ISBN 978-83-60810-22-4

Hybrid Immune-inspired Method for Selecting the Optimal or a Near-Optimal Service Composition

Ioan Salomie, Monica Vlad, Viorica Rozina Chifu, Cristina Bianca Pop
Department of Computer Science
Technical University of Cluj-Napoca
26-28 Baritiu str., Cluj-Napoca, Romania
Email: {Ioan.Salomie, Viorica.Chifu, Cristina.Pop}@cs.utcluj.ro

***Abstract*—The increasing interest in developing optimization techniques that provide the optimal or a near-optimal solution of a problem in an efficient way has determined researchers to turn their attention towards biology. It has been noticed that biology offers many clues regarding the design of such optimization techniques, since biological systems exhibit self-optimization and self-organization capabilities in a decentralized way without the existence of a central coordinator. In this context we propose a bio-inspired hybrid method that selects the optimal or a near-optimal solution in semantic Web service composition. The proposed method combines principles from immune-inspired, evolutionary, and neural computing to optimize the selection process in terms of execution time and explored search space. We model the search space as an Enhanced Planning Graph structure which encodes all the possible composition solutions for a given user request. To establish whether a solution is optimal, the *QoS* attributes of the services involved in the composition as well as the semantic similarity between them are considered as evaluation criteria. For the evaluation of the proposed selection method we have implemented an experimental prototype and carried out experiments on a set of scenarios from the trip planning domain.**

I. Introduction

WEB services are software components exposing atomic functionalities over the Internet that are often composed to address complex user requests. As there might be more than one service offering the same functionality the problem of manually composing Web services is unfeasible and automatic strategies need to be considered. The composition process focuses mainly on finding the appropriate services that composed satisfy the user functional requirements without considering the non-functional ones. These non-functional requirements are taken into consideration in the process of selecting the optimal composition of services. The large number of available services providing the same functionality leads to many composition solutions and the choice of the best one according to the functional and non-functional requirements becomes an optimization problem. Such an optimization problem requires appropriate selection strategies which provide the optimal or a near-optimal solution in a short time and without processing the entire search space. Generally, selection strategies fall into the category of exhaustive strategies or approximate ones. In the context of selecting the optimal Web service composition solution the most appropriate from the practical point of view are the approximate ones. These methods guarantee to find an optimal or a near optimal solution in a time efficient manner. Meta-heuristics have emerged as a promissing type of approximate algorithms as they represent algorithmic frameworks defining general-purpose concepts and strategies that can be easily adapted and used to solve various combinatorial optimization problems [3], such as Web service composition.

This paper presents how principles inspired from immune and evolutionary systems can be combined with reinforcement learning to design a hybrid method for selecting the optimal or a near-optimal composition solution. The search space for the 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. The functional requirements are expressed using ontological concepts that semantically describe the inputs and outputs of the requested composed service. The non-functional requirements represent weights associated to user preferences regarding the relevance of a composition solutions semantic quality and its QoS attributes. To identify the optimal composition solution encoded in the EPG, we define a fitness function which uses as evaluation criteria both the QoS attributes and the semantic quality of the services involved in composition as opposed to other research approaches which focus only on QoS. The proposed selection method has been tested and validated on a set of scenarios from the trip planning domain.

The paper is structured as follows. Section II presents related work. In Section III we overview the formal model for representing the search space of our hybrid selection method. Section IV introduces the hybrid selection method, while its evaluation is discussed and analyzed in Section V. We end our paper with conclusions and future work proposals.

II. Related Work

This section reviews bio-inspired methods for selecting the optimal service composition solution available in the research literature.

In [1] authors present a genetic-based algorithm for binding concrete services to an abstract composition workflow. A genetic chromosome is mapped to a service composition solution encoded using an integer array representation. Each composition solution is evaluated using a fitness function that considers only the QoS attributes as binding criteria. The crossover and mutation operators are used to ensure the evolution of the available service composition solutions towards the optimal or a near-optimal one. When applying the crossover and mutation operators authors are guided only by the dependency constraints between the services, which means that if a concrete service is assigned to an abstract task then all the abstract tasks that depend on the first one must have assigned only services from the same endpoint address [1]. The crossover and mutation points are chosen randomly. To select the service composition solutions that will be part of the next population, a roulette wheel selection operator is used. Authors introduce a re-binding algorithm used at runtime to re-evaluate the QoS score of a composition solution and if it is below a specified threhold then the services affecting the QoS score are replaced with other similar services which do not violate the dependency constraints.

A genetic-based algorithm for selecting the optimal or a near-optimal solution in multi-path Web service composition is proposed in [6]. A genetic chromosome is mapped to a service composition solution and each chromosome unit is represented as a triple containing the task context, workflow sign (specifies if the task is part of an AND/OR workflow) and the pointer towards the candidate service [6]. Initially, a set of service composition solutions having different topologies (corresponding to different paths) are randomly generated. Then the actual selection process is performed by iteratively (i) selecting the best solutions based on QoS attributes, (ii) applying a crossover operator between solutions with different topologies, (iii) and randomly mutating a randomly chosen service composition solution.

A hybrid method combining Particle Swarm Optimization (PSO) [7] with Simmulated Annealing is proposed in [4] for selecting the optimal or a near-optimal service composition solution based on QoS attributes. Authors model service composition using an abstract workflow on which concrete services are mapped. A composition solution is considered as the position of a particle in PSO, while velocity is used to modify a composition solution. To avoid the problem of premature stagnation in a local optimal solution, a Simmulated Annealing-based strategy is introduced which produces new composition solutions by randomly perturbing an initial solution.

Another hybrid method based on PSO is presented in [10] which introduces a non-uniform mutation strategy that aims to modify the global best optimal composition solution for ensuring diversity - i.e. the exploration of new areas of the search space. In addition, to improve the convergence speed the authors use an adaptive weight strategy for adjusting the particle velocity. A local best first strategy is used to replace the services having a low QoS score from a composition solution with others having a higher QoS score.

In [11], authors apply the Ant Colony Optimization (ACO) meta-heuristic to select the optimal or a near-optimal service composition solution based on QoS attributes. The service composition problem is modeled as an abstract workflow on which concrete services are mapped similar to [1] and [4]. The resulting composition graph represents the search space for the proposed ACO-based selection algorithm. To identify the optimal values of the adjustable parameters introduced by the ACO-based selection algorithm, authors employ a genetic-based strategy.

The differences between our approach and the ones presented above are the following: (i) our composition graph is generated dynamically based on a user request and it does not start from a predefined workflow, (ii) our criteria for evaluating the quality of a candidate composition solution includes not only QoS attributes but also the property of semantic quality between the services involved in the composition solution, (iii) the replacement of a concrete candidate service is not done randomly, but according to QoS attributes and semantic quality. The chances of randomly choosing a certain service as a replacement are directly proportional to the quality of the solution obtained by using the respective service.

III. Formal Model for Representing Semantic Web Service Composition

The Web service composition has been modeled using an *Enhanced Planning Graph* (EPG) structure which we described in [8]. The EPG model is obtained by mapping the classical AI planning graph problem [9] to semantic Web service composition and also by enhancing the mapped concepts with the new abstractions of service cluster and parameter cluster. A service cluster 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. For simplicity, we consider that a Web service has only one operation. A parameter cluster groups similar input and output service parameters annotated with *is-a* related ontological concepts. Consequently, the AI planning graph concepts are mapped to the Web service composition concepts as follows:

- An action becomes an ontology concept annotating a service operation.
- A precondition becomes an ontology concept annotating an input parameter of a service operation.
- An effect becomes an ontology concept annotating an output parameter of a service operation.
- The initial state becomes the set of ontology concepts semantically describing the user provided input parameters.
- The goal state becomes the set of ontology concepts semantically describing the user requested output parameters.

The EPG construction is an iterative process which operates at the semantic level by considering the ontology concepts that annotate the service functionality and the input/output parameters. At each step, a new layer i consisting of a tuple $(A_i,\ L_i)$ 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 tuple $(A_0,\ L_0)$ where A_0 is an empty set of services (actions) and L_0 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 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, based on the semantic matching between the services' inputs and the set of parameters of the previous graph layer. The L_i set is built as a union of the L_{i-1} set and the set of the outputs of the services in A_i.

The construction of the 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 services and parameters are the same for the last two consecutive generated layers.

A composition solution encoded in the EPG consists of a set of services, one from each cluster from each EPG layer.

IV. The Immune-inspired Hybrid Method

This section presents how immune-inspired principles can be applied to the problem of selecting the optimal service composition solution as well as a hybrid selection algorithm. This algorithm adapts and enhances a version of the CLONALG algorithm [2] (proposed for general purpose optimization problems) by combining immune-inspired principles with evolutionary computing and reinforcement learning to ensure that the optimal or a near-optimal composition solution is obtained in a short time and without processing the entire search space.

A. Applying Immune-inspired Principles to Select the Optimal Composition Solution

Clonal selection is one of the most important processes of the immune system. It is triggered when a B-cell has high affinity to an invading pathogen (antigen presenting cell), and as a result, the B-cell is stimulated to clone itself. Through cloning, a number of identical copies of B-cells, proportional to the affinity value, are obtained. The number of copies is proportional to the affinity value. The clones are involved in an affinity maturation process which helps in improving their specificity to the invading pathogen by means of somatic hypermutation. The degree of mutation is inverse proportional to the affinity value, meaning that the clones having high affinity do not need to be mutated as much as the ones with low affinity. The affinity matured clones pass through new selection processes aiming at (*i*) keeping the clones having high affinity to the pathogen, and (*ii*) eliminating the clones with low affinity. The selected clones are then differentiated into memory cells and effector cells.

Generally, the biological clonal selection process is mapped to the Web service composition problem as follows: (*i*) a B-cell (or antibody) is represented by a service composition solution, (*ii*) a pathogen (or antigen) is represented by a function *f* that evaluates the set of composition solutions in order to find the optimal one in terms of *QoS* attributes, and (*iii*) the affinity between an antibody and an antigen is represented by the value of the function *f* for a composition solution [5][12].

In our approach, the pathogen (or antigen) is represented by the following multi-criteria function *QF* which evaluates a composition solution, sol, in terms of *QoS* and semantic quality:

$$QF(sol) = \frac{w_{QoS} * QoS(sol) + w_{Sem} * Sem(sol)}{(w_{QoS} + w_{Sem}) * |sol|} \quad (1)$$

where:

- $QoS(sol)$ [8] is the QoS score of the composition solution sol.
- $Sem(sol)$ [8] is the semantic quality score of the composition solution sol.
- w_{QoS} and w_{Sem} are the weights corresponding to user preference related to the relevance of QoS and semantic quality.

In our case, the objective of applying the clonal selection principle is to obtain the optimal or a near-optimal solution which maximizes the QF function. Thus, the affinity between the antibody and antigen is considered to be the value of the QF function for a solution sol.

In our approach, we represent a composition solution by using a discrete type of genetic representation. Consequently, each Web service has associated an identification number in the format $n_l n_c n_s$, where n_l is the service's layer number in EPG, n_c is the number of the cluster to which the service belongs, and n_s is the service's identification number within the cluster. A solution will be a set of service identification numbers.

We model the processes of cloning and somatic hypermutation by keeping the current solution set and adding new different solutions as a result of a mutation process over the older ones. Customized to our problem, cloning implies duplicating for a number of times each solution in the solution set, while somatic hypermutation implies replacing some of the services part of a solution with other services from the same cluster, according to specific criteria. Thus the affinity maturation process is considered to enlarge the search space and to ensure diversity but in a controlled way such that only the mutated clones that are relevant to the problem to be solved are kept. Consequently, if the somatic hypermutated clone is not better in terms of the *QF* function than the parent solution (the original solution) it will be ignored (i.e. will die), otherwise it will be added to the solutions set (i.e. will survive).

B. The Immune-inspired Hybrid Algorithm

Our bio-inspired hybrid selection algorithm (see Algorithm 1) determines the optimal composition solution, sol_{opt}, according to the fitness function QF (see Formula 1) by processing the EPG. Besides the EPG graph, the algorithm takes as inputs the following adjustable parameters: (i) the number n of solutions that can be selected for the somatic hypermutation and affinity maturation process, (ii) the number m of the worst solutions that will be replaced by new ramdomly generated solutions, (iii) the number β that is used to decide how many clones should be generated for each solution in an iteration, (iv) the number noS of allowed stagnations, and (v) the restart iteration number, r. The algorithm returns the optimal or a near-optimal composition solution.

Algorithm 1: Hybrid_Selection

```
1  Input: $EPG$, $n$, $m$, $\beta$, $noS$, $r$
2  Output: $sol_{opt}$
3  Comments: $M_L$ - learning memory, $M$ - solution
   frequency memory, $it_c$ - current iteration, $f_{opt}$ - $sol_{opt}$
   appearance frequency.
4  begin
5    $Sol$ = Initialize_Solution_Set($EPG$)
6    $sol_{opt}$ = Max_QF($Sol$)
7    $M = \emptyset, f_{opt} = 1, it_c = 1$
8    $M$ = Update_Frequency_Memory($Sol, M$)
9    while (!Stop_Cond($it_c, f_{opt}, noS, sol_{opt}$)) do
10     $topN$ = Calculate_TopN($Sol, n$)
11     $SelectedSols$ = Select_Solutions($Sol, topN$)
12     $n_c$ = Calculate_Number_of_Clones($\beta, topN$)
13     $Sol^* = \emptyset$
14     foreach $sol$ in $SelectedSols$ do
15       $Sol^* = Sol^* \cup$ Generate_Clones($sol, n_c$)
16       foreach $cl$ in $Sol^*$ do
17         $cl$ = Somatic_HyperMut($cl, M, M_L$)
18         if ($sol! = cl$ and $QF(sol) < QF(cl)$) then
19           $Sol = Sol \cup \{cl\}$
20           $M_L$ = Reward($sol, cl, M_L$)
21           if ($QF(sol_{opt}) \leq QF(cl)$)
22           then Replace_Optim($sol_{opt}, cl$)
23           else $M_L$ = Penalize($sol, cl, M_L$)
24         end if
25       end for
26     end for
27     $lastR$ = Calculate_LastR($m, topN, Sol$)
28     $Sol$ = Sort($Sol$)
29     Replace_Solutions($Sol, lastR, M$)
30     if (Restart_Condition($it_c, r$)) then Restart($Sol, M_L$)
31     $f_{opt} = f_{opt} + 1, it_c = it_c + 1$
32   end while
33   return $sol_{opt}$
34 end
```

The algorithm starts with an initialization stage (lines 5-8) in which two composition solutions, sol_1 and sol_2, are randomly generated by processing the EPG. These solutions are added to the set of solutions Sol and the one that has the highest QF value is declared the current optimal solution, sol_{opt}. This solution will be considered as the model of the set and will not take part in any other processes. The aim of the other solutions in the set Sol is to become similar to the model. In addition, the appearance frequency of these two solutions is initialized with 1 and stored in a frequency memory. The frequency memory is used to record the appearance frequency of every generated composition solution.

Next, the following main steps are repeated until the stopping condition is satisfied (line 9), namely whether the number of stagnations is equal to noS:

1) Select and clone the topN best solutions: The number $topN$ of solutions that will be selected in order to be cloned and affinity maturated is computed (line 10) according to the following formula:

$$topN = \begin{cases} |Sol| - 1, & if\ |Sol| - 1 < n \\ n, & otherwise \end{cases} \tag{2}$$

Afterwards, the first $topN$ best solutions in the set Sol are selected (line 11) for further processing.

The number n_c of clones that will be generated for each selected solution is calculated (line 12) according to the following formula adapted from [2]:

$$n_c = Round(\beta * topN) \tag{3}$$

where β is a problem specific parameter which influences the selection algorithm's convergence speed towards the optimal solution. In our approach we have considered that $\beta \in [0, 1]$ and its optimal value is determined experimentally.

Afterwards, for each selected solution, a number of n_c clones are generated (line 15).

2) Somatic Hypermutation: Each clone is passed through a somatic hypermutation process (line 17) in which a combined genetic operator is applied between the clone and the current optimal solution. We introduced a combined genetic operator in the selection method to ensure diversity by allowing the exploration of other possible solutions encoded in the EPG. This operator is applied between two composition solutions, one being the current optimal solution sol_{opt}, and the other one being the currently processed solution sol_i, and consists of the following processing steps: (1) the two solutions are compared service by service and as a result, a new solution is obtained which keeps the services that are common to both sol_{opt} and sol_i on one hand and keeps the services from sol_i which do not appear in sol_{opt} on the other hand, (2) the new solution is subject to a mutation process in which the services that belong only to sol_i are replaced with other services from the same cluster. In the mutation step it is important to constantly improve the current local optimum solution. This can be efficiently achieved by taking into account the result of previous decisions. For example, if once we have replaced a service s_1 by another service s_2 and obtained a better solution then it is obvious that by using this replacement in the future, better solutions will be obtained. This is the motivation that lead us to choose reinforcement learning as an important component of the bio-inspired hybrid model. In our case, we have adopted an active type of reinforcement learning.

In what follows, we present the reinforcement learning strategy applied for selecting the optimal Web service composition solution. First, we have defined a special data structure, called the *learning memory*, M_L, defined as follows:

$$M_L = \{m_l | m_l = ((s_i, s_j), reward)\} \tag{4}$$

where s_i is the service that will be replaced by s_j and $reward$ is a numerical value used for recording awards and penalties for the tuple (s_i, s_j). Suppose sol is the current processed solution. After the affinity maturation process completes, some feedback information needs to be sent. If the clone sol_c of sol survives after the affinity maturation process, which means that the clone's affinity value is higher than the parent's affinity value, then the learning process starts and some rewards are given. Consequently, the structure of the mutated clone sol_c is compared against the structure of its parent sol at the cluster level to identify the sol's services that need to be analyzed as follows: if a service s_j belonging to cluster C in sol_c is different than its adjacent service s_i from sol, and the service s_j positively affects the quality of sol compared to s_i, then it will be learnt that by replacing service s_i with service s_j a better solution is obtained and s_j will receive a reward. In other words, the pair (s_i, s_j) is added to M_L if it does not already exist and the reward associated to it will be 1, otherwise it will not be added to M_L and the associated reward will be incremented by 1. If sol_c will not survive after the affinity maturation process, then the learning process also starts, but this time some penalties will be considered. The solutions sol and sol_c are compared as was described above. If the pair (s_i, s_j) is stored in M_L then the reward associated to it will be decremented by 1. In our hybrid method, the learning memory M_L will be used in the somatic hypermutation process every time s_i needs to be mutated. In this process, a pair (s_i, s_j) is searched in M_L and if it is found then s_j will take the place of s_i. Otherwise s_j will be chosen from the s_i's cluster, providing that sol_c will have a higher value of the affinity than sol.

3) Update Sol and Replace $lastR$ Worst Solutions from Sol: If the new clone, *cl*, is different from its parent sol and *cl*'s QF value is better than sol's QF value, then *cl* is added to the set of solutions and rewards are given (line 20) to those pairs of services that lead to a solution that is closer to the optimal one. Also, if *cl*'s QF value is greater than sol_{opt}'s QF value, then sol_{opt} is replaced with *cl*. In case the new clone is not better than its parent, then some penalties are applied.

After processing each solution selected for affinity maturation, the number $lastR$ of worst solutions, in terms of the QF function, is computed (line 27) according to the formula:

$$lastR = \begin{cases} val_1, & if\ topN < n \\ val_2, & otherwise \end{cases} \tag{5}$$

In Formula 5, val_1 and val_2 are computed with Formulas 6 and 7, respectively:

$$val_1 = \begin{cases} 1, & if\ \frac{m*n}{topN} < 0.5 \\ Round(\frac{m*n}{topN}), & otherwise \end{cases} \tag{6}$$

$$val_1 = \begin{cases} |Sol| - 1, & if\ |Sol| - 1 \leq m \\ m, & otherwise \end{cases} \tag{7}$$

where n is the number of solutions that can be selected for the somatic hypermutation and affinity maturation process, and m is the number of the worst solutions that will be replaced by new ramdomly generated solutions.

Consequently, the set Sol is sorted (line 28) and the last $lastR$ worst solutions in the set Sol will be replaced by other randomly generated solutions (line 29).

Formulas 5, 6 and 7 have been validated experimentally.

4) Verify Restart Condition: If the restart condition is true (line 30), namely if it_c is a multiple of r, then the algorithm is restarted (line 30). Consequently, the set of solutions Sol will be initialized with the first two solutions in the current set Sol and the learning memory content will be deleted.

The restart strategy works as follows. Let's consider a predefined parameter t called the restart iteration. Whenever the current iteration of the selection algorithm is a multiple of t, the algorithm is restarted in this way:

- The first two composition solutions of the initial set will not be randomly generated (as when starting the selection algorithm); they will be the current local optimal solution and the solution that follows after it (in terms of affinity values - see Formula 1).
- The learning memory M_L is deleted.

V. Experimental Results

We have evaluated the proposed selection method on a set of scenarios from the trip planning domain. The set of services used in our experiments was developed in-house and annotated according to the SAWSDL specification. A user request is specified by a set of ontological concepts semantically describing the provided inputs and requested outputs as well as by weights indicating the relevance of *QoS* related to the property of semantic quality. Services are semantically described with concepts annotating their functionality, the input and the output parameters. In what follows we present the experimental results obtained for the user request in Table I which aims to make the travel arrangements for a trip to Szczecin, Poland. The EPG for this user request is organized on 3 layers consisting of 51 services grouped in 11 clusters.

A challenge we faced while testing the proposed hybrid method was setting the optimal values of the following parameters: n - the number of composition solutions selected

TABLE I
User request for planning a holiday

User inputs	Requested outputs	*QoS* weights	Semantic quality weight
SourceCity	AccomodationInvoice	Total QoS = 0.55	0.35
DestinationCity	FlightInvoice	Availability = 0.30	
StartDate	CarInvoice	Reliability = 0.30	
EndDate		Cost = 0.15	
HotelType		ResponeTime = 0.25	
NumberOfPersons			
NumberOfRooms			
CarType			
ActivityType			

to be cloned, m - the number of worst composition solutions to be replaced with randomly generated solutions, noS - the number of stagnations in a local optimal solution, r - restart iteration number, β - mutation rate.

The methodology used for establishing the optimal configuration (i.e. a 5-tuple containing a value for each adjustable parameter) of the adjustable parameters consists of three steps. In the first step, an exhaustive search is performed to identify the score of the optimal composition solution which is 6.8 for the considered scenario. This score is further used to identify the most appropriate configuration of the adjustable parameters which ensures that the optimal or a near-optimal composition solution is obtained without processing the entire search space. In the second step, we chose the following initial configuration of the adjustable parameters based on some suppositions: $\beta = 0.25$, $r = 11$, $n = m = 4$ and $noS = 33$. In the third step, the initial configuration of the adjustable parameters is fine-tuned iteratively to identify the optimal configuration. For the considered scenario we have chosen 43 configurations and for each one we have performed 100 runs of the selection algorithm analyzing the following aspects: the global optimal solution frequency, the number of iterations, the number of generated distinct solutions, the execution time measured in seconds, and the standard deviation. In Table II we present a fragment of the average experimental values obtained for each configuration, where: (i) $fopt_{av}$ is the average global optimal solution frequency, (ii) $noIt_{av}$ is the average number of iterations, (iii) $noSG_{av}$ is the average number of generated distinct solutions, (iv) tEx_{av} is the average execution time measured in seconds, and (v) stD_{av} is the standard deviation.

By analyzing the experimental results from Table II we reached the conclusion that the optimal configuration of the adjustable parameters is $n = 7$, $m = 8$, $\beta = 0.5$, $r = 6$, $noS = 24$. We chose these values because they ensure a tradeoff between obtaining the optimal solution and a very low execution time.

In the final stage of the hybrid algorithm's evaluation we performed 2100 simulations on the scenario presented in Table I and by considering the optimal configuration of the adjustable parameters. In our experiments we focused on the following issus: the average number of processed solutions, the average percent of explored search space, the average simulation time, the average number of cases in which the optimal solution has been obtained, and the standard deviation of the score of the best solution returned by the algorithm related to the score of the global optimal composition solution for the considered scenario. By analyzing the experimental results we conclude that:

- The hybrid selection algorithm explores aproximatly 0.001% of the search space (i.e. approximatly 205 distinct composition solutions explored out of 13996800).
- The average execution time is about 27 seconds.
- The hybrid selection algorithm returns the global optimal solution in 95% of the simulations (i.e. 2015 simulations out of 2100 simulations).
- The average standard deviation on all 2100 simulations of the hybrid selection algorithm is 0.0344.

TABLE II
TESTS SUMMARY

No.	n	m	β	r	noS	$fopt_{av}$	$noIt_{av}$	$noSG_{av}$	tEx_{av}	stD_{av}
1	4	4	0.25	11	33	0.95	54	207	19	0.042
4	4	4	0.5	11	44	0.96	70	271	48	0.031
3	4	4	0.25	11	44	0.96	68	261	32	0.028
7	6	6	0.25	11	44	0.97	66	351	36	0.027
8	6	6	0.25	11	33	0.91	54	287	28	0.043
9	6	6	0.5	11	33	0.89	54	286	34	0.065
11	8	8	0.25	22	11	0.23	18	134	9	0.175
12	8	8	0.25	22	22	0.55	38	272	18	0.128
13	8	8	0.5	11	22	0.87	41	266	30	0.056
15	8	8	0.5	11	44	0.91	66	439	50	0.052
17	8	8	0.5	11	33	0.96	53	352	39	0.035
18	6	4	0.5	11	33	0.96	56	205	39	0.031
19	6	4	0.5	11	22	0.8	42	156	29	0.079
21	6	4	0.75	11	33	0.93	55	202	52	0.039
23	6	4	0.75	11	11	0.51	23	82	23	0.155
24	8	6	0.75	11	11	0.57	23	114	23	0.131
26	6	5	0.75	11	33	0.93	56	245	51	0.049
27	6	5	0.75	11	22	0.79	39	174	37	0.078
28	6	5	0.75	11	11	0.6	24	106	25	0.121
30	6	5	0.75	6	12	0.86	25	97	24	0.076
34	7	8	0.5	11	33	0.93	51	335	37	0.038
35	7	8	0.5	11	22	0.85	38	245	27	0.083
37	7	8	0.5	6	18	0.93	33	173	23	0.038
41	5	6	0.75	11	33	0.87	53	281	38	0.060
42	4	6	0.75	11	44	0.93	70	380	37	0.040
43	7	8	0.5	6	24	0.95	39	204	28	0.033

VI. CONCLUSIONS AND FUTURE WORK

This paper proposed a hybrid method for selecting the optimal or a near-optimal solution in semantic Web service composition. The method was designed according to principles from the clonal selection process which occurs in biological immune systems. To increase the algorithm's convergence speed towards the optimal solution as well as to avoid the problem of stagnation in a local optimum we introduce a combined genetic operator which generates new better composition solutions by querying a learning memory. The learning memory stores a history of service replacements as well as a score which indicates which replacement can lead to a new better service composition solution.

The hybrid method has been tested on a set of scenarios from the trip planning domain and its performance has been evaluated according to the following criteria: the number of processed solutions, the percent of explored search space, the simulation time, the number of cases in which the optimal solution has been obtained, and the standard deviation.

As future work we intend to comparatively analyze the hybrid method with other selection methods using the same test scenarios.

References

[1] G. Canfora, M. Di Penta, R. Esposito, M. L. Villani, *A Framework for QoS-Aware Binding and Re-Binding of Composite Web Services*, Journal of Systems and Software, Volume 81, Issue 10, pp. 1754-1769, 2008.

[2] L. Castro, F. von Zuben, *Learning and Optimization using the Clonal Selection Principle*, IEEE Transactions on Evolutionary Computation, Volume 6, Number 3, pp. 239-251, 2002.

[3] M. Dorigo, M. Birattari, Thomas Stutzle, *Ant Colony Optimization - Artificial Ants as a Computational Intelligence Technique*, IEEE Computational Intelligence Magazine, Volume 1, Number 4, pp. 28-39, 2006.

[4] X. Fan, X. Fang, *On Optimal Decision for QoS-Aware Composite Service Selection*, Information Technology Journal, Volume 9, Issue 6, pp. 1207-1211, 2010.

[5] G. Yan, N. Jun, Z. Bin, Y. Lei, G. Qiang, D. Yu, *Immune Algorithm for Selecting Optimum Services in Web Service Composition*, Wuhan University Journal of Natural Sciences, Volume 11, Number 1, pp. 221-225, 2006.

[6] H. Jiang, X. Yang, K. Yin, S. Zhang, J. A. Cristoforo *Multi-path QoS-aware Web Service Composition using Variable Length Chromosome Genetic Algorithm*, Information Technology Journal, Volume 10, Issue 1, pp. 113-119, 2011.

[7] J. Kennedy, R. C. Eberhart, *Particle Swarm Optimization*, Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ. pp. 1942-1948, 1995.

[8] C. B. Pop, V. R. Chifu, I. Salomie, M. Dinsoreanu, *Immune-Inspired Method for Selecting the Optimal Solution in Web Service Composition*, LNCS, Volume 6162/2010, pp. 1-17, 2010.

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

[10] W. Wang, Q. Sun, X. Zhao, F. Yang, *An improved Particle Swarm Optimization Algorithm for QoS-aware Web Service Selection in Service Oriented Communication*, International Journal of Computational Intelligence Systems, Volume 3, Supplement 1, pp. 18 - 30, 2010.

[11] Z. Yang, C. Shang, Q. Liu, C. Zhao, *A Dynamic Web Services Composition Algorithm Based on the Combination of Ant Colony Algorithm and Genetic Algorithm*, Journal of Computational Information Systems, Volume 6, Issue 8, pp. 2617-2622, 2010.

[12] J. Xu, S. Reiff-Marganiec, *Towards Heuristic Web Services Composition Using Immune Algorithm*, Proceedings of the International Conference on Web Services, Beijing, China, pp. 238-245, 2008.