A Bee-inspired Approach for Selecting the Optimal Service Composition Solution

Cristina Bianca POP, Viorica Rozina CHIFU, Ioan SALOMIE, Mihaela DINSOREANU, Mihaly FODOR, Irina CONDOR

Department of Computer Science, Technical University of Cluj-Napoca
26-28 Baritiu Street, Cluj-Napoca, Romania

Abstract — This paper presents a bee colony optimization method for selecting the optimal solution in semantic Web service composition. The bee-inspired selection method uses an enhanced planning graph model and a matrix of semantic links to incrementally search the optimal solution. We use a multi-criteria function which evaluates whether a solution is optimal or not in terms of its QoS attributes and the quality of the semantic match between the services involved in the solution. The selection method was validated by making experiments on a set of semantic Web services from the trip planning domain.

Index Terms — bee colony optimization, enhanced planning graph, ontology, semantic Web service, service composition

I. INTRODUCTION

Web Services have grown in importance for the past years because they provide business functionality to other applications using Internet connections in a modular and self-contained way. In the real world, a single Web-service is rarely used as the answer to a complex request and so, more services need to work together in order to accomplish the task. Manually discovering and composing Web services is a tedious work, if the growing number of available Web services and their diversity in terms of functionality or Quality of Service (QoS) is taken into account. Web Services can be described at a syntactic and at a semantic level. In the case of the syntactic description, the WSDL (interface-based description of a Web service) and SOAP (XML-based formalization of Web services inter-communication) standards are used. In the semantic Web view, Web services are described using ontology concepts which annotate the service inputs, outputs, preconditions and effects. As a result, computers can understand not only the syntax of Web services, but also their content and meaning thus favoring an automatic process of service discovery and composition. In case the number of services involved in composition is large, a lot of composition solutions may be obtained and the search for the optimal solution can be seen as an optimization process.

This paper presents a new technique for automatic Web service composition inspired by the behavior of bees. The proposed technique uses an enhanced AI planning graph model combined with a bee-inspired optimization algorithm to find the composition solution which satisfies the user request. The user request is described in terms of functional and non-functional requirements. The functional requirements are expressed using ontological concepts that annotate the provided inputs and requested outputs. The non-functional requirements represent weights associated to user preferences regarding the relevance of a solution’s semantic quality compared to its QoS attributes. The bee-inspired optimization algorithm is used to select the optimal composition solution according to QoS attributes and semantic quality.

The paper is organized as follows. Section II presents the proposed Web service composition method, while section III details the bee-inspired selection method for identifying the optimal composition solution. Experimental results are presented in section IV. Section V surveys the related work of bio-inspired service composition methods. We end our paper with conclusions and future work proposals.

II. THE WEB SERVICE COMPOSITION METHOD

To obtain the optimal Web service composition solution we combine a bee-inspired selection method with an Enhanced Planning Graph (EPG) model and a Matrix of Semantic Links (MSL). In this section we briefly present the composition approach that we have proposed in [6] which aims at building the EPG and MSL models.

A. The Enhanced Planning Graph Model

We obtained the EPG model by mapping the classical AI planning graph problem [7] to semantic Web service composition (see Table 1) and by adding new structures.

<table>
<thead>
<tr>
<th>TABLE 1: CONCEPTS FROM MAPPING THE AI PLANNING GRAPH PROBLEM TO SEMANTIC WEB SERVICE COMPOSITION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AI planning graph concepts</strong></td>
</tr>
<tr>
<td>Action</td>
</tr>
<tr>
<td>Precondition</td>
</tr>
<tr>
<td>Effect</td>
</tr>
<tr>
<td>Initial state</td>
</tr>
<tr>
<td>Goal state</td>
</tr>
</tbody>
</table>

The EPG construction is an iterative process. In each iteration, a new layer consisting of a tuple (A, L) is added to the graph where A represents a set of service clusters and L is a set of clusters of service output parameters. Layer 0 consists of a tuple (A₀, L₀) where A₀ is an empty set of services (actions) and L₀ contains the input parameters of the
user request. For each layer \( i > 0 \), \( A_i \) consists of a set of clusters of services for which the input parameters are literals from \( L_{i-1} \). A cluster of services groups services which provide the same functionality. The functionalities of the services belonging to the same cluster are annotated only with is-a related ontological concepts. The services which contribute in each step to the extension of the EPG are provided by a discovery process. The discovery process finds the appropriate Web services in a repository of services, based on the semantic matching between the services’ inputs and the set of literals of the previous graph layer. The \( L_i \) set is constructed as a union of the \( L_{i-1} \) set and the sets of outputs of the services in \( A_i \). Parameters of \( L_i \) (inputs and outputs) are grouped in clusters of literals where each cluster contains only is-a related ontological concepts. The construction of the EPG ends either when the user requested outputs are contained in the current set of literals or when the graph reaches a fixed point. Reaching a fixed point means that the sets of actions and literals are the same for the last two consecutive generated layers. In figure 1 we present the step-by-step construction of an EPG.

![Figure 1. An example of step-by-step construction of the EPG.](image)

In the figure, a service is represented with a rectangle and a literal with an oval. The services as well as the literals are grouped in clusters. The clusters are represented as rectangles with rounded corners. An example of a cluster of literals would be \( \{E, EE, EEE\} \) in \( L_1 \) which is linked to the second cluster of \( A_1 \) services from the first layer. The link indicates that the \( s_2, s_3 \) and \( s_4 \) services in this cluster, have \( E, EE \) and, \( EEE \) as outputs. A composition solution for the EPG of figure 1 consists of the following subsets of services: \( \{[s_1, s_3], [s_2, s_4]\} \). In each solution subset, only one service of its cluster can be considered.

B. The Matrix of Semantic Links

The Matrix of Semantic Links (MSL) is built in parallel with the construction of the EPG. The matrix stores the semantic links established between services on different layers in the EPG. One Web service could be linked to several other Web services. We say that there is a semantic link between two services \( s_i \) and \( s_j \) if there is a Degree of Match (DoM) [8] between the set of the output parameters of service \( s_i \) and the set of the input parameters of service \( s_j \).

When computing the DoM between an output concept of a service and an input concept of another service we consider three types of semantic matching: EXACT, PLUG IN, and SUBSUME [8].

In MSL, both the columns and the rows are labeled with services from the EPG. A column and a row having the same index will be labeled with the same service. MSL is formally defined as:

\[
MSL = [m_{ij}], i=1...n, j=1...m
\]

where an element \( m_{ij} \) of MSL is represented as follows:

\[
m_{ij} = \begin{cases} 
\phi, & \text{if } simS(s_i.out_k, s_j.in_l) = 0 \\
s_{ij}, & \text{otherwise}
\end{cases}
\]

In (2), \( s_{ij} \) represents the semantic similarity link between the service on row \( i \) and the service on column \( j \). A semantic similarity link is formally defined as a tuple

\[
s_{ij} = (V, simS)
\]

where:

- \( V \) is a set of pairs \( (s_i.out_k, s_j.in_l) \) where \( s_i.out_k \) is an output parameter of service \( s_i \) and \( s_j.in_l \) is an input parameter of service \( s_j \), and the degree of match between \( s_i.out_k \) and \( s_j.in_l \) is greater than 0.
- \( simS \) is the semantic similarity score computed between the subset of output parameters of service \( s_i \) and the subset of input parameters of service \( s_j \).

We compute \( simS \) between the subset out of output parameters of service \( s_i \) and the subset in of input parameters of service \( s_j \) with the following formula:

\[
simS(s_i.out_k, s_j.in_k) = \frac{\sum_{i=1}^{m} F_i - MeasureF(s_i.out_k, s_j.in_k)}{m}
\]

\( F \) is an adapted version of the similarity measure from information retrieval [12] and is computed as:

\[
F_i - Measure(s_i.out_k, s_j.in_k) = \frac{2 \cdot prc_g(s_i.out_k, s_j.in_k) \cdot rec_g(s_i.out_k, s_j.in_k)}{prc_g(s_i.out_k, s_j.in_k) + rec_g(s_i.out_k, s_j.in_k)}
\]

where:

- \( prc_g \) is the precision between an output parameter of service \( s_i \) and an input parameter of service \( s_j \) and is computed with the formula:

\[
prc_g(s_i.out_k, s_j.in_k) = \sqrt{prc_i(s_i.out_k, s_j.in_k) \cdot prc_j(s_i.out_k, s_j.in_k)}
\]

- \( rec_g \) is the recall between an output parameter of service \( s_i \) and an input parameter of service \( s_j \) and is computed with the formula:

\[
rec_g(s_i.out_k, s_j.in_k) = \sqrt{rec_i(s_i.out_k, s_j.in_k) \cdot rec_j(s_i.out_k, s_j.in_k)}
\]

In formulas (6) and (7), \( prc_i \) and \( rec_i \) represent the precision and recall between an output of a service and an input of another service, while \( prc_j \) and \( rec_j \) represent the global precision and recall between all the properties of a service output and of another service input. The precision
and recall have been evaluated according to the formulas presented in [12].

In the case of the EPG in figure 1, a semantic similarity link between services \( s_2 \) and \( s_7 \) is established because there is a degree of match greater than 0 between the output of service \( s_2 \) and an input of service \( s_7 \).

III. THE BEE-INSPIRED SELECTION METHOD

For finding the optimal composition we adapted and enhanced the Bee Colony Optimization Metaheuristic defined in [9]. As any metaheuristic, the one proposed in [9] defines a general-purpose algorithmic framework that can be adapted to solve different optimization problems [10]. Our bee-inspired selection method uses the EPG and MSL generated by the composition method (see section II) as well as a multi-criteria function (fitness function) to find the optimal composition solution. The fitness function considers the QoS user preferences and semantic quality in the evaluation of a composition solution.

A. The Foraging Behavior of Bees

Bees represent complex collective systems in which individuals follow simple rules to build hives and search for food. In the case of nectar gathering, scout bees are sent to find the most satisfying food sources. They randomly fly on varying distances in search of food. Upon returning to the hive, they have three possibilities: to abandon their service because it was not good enough, to continue to collect nectar from the current discovered source or to communicate to the other bees what they have found so far and thus recruiting followers [9]. This type of communication is achieved through a waggle dance. Each returning bee performs this dance several times in front of the other bees in the hive. The dance encodes information about the distance, the direction and the quality of the food source. The bees in the hive can decide to follow one of the waggle dancing bees. Through this mechanism, the flower patches closer to the nest and with more quality nectar are visited by more bees enabling an efficient pollen harvesting.

B. Mapping the Foraging Behavior of Bees to the Web Service Composition Problem

We mapped the concepts from the foraging behavior of bees to the concepts of the Web service composition problem as illustrated in table II.

<table>
<thead>
<tr>
<th>Concepts from the foraging behavior of bees</th>
<th>Web service composition concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bee</td>
<td>Artificial bee</td>
</tr>
<tr>
<td>Hive</td>
<td>The layer ((L_0, L_1)) from the EPG</td>
</tr>
<tr>
<td>Food source</td>
<td>A Web service composition solution</td>
</tr>
<tr>
<td>Quality of the food source</td>
<td>Information about the services and quality of a composition solution</td>
</tr>
<tr>
<td>Waggle dance</td>
<td>Artificial bee that continues expanding its Web service composition solution</td>
</tr>
<tr>
<td>Scout bee</td>
<td>Artificial bee which abandons its service composition solution and adopts the solution of a scout bee</td>
</tr>
<tr>
<td>Follower bee</td>
<td></td>
</tr>
</tbody>
</table>

In the Web service composition problem, we consider that the biological bee becomes an artificial bee [9] which explores the EPG in search of the optimal composition solution (the best quality food source). We define an artificial bee as follows:

\[
bee = (bSol, loyalty) \tag{8}
\]

where

- \( bSol \) is a set of \( n \) elements \( bSolElem \), where a \( bSolElem \), is defined as a set of services \( \{s_{kl}^i | s_{kl}^i \text{ is the } i\text{-th service from cluster } k \text{ on layer } i \text{ of the EPG} \} \) and \( n \) is the total number of layers of the EPG;
- \( loyalty \) is the quality of the \( bSol \) (similar to the quality of a discovered food source) in terms of QoS attributes and semantic quality.

To evaluate the loyalty we define and use the QF function below:

\[
QF(bSol) = \frac{w_{QoS} \cdot QoS(bSol) + w_{Sem} \cdot Sem(bSol)}{w_{QoS} + w_{Sem}} \tag{9}
\]

where

- \( QoS(bSol) \) is the QoS score of the Web service composition solution, \( bSol \).
- \( Sem(bSol) \) is the semantic quality score of the Web service composition solution \( bSol \).
- \( w_{QoS} \) and \( w_{Sem} \) are the weights which illustrate the user preference related to the relevance of QoS and semantic quality during the evaluation of a composition solution.

The QoS score of a composition solution \( bSol \) is computed using formula 10:

\[
QoS(bSol) = \frac{\sum_{i=1}^{n} \omega_i \cdot qos_i(bSol)}{\sum_{i=1}^{n} \omega_i} \tag{10}
\]

where

- \( qos_i(bSol) \) represents the value of a QoS attribute computed for the composition solution \( bSol \).
- \( \omega_i \) is the weight associated to the relevance of the \( qos_i \) attribute.
- \( n \) is the total number of QoS attributes considered.

The semantic quality score of a composition solution \( bSol \) is computed as follows:

\[
Sem(bSol) = \frac{\sum_{i=1}^{n} \text{simS}(s_{kl}^i, s_{qr}^p)}{n - 1} \tag{11}
\]

where

- \( s_{kl}^i \) is the service \( l \) in cluster \( k \) from layer \( j \);
- \( s_{qr}^p \) is the service \( r \) in cluster \( q \) from layer \( p \);
- \( s_{kl}^i, s_{qr}^p \) are part of the solution \( bSol, j < p \);
- \( n \) is the total number of services involved in the solution \( bSol \);
- \( \text{simS} \) computes the degree of match between \( s_{kl}^i \) and \( s_{qr}^p \) (see formula 4).

In Web service composition, scout bees are mapped to artificial bees that have identified a good solution and that will continue expanding it. On the other hand, follower bees are mapped to artificial non-loyal bees which have identified...
a poor solution and will adopt the solutions of waggle bees which they will expand.

C. The BEE-INSPIRED SELECTION ALGORITHM

The bee-inspired selection algorithm (ALGORITHM_1) determines a set SOL of high quality composition solutions evaluated according to the function QF (see formula 9), the first solution being the optimal one, by considering the following: (i) the EPG and MSL structures resulted from Web service composition, (ii) the weights \( w_{QoS} \) and \( w_{Sem} \) which state the relevance of a solution’s QoS quality compared to its semantic quality, and (iii) a number \( n_{Bees} \) of artificial bees used in the search for the optimal solution.

\[ QF = \sum_{i} w_{QoS} \cdot Q_{QoS}^i + \sum_{j} w_{Sem} \cdot Q_{Sem}^j \]

\[ Q_{QoS}^i = \frac{1}{n} \sum_{k=1}^{n} \text{QoS}_k \]

\[ Q_{Sem}^j = \frac{1}{n} \sum_{k=1}^{n} \text{Sem}_k \]

\( n \) is the number of services in the composition.

Before performing the actual search, an initial solution is identified using a greedy approach (GET-INITIAL-SOLUTION). The selection algorithm consists of a number of stages equal to the number of layers (GET-NR-OF-LAYERS) in the EPG. Within each stage, the artificial bee performs a forward (FORWARD-PASS) and a backward pass (BACKWARD-PASS) aiming at identifying new candidate composition solutions. At the end of an iteration each artificial bee obtains a solution. The set of solutions obtained within an iteration is evaluated according to the QF function (see formula 9). Then, the best solution identified (GET-BEST-SOLUTION) becomes the recommended one which will guide artificial bees in taking decisions when exploring the EPG in the next iteration. During the search process, the artificial bees communicate directly by exchanging information about the quality of the partial solutions they have built so far.

Within the forward pass (ALGORITHM_2), each bee selects a new service (SELECT-NEXT-SERVICE) from each cluster of the currently explored EPG layer and adds the selected services to the candidate solution.

\[ \text{SOL} = \{ \text{service}_1, \ldots, \text{service}_n \} \]

services defined as a set \( \{(s, \text{quality}) | s \in \text{EPG}\} \)
The candidate services are further sorted according to their probability to be chosen (SORT-BY-PROBABILITY). The service that will be added to the partial solution of the considered bee is selected according to the service probability and a destiny value. The destiny value is introduced to avoid the algorithm stagnancy in a local optimum and to add diversity (ADD-SERVICE).

Within the backward pass (ALGORITHM 4), all artificial bees return in the origin of the EPG where they have to make two decisions: abandon the current partial solution or expand the partial solution without recruiting new artificial bees.

for \( j = 0 \) to \( wBees \).size do
begin
\( lSum = lSum + wBees[j].loyalty \)
end
for \( j = 0 \) to \( wBees \).size do
begin
\( wBees[j].loyalty = wBees[j].loyalty / lSum \)
\( wBee = GET-FIRST(uBees); uBees = uBees - wBee \)
\( uBee.bSol = wBees[j].bSol \)
end
end
return \( BEES \)

The first decision is influenced by the loyalty of an artificial bee towards the solution (the loyalty representing the quality of the partial solution evaluated with the \( QF \) function). By interpreting the loyalty according to some pre-established thresholds (\( \Delta_1 \) and \( \Delta_2 \), the bee will either expand its solution (the bee becomes waggle) or abandon it (the bee becomes a follower). The second decision which determines whether the partial solution will be further expanded without recruiting new hive mates is also based on the loyalty interpretation. We modeled the recruiting activity as a simple even distribution of non-loyal bees between the waggle ones.

IV. EXPERIMENTAL RESULTS

We have experimented and evaluated our bee-inspired selection method on a scenario from the trip planning domain. We developed and used in our experiments a set of 110 semantic Web services. The set of services was annotated according to the SAWSDL [11] specification. To semantically describe the user requests and the service capabilities we developed a domain ontology which stores 200 concepts organized on 8 hierarchic levels. As a result, the user request is specified by a set of ontological concepts annotating the provided inputs and requested outputs and by weights indicating the relevance of quality of service compared to semantic quality. Services are semantically described with concepts annotating their functionality, input and output parameters. In this section we partially trace the composition and selection algorithms for the user request given in table III and evaluate the experimental results. The user request specifies the need for a composed service able to make travel arrangements (search and book accommodation, flight and car). In our experiments we have considered four QoS attributes: availability (Av), reliability (Rel), cost (Ct) and response time (Rt).

<table>
<thead>
<tr>
<th>TABLE III USER REQUEST</th>
<th>( in_1, in_2, \ldots )</th>
<th>( out_1, out_2, \ldots )</th>
<th>QoS weights</th>
<th>SemQ weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>SourceCity, DestinationCity; StartDate; EndDate; Hotel; MediterraneanFood; NumberOfPersons, NumberOfRooms</td>
<td>Vacation</td>
<td>Total QoS: 0.75; Av: 0.15; Rel: 0.25; Ct: 0.45; Rt: 0.15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Price</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
EPG along with the MSL. The resulting EPG contains four layers of service clusters. Tables IV and V illustrate two service clusters, one from the second layer which groups services that search accommodation and the other one from the third layer which groups services that book accommodation.

By analyzing the experimental results we noticed that if the number of bees used in the search process is high, the optimal solution is obtained in less iterations but with a higher execution time.

V. RELATED WORK

In the past few years, biologic behaviors, such as foraging in the case of insects or birds, have inspired several optimization techniques. This section reviews biologically-inspired techniques applied to the problem of selecting the optimal service composition solution.

In [1] and [2], 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 Web services with different associated QoS values. 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 [1] and as a binary string in [2], 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. In the case of genetic operators, both approaches perform random mutations to generate new workflow instances. In the case of fitness functions, [1] proposes the use of a dynamic fitness function which penalizes the individuals that do not meet the constraints in each generation, thus favoring a quicker convergence toward an optimal solution. Compared to [1], the approach in [2] proposes three fitness functions, one associated to each considered QoS attribute, in order to increase the probability of finding the optimal solution. Still, genetic algorithms are not scalable due to the fixed way in which the operations are performed on the genome. This problem is solved in [2] by introducing a dynamic fitness function which allows for a more flexible approach to the optimization problem. The resulting composition solution is obtained in a lower number of iterations but with a higher execution time.

Table VI presents a subset of the semantic similarity links stored in the MSL established between the services from tables IV and V.

For the trip planning scenario we performed two experiments. In the first experiment we generated the entire set of solutions to identify the optimal composition solution which has a score of 9.556. Then, we performed several experiments on our bee-inspired selection technique to check whether the optimal solution identified is indeed the best or is very close to the best. In our experiments we varied the number of bees, from 10 to 150. We noticed that for a stopping condition threshold of 9.5, and for the following parameter values: $\Delta_1 = 0.5$, $\Delta_2 = 0.2$, $w_{QoS} = 0.25$, $w_{Sem} = 0.75$, the selection algorithm identifies the optimal solution (for the most of the cases) or identifies a solution very close to the optimal one. Table VII presents the number of iterations in which the optimal solution or a solution close to the optimal one was identified by different numbers of bees.

<table>
<thead>
<tr>
<th>Number of bees</th>
<th>Number of iterations</th>
<th>Optimal solution score</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>52</td>
<td>9.556</td>
</tr>
<tr>
<td>10</td>
<td>75</td>
<td>9.556</td>
</tr>
<tr>
<td>50</td>
<td>11</td>
<td>9.556</td>
</tr>
<tr>
<td>70</td>
<td>6</td>
<td>9.549</td>
</tr>
<tr>
<td>80</td>
<td>18</td>
<td>9.556</td>
</tr>
<tr>
<td>90</td>
<td>10</td>
<td>9.556</td>
</tr>
<tr>
<td>100</td>
<td>11</td>
<td>9.556</td>
</tr>
<tr>
<td>110</td>
<td>12</td>
<td>9.556</td>
</tr>
<tr>
<td>140</td>
<td>7</td>
<td>9.556</td>
</tr>
<tr>
<td>150</td>
<td>5</td>
<td>9.549</td>
</tr>
</tbody>
</table>

By analyzing the experimental results we noticed that if the number of bees used in the search process is high, the optimal solution is obtained in less iterations but with a higher execution time.
genomes and chromosomes need to be encoded. Moreover, there is the risk that the genetic algorithm may get stuck in a local optimum.

Particle Swarm Optimization (PSO) [3], an optimization technique inspired by the behavior of foraging birds is a viable alternative to genetic algorithms since it converges more rapidly and is more scalable. In [4], PSO is used to obtain the optimal service composition solution. As in [1] and [2], the composition approach is based on a predefined workflow associated to concrete candidate services. The position of the particle is represented by the concrete services mapped on workflow tasks, while velocity indicates how the concrete services should be changed. By performing the operations of addition, subtraction and multiplication (adapted to the case of Web service composition) on positions and velocities, the proposed selection algorithm identifies optimal composition solutions based on QoS attributes. However, similar with genetic algorithms, by applying PSO when searching for the optimal composition solution, the problem of early stagnation in a local optimum cannot be avoided. To address this issue, the authors have introduced a variation operator to enlarge the searching space.

In [5], genetic algorithms and PSO are combined for service composition aiming at balancing the local and the global search and thus avoiding early stagnation. The mapping of service composition to PSO is similar to the one in [4], but in this case the crossover operation is used to add diversity by introducing new services and favoring the exploration of new search spaces.

Our approach to selecting the optimal composition solution is inspired by the foraging behavior of honey-bees. The differences between our approach and the ones presented above are the following:

- Our composition method does not start from a predefined workflow as in [1], [2], [4] and [5] but from the user request. As a result, a multi-layered planning graph of services is obtained, each layer containing sets of services providing different functionalities.
- In our approach, the problem of local stagnancy in selecting the optimal composition solution encountered in [1], [2] and [4] is overcome by using a bee-inspired technique which favors the exploration of several candidates for the optimal solution.
- Our criteria for evaluating the quality of a candidate solution include not only QoS attributes as in [1], [2], [4] and [5] but also the property of semantic quality.
- In our case, the replacement of a concrete candidate service is not done randomly as in [1], [2] and [4], but according to QoS attributes and semantic quality.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we have proposed a new method for selecting the optimal composition solution which uses a technique inspired by the behavior of bees in nature. The bee-inspired technique optimizes the selection process without considering the entire search space and avoids the local optimum stagnancy problem. The selection algorithm uses two structures, namely an enhanced planning graph and a matrix of semantic links, resulted from Web service composition, to incrementally build better solutions until an acceptable one is found taking into consideration the QoS and semantic quality. The enhanced planning graph and the matrix of semantic links store the set of services that can be composed as well as information about their degree of match. We obtained promising results in testing our bee-inspired selection method on a case study for making travel arrangements involving 110 semantic Web services.

As future work we intend to test the selection method on larger and more complex sets of Web services and to speed-up the process by parallelizing the bees search operations.

REFERENCES