results with fewer services or simpler work plans.

The problem of selecting composition solutions that optimize the overall QoS, while satisfying multiple global QoS constraints in automated semantic web service composition becomes a -combinatorial-optimization problem which is known to be NP-hard [41]. In this type of problem, optimization techniques are the most appropriate because due to the large number of similar services that would lead to a large number of candidate solutions, the search for the optimal or near optimal solution should be done in a short time and without processing the entire search space. Meta-heuristic algorithms are well known approximate algorithms which can solve optimization problems with satisfying results [65, 39]. The works in [44, 49, 21, 46] address the problem partially by exploiting AI planning or graph based automated service composition algorithms and selecting the optimal composition based on QoS. However, owning to not considering QoS constraints, the optimal solution in their approach may not satisfy user's needs, such as price and reputation constraints.

The Harmony Search (HS) algorithm [14, 25] is one of the most recently developed optimization algorithms that mimics the behavior of a music orchestra when aiming at composing the most harmonious melody, as measured by an aesthetic standard. HS has obtained excellent results in many optimization problems [69, 22, 54] and has become a popular algorithm in the evolutionary computation field due to its superiority to many other algorithms. However, it has never been used for identifying the optimal composition in QoS aware automated service composition problem.

In this paper, we propose a novel method based on planning graph [44] and Harmony Search algorithm [14] for selecting the optimal composition solution. The user can specify multiple global QoS constraints and our method finds a composite service that optimizes the overall QoS, while satisfying those specified global constraints.

Our main contributions are :

1. We present an integrated framework for QoS aware optimal and automated semantic web service composition.
2. We propose a new mechanism for computing the semantic matching scores (degrees) between the services involved in the composition.
3. We also propose an efficient Harmony Search algorithm for finding the optimal composition that respects the global QoS constraints.
4. We compare our selection algorithm based on Harmony search with its recently developed variants Improved Harmony Search (IHS) algorithm and Global Best Harmony Search (GHS) algorithm in order to improve the selection process.

We experimented our proposal using the Web Service Challenge 2009 dataset[1] (a famous world competition about automatic composition of web services). The experimental study demonstrates the efficiency and the effectiveness of our approach in different settings. The remainder of the paper is organized as follows. Section 2 gives an overview of related work. In Section 3, we present a motivating scenario.In Section 4, we formalize the web service composition problem. In Section 5, we give the details on the proposed framework and in Section 6, we present the results of a set of experimental evaluations. Finally, in Section 7, we conclude with a summary of our contributions and the perspectives of this work.

2 State of the Art

Recently, research on web service composition has been receiving a lot of attention. Existing approaches can broadly be classified into one of the following categories: Automated web service composition approaches, QoS-aware service composition approaches and Hybrid approaches.

2.1 Automated web service composition

The automatic composition of services is a fundamental and complex problem in the field of Service Oriented Computing. Most approaches can be categorized into: 1) classical AI planning approaches [42], where the composition problem is translated into the planning domain and solved using general planners, and 2) graph-based I/O driven approaches [60] that build a graph with the services and their input/output semantic relations, and apply graph search techniques to extract service compositions from the graph. The approaches discussed below are representative of these two distinct research paths.

2.1.1 AI planning approaches

Theoretical works, such as the Causal Link Matrix (CLM)[31], provide a solid background for semantic web service composition through AI techniques. CLM is a formal theoretical model accommodating AI planning for web service composition. It involves precomputing all causal relations between semantic web services and using them to formulate valid compositions. Although it takes into account semantics, the lack of an implementation and experimental results does not allow us to draw conclusions about its scalability.

SHOP2 [53] was initially created as a general-purpose, heuristic-driven Hierarchical Task Network (HTN) planning system. It was later used for automated web service composition. OWL-S process models are encoded as SHOP2 domains, while the web service composition problem is encoded as a planning problem. Solutions are acquired by HTN planning. The main disadvantage of this approach is that the planning process, due to its hierarchical nature, requires certain decomposition rules to be encoded in advance with the help of a DAML-S process ontology. For hearing sound decomposition rules, prior expert knowledge of the domain is required.

The work in [37] represents atomic services as state transition operators and employs estimated-regression planning with heuristics to perform composition. In order to be used, it requires extension to current standards, while scalability results are not encouraging.

The approach presented in [38] attempts the modification of GOLOG [34] to adjust it to web service composition standards. The approach is based on intelligent agents having the ability to reason for automated service discovery and composition. User requirements and constraints are modeled through situation calculus. Consequently, GOLOG is used to find an appropriate composition plan. Encoding and translation processes in this approach are generally complex, while interoperability with existing systems and standards is decreased.

The SWORD system [43] describes available web services with the aid of EntityRelationship Models and Horn rules. Therefore, domain-specific knowledge is required. The final composition plan is derived through a rule-based expert system, requiring user involvement.

OWLS-XPlan [27] uses semantic descriptions of web services in OWL-S to derive planning domains and problems, and then invokes a planning module, called XPlan, to generate composite services. The system is compliant with an XML dialect of PDDL. However, semantic information provided from domain ontologies is not utilized therefore, the planning module requires exact matching between service inputs and outputs.

The work in [17] presents an integrated approach for automated semantic Web service composition using AI planning techniques. An important advantage of this approach is that the com-

[1] http://ws-challenge.georgetown.edu/wsc09/

position process, as well as the discovery of the atomic services that take part in the composition, are significantly facilitated by the incorporation of semantic information. The implementation was performed through the development and integration of two software systems, namely PORSCE II and VLEPPO. However user constraints and QoS were not addressed in this approach and just outlined as future goal.

The authors in [67] provided four strategies to remove the redundant Web services during a forward planning-graph generating process. They attempted to find a solution in the shortest time. However, without the backward pruning, the solutions may contain some redundant Web services. Moreover, they did not consider QoS and Semantics.

[35] focused on the semantic matching problem in the planning graph-based composition algorithms. In the forward search, it used the concept similarity and some predefined threshold to calculate the service matching degrees, which were added into the planning-graph as the weights of services. In addition, it removed services producing the same parameter but with lower weights to keep the planning-graph as simple as possible. It adopted a backward search to prune the redundant services. Unfortunately, the threshold is not easy to define, and this limits the application of this method.

The authors in [32] present an approach for performing automated data flow in web service composition by i) exploiting semantic matchmaking between web service parameters (i.e., outputs and inputs) to enable their connection and interaction,and ii) adapting XML database solutions, specifically XML Schema mapping, to perform syntactic data transformation and integration of exchanged messages. The system is implemented and interacting with web services dedicated on a Telecom scenario but QoS requirements were not considered in their approach.

2.1.2 Graph-based approaches

In [33], the authors develop an integrated framework for dynamic Web service composition. The framework exploits the semantic input-output matchmaking to discover relevant services and performs automatic composition using a graph-based approach, taking into account functional requirements. However, graph optimization are not considered and the composition search is non-optimal, since the selection of the services is merely greedy-based.

In [50], the authors present an approach to automatic service composition with semantic matching. Given a request (goals, inputs and outputs), a set of matching services are discovered from the repository, applying semantic matching between service properties and the composition request. Then, a graph is created dynamically by connecting semantically similar nodes (single services) to each other. Once the graph is created, a search over it is performed building acyclic tree structures from goal nodes to start nodes. However, there are no experimental results to validate the model.

In [57], three different approaches for service composition were used and compared: 1) an Iterative Depth-First Search approach; 2) a Greedy approach; 3) and an Evolutionary approach. These approaches consider a subsumption-based matching of services, but none of them considers non-functional characteristics or QoS attributes.

The authors in [60] present an automatic service composition algorithm using AND/OR graph. In this proposal, an AND/OR graph is created from a request, connecting services by their inputs and outputs. Then, a search over the graph is performed using the AO* search algorithm. Although this proposal shows a good performance over large repositories, the autors have not implemented optimization techniques in order to improve the scalability of the algorithm.

The work in [47] define a composition framework by means of integration with fine-grained I/O service discovery that enables the generation of a graph-based composition which contains the set of services that are semantically relevant for an input-output request. The proposed framework also includes an optimal composition search algorithm to extract the best composition from the graph minimizing the length and the number of services, and different graph optimisations to improve the scalability of the system but with no QoS support.

Another interesting graph-based approach has been presented in [29]. In this paper the authors present an efficient framework for Web service composition that supports semantic Web service discovery. The composition is generated by performing a forward chaining of operators to find a feasible composition. The authors also evaluated the system with the datasets of the Web Service Challenge 2006 and presented a detailed experimentation. Their results demonstrate the capabilities and the good performance of this system, however it does not analyse service optimisations to remove redundant information.

2.2 QoS-aware service composition

To guarantee local or global QoS requirements of service composition, the QoS-aware service composition has attracted the attention of a lot of researchers from different fields. Compared to automatic service composition, it assumes the existence of a predefined work plan with a set of abstract tasks, while the objective is to select service for each task from its candidate services to meet local or global QoS constraints. The QoS-aware service composition also requires that the QoS information attached to the services is predicted in a reliable manner. To do so, both the provider and the customer of the services have to use prediction algorithms such as collaborative filtering (CF) to facilitate the composition task.

In [61] the authors predict the QoS of the target service for a given location and time information. The proposed solution combine the advantages of memory-based techniques and model-based techniques, in particular the adoption of small clusters allows the scalability of the similarity computation, on the other hand the time information ensures the prediction accuracy. In the same line of thought the authors of [6] leverage model driven engineering to automatically instantiate the SLA (Service Level Agreements) of SaaS (Software as a Service) compositions. They also enrich the resulting model with cloud concepts such as storage, networks and computation ressources.

Assuming that the prediction of the QoS information is correctly performed, the next task consists in reducing the composition problem into a combinatorial Knapsack problem, which is generally solved using constraint satisfaction algorithms (such as Integer Programming) [64, 63] or Evolutionary Algorithms [7, 56, 28].

In [64] the authors present AgFlow, a QoS middleware for service composition. They analyze two different methods for QoS optimization, a local selection and a global selection strategy. The second strategy is able to optimize the global end-to-end QoS of the composition using an Integer Linear Programming method, which performs better than the suboptimal local selection strategy.

In [63], the authors model the problem as a multichoice 0-1 knapsack problem or a multiconstraint optimal path problem, and then compute optimal result according to the objective function. Similarly, in [3] the authors propose a hybrid QoS selection approach that combines a global optimization strategy with local selection for large-scale QoS composition.

The authors in [4] propose an approach for facilitating web service selection according to user requirements. These requirements specify the needed functionality and expected QoS, as well as the composability between each pair of services. The originality of their approach is embodied in the use of Relational Concept Analysis (RCA), an extension of Formal Concept Analysis (FCA). They classify services by their calculated QoS levels and composability modes. They use a real case study of 901 services to show how to accomplish an efficient selection of services satisfying a specified set of functional and non-functional requirements.

In [7] and [55], genetic algorithms are used to find the optimal composition solution.The composition method is based on a given service abstract workflow, where each abstract service has a set of candidate concrete services with different QoS values associated. Genetic algorithms are used to bind concrete services to the abstract ones aiming at identifying the optimal workflow instance in terms of QoS attributes. The genome is encoded as an integer array in [7] and as a binary string in [55], where each position is associated to an abstract service in the workflow and indicates the concrete service which is selected to be used. Both approaches make use of genetic operators and fitness functions applied on the genome to find the optimal composition solutions.

A hybrid method combining Particle Swarm Optimization (PSO)[23] with Simulated Annealing is proposed in [56] for selecting the optimal service composition solution based on QoS attributes. A composition solution is considered as the position of a particle in PSO, while velocity is used to modify a composition solution. To avoid the problem of premature stagnation in a local optimal solution, a Simulated Annealing-based strategy is introduced which produces new composition solutions by randomly perturbing an initial solution.

In [59] the authors use an immune algorithm to tackle the composition problem. First, they encode the problem into an antibody. This method has two steps: the Immune selection operation and Clonal selection operation. In immune selection, antibodies are proliferated and suppressed in order to control their density in the mating pool and also to make sure antibodies that are helpful and potential (vaccine) will not be destroyed. In Clonal operation, they use antibodies with a high fitness as a heuristic information for speeding up conver-

gence. They consider several control flow operators of business process, and take into account several QoS attributes in the fitness function like service cost and service response time. Antibodies with the best fitness are considered as vaccine in the algorithm.

A hybrid meta-heuristic method that combines Tabu Search and simulated annealing is proposed by [28] to search for a high quality constraint-compliant service composition plan. Applying Tabu list and probabilistic move to inferior plans enables this approach to find solutions quickly.

In [10], the authors suggest an efficient service selection mechanism based on the social harmony search algorithm to help clients select services with considering the quality attributes. In order to demonstrate the capabilities of the applied algorithm in the quality optimization of a composite service it is compared to a similar optimization problem with a genetic algorithm approach and the corresponding results revealed that the presented algorithm consumes less time finding the optimum combination and therefore is more efficient.

In [20], a harmony search based method is proposed for web service selection. This research lacks pitch adjustment parameter in improvisation phase -which is one of the important parameters of the algorithm- and this lack is justified by this statement that QoS values of equivalent services are not relevant to each other. Therefore, pitch adjustment does not give rise to an improvement of an objective function value and therefore, it acts just the same as random selection in improvisation.

2.3 Hybrid approaches

Hybrid approaches combine elements of AI planning or graph based and optimisation techniques for solving the composition problem with correct functionally, optimised QoS solutions [46, 21, 49].

In [46] the authors present a graph-based approach for automatic composition of web services that generates semantic input output based compositions with optimal end-to-end QoS, minimizing the number of services of the resulting composition. The proposed approach has four main steps: 1) generation of the composition graph for a request; 2) computation of the optimal composition that minimizes a single objective QoS function; 3) multi-step optimizations to reduce the search space by identifying equivalent and dominated services; and 4) hybrid local-global search to extract the optimal QoS with the minimum number of services.

In [5] the authors propose an approach by considering a two-phase composition process. The composition first proceeds to generate an abstract plan based on web service types using Dynamic Description Logics (DDL). This abstract plan is then concretized into an executable plan by selecting the appropriate Web service instances based on non-functional requirements.

In [9] The authors propose a novel approach based on the planning-graph to solve the top-k QoS-aware automatic composition problem of semantic Web services. The approach includes three sequential stages: a forward search stage to generate a planning-graph to reduce the search space of the following two stages greatly, an optimal local QoS calculating stage to compute all the optimal local QoS values of services in the planning, and a backward search stage to find the top-K composed services with optimal QoS values according to the planning-graph and the optimal QoS value.

Qsynth in [21] is proposed to address both scalability and accuracy by using QoS objectives of a service request as the search directives. This approach effectively prunes the search space and significantly improves the accuracy of the search results. However, these methods can maximize only one QoS attribute during the planning process and QoS constraints are not taken into consideration.

In [44], an approach that combines AI planning and an immune-inspired algorithm is used to perform fully automated QoS-aware Web service composition, also considering semantic properties. One significant contribution of this work is the proposal of an enhanced planning graph, which extends the traditional planning graph structure by incorporating semantic information such as ontology concepts. Given this data structure, the composition algorithm selects the best solution configuration from a set of candidates. A fitness function considering QoS values and semantic quality is used to identify the best solution, and a clonal selection approach is employed to perform the optimisation. Candidate cells (solutions) are cloned, matured (mutated by replacing services with others from the same cluster in the EPG) and the cell most suited to combating the invading organism (i.e. the best solution) is discovered. Similarly, in [49] the authors propose two hybrid algorithms: Cuckoo Search and Firefly Algorithms, for selecting the optimal solution in semantic web service composition using the same model of enhanced

planning graph. The proposed algorithms combine principles from population-based meta-heuristics with principles from trajectory-based meta-heuristics and reinforcement learning to optimize the search process in terms of execution time and fitness value reflecting the QoS and semantic quality of a solution.

2.4 Discussion

In general all these approaches for web service composition only consider one or two aspects of QoS aware automated service composition problem.

Most of the automated web service composition approaches (Section 2.1) have mainly focused on exploiting semantic techniques and developing heuristics to generate valid composition plans. Unfortunately, none of these methods generate optimal QoS aware compositions nor respect the user constraints.

Furthermore, both the composition workflow and the service candidates for each abstract task in QoS-aware service composition approaches (Section 2.2) are assumed to be predefined beforehand. So these techniques are not able to produce compositions with variable size. In addition, they usually optimize only two or three QoS attributes.

Although the hybrid approaches (Section 2.3) propose an efficient way to produce optimized QoS composition solutions, they do not include any discussion on the issue of producing solutions that satisfy global QoS constraints, which is crucial in such service-oriented environments with real business settings. The second limit of these methods is that they focus only on sequential compositions and they do not deal with more complex composition structures such as the parallel composition construct in which multiple web services are executed concurrently.

To address the above issues, we propose a novel approach that simultaneously considers three composition dimensions producing solutions that are: (1) fully functional by using a mechanism of semantic matching between the services involved in the solutions, (2) are optimised according to non-functional properties (our approach can handle n QoS attributes but in the experiments we have considered 5 attributes), and (3) fulfil global QoS constraints.

3 Motivating Scenario

In this section we present a motivating scenario to illustrate our approach for automatic web service composition. We use an application from the car brokerage domain [65]. A typical scenario would be of a customer, say Bob, planning to buy a used car having a specific model, mark, and mileage. To purchase an entire car package, Bob would first like to know the price quote of the selected car and the vehicle history report. He then needs to get the insurance quote. Finally, since Bob needs the financing assistance, he also wants to know the financing quote. Consider that all these subtasks are published by businesses as web services and our goal is to compose them for purchasing a car. Examples of Web services that need to be accessed include Car Purchase, Car Insurance, and Financing. We also anticipate that there will be multiple competitors to provide each of these web services. For instance, the customers have to choose the most suitable Insurance company that best meets his preference and price requirements (i.e. the sum of the price quote and the insurance cost must not exceed the predefined budget (the global QoS constraints has to be preserved). In addition, Bob may choose an Insurance company that provides the shortest periods of accident repayments. This QoS criterion must be optimized during the composition process (the QoS dimension is very important in this case).

4 Web Service Composition Problem

The automatic composition of services requires a mechanism to select appropriate services based on their functional descriptions, as well as to automatic match the services together by linking the outputs of some services to the inputs of others to generate executable compositions. To this end, we introduce in this section the main concepts that we use to tackle the QoS-aware automatic web service composition problem.

Definition 1 (QoS) *A quality-of-service attribute Q^i is a value representing the non-functional property of a web service, such as response time, throughput, cost, reputation and so on. The QoS attributes can be categorized into two classes [45]. One is negative, the higher is the value, the lower is the quality, such as response time and cost. The other is positive, the higher is the value, the higher is the quality, such as throughput and reputation.*

Definition 2 (Global QoS Constraints) *Given an n-tuple of QoS attributes $\langle Q^1, Q^2, ...Q^n \rangle$, global*

QoS constraints, denoted as G, is an n-tuple of QoS values $\langle g_1, g_2, ...g_n\rangle$. Each $g_i \in G$ is a lower bound on a positive QoS attribute Q^i and/or an upper bound on a negative QoS attribute Q^i.

Each global QoS constraint in G is used to restrict its corresponding QoS attribute as the global QoS value of a composition solution.

Definition 3 (Ontology Tree) *An ontology tree is defined as a 2-tuple Γ=$\langle C,R\rangle$, where:*

1. *C is the set of concepts represented by nodes in the ontology tree.*
2. *R is the set of hierarchical relationships represented by edges in the ontology tree. If a concept c_2 is a direct sub-concept of another concept c_1, it can be denoted as $c_2 \sqsubseteq c_1$, (see Definition 5).*

Definition 4 (Semantic Web Service) *Given an ontology Γ=$\langle C,R\rangle$, a semantic web service is defined as a triple w_i=$\langle$ I_{w_i}, O_{w_i}, QoS_i $\rangle$, where :*

1. *$I_{w_i} \subseteq C$ represents the set of concepts that semantically describe the input parameters of the web service.*
2. *$O_{w_i} \subseteq C$ represents the set of concepts that semantically describe the output parameters of the web service.*
3. *QoS_i is an n-tuple $\langle Q_i^1, Q_i^2, ...Q_i^n\rangle$, where each Q_i^j denotes the corresponding value of a QoS attribute of the n-tuple $\langle Q^1, Q^2, ...Q^n\rangle$.*

Semantic inputs and output parameters are used to compose the functionality of multiple services by matching their inputs and outputs together. In order to measure the quality of the match, we need a matching mechanism that exploits the semantic I/O information of the services. In our work we use the different matching degrees that are introduced in [26]:

- **Exact($\equiv$)**: An input $i_{w_j} \in I_{w_j}$ of a service w_j matches an output $o_{w_i} \in O_{w_i}$ of a service w_i with a degree of exact match if both concepts are equivalent; denoted by, $i_{w_j} \equiv o_{w_i}$.
- **Plugin($\sqsubseteq$)**: An input $i_{w_j} \in I_{w_j}$ of a service w_j matches an output $o_{w_i} \in O_{w_i}$ of a service w_i with a degree of plugin if i_{w_j} is a direct sub-concept of o_{w_i}; denoted by, $i_{w_j} \sqsubseteq o_{w_i}$.
- **Subsume($\sqsupset$)**: An input $i_{w_j} \in I_{w_j}$ of a service w_j matches an output $o_{w_i} \in O_{w_i}$ of a service w_i with a degree of subsume if i_{w_j} is an indirect sub-concept of o_{w_i}; denoted by, $i_{w_j} \sqsupset o_{w_i}$.
- **Subsumed by($\sqsupseteq$)**: An input $i_{w_j} \in I_{w_j}$ of a service w_j matches an output $o_{w_i} \in O_{w_i}$ of a service w_i with a degree of subsumed by if i_{w_j} is a direct super-concept of o_{w_i}; denoted by, $i_{w_j} \sqsupseteq o_{w_i}$.
- **Fail($\perp$)**: When none of the previous matches are found, then both concepts are incompatible and the match has a degree of fail; denoted by $i_{w_j} \perp o_{w_i}$.

Definition 5 (Compatible Match) *Given two sets of concepts parameters O_{w_i} and I_{w_j} for services w_i and w_j respectively, a compatible match between O_{w_i} and I_{w_j} exists if $\forall\ i_{w_j} \in I_{w_j}$, $\exists\ o_{w_i} \in O_{w_i}$ ($i_{w_j} \equiv o_{w_i} \vee i_{w_j} \sqsubset o_{w_i} \vee i_{w_j} \sqsubseteq o_{w_i} \vee i_{w_j} \sqsupseteq o_{w_i}$), denoted as $O_{w_i} \otimes I_{w_j}$.*

Definition 6 (Composition Request) *Given an ontology Γ=$\langle C,R\rangle$, a composition request is defined as a quadruple $\langle R = I_R, O_R, G, W\rangle$, where:*

1. *$I_R \subseteq C$ represents the set of requested input parameters.*
2. *$O_R \subseteq C$ represents the set of requested output parameters.*
3. *G=$\langle$ $g_1, g_2, ...g_n$ $\rangle$ is an n-tuple of global QoS constraints for $\langle Q^1, Q^2, ...Q^n\rangle$.*
4. *$W = (W_{Q_{ind}}, W_{Q_{glob}}, W_{sem})$ such that $W_{Q_{ind}}$, $W_{Q_{glob}}, W_{sem} \in [0,1]$ is the set of user provided weights such that:*
   - *$W_{Q_{ind}} = \langle W_{Q^1}, W_{Q^2}, ...W_{Q^n}\rangle$ represents the user preferences regarding to the relevance of an individual value of a QoS attribute for a composition solution.*
   - *The $W_{Q_{glob}}, W_{sem}$ weights represent respectively the user preferences regarding to the relevance of:*
     - *The overall QoS value of a composition solution;*
     - *The value of the global score of semantic similarity of a composition solution.*

Definition 7 (QoS-aware Automatic Service Composition
Given a set of semantic web services W and a composition request $\langle R = I_R, O_R, G, W\rangle$, the composition problem considered in this paper consists of searching for an optimal composition solution that defines an invocation order over (workflow) a set of web services $\{w_1, w_2, ..., w_n\}$ and satisfies the conditions:

1. *$\{I_R \cup O_{w_1} \cup ... \cup O_{w_i}\} \otimes I_{w_{i+1}} (1 \leq i \leq n-1)$.*
2. *$\{O_{w_1} \cup O_{w_2} \cup ... \cup O_{w_n}\} \otimes O_R$.*
3. *The overall QoS of the composition solution is optimal.*
4. *The global QoS constraints G are satisfied.*

Our approach models QoS-aware automatic service composition as a multi-layered process which creates a planning-graph structure. The graph represents the space of all candidate composition solutions, for a composition request R, which are generated by computing the semantic similarity scores

between services on different layers. Our objective is not to select any composition solution, but the one that gives the global optimal QoS and respects the QoS constraints.

5 The Proposed Framework

In this section we present an integrated framework for QoS aware automated semantic web service composition with user's constraints. Figure 1 shows the overview of the framework with the different steps involved. The process is triggered by a composition request that specifies the user needs in terms of functional and non-functional requirements. In the composition planning-graph generation phase the functional requirements are used to build a planning-graph with all the relevant services and the semantic relations between their inputs and outputs. The planning-graph [44] is a directed graph of layers, each layer contains all services that can be executed with the outputs of the services of the previous layer. In order to retrieve the relevant services, we propose a semantic matching for calculating the semantic similarity scores between services on different layers. The planning graph contains all possible service compositions that satisfy the composition request. Then these candidate solutions are ranked according to user QoS requirements and preferences in the selection phase by using a Harmony Search (HS) Algorithm. Finally the optimal composition is returned.

In this section, we explain each step of the framework based on the problem formulation presented in the previous section.

5.1 User Requirements specification

In this phase, the user can specify her/his composition request in terms of functional and non-functional requirements:

1. Ontological concepts representing the semantic description of the inputs and outputs of the composed web services.
2. A set of weights that indicate the user preferences regarding the importance of QoS attributes and semantic quality established between the services involved in the composition. We consider two categories of weights: one category refers to the relevance of QoS attributes compared to the semantic quality, while the other category establishes the relevance of each individual QoS attribute.
3. A set of global QoS constraints on the composition solution.(i.e., composite service price not going beyond 100 $, composite service availability not being under 99.9 %, etc.).

5.2 Composition planning-graph generation

The goal of web service composition is to obtain an inter-connected set of web services satisfying the composition request. To achieve this goal, we adopt the planning-graph structure which is defined in [44]. The planning graph [15] provides a unique search space where connections between layers are expressed in a compact way. It has features of soundness, completeness, termination, and can be constructed in polynomial time. We have chosen the planning graph structure for the following reasons:

- It provides the whole set of composition solutions as response to the composition request.
- It considers multiple web services providing the same functionality (the functionalty being defined by the inputs/outputs).
- It takes into consideration QoS attributes.

The construction of the planning-graph takes into account the semantic matching between the output parameters of some services and the input parameters of other services. First we define how to compute the semantic similarity scores between services, then we give the basic concepts and principles of the planning-graph.

5.2.1 Computing the Score of Semantic Similarity between two services

For evaluating the semantic similarity between two services we first define a matching function to compute the degree of match between the concepts describing the input/output parameters of the services.

Definition 8 (Degree Of Match) *Given two concepts $o_{w_i} \in O_{w_i}$ and $i_{w_j} \in I_{w_j}$ for services w_i and w_j respectively, the degree of match (DoM) between o_{w_i} and i_{w_j} returns a value in [0, 1] such as:*

$$DoM(o_{w_i}, i_{w_j}) = \begin{cases} 1 & \text{if } i_{w_j} \equiv o_{w_i} \\ 0.9 & \text{if } i_{w_j} \sqsubseteq o_{w_i} \\ 0.8 & \text{if } i_{w_j} \sqsubset o_{w_i} \\ 0.7 & \text{if } i_{w_j} \sqsupseteq o_{w_i} \\ 0 & \text{Otherwise} \end{cases} \quad (1)$$

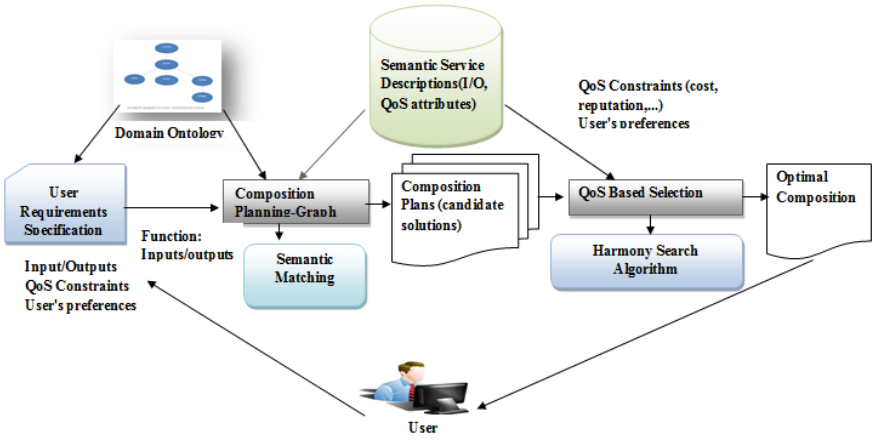

Fig. 1: Overview of the proposed framework

Definition 9 (Semantic Similarity Score) *The semantic similarity score (Sim_s) between the output concepts parameters O_{w_i} of service w_i and the input concepts parameters I_{w_j} of service w_j is computed as follows:*

$$Sim_s(O_{w_i}, I_{w_j}) = \frac{\sum_{k=1}^{|I_{w_j}|} \max_{l=1}^{|O_{w_i}|} DoM(o^l_{w_i}, i^k_{w_j})}{|I_{w_j}|} \quad (2)$$

Where $o^l_{w_i} \in O_{w_i}$ and $i^k_{w_j} \in I_{w_j}$.

5.2.2 Planning-Graph Structure

The Planning-graph is obtained by mapping the concepts of AI-planning graph [48] to semantic web service composition and also by enhancing the mapped concepts with the new abstractions of *service cluster* and *literal*[44].

Definition 10 (Service Cluster) *A service cluster SC groups services which have the same set of inputs parameters and are in one of the following matching relations: exact, plugin, subsume or subsumed by. Formally [44]:*

$\forall w_i, w_j \in SC, \forall i_{w_i} \in I_{w_i}, \forall i_{w_j} \in I_{w_j}$, *on a:* $i_{w_i} \equiv i_{w_j} \vee i_{w_i} \sqsubseteq i_{w_j} \vee i_{w_i} \sqsubset i_{w_j} \vee i_{w_i} \sqsupseteq i_{w_j}$.

Definition 11 (Literal) *Let be W a web service repository, a literal is a set of concepts that semantically describes a set of output and/or input parameters of a set a web services, formally [44]:*

$l = l_i \mid \forall w_i \in W, l_i \in \{I_{w_i} \cup O_{w_i}\}$.

The construction of planning-graph is an iterative process. At each step, a new layer consisting of a tuple $\langle A_i, L_i \rangle$ is added to the graph where A_i represents a set of *service clusters* and L_i is a *literal*. In the tuple $\langle A_0, L_0 \rangle$ from layer 0, A_0 is an empty set of services and L_0 contains the set of requested input parameters. For each layer $i > 0$, A_i is a set of clusters of services for which the input parameters are in L_{i-1}, and L_i is a *literal* obtained by adding the outputs of the services in A_i to the set L_{i-1}. The construction of the planning-graph ends either when the requested outputs are contained in the current *literal* or when the sets of *service clusters* and *literals* are the same on two consecutive layers. A composition solution encoded in the planning-graph consists of a set of services such that one service is selected from each cluster of each layer.

An example of a service composition planning graph for a composition request: $I_R = \{c1, c2\}$ and $O_R = \{c4, c6, c7, c8\}$ is shown in Figure 2. As we can see, this graph contains 5 service clusters (surounded by cicles) and 3 literals ($1^{st}, 3^{rd}$ and 5^{th}*columns*).

Definition 12 (Service Composition planning Graph) *A service composition planning-graph is defined as a set of tuples [44]:*

$SCPG = \{\langle A_i, L_i \rangle\}$, $\phi \leq i \leq n$ *where:*

- $i \in [1...n]$ *represents the index of layer* lay_i *in* $SCPG$.
- $A_i = \{sc^i_j \mid sc^i_j \in lay_i\}$ *represents the set of service clusters from the layer* lay_i *in* $SCPG$.
- $L_i = L_{i-1} \cup \{c \mid c \in \cup O_{w_k}$, *with* $w_k \in sc^i_j$ *and* $sc^i_j \in lay_i\}$, $i \geq 1$.

Note: (A_0, L_0) is a particular case where $A_0 = \phi$ and $L_0 = I_R$.

Definition 13 (A Composition Solution) *A composition solution sol for the SCPG contains a service for each cluster belonging to a layer, formally*

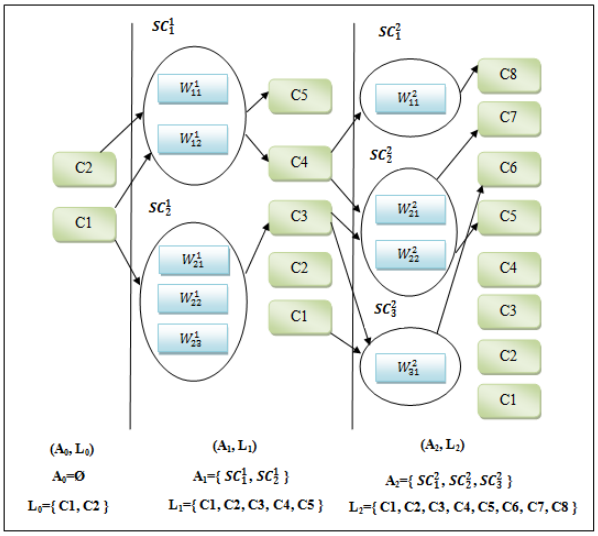

Fig. 2: Example of a service composition planning graph, $SCPG$=$\{\langle A_0, L_0\rangle, \langle A_1, L_1\rangle, \langle A_2, L_2\rangle\}$

[44]: $\forall\ lay_i \in SCPG \wedge \forall sc^i_j \in lay_i,\ \exists!\ w^i_{jk} \in sc^i_j \mid w^i_{jk} \in sol$ *where:*

- i *represents the index of layer* lay_i *in* $SCPG$*;*
- sc^i_j *is the cluster* j *on the layer* lay_i *in the* $SCPG$*;*
- w^i_{jk} *is the service* k *in cluster* j *on the layer* lay_i *in the* $SCPG$*.*

A composition solution for the planning-graph of Figure 2 consists of a set of subsets of services: sol= $< \{w^1_{11}, w^1_{21}\}\ ,\ \{w^2_{11}, w^2_{21}, w^2_{31}\} >$. In each solution subset, only one service of its cluster is considered. The services which contribute to the extension of the planning graph are provided by a discovery process that finds the appropriate web services in a repository of services, based on the semantic matching between the services' inputs and the set of literals corresponding to the previous layer. *The Matrix of Semantic Links (MSL)* stores the *Semantic Links* established between the services on different layers. One web service could be linked to several other web services.

Definition 14 (Semantic Similarity Link) *A Semantic similarity Link between two services* w_i, w_j *is defined as a tuple [44]:*

sl_{ij}=$(V, Sim_s(O_{w_i}, I_{w_j}))$ *where* V *is a set of pairs of output parameters of service* w_i *and input parameters of service* w_j *such as:*

V=$\{(o_{w_i}, i_{w_j}) \mid DoM(o_{w_i}, i_{w_j}) > 0, o_{w_i} \in O_{w_i}, i_{w_j} \in I_{w_j}\}$.

Definition 15 (Semantic Link Matrix) *A Semantic Link Matrix is defined in [44]:*

$$SLM = [slm_{ij}], i = 1, ...n; j = 1, ...m;\ such\ as:$$

$$slm_{ij} = \begin{cases} (\phi, 0), & if\ Sim_s(O_{w_i}, I_{w_j}) = 0 \\ sl_{ij}, & Otherwise \end{cases} \quad (3)$$

Where sl_{ij} *represents the semantic similarity link between the service on row* i *and the service on column* j*.*

5.3 QoS Based Selection

In this section, we describe our selection method based on Harmony Search (HS) algorithm for solving the QoS-aware optimal composition problem. According to the previous studies, HS algorithm proved to be very successful in a wide range of optimization problems, such as water distribution and games [12, 13], and showed better performance in comparison with other traditional optimization techniques. The search space of the selection method is the set of service composition solutions in a compact form of the previous phase.

A Principle of The Harmony Search Algorithm

Harmony Search (HS) is a meta-heuristic evolutionary optimization algorithm, recently developed by Geem et al [14]. It imitates musical process of searching for a perfect state of harmony. The Harmony Search algorithm has simple concepts and few parameters [24]. It imposes a few mathematical requirements and can easily be implemented. In Harmony Search algorithm, the *Harmony Memory (HM)* is a memory location where all current solution vectors (sets of decision variables) are stored. In each evolution of Harmony Search, if a solution vector with relatively good fitness is generated, it will be saved in HM and might be used in next generations. The HS steps are as follows [14]:

Step 1 : Initialize the problem and algorithm parameters.
Step 2 : Initialize the harmony memory.
Step 3 : Improvise a new harmony.
Step 4 : Update the harmony memory.
Step 5 : Repeat **Steps 3** and **4** until satisfying termination criterion.

In **Step 1**, the objective function with N decision variables and the set of possible values for each decision variable are defined. The HS algorithm parameters are also specified in this step. HS parameters include: *Harmony Memory Size(HMS)*, *Harmony Memory Considering Rate (HMCR)*, *Pitch*

Adjusting Rate (PAR), and *the termination criterion.*

In **Step 2**, the HM matrix is filled with as many randomly generated solution vectors as the HMS.

In **Step 3**, a new harmony vector like $X=(x_1, x_2, ..., x_n)$ is generated based on three rules: (1)*Harmony Memory Consideration(HMC)*,(2)*Pitch Adjustment(PA)* and (3) *Random Selection(RS)*. In HMC, the value of a decision variable for the new vector is chosen from any of the values for that decision variable in the specified HM. The HMCR, which varies between 0 and 1, is the rate of choosing one value from the historical values stored in the memory while (1- HMCR) is the rate of randomly selecting one value from the possible range of values. Every component obtained by HMC is examined to determine whether it should be pitch-adjusted. In PA, the current value of a decision variable is replaced with one of its neighboring values. The probability of pitch adjustment is specified by PAR parameter. During improvisation (generating a new harmony), HMCR and PAR are used to improve the solution vector globally and locally respectively. HMC, PA or RS is applied to each variable of the new harmony vector in turn.

In **Step 4**, if the new harmony vector is better than the worst harmony in the HM, evaluated in terms of the objective function value, the new harmony is included in the HM and the existing worst harmony is excluded from the HM.

In **Step 5**, if the stopping criterion (maximum number of improvisations) is satisfied, evolution is terminated. Otherwise, **Steps 3** and **4** are repeated. Finally, the best vector of memory in terms of objective function value is the (near)-optimal solution.

B Mapping HS algorithm to the Web Service Composition Problem

In our web service composition problem, HS is mapped as follows: (i) an harmony represents a candidate composition solution, (ii) a music pitch is represented by a service cluster and (iii) an aesthetic standard is represented by a fitness function that evaluates the composition solution. The HS algorithm uses the functional and non-functional (QoS attributes) properties of the web services in order to find the optimal composition solution. Such a composition needs to:

- Optimize a function that calculates the score of semantic similarity between services.
- Optimize the aggregated QoS of the composition solution.
- Satisfy the QoS global constraints.

B.1) **Fitness Evaluation**

The fitness function F is a multi-criteria function that optimizes:

1. The global score of semantic similarity of a composition solution.
2. The aggregated QoS of a composition solution.

Definition 16 (Global Score of Semantic Similarity) *The global score of semantic similarity (Sim_g) of a composition solution sol is computed as follows:*

$$Sim_g(sol) = \frac{\sum_{k=1}^{nlinks} Sim_s(O_{w^l_{qi}}, I_{w^p_{rj}})}{nlinks} \quad (4)$$

Where :

- $O_{w^l_{qi}}$ *represents the output concepts of the service i in cluster q from layer l ;*
- $I_{w^p_{rj}}$ *represents the input concepts of the service j in cluster r from layer p ;*
- w^l_{qi}, w^p_{rj} *are parts of the solution sol, $p = l+1$;*
- *nLinks is the total number of semantic similarity links (see Definition 14) in the composition solution sol.*

The overall QoS of a composition solution is calculated by applying QoS aggregation formulas to normalized QoS values of its component services. In this work we consider five QoS attributes: response time, cost, availability, fiability and reputation. The definitions of these properties and their computational methods are the same as [64].

Definition 17 (QoS Score of a composition solution) *The overall QoS of a composition solution is computed as follows:*

$$QoS(sol) = \sum_{k=1}^{n} w_{Q_k} * Q_k \quad (5)$$

where Q_k is the aggregated QoS of a solution for QoS attribute k, w_{Q_k} is the weight of each attribute (its importance degree defined by user preferences), n is the number of QoS attributes and $\sum_{k=1}^{n} w_{Q_k} = 1$.

By combining the two scores defined by the formulas 4 and 5, we obtain the fitness function enabling the evaluation of a composition solution which is defined as:

$$F(sol) = W_{Q_{glob}} * QoS(sol) + W_{sem} * Sim_g(sol) \quad (6)$$

Where the weights $W_{Q_{glob}}, W_{sem} \in [0,1]$ represent the user preferences regarding to:

- The overall QoS value of a composition solution;
- The value of the global score of semantic similarity of a composition solution.

In addition, the fitness function $F(sol)$ must drive the evolution towards global QoS constraints satisfaction. To this end compositions that do not meet the constraints are penalized. Several penalty functions are proposed in the literature (static, dynamic, adaptive functions, etc.), we have chosen a static function which is adopted by [30], because the two others functions does not give a significant improvement (they only increase the execution time)(see formula 7).

$$F'(sol) = F(sol) - (\sum_{k=1}^{n}(\frac{\Delta Q}{g_k^{max} - g_k^{min}}))^2 \quad (7)$$

Where g_k^{max} and g_k^{min} are respectively the maximum and the minimum value of the k^{th} QoS constraint. n represents the number of QoS constraints and ΔQ is defined by formula 8.

$$\Delta Q = \begin{cases} Q_k - g_k^{max} & \text{if } Q_k > g_k^{max} \\ 0 & \text{if } g_k^{min} \leq Q_k \leq g_k^{max} \\ g_k^{min} - Q_k & \text{if } Q_k < g_k^{min} \end{cases} \quad (8)$$

B.2) **The HS-based Selection Algorithm**

Our Harmony Search-based selection algorithm (see Algorithm 1) determines the optimal solution by considering the set of solutions encoded in the composition planning-graph generation phase. The inputs of the selection algorithm are: i) the $SCPG$ for a composition request, ii) the fitness function F'(see formula 7), iii) the adjustable parameters HMS, $HMCR$, PAR, IV) $pitch\_num$ denotes the set of service clusters, V) $pitch\_bounds$ represents the set of services in a given cluster, and VI) $iteration\_max$ is the maximum number of iterations. The algorithm proceeds as follows:

Step 1. Initialize the Harmony Memory HM with random solutions generated by the $SCPG$ *(Line 1)*. The HM is encoded as a matrix where each column represents a service cluster $pitch\_j \in pitch\_num$ and each row is a harmony (solution). The solutions initialized in this stage are improved in an iterative stage which is executed until the maximum number of iterations is reached *(Lines 3-23)*.

Step 2. Evaluate the harmonies in HM according to F' *(Line 2)*.

Algorithm 1: Harmony Search Algorithm

```
   Input : SCPG, F', HMS, HMCR, PAR,
           pitch_num, pitch_bounds, iteration_max
   Output : Best_Harmony(optimal solution)
1  HM =
     Initialize_Harmony_Memory(SCPG, HMS,
     pitch_num, pitch_bounds);
2  Evaluate_Harmony_set(HM, F');
3  for i ← iteration_max do
4      Harmony ← Nil;
5      foreach pitch_j ∈ pitch_num do
6          if Rand(0,1) ≤ HMCR then
7              RandomPitch =
                 Select_Pitch_Random_Harmony(HM, pitch_j);
8              if Rand() ∈ PAR then
9                  Pitch =
                     Pitch_Adjustment(RandomPitch)
10             else
11                 Pitch = RandomPitch
12             end
13         else
14             Pitch =
                 Get_Random_Pitch(pitch_bounds(pitch_j))
15         end
16         Update(Harmony, Pitch, pitch_j)
17     end
18     Harmony =
         Evaluate_Harmony(Harmony, F');
19     if F'(Harmony) ≤
         F'(Worst_Harmony(HM)) then
20         HM = HM − (Worst_Harmony(HM));
21         HM = HM ∪ (Harmony)
22     end
23 end
24 Best_Harmony = Select_Best(HM)
25 return Best_Harmony
```

Step 3. Improvise (create) a new harmony as follows *(Line 3-23)* :

(i) Each component $pitch\_j$ of the solution is initialized with a web service w_i taken from a random harmony h such that $h \in HM$ with a probability $HMCR$ *(Line 6-7)*.

(ii) With a probability PAR, the previous web service component is replaced with a neighbor web service w_i' of the same service cluster such that $F'(w_1, ...w_i'...w_n)$ is maximized *(Line 8-12)*.

(iii) If the statements i) and ii) are not executed, then the component $pitch\_j$ is initialized with a random web service w_j, such that $w_j \in pitch\_bounds(pitch\_j)$ *(Line 14)*.

(iv) We update the $pitch\_j^{th}$ component of the harmony by using the value of $Pitch$ *(Line 16)*.

Step 4. Evaluate the constructed solution denoted $Harmony$ *(Line 18)*.

Step 5. The worst harmony of HM is replaced by $Harmony$ if this later is more effective in terms of fitness value *(Lines 19-22)*.

Step 6. Return the best solution of the HM ($Best\_Harmony$) *(Lines 24-25)*.

6 Experimental Evaluation

This section presents an experimental evaluation of our approach, focusing on the following research questions:

- **RQ1**: What is the performance of our optimization method in terms of optimality and execution time?
- **RQ2**: What are the effects of QoS constraints on the retrieved composition solutions?
- **RQ3**: How much the variants of the HS algorithm IHS and GHS improve the selection process?

In order to perform a standard evaluation, we selected the Web Service Challenge 2009 dataset that focuses on the semantic composition of web services with QoS. WS-Challenge dataset provides a set of standard testing tools and data sets. This dataset is well suited to the semantic web service composition problem since it involves a large number of web services (˜4000 instances), a large number of concepts (˜40 000 concepts), and different solution depth (3 up to 16).

This dataset contains three different files:

- A WDSL file that contains a set of semantically annotated web services along with annotations of their input and output parameters. Every web service has an arbitrary number of parameters and the numbers of services vary from 500 to 4000.
- An OWL file containing the ontology relating the different concepts. The 2009 WSC features ontologies with between 5000 and 40 000 concepts.
- A WSLA file with the QoS attributes of the services. In the dataset, two only QoS attributes are considered: the response time and the throughput. In order to take into consideration other non-functional attributes, we have enriched this file with four metrics: the cost, the availability, the fiability and the reputation as suggested in [62]: Cost: [0, 30]$, Availability : [0.7, 1]%, Fiability: [0.5, 1]% , Reputation: [0, 5]%, and Response time: [0, 300] ms.

Our framework for QoS-aware automated semantic web service composition is implemented in Java. We have used *JDOM*[2] for handling the web services and *JENA*[3] for semantic reasoning. The experiments were conducted on a 1.8 GHZ Intel i5 core CPU and 6 GB of RAM, running under Windows 7. This experimentation can be reproduced using the programs and the dataset which are provided in the archive which is downloadable using the following link:

`http://www.lirmm.fr/~tibermacin/experiment.zip`

6.1 Experiment for RQ1

In order to answer the first research question, we present our methodology for evaluating the selection algorithm as well as an analysis of the experimental results obtained in different settings.

A Evaluation Methodology

The convergence of an optimization algorithm towards the optimal solution is influenced by a set of adjustable parameters specific to the algorithm. We consider that a proper methodology for evaluating an optimization algorithm should consist of two steps: one step for establishing the optimal values of the adjustable parameters, and another step for evaluating the algorithm using the optimal configuration of the adjustable parameters. To establish the optimal values of the adjustable parameters the following two steps need to be addressed [49].

In the first step, an exhaustive search in the composition model should be performed to identify the score of the optimal composition solution. This score is further used to identify the most appropriate configuration of the adjustable parameters which ensures that the optimal or a near-optimal composition solution is obtained without processing the entire search space.

In the second step, the initial configuration of the adjustable parameters is fine-tuned iteratively to identify their optimal values. During these experiments, *the execution time* and *the deviation* of the optimal solution returned by the algorithm are compared with *the execution time* and *the optimal solution* returned by the exhaustive search.

[2] http://www.jdom.org

[3] https://jena.apache.org

B Experimental results

We have tested our approach on three scenarios with different complexities. In Table 1, we specified for each scenario: (i) the scenario id, (ii) the graph configuration in terms of the number of layers (given by the number of subsets), the number of clusters from each layer (given by the cardinality of each subset) and the number of services per cluster (given by the value of each element in a subset), (iii) the search space complexity in terms of total number of possible solutions encoded in the planning-graph structure (the number has been obtained by counting the number of solutions generated in an exhaustive search procedure), (iv) the global optimal fitness value identified by performing an exhaustive search, and (v) the execution time in which the optimal solution has been found when performing the exhaustive search.

Table 1: The configurations of the graph for each Scenario

Scenario Id	**Graph Configuration**	**Search Space Complexity**	**Global Optimal Fitness**	**Execution Time (hour:min:s)**
A	Layer1:{3 5 6} Layer2:{6 4 3} Layer3:{4 6 5}	777600	0.891	00:02:10
B	Layer1:{2 5 4 6} Layer2:{6 4 6 4} Layer3:{4 5}	2764800	0.950	00:07:47
C	Layer1:{6 6 3 3} Layer2:{5 6 5 3} Layer3:{6 5 6}	26244000	0.920	01:13:18

The adjustable parameters of the HS Algorithm are the following: *HMS*, *HMCR*, and *PAR*. We have varied the values of these parameters by considering the same ranges as in other works from the literature [24]: $HMS \in \{200, 400\}$, $HMCR \in \{0.70, 0.95\}$, and $PAR \in \{0.01, 0.30\}$. Tables 2, 3, and 4 illustrate a fragment of the best experimental results (average optimal fitness (*Fitness*), average execution time (*Time*), average deviation (*Deviation*)) obtained while varying the values of the adjustable parameters for scenarios A,B, and C (each table row represents an average value of the results obtained while running the algorithm for 50 times on the same configuration of adjustable parameters). The rows highlighted in bold indicate the optimal configuration of adjustable parameters (best compromise between fitness and time).
By analyzing the experimental results, we conclude that the HS selection algorithm presents a good performance in terms of optimality (the average optimal fitness exceeds 0.8 in the case of scenario B). Besides, it returns the optimal or a near-optimal solution on average in 7 seconds (in the case of the three scenarios) which is an acceptable execution time. This confirms the ability of HS algorithm in exploring the solution space within a reasonable time.

Table 2: Fragment of the best experimental results for Scenario A

#	HMS	HMCR	PAR	Fitness	Time	Deviation
1	250	0.70	0.1	0.720	5000	0.171
2	250	0.70	0.2	0.725	5500	0.166
3	250	0.70	0.3	0.730	5500	0.161
4	250	0.80	0.1	0.720	5000	0.171
5	250	0.80	0.2	0.726	5500	0.165
6	250	0.80	0.3	0.735	5500	0.156
7	250	0.95	0.1	0.720	5800	0.171
8	250	0.95	0.2	0.735	6100	0.156
9	250	0.95	0.3	0.730	6200	0.161
10	300	0.70	0.1	0.729	5500	0.162
11	300	0.70	0.2	0.740	5500	0.151
12	300	0.70	0.3	0.741	6000	0.15
13	300	0.80	0.1	0.729	4000	0.162
14	300	0.80	0.2	0.730	4100	0.161
15	300	0.80	0.3	0.732	5000	0.159
16	300	0.95	0.1	0.733	4000	0.158
17	**300**	**0.95**	**0.2**	**0.745**	**5000**	**0.146**
18	300	0.95	0.3	0.744	5910	0.147
19	400	0.70	0.1	0.730	5000	0.161
20	400	0.70	0.2	0.731	6000	0.160
21	400	0.70	0.3	0.732	6100	0.159
22	400	0.80	0.1	0.720	6500	0.171
23	400	0.80	0.2	0.725	6800	0.166
24	400	0.80	0.3	0.725	6800	0.166
25	400	0.95	0.1	0.700	7000	0.191
26	400	0.95	0.2	0.710	7000	0.181
27	400	0.95	0.3	0.720	7100	0.171

Table 3: Fragment of the best experimental results for Scenario B

#	HMS	HMCR	PAR	Fitness	Time	Deviation
1	250	0.70	0.1	0.8	4500	0.15
2	250	0.70	0.2	0.8	4510	0.15
3	250	0.70	0.3	0.810	4520	0.14
4	250	0.80	0.1	0.810	5000	0.14
5	250	0.80	0.2	0.820	5200	0.13
6	250	0.80	0.3	0.830	5300	0.12
7	250	0.95	0.1	0.80	5100	0.15
8	250	0.95	0.2	0.80	5000	0.15
9	250	0.95	0.3	0.810	5100	0.14
10	300	0.70	0.1	0.810	5000	0.14
11	300	0.70	0.2	0.830	5500	0.12
12	300	0.70	0.3	0.820	5200	0.13
13	300	0.80	0.1	0.828	4900	0.122
14	300	0.80	0.2	0.830	5000	0.12
15	300	0.80	0.3	0.835	5500	0.115
16	300	0.95	0.1	0.82	6000	0.13
17	300	0.95	0.2	0.84	6200	0.11
18	**300**	**0.95**	**0.3**	**0.85**	**6300**	**0.1**
19	400	0.70	0.1	0.821	6300	0.129
20	400	0.70	0.2	0.828	6400	0.122
21	400	0.70	0.3	0.810	6500	0.14
22	400	0.80	0.1	0.829	6800	0.121
23	400	0.80	0.2	0.83	6900	0.12
24	400	0.80	0.3	0.831	7000	0.119
25	400	0.95	0.1	0.810	7100	0.14
26	400	0.95	0.2	0.820	7200	0.13
27	400	0.95	0.3	0.825	7200	0.125

Table 4: Fragment of the best experimental results for Scenario C

#	HMS	HMCR	PAR	Fitness	Time	Deviation
1	250	0.70	0.1	0.77	5000	0.15
2	250	0.70	0.2	0.770	5000	0.15
3	250	0.70	0.3	0.772	5100	0.148
4	250	0.8	0.1	0.776	5200	0.144
5	250	0.8	0.2	0.78	5300	0.14
6	250	0.8	0.3	0.782	5310	0.138
7	250	0.95	0.1	0.79	5300	0.13
8	250	0.95	0.2	0.79	5300	0.13
9	250	0.95	0.3	0.78	5200	0.14
10	300	0.70	0.1	0.771	5000	0.149
11	300	0.70	0.2	0.772	5100	0.148
12	300	0.70	0.3	0.782	5200	0.138
13	300	0.8	0.1	0.77	5000	0.15
14	300	0.8	0.2	0.783	6000	0.137
15	300	0.8	0.3	0.773	5900	0.147
16	300	0.95	0.1	0.79	6700	0.13
17	300	0.95	0.2	0.782	6400	0.138
18	300	0.95	0.3	0.783	6500	0.137
19	400	0.70	0.1	0.787	6600	0.133
20	300	0.70	0.2	0.789	6700	0.131
21	300	0.70	0.3	0.789	6700	0.131
22	400	0.8	0.1	0.790	6800	0.13
23	400	0.8	0.2	0.792	6800	0.128
24	400	0.8	0.3	0.793	6830	0.127
25	400	0.95	0.1	0.789	6800	0.131
26	**400**	**0.95**	**0.2**	**0.800**	**7000**	**0.12**
27	400	0.95	0.3	0.790	7000	0.13

In order to evaluate the performance of the proposed selection algorithm we have used the *fitness graph* measure [11], which gives information about the algorithm performance across several runs of the algorithm using the same configuration of adjustable parameters and different initial populations (randomly generated). For an algorithm to be efficient it is desirable that the average fitness of the individuals in the population as well as the best fitness values vary little across several runs of the algorithm. In Figures 3, 4, and 5 we plot the fitness graphs for the three considered scenarios by using the optimal configuration of the adjustable parameters.

By analyzing Figures 3, 4, and 5,we can observe that the number of clusters as well as the size of clusters will largely influence the convergence duration. For instance, Scenario A which contains 09 clusters will start the convergence after 60 iterations. However, Scenario B which contains 10 clusters, will start the convergence after 70 iterations, lastly Scenario C which contains 11 clusters will start the convergence after 75 iterations. It can be also noticed that the variation of the average and best fitness values is little when running the algorithm multiple times using the same configuration for each scenario. This proves the efficiency of our selection method for identifying the optimal composition solution in QoS-aware semantic web service composition problem.

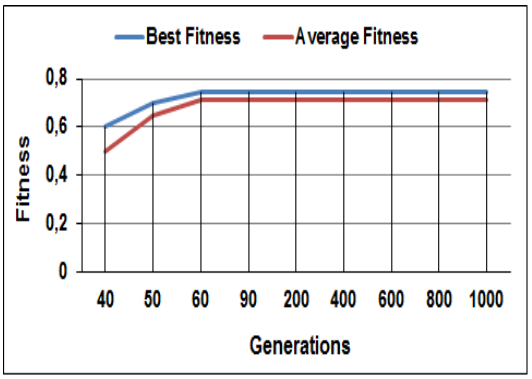

Fig. 3: Fitness vs. Generations (Scenario A)

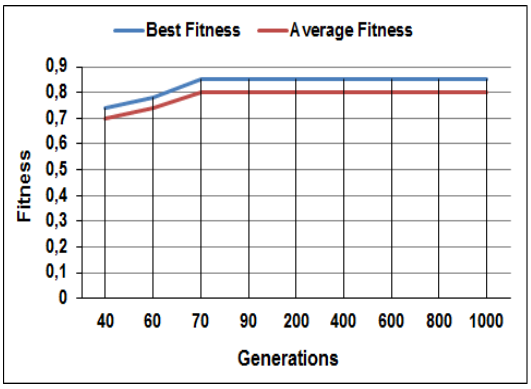

Fig. 4: Fitness vs. Generations (Scenario B)

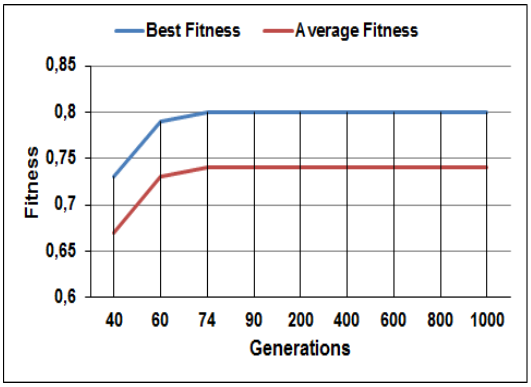

Fig. 5: Fitness vs. Generations (Scenario C)

6.2 Experiment for RQ2

To evaluate the effects of the global QoS constraints on the quality of composition solutions in our approach, we conducted two experiments. In the first experiment, we have defined the *fitness graphs* in the first place without taking into consideration the constraints. Then, we incorporate them in the selection process. Figures 6, 7, and 8 illustrate the results obtained in the three considered scenarios by using the same optimal configuration as in the previous Section. As we can see, the performance of HS algorithm in terms of optimality without global constraints is better than its performance with constraints (the best fitness value exceeds 0.9 in scenario B). In the second experiment, we have evaluated the performance of our selection method against the number of QoS constraints. For this purpose, we varied the number of constraints from 1 to 5 (recall that the number of QoS attributes considered in our work is 5). The results of this experiment are shown in Figure 9. In the three Scenarios, we observe that the second QoS constraint adds a little overhead with respect to the first QoS constraint cost. However, the overhead of the third, the forth and the fifth QoS constraints will gradually raise with respect to the previous computational cost. In summary, we conclude that the execution time to find the optimal composition solution increases linearly with the number of constraints in the three scenarios.

The results obtained in the two experiments demonstrated that the problem of finding the composition that optimizes the overall QoS while satisfying multiple QoS constraints is much more difficult than just optimizing the QoS fitness function, in fact the problem is known to be NP-hard. Any exact solution to this problem is expected to have an exponential computational complexity. This is the main reason why we have proposed an optimization method based on metaheuristic algorithms.

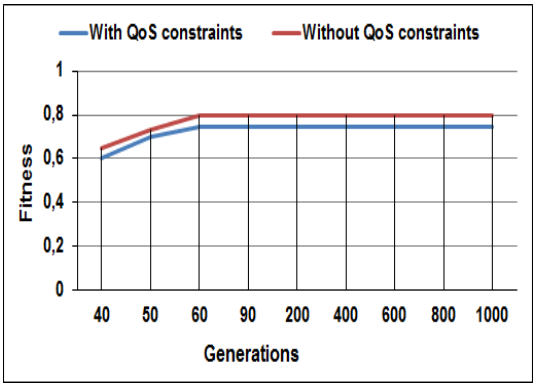

Fig. 6: Fitness vs. Generations (Scenario A)

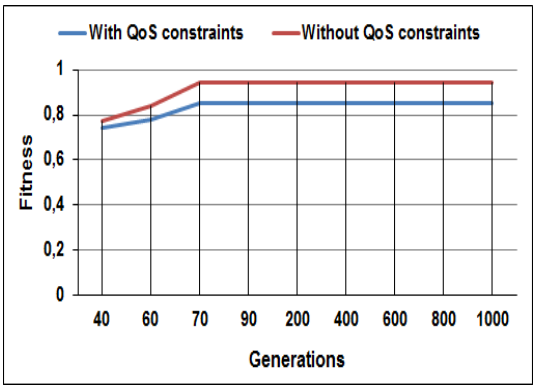

Fig. 7: Fitness vs. Generations (Scenario B)

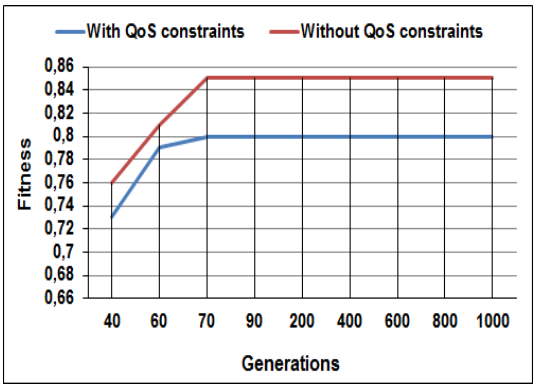

Fig. 8: Fitness vs. Generations (Scenario C)

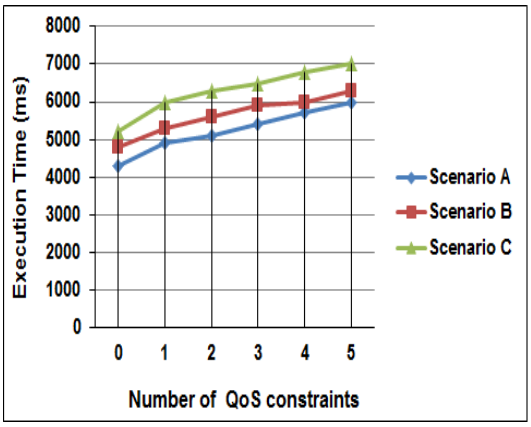

Fig. 9: Execution Time vs. Number of QoS constraints

Table 5: Comparative Analysis (Scenario A)

Algorithm	Fitness	Time	Deviation
HS (Harmony Search)	0.745	6000	0.146
IHS (Improved Harmony Search)	0.80	5900	0.091
GHS (Global Best Harmony Search)	0.87	7500	0.021

Table 6: Comparative Analysis (Scenario B)

Algorithm	Fitness	Time	Deviation
HS (Harmony Search)	0.850	6300	0.10
IHS (Improved Harmony Search)	0.90	6000	0.05
GHS (Global Best Harmony Search)	0.947	8000	0.03

Table 7: Comparative Analysis (Scenario C)

Algorithm	Fitness	Time	Deviation
HS (Harmony Search)	0.800	7000	0.12
IHS (Improved Harmony Search)	0.89	6500	0.03
GHS (Global Best Harmony Search)	0.919	9000	0.01

6.3 Experiment for RQ3

In order to answer the third research question, we have compared the HS algorithm with its recently developed variants IHS (Improved HS) algorithm [36] and GHS (Global Best HS) [40] algorithm. The IHS algorithm is a new harmony algorithm proposed in 2007 by Mahdavi et al. [36], it applies a method for generating new solution vectors based on the dynamic adjustment of the *PAR* and *HMCR* parameters (instead of considering their fixed values as in the original HS algorithm), thus improving accuracy and convergence speed. In this deviation, only the step that creates a new harmony is adjusted. *PAR* and *HMCR* change dynamically with the number of generations. Besides, the GHS [40] algorithm was inspired by PSO (Particule Swarm Optimization) concepts [23], it has exactly the same steps as the IHS algorithm with the exception that in the new harmony improvisation step the new harmony vector imitates the best harmony (in terms of fitness value) in the Harmony Memory (*HM*).

The three selection algorithms have been comparatively evaluated according to the following criteria: average optimal fitness (*Fitness*), average execution time (*Time*), and average deviation (*Deviation*). For IHS and GHS algorithms we have performed the same procedure in Section 6.1 to establish the optimal values of adjustable parameters in the three considered scenarios. The experimental results obtained by the algorithm versions are illustrated respectively, in Table 5 and Figure 10 for scenario A, in Table 6 and Figure 11 for scenario B, and finally in Table 7 and Figure 12 scenario C. Each row in Tables 5, 6, 7 represents an average obtained for 50 runs of an algorithm.

Concerning the Scenario A, we notice that the three variants of HS algorithm have almost the same duration of convergence, moreover we notice that GHS algorithm is more effective than IHS algorithm and IHS algorithm is more effective than HS algorithm. In contrast, in Scenario B, we observe different durations of convergence. First GHS algorithm has the largest duration of convergence

(90 generations) but it ensures the largest fitness (0.947), on the other hand IHS has the smallest duration of convergence (60 generations), however it gives the second best fitness which is equal to 0.9. Finally HS presents a medium convergence time (70 generations), but it gives the smallest fitness which is equal to 0.85. The Scenario C confirms the same findings of scenario B, i.e, the IHS algorithm has the smallest duration of convergence and GHS has the largest duration of convergence but it ensures the highest fitness. The worst behavior is performed by the HS algorithm. We also notice that the highest fitness obtained in this Scenario (0.91) is lesser than the highest fitness obtained in Scenario B (0.947).

In summary, we can say that both variants IHS and GHS have brought improvements in terms of fitness and execution time compared to the original version of the algorithm. Considering the three scenarios, we can conclude that GHS algorithm is more efficient than the other two HS algorithms in terms of optimality, but in terms of time it provides slightly higher values, which are acceptable (9 seconds in the case of Scenario C). This can be explained by the fact that GHS takes a little more time for searching the best harmony in the improvisation process. In contrast, execution time values of IHS are better than those of HS and GHS (less than 7 seconds in the three scenarios), which implies that IHS provides a better convergence than the HS and GHS algorithms. This is due to the mechanism for dynamically adjusting algorithm parameters which is benefical to escape from local optimum solutions and improves the convergence speed.

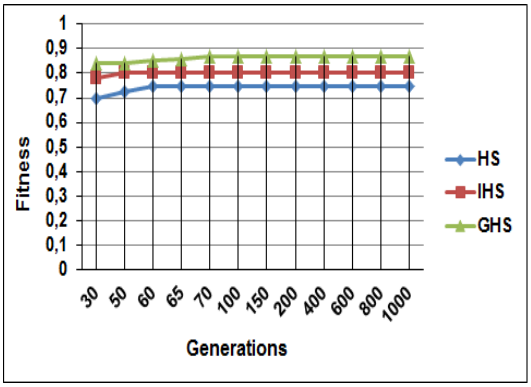

Fig. 10: Fitness vs. Generations (Scenario A)

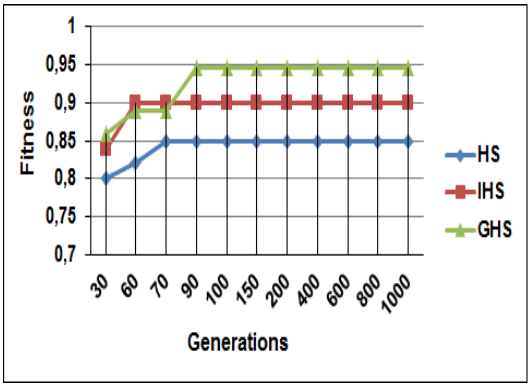

Fig. 11: Fitness vs. Generations (Scenario B)

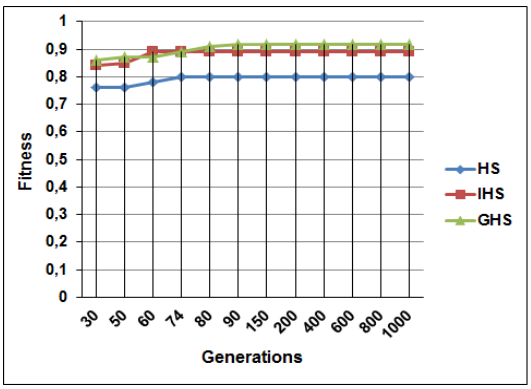

Fig. 12: Fitness vs. Generations (Scenario C)

7 Conclusion

This paper proposes an approach based on AI-planning and optimization techniques to address the QoS-aware automated web service composition problem, which has emerged as a significant issue in the web services research community. The novelty of our approach is that it addresses three composition dimensions simultaneously: the functional correctness of solutions, the optimization of solutions according to non-functional properties, and the fulfilment of user's QoS constraints. The first dimension was handled by the use of a mechanism for computing the semantic matching scores between the services involved in the composition solution. The last two dimensions were handled by the use of three HS algorithms, wherein a fitness function was proposed to rank solutions according to their overal QoS and global constraints. The experimental results showed that the proposed approach is efficient and effective to identify the op-

timal or near optimal composition in diverse scenarios.

Our ongoing research effort aims at introducing context-awareness to our approach and at evaluating the efficiency and performance of our solution to context-awre service composition. We also intend to extend the matching mechanism to handle web service preconditions and effects in order to make the search results more accurate. Furthermore, we plan to use recent optimization algorithms like Invasive Weed optimization (IWO), Biogeography Based Optimization (BBO) and Bat-inspired Algorithms (BA) to improve the selection process. Finally, we plan to experiment our composition approach on more complex and real scenarios.

References

1. Akkiraju R, Srivastava B, Ivan A, Goodwin R, Syeda-Mahmood TF (2006) SEMAPLAN: combining planning with semantic matching to achieve web service composition. In: IEEE International Conference on Web Services (ICWS), IEEE, pp 37–44
2. Alonso G, Casati F, Kuno HA, Machiraju V (2004) Web Services - Concepts, Architectures and Applications. Data-Centric Systems and Applications, Springer
3. Alrifai M, Risse T (2009) Combining global optimization with local selection for efficient qos-aware service composition. In: International conference on World Wide Web (WWW), ACM, pp 881–890
4. Azmeh Z, Driss M, Hamoui F, Huchard M, Moha N, Tibermacine C (2011) Selection of composable web services driven by user requirements. In: IEEE International Conference on Web Services (ICWS), IEEE, pp 395–402
5. Baccar S, Rouached M, Abid M (2013) A user requirements oriented semantic web services composition framework. In: IEEE Ninth World Congress on Services (SERVICES), IEEE, pp 333–340
6. Boukadi K, Grati R, Ben-Abdallah H (2016) Toward the automation of a qos-driven sla establishment in the cloud. Service Oriented Computing and Applications 10(3):279–302
7. Canfora G, Di Penta M, Esposito R, Villani ML (2005) An approach for QoS-aware service composition based on genetic algorithms. In: 7th annual Genetic and Evolutionary Computation COnference (GECCO), ACM, pp 1069–1075
8. Carman M, Serafini L, Traverso P (2003) Web service composition as planning. In: ICAPS 2003 Workshop on Planning for Web Services
9. Deng S, Wu B, Yin J, Wu Z (2013) Efficient planning for top-k web service composition. Knowledge and information systems 36(3):579–605
10. Esfahani PM, Habibi J, Varaee T (2012) Application of social harmony search algorithm on composite web service selection based on quality attributes. In: Sixth International Conference on Genetic and Evolutionary Computing (ICGEC), IEEE, pp 526–529
11. Floreano D, Mattiussi C (2008) Bio-inspired artificial intelligence: theories, methods, and technologies. MIT press
12. Geem ZW (2000) Optimal design of water distribution networks using harmony search. PhD thesis, Korea University
13. Geem ZW (2007) Harmony search algorithm for solving sudoku. In: Knowledge-Based Intelligent Information and Engineering Systems, Springer, pp 371–378
14. Geem ZW, Kim JH, Loganathan G (2001) A new heuristic optimization algorithm: harmony search. Simulation 76(2):60–68
15. Ghallab M, Nau D, Traverso P (2004) Automated planning: theory & practice. Elsevier
16. Gu Z, Li J, Xu B (2008) Automatic service composition based on enhanced service dependency graph. In: IEEE International Conference on Web Services (ICWS), IEEE, pp 246–253
17. Hatzi O, Vrakas D, Nikolaidou M, Bassiliades N, Anagnostopoulos D, Vlahavas I (2012) An integrated approach to automated semantic web service composition through planning. Services Computing, IEEE Transactions on 5(3):319–332
18. Hwang SY, Lim EP, Lee CH, Chen CH (2008) Dynamic web service selection for reliable web service composition. IEEE Transactions on Services Computing 1(2):104–116
19. Jaeger MC, Rojec-Goldmann G, Muhl G (2004) QoS Aggregation for Web Service Composition using Workflow Patterns. In: 17th IEEE International Enterprise Distributed Object Computing Conference (EDOC), IEEE, pp 149–159
20. Jafarpour N, Khayyambashi MR (2010) QoS-aware selection of web service compositions us-

ing harmony search algorithm. Journal of Digital Information Management 8(3):160–166
21. Jiang W, Zhang C, Huang Z, Chen M, Hu S, Liu Z (2010) Qsynth: A tool for qos-aware automatic service composition. In: IEEE International Conference on Web Services (ICWS), IEEE, pp 42–49
22. Kaveh A, Ahangaran" M (2012) Discrete cost optimization of composite floor system using social harmony search model. Applied Soft Computing 12(1):372 – 381
23. Kennedy J (2011) Particle swarm optimization. In: Encyclopedia of machine learning, Springer, pp 760–766
24. Kim JH, Geem ZW (2015) Harmony Search Algorithm: Proceedings of the 2nd International Conference on Harmony Search Algorithm (ICHSA2015), vol 382. Springer
25. Kim JH, Geem ZW, Kim ES (2001) Parameter estimation of the nonlinear muskingum model using harmony search1. JAWRA Journal of the American Water Resources Association 37(5):1131–1138
26. Klusch M, Kapahnke P (2008) Semantic web service selection with sawsdl-mx. In: 7th International Semantic Web Conference, Citeseer, p 3
27. Klusch M, Gerber A, Schmidt M (2005) Semantic web service composition planning with owls-xplan. In: AAAI Fall Symposium on Semantic Web and Agents, AAAI Press
28. Ko JM, Kim CO, Kwon IH (2008) Quality-of-service oriented web service composition algorithm and planning architecture. Journal of Systems and Software 81(11):2079–2090
29. Kona S, Bansal A, Blake MB, Gupta G (2008) Generalized semantics-based service composition. In: IEEE International Conference on Web Services (ICWS), IEEE, pp 219–227
30. Lécué F (2009) Optimizing qos-aware semantic web service composition. Springer
31. Lécué F, Léger A (2006) A formal model for semantic web service composition. In: The Semantic Web-ISWC 2006, Springer, pp 385–398
32. Lécué F, Salibi S, Bron P, Moreau A (2008) Semantic and syntactic data flow in web service composition. In: IEEE International Conference onWeb Services (ICWS), IEEE, pp 211–218
33. Lécué F, Silva E, Pires LF (2008) A framework for dynamic web services composition. In: Emerging Web Services Technology, Volume II, Springer, pp 59–75
34. Levesque HJ, Reiter R, Lesperance Y, Lin F, Scherl RB (1997) Golog: A logic programming language for dynamic domains. The Journal of Logic Programming 31(1):59–83
35. Li W, Dai X, Jiang H (2010) web services composition based on weighted planning graph. In: First International Conference on Networking and Distributed Computing (ICNDC), IEEE, pp 89–93
36. Mahdavi M, Fesanghary M, Damangir E (2007) An improved harmony search algorithm for solving optimization problems. Applied mathematics and computation 188(2):1567–1579
37. McDermott DV (2002) Estimated-regression planning for interactions with web services. In: AIPS, vol 2, pp 204–211
38. McIlraith S, Son TC (2002) Adapting golog for composition of semantic web services. KR 2:482–493
39. Menascé DA (2004) Composing Web Services: A QoS View. IEEE Internet Computing 8(6):88–90
40. Omran MG, Mahdavi M (2008) Global-best harmony search. Applied Mathematics and Computation 198(2):643–656
41. Papadimitriou CH, Steiglitz K (1982) Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall, Inc., Upper Saddle River, NJ, USA
42. Peer J (2005) Web service composition as ai planning-a survey, university of st. Gallen, Switzerland
43. Ponnekanti SR, Fox A (2002) Sword: A developer toolkit for web service composition. In: Eleventh International World Wide Web Conference (WWW), vol 45
44. Pop CB, Chifu VR, Salomie I, Dinsoreanu M (2009) Immune-inspired method for selecting the optimal solution in web service composition. In: Resource Discovery, Springer, pp 1–17
45. Ran S (2003) A model for web services discovery with qos. ACM Sigecom exchanges 4(1):1–10
46. Rodriguez-Mier P, Mucientes M, Lama M (2015) Hybrid optimization algorithm for large-scale qos-aware service composition. IEEE Transactions on Services Computing
47. Rodriguez Mier P, Pedrinaci C, Lama M, Mucientes M (2016) An integrated semantic web service discovery and composition framework. IEEE Transactions on Services Computing 9

48. Russell S, Norvig P, Intelligence A (1995) A modern approach. Artificial Intelligence Prentice-Hall, Egnlewood Cliffs 25:27
49. Salomie I, Chifu VR, Pop CB (2014) Hybridization of cuckoo search and firefly algorithms for selecting the optimal solution in semantic web service composition. In: Cuckoo Search and Firefly Algorithm, Springer, pp 217–243
50. Shiaa MM, Fladmark JO, Thiell B (2008) An incremental graph-based approach to automatic service composition. In: IEEE International Conference on Services Computing (SCC), IEEE, vol 1, pp 397–404
51. Sirin E, Parsia B (2004) Planning for semantic web services. In: Semantic Web Services Workshop at 3rd International Semantic Web Conference, pp 33–40
52. Sirin E, Parsia B, Wu D, Hendler J, Nau D (2004) HTN Planning for Web Service Composition Using SHOP2. Web Semant 1(4):377–396
53. Sirin E, Parsia B, Wu D, Hendler J, Nau D (2004) Htn planning for web service composition using shop2. Web Semantics: Science, Services and Agents on the World Wide Web 1(4):377–396
54. Tangpattanakul P, Meesomboon A, Artrit P (2010) Optimal trajectory of robot manipulator using harmony search algorithms. In: Recent Advances In Harmony Search Algorithm, Springer, pp 23–36
55. Wang J, Hou Y (2008) Optimal web service selection based on multi-objective genetic algorithm. In: International Symposium on Computational Intelligence and Design (ISCID), IEEE, vol 1, pp 553–556
56. Wang P, Chao KM, Lo CC (2010) On optimal decision for qos-aware composite service selection. Expert Systems with Applications 37(1):440–449
57. Weise T, Bleul S, Comes D, Geihs K (2008) Different approaches to semantic web service composition. In: Third International Conference on Internet and Web Applications and Services (ICIW), IEEE, pp 90–96
58. Wu B, Chi C, Xu S (2007) Service selection model based on qos reference vector. In: IEEE International Conference on Services Computing - Workshops (SCW 2007), IEEE, pp 270–277
59. Xu J, Reiff-Marganiec S (2008) Towards heuristic web services composition using immune algorithm. In: IEEE International Conference on Web Services (ICWS), IEEE, pp 238–245
60. Yan Y, Xu B, Gu Z (2008) Automatic service composition using and/or graph. In: 10th IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, IEEE, pp 335–338
61. Yu C, Huang L (2016) A web service qos prediction approach based on time-and location-aware collaborative filtering. Service Oriented Computing and Applications 10(2):135–149
62. Yu Q, Bouguettaya A (2009) Foundations for efficient web service selection. Springer Science & Business Media
63. Yu T, Zhang Y, Lin KJ (2007) Efficient algorithms for web services selection with end-to-end qos constraints. ACM Transactions on the Web (TWEB) 1(1):6
64. Zeng L, Benatallah B, Ngu AH, Dumas M, Kalagnanam J, Chang H (2004) Qos-aware middleware for web services composition. Software Engineering, IEEE Transactions on 30(5):311–327
65. Zeng L, Benatallah B, Ngu AHH, Dumas M, Kalagnanam J, Chang H (2004) Qos-aware middleware for web services composition. IEEE Trans Software Eng 30(5):311–327
66. Zhang W, Yang Y, Tang S, Fang L (2007) Qos-driven service selection optimization model and algorithms for composite web services. In: 31st Annual International Computer Software and Applications Conference (COMPSAC) Volume 2, pp 425–431
67. Zheng X, Yan Y (2008) An efficient syntactic web service composition algorithm based on the planning graph model. In: IEEE International Conference on Web Services (ICWS), IEEE, pp 691–699
68. Zhou A, Huang S, Wang X (2007) BITS: A binary tree based web service composition system. Int J Web Service Res 4(1):40–58
69. Zou D, Gao L, Li S, Wu J (2011) Solving 0–1 knapsack problem by a novel global harmony search algorithm. Applied Soft Computing 11(2):1556–1564