Computing and Informatics, Vol. 36, 2017, 1143–1172, doi: 10.4149/cai_2017_5_1143

HYBRID HONEY BEES MATING OPTIMIZATION ALGORITHM FOR IDENTIFYING THE NEAR-OPTIMAL SOLUTION IN WEB SERVICE COMPOSITION

Viorica Rozina Chifu, Cristina Bianca Pop
Ioan Salomie, Emil Stefan Chifu

Department of Computer Science
Technical University of Cluj-Napoca
Baritiu Street, No. 26-28, Cluj-Napoca, Romania
e-mail: {viorica.chifu, cristina.pop, ioan.salomie}@cs.utcluj.ro

Abstract. This paper addresses the problem of optimality in semantic Web service composition by proposing a hybrid nature-inspired method for selecting the optimal or near-optimal solution in semantic Web Service Composition. The method hybridizes the Honey-Bees Mating Optimization algorithm with components inspired from genetic algorithms, reinforcement learning, and tabu search. To prove the necessity of hybridization, we have analyzed comparatively the experimental results provided by our hybrid selection algorithm versus the ones obtained with the classical Honey Bees Mating Optimization algorithm and with the genetic-inspired algorithm of Canfora et al.

Keywords: Honey Bees Mating Optimization, genetic algorithms, tabu search, reinforcement learning, web service composition

1 INTRODUCTION

Web services have emerged as a flexible and advantageous technology for exposing software functionality over the Internet. However, there are many cases in which the available atomic Web services cannot address complex user requests and this leads to the necessity of developing methods for composing them. The composition of Web services is a difficult task if only their standard descriptions provided in the WSDL

documents are considered. The difficulty arises from the lack of semantics which is a necessity for automatically processing Web services. In this context, Semantic Web technologies seem to provide the necessary resources to develop methods for automatically composing Web services. The steps required for developing such composition methods are the following:

1. develop a semantic Web service ontology,
2. semantically annotate Web services using the ontology defined,
3. define a method for matching semantic Web service descriptions,
4. develop a method for composing semantic Web services, and
5. develop a method for selecting the optimal composition solution.

Among these steps, the one for developing a method for selecting the optimal composition reduces to identifying the optimal configuration of services that, when composed, would satisfy in a great percent the user requirements.

Because of the large number of available services providing similar functionality and because of the set of user constraints that must be satisfied by a composition solution, the problem of identifying the complete configuration of services that satisfies all the constraints imposed (i.e., maximizes an objective function) can be considered as a combinatorial optimization problem. Such problems can be solved through exhaustive or approximate approaches, the latter ones being preferred from the practical point of view due to their ability to provide the optimal or a near optimal solution in a short time and without processing the entire search space. A special type of approximate algorithms are the meta-heuristics which can be easily adapted to solve various combinatorial optimization problems. Some of these meta-heuristics, the population-based ones, focus mainly on diversification – the exploration of new areas of the search space, while others, the trajectory-based ones, focus mainly on intensification – the exploitation of a region in the neighborhood of a solution. Recently, a new class of meta-heuristics has emerged, the hybrid metaheuristics, which aim to maintain a dynamic balance between diversification and intensification [1].

Hybrid metaheuristics combine algorithmic components from various optimization algorithms aiming to improve the performance of the original metaheuristics in solving hard optimization problems [2]. Hybridizing metaheuristic for performance improvement does not necessarily guarantee that it will work well for all optimization problems, but only for some specific problems [2].

Generally, the approaches that hybridize metaheuristics with components from other metaheuristics use trajectory-based metaheuristics in population-based metaheuristics. This hybridization is motivated by the fact that population-based metaheuristics can find promising search space areas (due to their focus on search space exploration) which can be exploited to find the best solutions quickly using trajectory-based metaheuristics (due to their focus on the search space exploitation) [2].

In this paper we present a method for selecting the optimal composition solution that hybridizes the Honey-Bees Mating Optimization algorithm [3] with prin-

ciples from genetic algorithms, reinforcement learning, and tabu search. The search space for our hybrid method is represented by an Enhanced Planning Graph (EPG) which encodes all the possible composition solutions for a given user request. In our approach, a user request is described in terms of functional and non-functional requirements. To identify the optimal solution encoded in the EPG, we define a fitness function which uses as evaluation criteria the *QoS* attributes and the semantic quality of the services involved in composition. To assess the performance of the hybrid method proposed, we have designed an evaluation methodology consisting of the following steps:

1. develop different algorithm variants by gradually hybridizing the Honey Bees Mating Optimization algorithm with Genetic, Reinforcement Learning and Tabu Search components,
2. develop a set of test scenarios having different complexities to prove the scalability of the hybrid algorithms proposed,
3. identify the optimal composition solution for each scenario considered, by performing an exhaustive search,
4. identify the configurations of the adjustable parameters which provides the best results on each scenario, and
5. comparatively analyze the hybrid algorithm variant with other state of the art bio-inspired algorithms by considering the optimal configurations of the adjustable parameters.

We chose to use and hybridize the Honey Bees Mating Optimization (HBMO) algorithm since the results obtained by HBMO are as good as those obtained with genetic algorithms [3]. In addition, HBMO has the advantage of quickly converging towards the optimal solution and having a small probability to get stuck into a local optimum.

The paper is structured as follows. Section 2 reviews state of the art nature-inspired selection methods. Section 3 presents the background concepts related to the problem of selecting the optimal or near-optimal solution in semantic Web service composition. Section 4 introduces the hybridization components, while Section 5 details the associated hybrid selection algorithms. Section 6 discusses the adjustable parameters of the selection methods and analyzes the experimental results. Section 7 performs a comparative analysis of our proposed approaches versus other similar approaches from the research literature. The paper ends with conclusions.

2 RELATED WORK

This section reviews some of the nature-inspired selection methods available in the research literature for identifying the optimal or near-optimal solution in Web service composition.

2.1 PSO-Based Approaches

In [4], the authors present a Particle Swarm Optimization-based method for selecting the optimal Web service composition. The approach proposed models a particle as a vector of n integers where each component represents the candidate Web service for performing a certain task. The velocity of a particle is represented as a vector of n terms, where a term represents the velocity associated to a service and takes values in the interval $[-n/2, n/2]$. The acceleration coefficients from the formulae for updating the velocity of a particle are computed by applying the Clerk and Kennedy method [5]. The quality of a composition solution is evaluated by a fitness function which considers as evaluation criteria the *QoS* attributes (e.g. execution time, availability, reputation, and successful execution rate) as well as the user defined constraints.

The authors of [7] propose a Particle Swarm Optimization-based approach for selecting the optimal Web service composition based on *QoS* attributes. The approach models the selection problem as a multi-objective composition optimization problem with *QoS* constraints which is then turned into a single objective optimization problem by using the ideal point method. In this approach, the Web service composition search space is represented as a directed graph where the nodes are groups of similar Web services (i.e. services having similar functionalities, but different *QoS* attributes) and the edges are the connections between services. For applying the Particle Swarm Optimization algorithm in the context of selecting the optimal Web service composition, the formulas for updating the position and velocity of a particle have been redefined.

In [10], the authors propose a new Particle Swarm Optimization-based algorithm, EMOSS, for solving the Web service selection problem. The approach models the selection problem as a constrained multi-objective optimization problem and then applies the EMOSS algorithm to solve this problem. The novelty of the EMOSS algorithm consists in the way in which the updating strategy from the Particle Swarm Optimization metaheuristic is redefined. More exactly, to select the global best position of each particle, the authors use an adaptive grid strategy combined with a shortest distance strategy based on the Euclidian distance.

2.2 Ant Colony Optimization-Based Approaches

In [6], the authors present an improved version of the Ant Colony Optimization algorithm for dynamic Web service composition. In this approach, the Web service composition is modeled as a graph where a node is a Web service and an edge connects two Web services. The weight associated to the edge represents the value of *QoS* attributes between the two nodes connected by the edge. The novelty of this approach consists in a new interpretation given to the pheromone value laid on the edge, which refers to the past learned experience involving the current service and its predecessor.

In [11], the authors propose an Ant Colony Optimization-based method for selecting the most appropriate concrete services for a given workflow, so that the user's *QoS* preferences and transactional constraints are satisfied. The search space is modeled as a directed acyclic graph whose nodes correspond to candidate services and whose edges represent connections between these services. The Ant Colony Optimization-based method is used to select the graph path that best satisfies the given preferences and constraints. The selection method considers a set of ants which iteratively explore the graph from the start node towards the end node by probabilistically choosing the next candidate services. The probability takes into account the *QoS* and transactional score of the candidate service as well as the pheromone quantity associated to the corresponding edge. At the end of an algorithm iteration, after all the ants have built a path (i.e. a composition solution), each path is evaluated from the *QoS* and transaction constraints perspectives, and, based on this evaluation, a pheromone evaporation takes place. In the end, the algorithm returns the path that best satisfies the *QoS* and transaction constraints.

In [23] the authors present an ant-inspired method for selecting the optimal or a near optimal solution in semantic Web service composition. The proposed method adapts and enhances the Ant Colony Optimization meta-heuristic and considers as selection criteria the QoS attributes of the services involved in the composition as well as the semantic similarity between them. To improve the performance of the proposed selection method a 1-OPT heuristic is defined which expands the search space in a controlled way so as to avoid the stagnation on local optimal solutions. The ant-inspired selection method has been evaluated on a set of scenarios and comparatively analyzed with a cuckoo-inspired selection method.

2.3 Genetic Approches

In [12], the authors propose a genetic-based approach for selecting the optimal or near optimal solution in Web service composition based on *QoS* attributes. The approach uses an adaptive crossover strategy combined with a population diversity measurement to improve the convergence of the classical genetic algorithms. The adaptive crossover strategy consists in probabilistically applying the crossover operation based on the population diversity and the fitness of an individual. When the population diversity or the fitness value of an individual will decrease then the crossover probability will increase, otherwise the crossover probability will decrease.

In [21], a genetic individual represents a composition solution encoded as an integer array. The size of the array is equal to the number of abstract tasks in the composition model, while an array element represents the index of the chosen concrete service. The authors propose a static and a dynamic fitness functions to evaluate a composition solution. Both functions aggregate two components: a *QoS* component for evaluating the *QoS* score and a component for evaluating the level of *QoS* constraint satisfaction. Additionally, the dynamic fitness function introduces a penalty which varies from one generation to another. The evaluation of the aggregate *QoS* score of a composition solution is based on the control flows like

sequence, switch, flow, and loop. To update a composition solution, the authors use the crossover, mutation, and selection operators. The crossover operator involves establishing randomly two crossover points and swapping the corresponding segments of the parent solutions. The mutation operator selects randomly a service from the currently processed solution, which is randomly replaced with a similar one from the available services. The approach considers the elitist and roulette wheel operators to select the individuals that will be part of the next generation.

2.4 Immune Approches

In [15], the authors use the negative selection algorithm for selecting the optimal solution in Web service composition. In this approach, a Web service composition solution is modeled as an integer string. An integer value in the string represents the identification number of a concrete Web service chosen to implement the functionality of an abstract task. In addition, the concept of gene is mapped to the identification number of a service, while the gene segments are mapped to the candidate services. The algorithm proposed consists in an iterative stage, which includes the following steps:

gene recombination – aims to generate new composition solutions by probabilistically choosing candidate services,

negative selection – aims to eliminate the composition solutions containing self patterns, namely patterns of services that are not good, and

gene warehouse management – aims to update the consistence of alleles, i.e. component tasks.

In [22], the authors propose an immune-inspired method for selecting the optimal or a near-optimal solution by considering the *QoS* attributes as evaluation criteria. A composition solution is represented as a set of binary strings, one string for each service selected to be part of the composition solution. The string associated to a service represents the binary encoding of its identification number within the set of services associated to an abstract task. The fitness function evaluates the quality of a solution based on the *QoS* attributes (the quality of a solution is computed by considering the cost, the response time, the availability and the reliability of the services involved in the composition solution). Each of QoS attributes was a separate calculation method, because they have different interpretations: for some of them a higher value is better, meanwhile for others smaller values are better.

Besides the affinity between the antigen and an antibody, the authors also model the affinity between two antibodies as a Hamming distance between the associated composition solutions. The algorithm proposed applies two immune operations. The first operation aims to generate a diverse population of antibodies (i.e. composition solutions) based on the affinity between antibodies and the antibodies concentration. The second operation aims to update the population of antibodies by using a clonal

selection inspired approach. This approach relies on a mutation operator which modifies the elements of a solution that alter the fitness of the solution.

2.5 Hybrid Approches

The authors of [8] present a hybrid method for selecting the optimal Web service composition based on the *QoS* attributes, that combines genetic algorithms with clonal selection principles from the artificial immune system. The clonal selection is used instead of the tournament selection to avoid the premature convergence of the classical genetic algorithm. In this approach, an antibody represents a composition solution, an antigen is a *QoS* value, and the fitness function for evaluating the quality of an antibody is defined by considering the *QoS* attributes as evaluation criteria.

In [9], the authors propose an approach for selecting the optimal Web service composition that combines the Particle Swarm Optimization with the Munkres algorithm. The Munkres algorithm is applied in every iteration of the Particle Swarm Optimization based algorithm in order to improve the position of a subset of particles from the population of particles. If the Munkres algorithm indeed improved the position of a particle, then the position of the particle is updated. For applying the Particle Swarm Optimization metaheuristic to the Web service selection problem, the concepts of position and velocity of a particle have been defined as: the position of a particle is defined as a vector having the Web services of a composition solution as elements and the size equal to the number of elements in the composition solution; the velocity of a particle is defined as a list of changes that can be applied to a particle in order to ensure the movement of the particle into a new position. Additionally, the subtraction between two particles, the addition of velocities, and the multiplication of a velocity have been redefined.

In [13], the authors address the problem of selecting the optimal Web service composition based on *QoS* attributes by proposing a hybrid selection method. The method combines immune principles (namely the clonal selection) with the Particle Swarm Optimization. In this approach, the position of a particle is mapped to a Web service composition solution, modeled as an integer vector, where a vector element indicates the identifier of the service selected for a certain task. A vector element might also have a value 0, indicating that the associated task is not part of the solution. The mathematical operators from the Particle Swarm Optimization are redefined. An antigen is mapped to the optimization problem, while an antibody is mapped to a Web service composition solution. In their method, the authors introduce the immune concept of crowd distance between two antibodies. This refers to the difference between the *QoS* values associated to the services part of each antibody. The immune concept of affinity is mapped to a value computed based on the fitness value associated to an antibody and based also on the distance between this antibody and the current global optimal solution. The authors also consider the immune concepts of clone proliferation and hypermuta-

tion. Clone proliferation refers to creating several copies of the highest affinity antibodies, whereas hypermutation refers to replacing probabilistically each service part of a solution with another one selected randomly. The authors also introduce two strategies, an improved local best first strategy and a perturbing global best strategy, used in the process of improving Web service compositions. The algorithm proposed consists of two phases, an initialization one and an iterative one. The initialization stage aims to create the initial population of particles (i.e. antibodies), by associating a random composition solution to each particle, together with an initial velocity. In the iterative stage, the following steps are executed:

1. the particles are cloned according to the fitness of their solutions, and then hypermutated,
2. the perturbing global best strategy is applied,
3. the velocity of each particle is updated,
4. the position of each particle is updated,
5. the personal best of each particle is updated if it is the case, and
6. the global best over all the particles is updated if it is the case.

In [14], the authors hybridize the Particle Swarm Optimization with a chaotic mutation and a local search strategy to select the optimal Web service composition in terms of *QoS* attributes. The position of a particle is mapped on a Web service composition solution. The hybrid algorithm proposed starts from a randomly generated set of composition solutions. In every iteration of the algorithm, the positions of the particles are updated by means of the chaotic mutation which aims to escape the algorithm from locally optimal solutions.

In [16], the authors hybridize the clonal selection algorithm with the Particle Swarm Optimization and other heuristic strategies (the local adaptation preemptive strategy, the elitist perturbation guiding strategy, and the global tournament strategy) in the context of selecting the optimal Web service composition solution. Web service composition is modeled as a set of abstract tasks, each task having a set of concrete services associated. In this approach, the antibody and the position of a particle are both mapped on a Web service composition solution modeled as an integer vector. An element of the vector represents the identification number of a concrete service selected for a particular abstract task. An antigen is mapped on the optimization problem being solved. The velocity of a particle is mapped on an integer vector containing 0s and 1s. The formulae from the Particle Swarm Optimization are redefined to work with the position and velocity structures proposed in the context of Web service composition. Hypermutation is mapped to an operation that is applied probabilistically on each task of a Web service composition solution to change randomly the current concrete service with another one. The hybrid algorithm proposed consists of two stages, an initialization stage aiming to generate the initial population of particles and their associated velocities, and an iterative stage aiming to:

1. clone the solutions in the current population, based on their affinity,
2. apply the local adaptation preemptive strategy on the current population, which ensures that a concrete service having a high QoS score has more chances to be chosen,
3. apply the elitist perturbation guiding strategy on the current population for diversification purposes,
4. execute the clonal selection operation or the global tournament selection operation, and
5. apply the elitist perturbation guiding strategy.

3 BACKGROUND

This section presents some background concepts related to the hybrid selection method proposed in this paper.

3.1 Enhanced Planning Graph Structure

The selection of the optimal or near-optimal solution in semantic Web service composition is considered as an optimization problem since its goal is to find the most appropriate configuration of Web services out of a very large set of possible configurations, given a composition request.

In our approach, the search space of the selection problem is modelled as an Enhanced Planning Graph (see Figure 1) previously introduced in [17]. The EPG model is an enhanced version of the classical AI planning graph [18] adapted in the context of the semantic Web service composition problem. In our EPG model, in addition to the classical AI planning graph, we introduce two new abstractions: the *service cluster* (SC) and *parameter cluster* (PC).

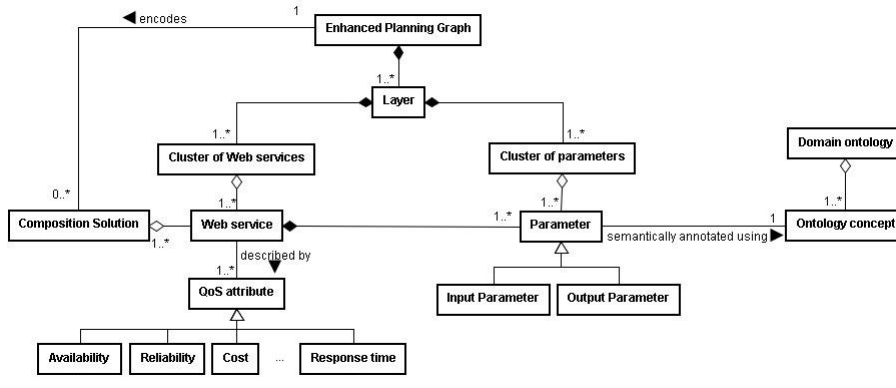

Figure 1. UML representation of the Enhanced Planning Graph

The construction of an EPG is triggered by a composition request issued by a user, and is formally defined as follows:

$$request = (In, Out, w_{Sem}, w_{QoS}) \tag{1}$$

where *request.In* is a set of ontological concepts semantically describing the user provided inputs, *request.Out* is a set of ontological concepts semantically describing the user requested outputs, $request.w_{Sem}$ is the weight of the semantic quality, while $request.w_{QoS}$ is the weight of the QoS quality.

A service cluster groups together services providing similar functionalities and having similar input parameters (i.e. services for which there is a degree of match between their semantic descriptions):

$$\begin{aligned} SC = \{sws_k | \forall\ sws_i,\ sws_k \in SC,\ (sws_i.f \subseteq sws_k.f \vee sws_i.f \supseteq sws_k.f) \\ \wedge (|sws_i.In| == |sws_k.In|) \wedge (\forall sws_i.In_j \in sws_i.In, sws_k.In_p \in sws_k.In, \\ sws_i.In_j \subseteq sws_k.In_p \vee sws_i.In_j \supseteq sws_k.In_p)\}. \end{aligned} \tag{2}$$

In Formula (2), sws is a semantic Web service, formally defined as:

$$sws_k = (f, In, Out, QoS) \tag{3}$$

where $sws.f$ is an ontology concept that annotates the functionality of the service, $sws.In$ is a set of ontological concepts annotating the input parameters of the service, $sws.Out$ is a set of ontological concepts annotating the output parameters of the service, while $sws.QoS$ is a set of values for the QoS attributes of the service considered. For simplicity, we consider that a Web service has only one operation.

The ontological concepts used for describing the user request and for annotating the semantic Web services belong to a domain ontology Γ.

A parameter cluster groups together similar input and output parameters of a service:

$$PC = \{p_k | \forall\ p_i,\ p_k \in PC, p_i \subseteq p_k \vee p_i \supseteq p_k\} \tag{4}$$

where p_k is an ontology concept that annotates an input/output parameter of a service.

The process of building an EPG operates at the semantic level by considering the ontological concepts that annotate the functionality of the services and the input and output parameters of the same services. At each step, a new layer i consisting of a pair (A_i, L_i) is added to the graph, where A_i represents a set of service clusters and L_i is a set of clusters of service input/output parameters. Layer 0 consists of a pair (A_0, L_0), where A_0 is an empty set of services, and L_0 contains the ontological concepts annotating the input parameters of the user request, which are contained in the set *request.In*. For each layer $i > 0$, A_i consists of a set of clusters of services for which the input parameters are contained in L_{i-1}. The services which contribute in each step to the extension of the EPG are provided by a discovery process which finds the appropriate Web services in a repository of services. The discovery is

based on the semantic matching between the inputs of the services and the set of parameters of the previous graph layer. The set L_i is built as a union of the set L_{i-1} and the set of ontological concepts annotating the outputs of the services in A_i. Consequently, an EPG can be formally defined as follows:

$$EPG = \{(A_i, L_i)\} \tag{5}$$

where

$$A_i = \{SC_{ij} | SC_{ij} \text{ is the service cluster } j \text{ from layer } i \text{ of the EPG}\}, \tag{6}$$

$$L_i = L_{i-1} \cup \{PC_{ij} | PC_{ij} \text{ is a cluster of output parameters of the services from } A_i\}. \tag{7}$$

The construction of an EPG ends either when the user requested outputs are contained in the current set of parameters or when the graph reaches a fixed point. Reaching a fixed point means that the sets of service clusters and parameter clusters are the same for the last two consecutively generated layers.

To illustrate how an EPG is built, we assume a very simple scenario where the user is interested in making the travel arrangements for attending a conference (i.e. flight and hotel reservation). Based on the user request (see Table 1) (expressed as a set of ontological concepts describing the requested inputs and the expected outputs for the composed service) and the set of available semantic Web services, an EPG (see Figure 2) is iteratively built. The construction of the EPG starts from the inputs specified by the user and adds, at each iteration, a new layer in the graph. This process ends when the expected user outputs are obtained. As can be seen in Figure 2, a solution obtained is relevant if there is a degree of semantic matching close to 1 between its semantic description and the semantic description of the requested composed service as specified by the user. For simplicity, we have illustrated in this example only one solution for the user request considered.

inputs	***outputs***
Airport, Airport, Date, Name	FlightNumber, SeatNumber
Country, City, CreditCardNumber	BookingNumber, Hotel

Table 1. User request for making the travel arrangements for attending a conference

3.2 Fitness Function

To evaluate the quality of a composition solution, the following fitness function is used [17]:

$$QF(sol) = \frac{w_{QoS} * QoS(sol) + w_{Sem} * Sem(sol)}{(w_{QoS} + w_{Sem}) * |sol|} \tag{8}$$

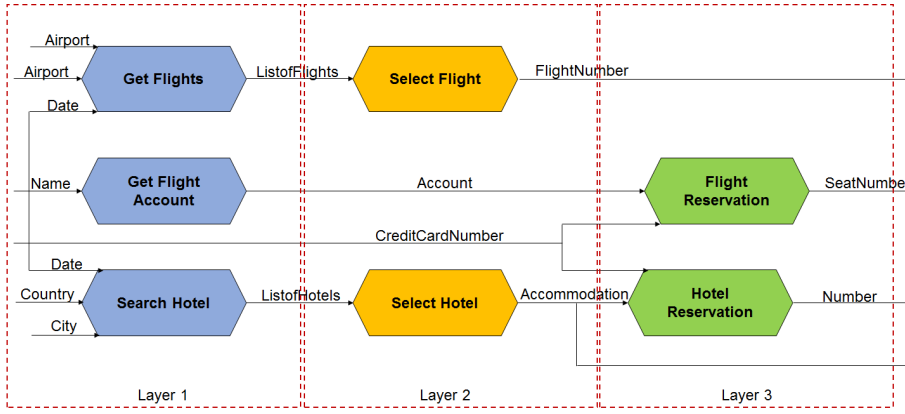

Figure 2. Example of constructing an EPG for making the travel arrangements for attending a conference

where

- $QoS(sol)$ [17] is the QoS score of the composition solution sol;
- $Sem(sol)$ [17] is the semantic quality score of the composition solution sol;
- w_{QoS} and w_{Sem} are weights representing the user preferences regarding the relevance of the QoS and semantic quality;
- $|sol|$ represents the number of services involved in a composition solution.

The QoS score of a composition solution is computed using the following formula:

$$QoS(sol) = \frac{\sum_{i=1}^{n} w_i * qos_i(sol)}{\sum_{i=1}^{n} w_i} \tag{9}$$

where

- $qos_i(sol)$ represents the value of a QoS attribute computed for the composition solution sol. The formulas for computing the values for each QoS attribute (i.e. availability, reliability, cost, and running time) have been previously introduced in [17].
- w_i is the weight representing the relevance of the qos_i attribute;
- n is the total number of QoS attributes considered.

The semantic quality score of a composition solution sol is computed as follows:

$$Sem(sol) = \frac{\sum_{i=1}^{nLinks} simS\left(s_{kl}^{j}.out, s_{qr}^{p}.in\right)}{nLinks} \tag{10}$$

where

- s_{kl}^{j} is the service l in cluster k of layer j;
- s_{qr}^{p} is the service r in cluster q of layer p;
- s_{kl}^{j}, s_{qr}^{p} are part of the solution *sol*, $j < p$;
- *simS* is the degree of semantic matching [17] computed between a subset of outputs of a service s_{kl}^{j} and a subset of inputs of a service s_{pq}^{r} (we compute the degree of semantic matching between the ontology concepts describing the input/output subsets of the two services). The formula used for computing the semantic similarity score is based on *FMeasure*, a metric from the information retrieval domain, and it takes into consideration the hierarchy of concepts as well as the hierarchy of properties in the ontology.
- *nLinks* is the total number of semantic similarity links in the composition solution *sol*.

3.3 Honey Bees Mating Optimization Algorithm

The Honey Bees Mating Optimization (HBMO) algorithm [3] is inspired by the mating behaviour of queen bees in nature. In the HBMO algorithm, the queen and the set of drones it mates represent solutions of the optimization problem being solved, while the workers represent heuristic strategies used to update the solutions. The HBMO algorithm consists of two stages, an initialization one and an iterative one. In the initialization stage, a set of initial solutions are randomly generated and then ranked according to their fitness values. The best solution is nominated as the queen, while the other solutions are nominated as drones that will participate in the mating process. Then, in the iterative stage, the following steps are performed [3]:

- A simulated annealing-based strategy is used to select the drone solutions that will mate with the queen solution.
- A new set of solutions are generated by applying different modification strategies between the queen solution and each selected drone solution.
- The solutions obtained in the previous step are further improved using different modification strategies.
- The queen solution is updated if there is another solution with a higher fitness in the newly generated solutions.
- The set of drone solutions that will be submitted to the next iteration is updated.

4 THE HYBRID NATURE-INSPIRED MODEL

This section presents the hybrid nature-inspired model which will be used in the development of our hybrid nature-inspired selection algorithm. The model is composed of one core constituent and multiple hybridization components.

4.1 Core Component

The **Honey Bees Mating inspired core component** is defined by mapping the concepts from the Honey Bees Mating Optimization metaheuristic [3] onto the concepts specific to selecting the optimal or near-optimal Web service composition solution (see Table 2).

Honey Bees Mating Optimization Concepts	**Concepts from Web Service Composition Selection**
Drone	Web service composition solution
Queen	Web service composition solution having the highest score in the population at the current moment
Process of generating new solutions for drones and queens	Perturbation of Web service composition solutions by replacing some of the solutions' services with other services from the same cluster, according to specific criteria
Objective function	Fitness function evaluating the semantic and QoS quality of a Web service composition solution

Table 2. Mapping the concepts from the HBMO metaheuristic onto the concepts of selecting the optimal or near-optimal Web service composition solution

The fitness function [17] used by the core component to evaluate a Web service composition solution considers the QoS attributes of the services part of a solution as well as the semantic quality of the connections between these services (see Formula (8)).

4.2 Hybridization Components

Our hybrid nature-inspired model includes two hybridization components that have been previously introduced in [10]. In what follows we briefly describe these hybridization components.

The Tabu Search and Reinforcement Learning Component. The Tabu Search meta-heuristic [20] relies on the principles of forbidding a set of elements for a period of time and turning them as allowable elements under certain circumstances. This is achieved by using two memory structures, namely a short-term memory and a long-term memory. The short-term memory stores information about the elements that are tabu for a predefined number of iterations, while the long-term memory stores information regarding the solutions identified so far, this latter information being used to improve the quality of the solutions generated in the future.

In the context of selecting the optimal Web service composition solution it is important to constantly improve the current locally optimal solution. This can be efficiently achieved by taking into account the results of previous decisions.

For example, if once we have replaced a service sws_1 with another service sws_2 and we have thus obtained a better solution, then it is obvious that by using the same replacement in the future, better solutions will also be obtained. This is the motivation that led us to choose the tabu short-term and long-term memories as important components of our nature-inspired hybrid model. Consequently, we have defined two special data structures, called the *short-term* and *long-term memory* structures. The short-term memory structure, M_S, is defined as follows:

$$M_S = \{m_s | m_s = ((sws_i, sws_j), iterations)\} \tag{11}$$

where sws_i is the service that was replaced by sws_j, and *iterations* specifies the number of iterations in which sws_i cannot be replaced by sws_j.

The long-term memory structure, M_L, is defined as:

$$M_L = \{m_l | m_l = ((sws_i, sws_j), rlscore)\} \tag{12}$$

where sws_i is the service that will be replaced by sws_j and *rlscore* is a numeric value used for recording rewards and penalties for the pair (sws_i, sws_j). The concepts of reward and penalty are borrowed from the reinforcement learning technique to provide feedback information assessing the correctness of the decisions taken by the selection algorithm in the past and to take thus better decisions in the future.

The Genetic Component. The genetic component uses a memory-based mutation operator to improve the quality of a Web service composition solution. This operator is similar to the classical genetic mutation operator. In addition to the latter, it adds some strategies for choosing the mutation points and for selecting the service that will replace the one as found in the mutation point chosen. To identify the mutation points, the currently processed solution is compared with the current locally optimal solution service by service, and the indices of the services that are different in the two solutions will represent the mutation points. To replace the services corresponding to the mutation points in the currently processed solution, the two memory structures part of the Tabu Search and Reinforcement Learning Component will be consulted. Figure 3 shows how the memory-based mutation operator is used.

When applying the memory-based mutation operator, the following steps are executed for each service s_i which belongs to the current solution $sol_{current}$ and which is not part of the locally optimal solution sol_{opt}:

- The long-term memory is searched for a triple $(s_i, s_j, rlScore)$ having the highest *rlScore*. If such a triple is found (i.e., s_i has been replaced with s_j in the past), then s_i is replaced with service s_j in $sol_{current}$; otherwise s_i is replaced with a service s_k randomly chosen from the same cluster.
- The updated $sol_{current}$ is evaluated using the fitness function in Formula (8). If the quality of the solution is improved (as compared with $sol_{current}$ before

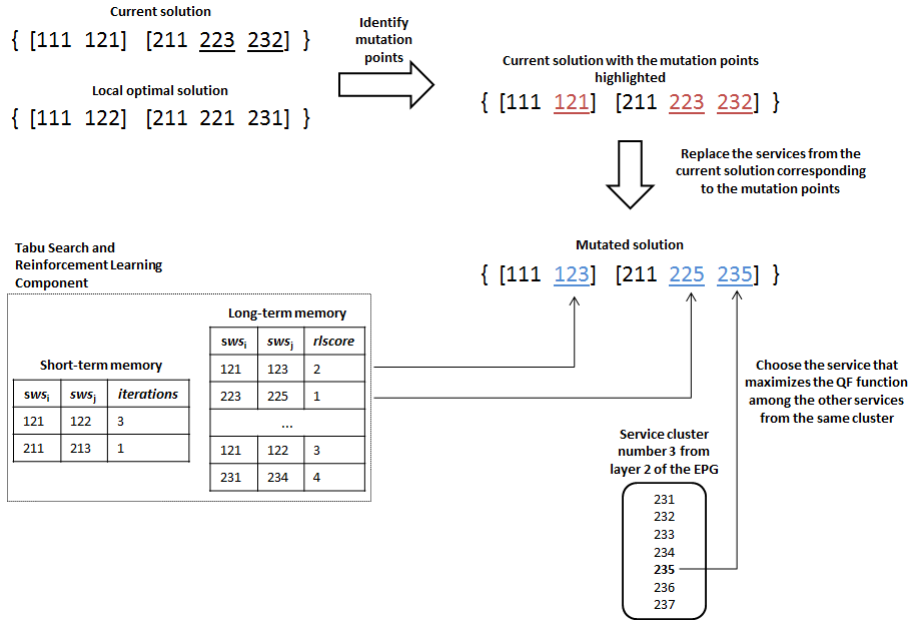

Figure 3. Example of applying the memory-based mutation operator

mutation) then rewards are granted to the pair (s_i, s_j) (if s_j was the substiute in the first step) or (s_i, s_k) (if a random s_k was the substitute); otherwise penalties are granted to the same pair.

5 THE HONEY BEES MATING INSPIRED ALGORITHM

Our hybrid honey bees mating inspired algorithm (see Algorithm 1) adapts, enhances, and hybridizes a version of the Honey Bees Mating Optimization algorithm [3], which was proposed for optimization problems. To do this, we injected the Tabu Search and Reinforcement Learning component as well as the Genetic component in the honey bees mating inspired core component (see Section 4). The algorithm takes as inputs the following parameters:

1. An enhanced planning graph structure, *EPG*;
2. The number of bees (i.e. composition solutions) used in the search process, *noBees*;
3. The queen's capacity to combine and to generate new solutions, *qCap*;
4. The number of crossover points, *noC*;
5. The number of mutation points, *noM*;

6. The percentage of the worst solutions that will be replaced, *wProc*; and
7. The maximum number of iterations, *noIt*.

The hybrid honey bees mating inspired algorithm consists of an initialization stage and an iterative stage. In the initialization stage, the set of bees is initialized with randomly generated Web service composition solutions, and the solution having the highest score is nominated as the queen.

The initialization phase continues by refactoring the number of mutation points and the number of crossover points provided as input arguments to the algorithm. The two arguments are adjusted according to the structure of the given EPG, by using the formulas below:

$$noM = \begin{cases} noM', & \text{if } noM' > |sol|/2, \\ noM' * 2 - 1, & \text{otherwise} \end{cases} \tag{13}$$

where noM' is computed as

$$noM' = noM \bmod |sol|, \tag{14}$$

$$noC = noC \bmod noL \tag{15}$$

where noL is the number of EPG layers.

Formulas (13) and (15) for computing the number of mutation points and the number of crossover points were introduced in order to adjust the values introduced by the user for the number of mutation/crossover points. The number of mutation points must be lower than or equal to the number of services involved in the composition solution. So, the mathematical operator modulo (i.e. **mod**) helps assign to the number of mutation points a value close to the one introduced by the user. We have also taken into account in the definition of Formula (13) that mutation must be applied on more than half the number of services involved in a solution (we reached this conclusion based on the experimental results).

In the case of crossover points, the number of crossover points must be lower than the numbers of layers in the EPG. This is because crossover is performed by interchanging services from the same layer in two different solutions.

In the iterative stage, the following main steps are repeated until the maximum number of iterations *noIt* is reached:

- For every bee, a probabilistic decision is made on whether the bee will be submitted to a crossover operation with the queen or not. The probability is computed as (adapted from [3]):

$$Mating\_Probability = e^{\frac{QF(queen)-QF(bee)}{qCap}} \tag{16}$$

where $qCap$ is the queen capacity, computed with the following formula:

$$qCap = qCap * \alpha. \tag{17}$$

Algorithm 1: Hybrid Honey Bees Mating Optimization

Inputs: EPG, $noBees$, $qCap$, noC, noM, $wProc$, $noIt$
Output: sol_{opt}
Comments: M_L – long term memory, M_S – short term memory it_c – current iteration number, $Broods$ – the set of broods.
begin
  $Bees =$ **Init_Sol_Set**$(EPG, noBees)$
  $sol_{opt} =$ **Get_Queen**$(Bees)$
  $noC =$ **Refactor_Crossover**(EPG, noC)
  $noM =$ **Refactor_Mutation**(EPG, noM)
  $M_S = \emptyset$, $M_L = \emptyset$, $it_c = 1$, $Broods = \emptyset$
  while$(it_c \leq noIt)$ **do**
    foreach *bee* **in** *Bees* **do**
      $r =$ **Random**$(0, 1)$
      if (**Mating**$(qCap) > r$) **then**
        $TBroods =$ **Crossover**(sol_{opt}, bee, noC)
        $Broods = Broods \bigcup TBroods$
        $qCap =$ **Update_QCap**$(qCap)$
      end if
    end for
    foreach *brood* **in** *Broods* **do**
      for $i = 1$ **to** noM **do**
        $ind =$ **Get_Mutation_Index**$(brood)$
        $brood_n =$ **Mutation**$(M_S, M_L, brood, ind)$
        $M_S =$ **UpdateMS**$(M_S, brood_n)$
        if $((brood_n \neq brood)$ &&
          (**Fitness**$(brood_n) >$ **Fitness**$(brood))$ &&
          $(brood_n \notin Broods)$ &&
          $(brood_n \notin Bees))$ **then**
            $M_L =$ **Reward**$(brood_n, brood, ind)$
            $brood = brood_n$
        else $M_L =$ **Penalize**$(brood_n, brood, ind)$
        end if
        if $(brood_n \notin Bees)$ **then**
            $Bees =$ **Replace**$(Bees, brood_n)$
        end if
        if (**Fitness**$(brood_n) >$ **Fitness**$(sol_{opt}))$ **then**
          $sol_{opt} = brood_n$
        end if
      end for
    end for
    $Bees =$ **Replace_Old**$(Bees, wProc)$
    $it_c = it_c + 1$
  end while
  return sol_{opt}
end

In Formula (17), the new value for the queen capacity is computed by considering its old value and a parameter α which takes values in the interval $(0, 1)$. If the new value is higher than a predefined threshold, then the bee will be involved in the mating process, otherwise not. The initial value for the queen capacity is determined experimentally.

The crossover operation (procedure **Crossover** in Algorithm 1) between two solutions is performed by interchanging services from the same layer in the two solutions. The crossover points are selected randomly and the number of crossover points is computed by using formula (15).

As a result of crossover, two brood solutions are obtained that will be added to the set of broods.

- The set of brood solutions are submitted to a mutation process, where the memory-based mutation operator is applied upon each brood solution. For each brood, the following steps are used (these steps are applied a number of times *noM*, where *noM* is computed with formula (13)):

  - Randomly select a point where the mutation is to be applied (procedure **Get_Mutation_Index** in Algorithm 1).
  - Apply mutation (procedure **Mutation** in Algorithm 1) on the brood, as follows: in the brood, the service corresponding to the mutation point is replaced with another service by applying the memory-based mutation operator (see the **Genetic Component** in section 4.2).
  - Update the short term memory.
  - If the mutated brood, $brood_n$, is different from the old brood, *brood*, and the fitness of $brood_n$ is better than the fitness of *brood*, and $brood_n$ is not identical with any bee in the *Brood* set nor in the *Bees* set, then the old brood *brood* becomes (i.e. is replaced with) the mutated brood, $brood_n$. The long-term memory structure is updated by giving rewards (computed using procedure **Reward** in Algorithm 1) to the pair of services that were interchanged. Otherwise penalties (computed using procedure **Penalize** in Algorithm 1) are given to the pair of services that were interchanged.
  - If the mutated brood, $brood_n$, is not identical with any bee in the set of bees, then it will replace the worst solution from the set of bees. Additionally, if the mutated brood $brood_n$ is better than the queen, (i.e. sol_{opt}), then the queen will be updated.
  - Finally, the new mutated brood is submitted to a new mutation process by selecting randomly another mutation point. This process is repeated for a number of times *noM*.

- A percentage of the worst bees are replaced with randomly generated solutions.

6 SETTING THE OPTIMAL VALUES FOR THE ADJUSTABLE PARAMETERS

In order to establish the optimal values for the adjustable parameters, we have performed experiments on four different scenarios involving EPG structures of various complexities. Two of the scenarios are inspired from the trip planning domain, whereas the remaining two are inspired from the medical care domain. Table 3 presents information about the Enhanced Planning Graphs involved in the four scenarios as well as the processing done on these graphs when using an exhaustive search method. For each of the scenarios considered, the following information is illustrated:

1. Code of scenario;
2. The configuration of the EPG, illustrating its layers, the clusters on each layer (the number of clusters on a layer is given by the cardinality of the set associated to the layer), and the services per cluster (the number of services in a cluster is given by the value of each element in the set associated to a layer);
3. The complexity of the search space, in terms of total number of possible solutions encoded in the EPG structure;
4. The optimal fitness value for the scenario considered;
5. The execution time in which the optimal solution has been found when using exhaustive search (it is the time required to go through all the possible solutions in order to identify the optimal one).

Scenario Code	*EPG Configuration*	*Search Space Complexity*	*Optimal Fitness*	*Time (min:sec)*
S	Layer 1: {4 5 6} Layer 2: {6 4 6} Layer 3: {4 6 5}	2 073 600	6.456	3:08
M	Layer 1: {3 5 4 6} Layer 2: {6 4 6 5} Layer 3: {4 6}	6 220 800	7.482	15:47
L	Layer 1: {6 5 3 3} Layer 2: {4 6 4 3} Layer 3: {6 5 6}	13 996 800	8.024	56:18
XL	Layer 1: {4 4 5 4} Layer 2: {5 4 5 4} Layer 3: {6 5 5}	19 200 000	7.577	63:23

Table 3. Configuration of EPGs and other associated information

The methodology for establishing the optimal configuration of values for the adjustable parameters consists of three steps. The first step performs an exhaustive

search in the EPG in order to identify the score of the optimal composition solution (see Table 3). This score is further used for identifying the most appropriate configuration for the adjustable parameters, one which would ensure that the optimal or a near-optimal composition solution is obtained without processing the entire search space. The second step identifies correlations between the adjustable parameters and the complexity of the search space. This is done by performing a set of try-error experiments and analyzing the impact of the values of the adjustable parameters on the performance of the algorithm in terms of fitness and execution time. The try-error experiments consist of two phases: first fixing different values for the adjustable parameters before the execution of the meta-heuristic; then determining empirically, based on the experimental results, the intervals for the adjustable parameters in relation to the search space complexity.

The third step in establishing the optimal parameters consists in fine-tuning iteratively the values of the adjustable parameters with the aim to identify their optimal configuration. For each of the scenarios considered, we have performed 100 runs of the selection algorithm for each configuration (we have considered about 100 configurations for the adjustable parameters). We analyzed the fitness value, the execution time, and the standard deviation which were obtained when using the different settings of the parameters.

The adjustable parameters of the Hybrid Honey Bees Mating Optimization algorithm are the following: noB – the number of bees, $qCap$ – the queen capacity, qS – the queen speed which in our approach corresponds to the number of crossover points, qE – the queen energy which corresponds to the number of mutation roc-points, and wPo – the percentage of the worst solutions that will be replaced with randomly generated ones.

In Tables 4–7 we illustrate the top fifteen experimental results (the best experimental results in terms of fitness values) obtained while tuning the values of the adjustable parameters for scenarios S, M, L, and XL, respectively. The experiments have been performed by applying the third step of the methodology for setting the values of the adjustable parameters. The results were achieved with our Hybrid Honey Bees Mating Optimization algorithm. In Tables 4–7, qS' and qE' are the adjusted values for speed (i.e. number of crossover points) and energy (i.e. number of mutation points), computed with Equations (13) and (15) as starting from the values introduced by the user for qS and qE.

For all the four scenarios, we have considered the same stopping condition (namely a predefined number of iterations of 100).

By analyzing the experimental results presented in Tables 4–7, we noticed that, for all the scenarios considered, our Hybrid Honey Bees Mating Optimization algorithm is able to identify a good solution (i.e. a solution whose standard deviation from the optimal solution is lower than or equal to 0.065) in a short execution time (the longest execution time for all the scenarios is 1.85 seconds).

However, in order to identify which are the adjustable parameters that affect the performance of our algorithm, we needed to perform a statistical analysis [24] on the experimental results. More exactly, from all the parameter configurations considered

noB	qC	qE	qS	$wProc$ (%)	t_{avg} (s)	fit_{avg}	qE'	qS'	stD_{avg}
35	4	7	5	0.9	0.435	6.456	7	2	0
50	2	3	6	0.5	0.665	6.456	5	0	0
40	2	3	6	0.5	0.512	6.456	5	0	0
38	8	3	8	0.9	0.63	6.456	5	2	0
25	8	3	8	0.9	0.42	6.456	5	2	0
20	2	3	6	0.5	0.293	6.456	6	2	0
29	4	3	8	0.95	0.425	6.451	5	2	0.0035
28	2	3	8	0.96	0.304	6.445	5	2	0.0077
28	2	7	5	0.96	0.363	6.44	7	2	0.0113
20	3	3	5	0.9	0.292	6.435	5	2	0.0148
5	8	4	4	0.95	0.142	6.435	7	1	0.0148
30	6	4	3	0.8	0.552	6.431	7	3	0.0176
30	5	4	3	0.96	0.545	6.431	7	3	0.0176
80	1	2	2	0.1	0.573	6.429	3	2	0.019
5	20	4	4	0.95	0.144	6.429	7	1	0.019

Table 4. Top fifteen experimental results for scenario S when running the Hybrid HBMO algorithm

in the experiments, we have selected 40 configurations, namely the configurations of parameters for which the standard deviation from the optimal solution is lower than or equal to 0.4. For the configurations selected, we have analyzed, by using a regression model [25], which are the critical parameters that affect the execution time of our Hybrid Honey Bees Mating Optimization algorithm.

noB	qC	qE	qS	$wProc$ (%)	t_{avg} (s)	fit_{avg}	qE'	qS'	stD_{avg}
23	8	10	6	0.9	1.111	7.477	10	0	0.0035
35	4	7	5	0.9	0.866	7.474	7	2	0.0056
50	2	3	6	0.5	0.907	7.463	5	0	0.0134
38	8	3	8	0.9	0.85	7.463	5	2	0.0134
28	4	4	3	0.95	0.666	7.463	7	3	0.0134
40	2	3	6	0.5	0.864	7.462	5	0	0.0141
28	2	7	5	0.96	0.487	7.459	7	2	0.0162
28	2	3	8	0.96	0.386	7.45	5	2	0.0226
5	8	5	3	0.95	0.209	7.444	9	3	0.0268
30	5	4	3	0.96	0.761	7.444	7	3	0.0268
30	5	4	3	0.8	0.792	7.44	7	3	0.0296
30	5	4	3	0.8	1.38	7.439	7	3	0.0304
29	4	3	8	0.95	0.583	7.434	5	2	0.0339
25	8	3	8	0.9	0.617	7.433	5	2	0.0346
30	6	4	3	0.8	0.796	7.433	7	3	0.0346

Table 5. Top fifteen experimental results for scenario M when running the Hybrid HBMO algorithm

noB	*qC*	*qE*	*qS*	*wProc* (%)	t_{avg} (s)	fit_{avg}	qE'	qS'	stD_{avg}
28	2	11	5	0.96	1.85	8.023	11	2	0.0007
30	4	11	5	0.95	1.557	8.023	11	2	0.0007
23	8	10	6	0.9	1.386	8.023	10	0	0.0007
30	4	11	6	0.95	1.652	8.022	11	0	0.0014
35	4	7	5	0.9	1.25	8.021	7	2	0.0021
50	2	3	6	0.5	1.394	8.016	5	0	0.0056
40	2	3	6	0.5	1.262	8.015	5	0	0.0063
29	4	3	8	0.95	0.832	8.014	5	2	0.0070
38	8	3	8	0.9	1.33	8.013	5	2	0.0077
25	8	3	8	0.9	0.841	8.013	5	2	0.0077
28	2	7	5	0.96	0.767	8.006	7	2	0.0127
28	2	3	8	0.96	0.563	8.002	5	2	0.0155
30	6	4	3	0.8	1.86	8	7	3	0.0169
5	8	5	3	0.95	0.342	7.999	9	3	0.0176
80	1	2	2	0.1	1.382	7.998	3	2	0.0183

Table 6. Top fifteen experimental results for scenario *L* when running the Hybrid HBMO algorithm

In what follows we present, for the case of scenario L, the interpretation of the experimental results with the help of the statistical analysis.

In the first step, we have defined a factorial design, by selecting the input factors that we wanted to analyze. In our case, we wanted to analyze the impact all the adjustable parameters will have on the execution time of the the algorithm, namely

noB	*qC*	*qE*	*qS*	*wProc* (%)	t_{avg} (s)	fit_{avg}	qE'	qS'	stD_{avg}
30	4	11	5	0.95	1.205	7.564	11	2	0.0091
23	8	10	6	0.9	1.17	7.563	10	0	0.0098
30	4	11	6	0.95	1.102	7.562	11	0	0.0106
50	2	3	6	0.5	1.27	7.553	6	0	0.0169
35	4	7	5	0.9	0.914	7.55	7	2	0.0190
28	2	11	5	0.96	0.719	7.544	11	2	0.0233
40	2	3	6	0.5	0.88	7.542	5	0	0.0247
30	5	4	3	0.8	1.72	7.528	7	3	0.0346
38	8	3	8	0.9	0.902	7.524	5	2	0.0374
25	8	3	8	0.9	0.528	7.518	5	2	0.0417
28	2	7	5	0.96	0.516	7.517	7	2	0.0424
28	4	4	3	0.95	0.642	7.511	7	3	0.0466
30	6	4	3	0.8	0.79	7.509	7	3	0.0480
30	5	4	3	0.96	0.72	7.492	7	3	0.0601
29	4	3	8	0.95	0.537	7.49	5	2	0.0615

Table 7. Top fifteen experimental results for scenario *XL* when running the Hybrid HBMO algorithm

the number of bees, the queen capacity, the queen speed, the queen energy, and the percentage of the worst solutions that will be replaced with randomly generated ones. For each of the input factors, we specified the low and high values, based on the experimental results.

In the second step, after having defined the factorial design, we performed a first analysis of the factorial design. We done this by selecting the Normal Plots and Residuals versus fits graph to be generated. When generating the graph, we have also set the maximum order for the factors in the model as 2. This means that the graphs will display the main effects of the adjustable parameters as well as 2-way interactions between them.

The normal plot of the standardized effects for scenario L (see Figure 4) shows that the factors that have a significant effect on the execution time of our Hybrid Honey Bees Mating Optimization algorithm are: the number of bees A, the queen capacity B, the queen energy D, and the combination of the number of bees and the queen energy AD.

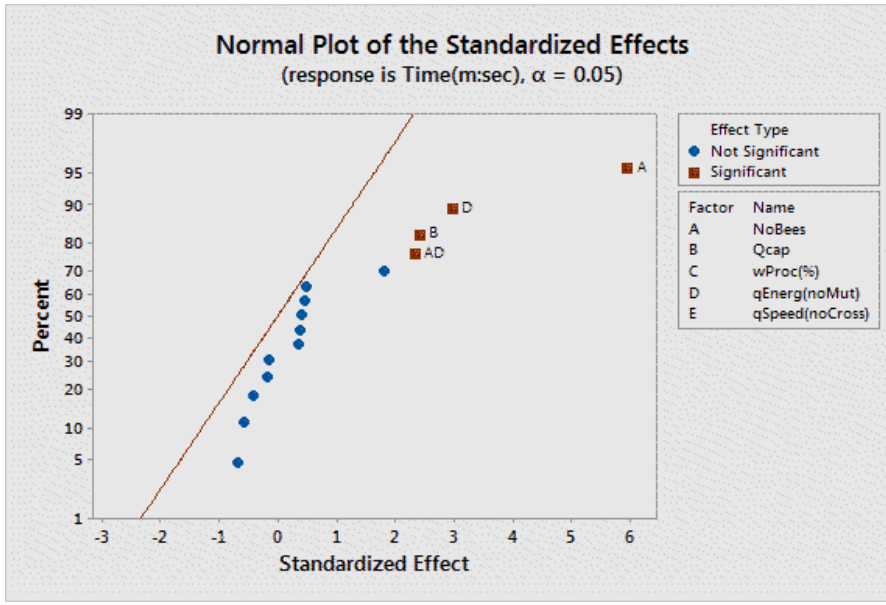

Figure 4. Normal plot of the standardized effects for our Hybrid Honey Bees Mating Optimization algorithm in the case of scenario L

Since the other terms or 2-way interactions of terms were insignificant, we dropped these terms from the model and re-analyzed the design by following steps 1 and 2 again. This time we only kept the significant factors (A, B, D, AD) in the model.

By analyzing the new model, we found that the normality and constant variance assumptions were met. Also, from the ANOVA table (see Figure 5) we notice that

three factors and two interactions have statistically significant effect on the execution time of the algorithm. The three main factors and two interactions are the ones for which P-Value is lower than 0.05.

Coded Coefficients

Term	Effect	Coef	SE Coef	T-Value	P-Value	VIF
Constant		2.108	0.164	12.81	0.000	
NoBees	3.706	1.853	0.183	10.15	0.000	7.72
Qcap	1.220	0.610	0.208	2.93	0.006	14.65
qEnerg(noMut)	1.552	0.776	0.139	5.57	0.000	8.37
NoBees*Qcap	1.118	0.559	0.214	2.61	0.014	11.65
NoBees*qEnerg(noMut)	1.245	0.622	0.166	3.74	0.001	6.14
Qcap*qEnerg(noMut)	0.007	0.003	0.134	0.03	0.979	4.33

Regression Equation in Uncoded Units

```
Time(sec) = -0.083 + 0.00390 NoBees - 0.0031 Qcap + 0.0167 qEnerg(noMut)
            + 0.001569 NoBees*Qcap + 0.00415 NoBees*qEnerg(noMut)
            + 0.00009 Qcap*qEnerg(noMut)
```

Figure 5. ANOVA table for our Hybrid Honey Bees Mating Optimization Algorithm in the case of scenario L

The ANOVA table (Figure 5) shows that all the three factors (the number of bees, the queen capacity, and the queen energy) are significant and there are significant interactions between number of bees (*noBees*) and queen capacity (*qCap*), between number of bees (*noBees*) and queen energy (*qEnerg*), as well as between queen capacity (*qCap*) and queen energy (*qEnerg*). Thus, Figures 6 and 7 show the interaction plot and the surface plot of number of bees and queen capacity, number of bees and queen energy, as well as queen capacity and queen energy.

7 PERFORMANCE EVALUATION

We have also assessed our Hybrid Honey Bees Mating Optimization algorithm by comparison with the state-of-the-art genetic-inspired algorithm for identifying the optimal solution in the composition of Web services, as proposed in [21], as well as with the classical Honey Bees Mating Optimization applied in the case of semantic Web service composition. For our algorithm, we have considered the configurations of the parameters that provide the results in the shortest execution time, for each of the four scenarios. The comparative analysis has been made on the same set of scenarios (S, M, L, and XL), as described in Table 3. Moreover, we have used the same stopping condition (namely 100 iterations) for the three algorithms under study. Table 8 presents the experimental results achieved with the three algorithms. In this table, fit_{avg} is the average optimal fitness value, $stDev_{avg}$

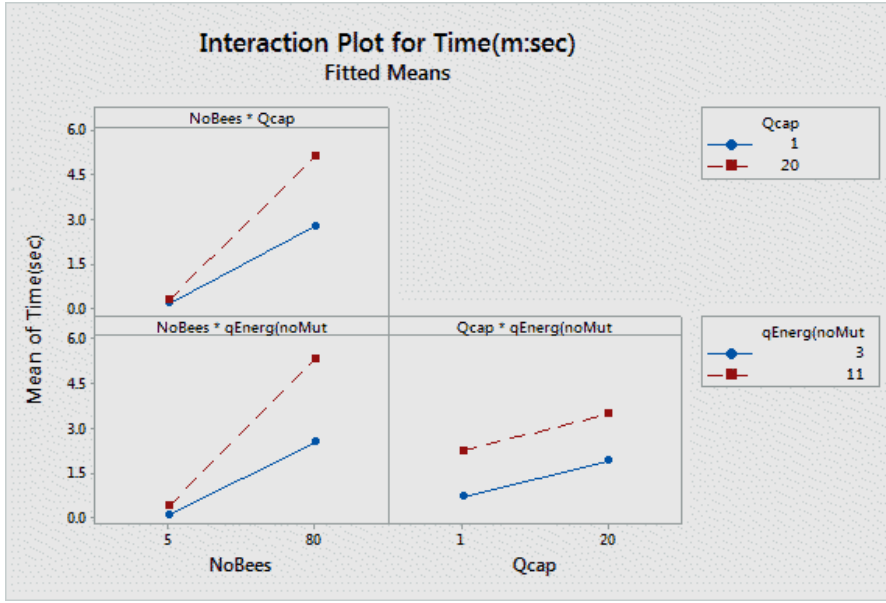

Figure 6. Interaction plots in the case of scenario L

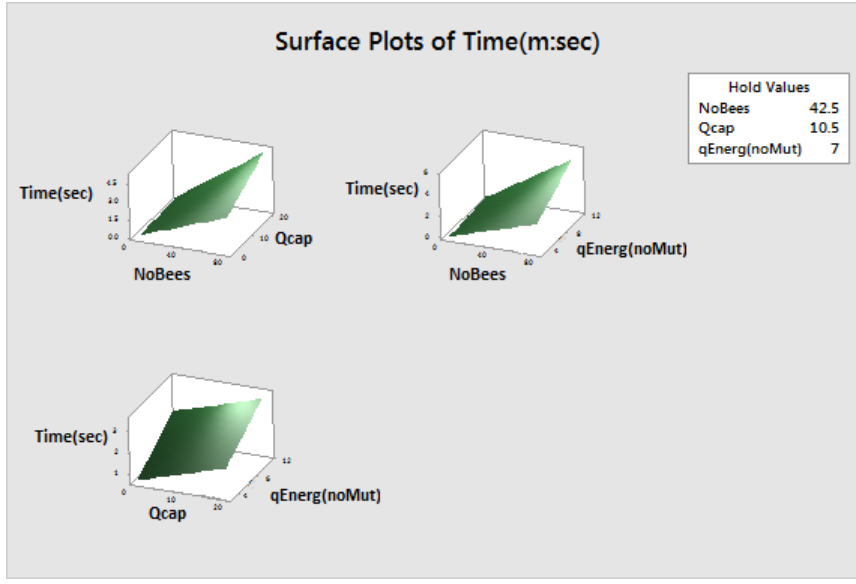

Figure 7. Surface plots in the case of scenario L

is the average standard deviation, and t_{avg} is the average execution time. All these values have been computed by running each algorithm version 100 times for each considered scenario for the optimal configuration of the adjustable parameters.

Algorithm	fit_{avg}	stD_{avg}	t_{avg} (s)
Canfora	6.056	0.2828	0.241
HBMO	6.227	0.1619	0.84
HBMOMG	6.373	0.0586	0.102

a)

Algorithm	fit_{avg}	stD_{avg}	t_{avg} (s)
Canfora	7.275	0.1463	0.340
HBMO	7.094	0.2743	0.98
HBMOMG	7.322	0.1131	0.1

b)

Algorithm	fit_{avg}	stD_{avg}	t_{avg} (s)
Canfora	7.36	0.4695	0.344
HBMO	7.896	0.0905	0.254
HBMOMG	7.921	0.0728	0.0309

c)

Algorithm	fit_{avg}	stD_{avg}	t_{avg} (s)
Canfora	7.332	0.1732	0.198
HBMO	7.219	0.2531	0.189
HBMOMG	7.403	0.123	0.148

d)

Table 8. Experimental results achieved for scenarios a) *S*, b) *M*, c) *L*, and d) *XL*

By analyzing the experimental results shown in Table 8, it can be noticed that for all the scenarios (S, M, L, and XL), our Hybrid Honey Bees Mating Optimization algorithm provides better results in terms of fitness values than both the genetic-inspired algorithm of Canfora et al. and the classical Honey Bees Mating Optimization. Also, the execution time of our algorithm in the case of scenarios S, M and L is shorter than the execution time of both the genetic-inspired algorithm of Canfora et al. and the classical Honey Bees Mating Optimization. In the case of scenario XL, even if the the execution time in the case of our algorithm is better than the execution time of the other two algorithms, the difference is insignificant. In conclusion, we can say that, in the case of scenario XL, the performance of the three algorithms is the same.

8 CONCLUSIONS

In this paper we have introduced a Hybrid Honey Bees Mating Optimization algorithm for selecting the optimal solution in semantic Web service composition. The algorithm proposed combines principles from population-based meta-heuristics with principles from trajectory-based meta-heuristics in order to improve the fitness value and to avoid the stagnation in locally optimal solutions. The hybrid selection algorithm has been evaluated on four scenarios of different complexities. We have also comparatively analyzed the experimental results provided by our hybrid selection algorithm versus the results obtained with the classical Honey Bees Mating Optimization algorithm and with the genetic-inspired algorithm of Canfora et al.

REFERENCES

[1] Blum, C.—Roli, A.: Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. ACM Computing Surveys Journal. Vol. 35, 2003, No. 3, pp. 268–308, doi: 10.1145/937503.937505.

[2] Blum, C.—Puchinger, J.—Raidl, G. R.—Roli, A.: Hybrid Metaheuristics in Combinatorial Optimization: A Survey. Applied Soft Computing, Vol. 11, 2011, No. 6, pp. 4135–415, doi: 10.1016/j.asoc.2011.02.032.

[3] Afshar, A.—Bozorg Haddad, O.—Mariño, M. A.—Adams, B. J.: Honey-Bee Mating Optimization (HBMO) Algorithm for Optimal Reservoir Operation. Journal of the Franklin Institute, Vol. 344, 2007, No. 5, pp. 452–462.

[4] Amiri, M. A.—Serajzadeh, H.: Effective Web Service Composition Using Particle Swarm Optimization Algorithm. Proceedings of the Sixth International Symposium on Telecommunications (IST 2012), Tehran, November 2012, pp. 1190–1194, doi: 10.1109/ISTEL.2012.6483169.

[5] Clerk, M.—Kennedy, J.: The Particle Swarm – Explosion, Stability, and Convergence in a Multidimensional Complex Space. IEEE Transactions on Evolutionary Computation, Vol. 6, 2002, No. 1, pp. 58–73, doi: 10.1109/4235.985692.

[6] Zhao, S.—Wang, L.—Ma, L.—Wen, Z.: An Improved Ant Colony Optimization Algorithm for QoS-Aware Dynamic Web Service Composition. Proceedings of the International Conference on Industrial Control and Electronics Engineering (ICICEE 2012), Xi'an, August 2012, pp. 1998–2001.

[7] Kang, G.—Liu, J.—Tang, M.—Xu, Y.: An Effective Dynamic Web Service Selection Strategy with Global Optimal QoS Based on Particle Swarm Optimization Algorithm. Proceedings of the 26th International Parallel and Distributed Processing Symposium Workshop (IPDPS 2012), Shanghai, May 2012, pp. 2280–2285, doi: 10.1109/IPDPSW.2012.281.

[8] Ludwig, S. A.: Clonal Selection Based Genetic Algorithm for Workflow Service Selection. Proceedings of the IEEE World Congress on Computational Intelligence (WCCI 2012), Brisbane, June 2012, pp. 1–7, doi: 10.1109/CEC.2012.6256465.

[9] Ludwig, S. A.: Applying Particle Swarm Optimization to Quality-of-Service-Driven Web Service Composition. Proceedings of the 26th International Conference on Advanced Information Networking and Applications (AINA 2012), Fukuoka, March 2012, pp. 613–620, doi: 10.1109/AINA.2012.46.

[10] Cao, J.—Sun, X.—Zheng, X.—Liu, B.—Mao, B.: Efficient Multi-Objective Services Selection Algorithm Based on Particle Swarm Optimization. Proceedings of the IEEE Asia-Pacific on Services Computing Conference (APSCC 2010), 2010, pp. 603–608.

[11] Wu, Q.—Zhu, Q.: Transactional and QoS-Aware Dynamic Service Composition Based on Ant Colony Optimization. Future Generation Computer Systems Journal, Vol. 29, 2013, No. 5, pp. 1112–1119.

[12] Zhang, C.: Adaptive Genetic Algorithm for QoS-Aware Service Selection. Workshops of International Conference on Advanced Information Networking

and Applications (AINA-2011), Singapore, March 2011, pp. 273–278, doi: 10.1109/WAINA.2011.43.

[13] Zhao, X.—Song, B.—Huang, P.—Wen, Z.—Weng, J.—Fan, Y.: An Improved Discrete Immune Optimization Algorithm Based on PSO for QoS-Driven Web Service Composition. Applied Soft Computing Journal, Vol. 12, 2012, No. 8, pp. 2208–2216.

[14] Zheng, K.—Xiong, H.: A Particle Swarm-Based Web Service Dynamic Selection Algorithm with QoS Global Optimal. Journal of Information and Computational Science, Vol. 9, 2012, No. 8, pp. 2271–2278.

[15] Zhao, X.—Wen, Z.—Li, X.: QoS-Aware Web Service Selection with Negative Selection Algorithm. Knowledge Information System, Vol. 40, 2013, No. 2, pp. 349–373.

[16] Zhao, X.—Huang, P.—Liu, T.—Li, X.: A Hybrid Clonal Selection Algorithm for Quality of Service-Aware Web Service Selection Problem. International Journal of Innovative Computing, Information and Control, Vol. 8, 2012, No. 12, pp. 8527–8544.

[17] Pop, C. B.—Chifu, V. R.—Salomie, I.—Dinsoreanu, M.: Immune-Inspired Method for Selecting the Optimal Solution in Web Service Composition. Resource Discovery, Lecture Notes in Computer Science, Vol. 6162, 2010, pp. 1–17.

[18] Russell, S.—Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall/Pearson Education, Upper Saddle River, NJ, 2003. ISBN 0137903952.

[19] Salomie, I.—Chifu, V. R.—Pop, C. B.: Hybridization of Cuckoo Search and Firefly Algorithms for Selecting the Optimal Solution in Semantic Web Service Composition. In: Yang, X. S. (Ed.): Book Chapter in Cuckoo Search and Firefly Algorithm: Theory and Applications, Vol. 516, 2014, pp. 217–243.

[20] Glover, F.—Laguna, M.: Tabu Search. Kluwer Academic Publishers, Norwell, MA, 1997, doi: 10.1007/978-1-4615-6089-0.

[21] Canfora, F.—di Penta, M.—Esposito, R.—Villani, M. L.: An Approach for QoS-Aware Service Composition Based on Genetic Algorithms. Proceedings of the 2005 Conference on Genetic and Evolutionary Computation (GECO '05), Washington, June 2005, pp. 1069–1075, doi: 10.1145/1068009.1068189.

[22] Xu, J.—Reiff-Marganiec, S.: Towards Heuristic Web Services Composition Using Immune Algorithm. Proceedings of the IEEE International Conference on Web Services (SCC '08), Honolulu, July 2008, pp. 238–245, doi: 10.1109/ICWS.2008.16.

[23] Chifu, V. R.—Salomie, I.—Pop, C. B.—Niculici, A.—Suia, S.: Exploring the Selection of the Optimal Web Service Composition Through Ant Colony Optimization. Computing and Informatics, Vol. 33, 2014, No. 5, pp. 1047–1064.

[24] Derrac, J.—García, S.—Molina, D.—Herrera, F.: A Practical Tutorial on the Use of Nonparametric Statistical Tests as a Methodology for Comparing Evolutionary and Swarm Intelligence Algorithms. Swarm and Evolutionary Computation Journal, Vol. 1, 2011, No. 1, pp. 3–18, doi: 10.1016/j.swevo.2011.02.002.

[25] `https://www.minitab.com/en-us/products/minitab/`.

Viorica Rozina Chifu received her Ph.D. in computer science from the Technical University of Cluj-Napoca in 2010. Currently, she is Associate Professor at the Technical University of Cluj-Napoca, Romania. Her research interests include parallel and distributed systems, big data techniques, computational intelligence, machine learning and data mining, Web service composition. She has published 69 research papers in refereed international journals or conference proceedings. The recognition of her professional merits by scientific community derives from the large number of citations in journals indexed in ISI or in other international databases and in the proceedings of international conferences.

Cristina Bianca Pop received her Ph.D. degree in computer science from the Technical University of Cluj-Napoca in 2013. Currently, she is Senior Lecturer at the Technical University of Cluj-Napoca, Romania. Her research interests include biologically-inspired distributed systems, ontologies and semantic Web, automatic Web service composition.

Ioan Salomie is currently Professor of computer science at the Technical University of Cluj-Napoca, Romania, being in the past years Invited Professor at Loyola College in Maryland (1996) and University of Limerick (2000–2004). His research interests focus on service oriented distributed computing, context awareness and autonomic computing, bio-inspired computing and intelligent systems. He is the Head of Distributed Systems Research Laboratory, which is an active partner in relevant national and EU projects.

Emil Stefan Chifu received his Ph.D. in computer science from the Technical University of Cluj-Napoca in 2010. Currently, he is Associate Professor at the Technical University of Cluj-Napoca, Romania. His research interests include opinion mining from text, AI planning, web mining with self-organizing maps, and text-based ontology learning.