Towards Heuristic Web Services Composition Using Immune Algorithm

Jiuyun Xu
School of Computer & Communication Engineering
China University of Petroleum
xujiuyun@ieee.org

Stephan Reiff-Marganiec
Department of Computer Science
University of Leicester
srm13@le.ac.uk

Abstract

One of the main benefits of web services is the dynamic composability, however how to achieve this is one of the current research challenges. Web service composition has been studied and, amongst other methods, the use of natural computing methods has been proposed previously. In this paper, we address the need for a fast response when computing the most suitable sequence of services. In particular, we propose a novel heuristic immune algorithm with an efficient encoding and mutation method. The algorithm involves two steps: an immune selection operation, which is maintaining antibody population diversity and a clonal selection. The use of a vaccine during the evolution provides heuristic information that accelerates the convergence. Our experimental results illustrate that the proposed heuristic immune algorithm is very effective in improving the convergence speed.

1. Introduction

Web services provide a good method for implementing Enterprise Application Integration (EAI), but their use is not restricted to this and they might form the application development framework of the future for any large scale distributed application that can be composed of functional units which are possibly owned by different stakeholders, such as banking systems or healthcare systems and the typical example of a travel agency. In the real world, a single web service usually cannot fulfill the requirement for a given enterprise task, however a composition of different web services might meet the complex requirements. Hence, there is the potential silver bullet that enterprises have been looking for and it has attracted the interest of many researchers. To date, there are two principal approaches for web service composition. The first one is considering web service composition in a dynamic environment essentially as a planning problem, mainly using semantic features of web services to determine whether two services are composable [1]. The alternative is to provide an abstract plan (a workflow) and select instances of services for fulfilling the tasks in the plan. In this context domain experts firstly establish the workflow of some business model using abstract web services and then concrete web services will be bound to every abstract service at runtime in order to fulfill the requirement of customers. It is intuitive that the latter is more readily achievable, and in fact BPEL [2] provides the de-facto specification language for this. However, this still leave the problem of selecting appropriate concrete services. To select suitable concrete service from a service candidates set with a range of QoS attributes is a non-trivial task. It is exactly this aspect were the current paper provides a novel solution. Clearly the problem is an optimization problem and thus is NP-complete. In previous work to address the issue the problem has been modeled as a knapsack problem [3] or alternatively linear and dynamic programming techniques [4,5] have been applied. More recently, researchers started to use natural computing techniques to address QoS-aware web service composition, because of its independence of specific web service knowledge [6,7,8].

In the world of web service composition using natural computing most effort has been focused on how to obtain the global optimal solution [6,8-10], with the focus on "optimal". However, in the real world, for example when considering telecommunication or medical operation services, it becomes crucially important that the solution is obtained quickly, even if they are then only approximation to the optimal solution. Natural computing techniques use populations of genes or antibodies as encoding of the problem space and a fitness function to measure how good a solution is; solutions are improved by evolution a la Darwin: the strongest genes survive and replicate.

For natural computing methods, the first step is to select a population of antibodies or genes for evolution. There are two methods to get the initial population: select the initial population manually or generate them randomly. Although the quality of initial antibodies does not affect the final solution much, a bad selection increases the time to convergence significantly.

In this paper, we address the problem of the convergence speed during evolution to suit the requirement of the real-time web service composition. We proposed a novel heuristic web service selection method using the immune optimization approach together with an efficient encoding to reduce the search space. Regarding the encoding, available services are encoded in binary mode in a concise representation which is maintained during the Immune evolution operation. Thus, any antibody will always be a solution to the problem (albeit not the optimal one) and hence the size of the search space is kept small – an issue of great importance when considering larger problems. Regarding the heuristics, we are capturing the genes that lead to low as well as high fitness in vaccines and use these during the evolution process to obtain faster convergence. In other words, mutation is not random for every gene of the whole antibody, but concentrates on the parts of genes which cause the low fitness of the antibody. With this heuristic information during evolution, the speed of convergence can be significantly improved, even when the initial antibody population is far from the global optimal one (i.e. the initial antibody was of low quality).

The reminder of this paper is arranged as follows: In section 2 some background knowledge is introduced. Section 3 provides and overview of the method and section 4 details the heuristic immune algorithm method; in section 5 a case study is introduced and experimental results are presented in section 6. And in section 7 some related work has listed. Finally, we conclude and provide an outline of further research.

2. Background

In this paper, we present a heuristic immune optimization method to handle QoS aware web service selection to be used as part of the service composition process. This is based on the immune optimization algorithm. We will now present some background on the immune algorithm to set the scene for our enhancements.

2.1. The natural immune system

In general, genetic algorithms encode a "solution" to a problem as a "gene", essentially a string with a predefined structure. The initial solution is chosen at random, and might in fact not be a solution at all. Assume the problem at hand: selecting concrete web services for a workflow; a solution would be a mapping where each task (or abstract service) is assigned a concrete service. The selected services might however not be the best solution (their interfaces might not match, they might not provide the best end to end security, etc). Hence, it is clearly necessary to determine how good the solution is, which is achieved by use of a fitness function. If the solution is not satisfactory, evolution (essentially mutation of the gene string) will lead to a new generation which might be fitter or less fit. This is continued for an undetermined number of generations, until a predefined acceptable level of fitness is achieved – clearly a time consuming process.

Compared to the typical genetic optimization approaches, the immune optimization approach is a population based optimization method. The difference between the immune algorithm and genetic algorithm has been detailed in [9, 11] and a detailed discussion is beyond the scope of this paper. Briefly, the immune algorithm has a property ensuring the diversity of the population during the evolution process. In recent years, some work has been dedicated to increasing the speed of the immune evolution approach [9-11]. Other work has concentrated on applying the immune algorithm in many application domains; web service being one of them. In [8] the immune algorithm was adopted to use notions of affinity and concentration to ensure population diversity. Population diversity is important as it prevents premature convergence.

2.2. Overview of Immune Optimization Method

There are two steps of evolution: the Immune selection operation and clonal selection operation. In the Immune selection operation, antibodies are proliferated and suppressed to regulate their density, and making sure that the potential helpful antibodies will not be destroyed. In the clonal selection operation, we use the potential genes which are in high quality antibodies as heuristic information for speeding up convergence.

Calculation of affinity and concentration. Genetic algorithms are useful for their independence of the concrete optimization problem tackled. However, the premature convergence problem often produces local optimal results rather than global ones. To obtain global optimization, the Immune algorithm has proposed the use of *affinity*, which means the similarity between two antibodies, and *concentration*, which is representing the density of population in the mating pool. Basically, when the mating population has a stable concentration, the evolution can easily maintain the population diversity. In this paper, we calculate the affinity between the antibody i and the antibody j using Hamming distance as per formula (1):

$$Affinity_{ij} = \frac{1}{N_d} \sum_{n=1}^{N_d} D_n \qquad (1)$$

Where N_d denotes the total number of genes in one antibody. And $D_n=1$, when the alleles are similar, and $D_n=0$ otherwise.

The concentration of antibody i is defined by:

$$C_i = \frac{1}{M} \sum_{j=1}^{M} \delta_j \qquad (2)$$

Where M is the size of the antibody population in the mating pool:

$$\delta_j = \begin{cases} 1 & Affinity_{ij} \geq \lambda \\ 0 & Affinity_{ij} \prec \lambda \end{cases}$$

and λ denotes the affinity threshold, $0.9 \leq \lambda \leq 1$.

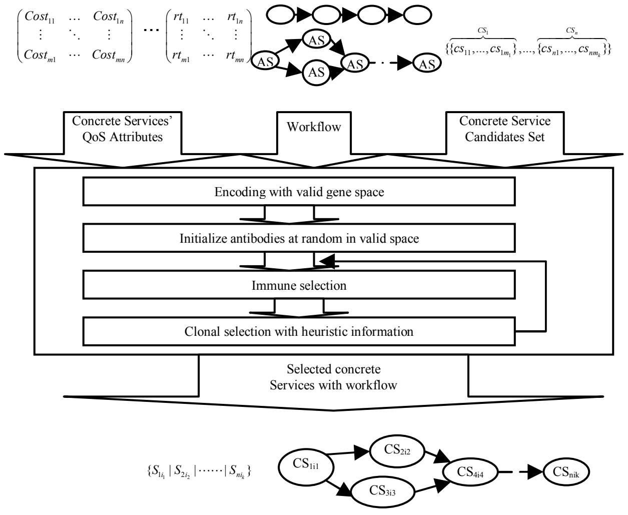
The Antibody Space for Encoding and Mutation. In genetic algorithms and Immune optimization algorithms, the first step is to encode the problem. Usually, one of two simple encoding methods is adopted and previous work in the web services field does the same. One possibility is an encoding where every concrete web service uses one bit. When this concrete web service occurs in the result, the code is “1”; otherwise, the code is “0”. However, this method often leads to enlarged search spaces. The other encoding is mapping all concrete web services with the same function into a binary number using the minimum of bits. This method can reduce the invalid search space much and hence helps during evolution.

Calculation of Fitness with Various Workflows. Workflow modeling, and BPEL is no exception, caters for a wide variety of different workflows. In the real world, workflows are much more complex than simple sequences that have often been considered for web service composition. Workflows contain many other control flow operators, for example parallel, case, fork and return flow are quite common. In fact, the same concrete services used within different workflows will result in different value of web service composition fitness. In [7], a series of formula of calculating the fitness to evaluate the fitness of various workflows has been given. We adopt those in our Immune optimization algorithm.

3. Overview of Approach

To ease the reader into the topic of web service composition with natural computing, let us consider a scenario: There are a set of concrete services which are provided by different service providers. Moreover, these services have different QoS attributes such as service cost, service response time, service availability and service reliability. However, all of these concrete services can be matched to some abstract services – hence we will have several sets of concrete services, one for each abstract service. Assuming a composition scenario following the BPEL approach, we expect an abstract service composition making use of different control flow operators having been provided (detecting this automatically is an as of yet unsolved planning problem). We also expect that the QoS attributes of the concrete services are known. An overview of the approach to instantiate web services compositions is shown in Fig 3.1.

In the next few paragraphs, we describe a high-level view of three key issues in our Immune algorithm for service selection, namely: the QoS attribute matrix, the encoding method and the adopted two-step selection operation. First, the QoS attribute matrix for concrete services, which determine the fitness of antibodies, includes four different aspects: service cost, service response time, service availability and service reliability (one could easily extend this to include further factors, or adapt the current ones). The fitness of each antibody can be calculated from the service QoS attributes and abstract service control flow.



Concrete Services' QoS Attributes
Workflow
Concrete Service Candidates Set
Encoding with valid gene space
Initialize antibodies at random in valid space
Immune selection
Clonal selection with heuristic information
Selected concrete Services with workflow
AS
$\{\underbrace{\{cs_{11},...,cs_{1m_1}\}}_{CS_1},...,\underbrace{\{cs_{n1},...,cs_{nm_k}\}}_{CS_n}\}$
$\{S_{1i_1} | S_{2i_2} | \cdots\cdots | S_{ni_k}\}$
CS_{1i1}
CS_{2i2}
CS_{3i3}
CS_{4i4}
CS_{nik}

Fig 3.1: Overview of the approach: the top presents the input, the core the process and the bottom part the output.

Secondly, a binary encoding method is adopted during the evolution of web service immune composition. Every concrete service is encoded into a binary string, with each set of concrete services functionally relating to one abstract service being encoded into one string. The maximum number of bits used for the string depends on the maximum number of concrete services in one abstract service. In the current implementation method, all binary strings have the same length (namely that of the longest such string, with blanks being padded). The valid binary strings corresponding to the concrete service have been labeled. The antibodies are composed of these binary strings according to the control flow for the evolution step. The detail of the encoding is presented in Fig 3.2.

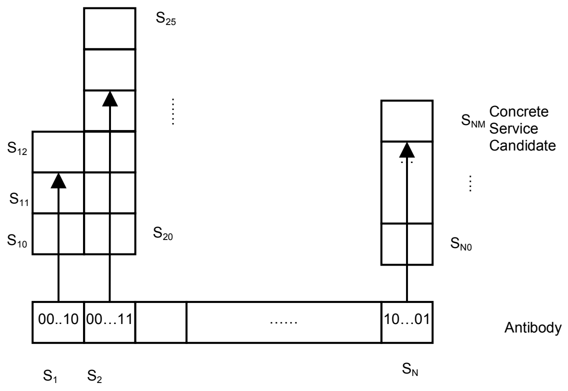


Fig. 3.2 Antibody encoding

Thirdly, during the generations of antibody evolutions, the immune selection algorithm and clonal selection algorithm are adopted to get a global optimal antibody. First, the Immune selection algorithm is applied to select the next generation of antibodies with the purpose of maintaining population diversity and increasing the opportunity of global optimization. And the clonal selection method is employed to enhance the local search, which intensifies exploitation of the known search space. Vaccines are developed based on experiential knowledge of the problem to be solved and inoculation is employed to improve the quality of the solution candidates to speed up convergence.
Finally, decoding the antibody provides the final output – the optimal service instantiation for the given composition problem.

4. The Heuristic Immune optimization approach for web services selection

4.1. Overview of some basic ideas

In the Immune optimization algorithm the problems to be solved are regarded as antigens, while antibodies are composed of genes. Generally, there are three types of parameters, which are: the fitness for measuring the antibody's quality, the affinity for similarity between antibodies, and the concentration for population diversity (we have introduced the latter two in section 2). In this paper, we take the common genes of antibodies with high fitness as heuristic knowledge, and thus construct a vaccine, to accelerate the speed of convergence.

Recall that a set of QoS matrices, a set of concrete web services and the workflow as an abstract web services composition based on BPEL are given as input. The goal is to find a suitable instantiation with concrete services. Generally, similar concrete web service instances used in different compositions (that is with different workflows) will have different fitness – it is not always the same service that will be the best choice.

Considering the Immune optimization algorithm, we use the two steps of Immune algorithm which is introduced by the paper [12]. The Immune algorithm, includes the Immune selection operation and clonal selection operation, and we will consider these in some more detail next.

4.2. Immune Selection Operation

Genetic algorithms, provide a good method for concrete optimization problems but premature convergence means that often the solution is a local optimum, not global one. The Immune optimization system uses the Immune selection operation (ISO) to overcome this problem. Generally, the Immune selection operation performs some measurements such as affinity and concentration. The affinity measures the similarity between antibodies, while the concentration measures the antibody population diversity.

We base our solution on ideas from [12], which uses Immune optimization in assembly planning. While assembly planning is quite close to our problem, and hence there are some differences between our method and theirs. Next, we will describe our Immune selection operation in detail.

The first step is concerned with calculating affinity and concentration. To calculate the affinity, a Hamming distance is used to calculate the difference between the fittest antibody and every other antibody. The concentration is based the affinity of antibodies, and is calculated according to formula (2).

4.3. Clonal selection Operation

The clonal selection operation is the second important step. We adopt the clonal selection operation that [10] proposed for pattern recognition. An overview of the algorithm is presented in Fig 4.1, and we will describe this here in more detail.

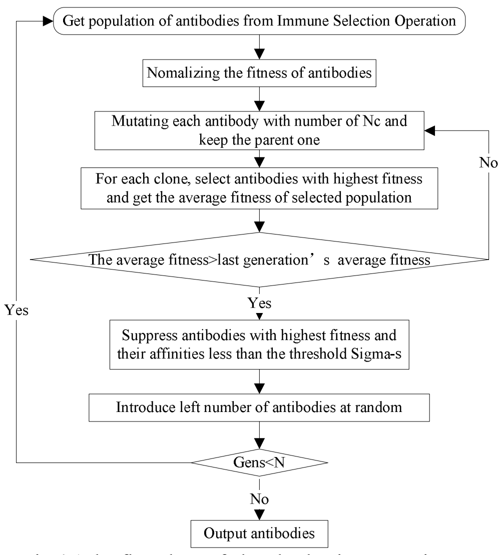

Get population of antibodies from Immune Selection Operation
Nomalizing the fitness of antibodies
Mutating each antibody with number of Nc and keep the parent one
For each clone, select antibodies with highest fitness and get the average fitness of selected population
The average fitness>last generation's average fitness
No
Yes
Suppress antibodies with highest fitness and their affinities less than the threshold Sigma-s
Introduce left number of antibodies at random
Gens<N
Yes
No
Output antibodies

Fig 4.1 the flowchart of clonal selection operation

In this method, the fitness of each antibody is determined first and then the vector of fitness is normalized. Next, a number Nc of clones is generated for each antibody which are mutated proportionally to the fitness of its parent antibody, but the parent antibody is kept. Next, the fitness of all individuals of the population is determined and for each clone, the antibody with the highest fitness is selected. The average fitness of the selected population is calculated. If the average divergence of the population is not significantly different from the previous iteration, then continue. Else, return to the beginning of the clonal selection operation to determine the affinity of all antibodies in the population. Suppress all, but the antibodies with the highest fitness whose affinities are less than the suppression threshold Sigma-s and determine the number of antibodies, named memory antibodies. After suppression, we introduce a percentage *d* of randomly generated antibodies.

4.4. Vaccine as Heuristic information in service selection

In the Immune algorithm, the initial cell population is important for the convergence of the evolution. Generally, if the initial selection is "good" the algorithm will converge very quickly on the final result, but otherwise it will take much more time. As the number of web services with same functionality are increasing, speed up convergence of the evolution will be essential (this is anyhow required, if we consider on-the-fly composition). For Web service selection, there is no prior information to obtain a "good" initial cell population, so the convergence of the algorithm has to be improved. In this paper, we enhanced the Immune algorithm with heuristic information to select suitable web services from the set of candidates of web service. First, an initial antibody population, represented by a set of antibodies, is generated at random. Second, the Immune selection operation measures the antibody population to select the best. Third, during the clonal selection operation, the mutation activity is not randomly applied to all cells as usual, but only to the parts of cells, which probably causes low fitness of antibodies.

4.5. Composite Web service with Workflow in BPEL

Considering our research work based on the BPEL specifications, we must investigate different workflows – that is in the shape and operators that they use as well as the abstract services. Generally, composition with different workflows leads to different fitness calculations. In this paper, we use the formulae from [7] to match the control flow elements of **sequence**, **while**, **switch** and **pick** in BPEL.

4.6. The heuristic Immune service selection algorithm

We will now describe the heuristic Immune optimization algorithm in detail. This algorithm reduces the reliance on a good initial population to obtain fast convergence. It achieves this by effectively using heuristic information – the latter is very interesting for QoS-aware web service composition in its own right.

In the clonal selection operation, all antibodies in the evolving pool will be divided into two groups according to their fitness. One of the groups contains antibodies with high fitness, while the other contains those with lower fitness. In the first group, we compare with allele in the set of all antibodies from left to right.

The heuristic Immune algorithm is described as follows:

Inputs: The workflow (i.e. the abstract web service composition), a set of concrete web services with QoS matrixes (According to [7], there are four matrixes to represent the costs, reliability, responsibility and time of concrete services, reliability – however more could be added for further criteria) and some required parameters for the procedure of the Immune algorithm, such as the elite, the number of memory pull of antibodies, etc.

Output: the instantiation of the workflow with concrete web services;

The description of heuristic Immune optimization method is as follows:

1. Encode the concrete web services in binary code. First, get the max number *X* of concrete services matching the same abstract service. Then make sure the integer *N* is chosen such that $2^{N-1} < X < 2^N$ allowing for an encoding of all concrete services in *N* bits. Remember the binary number chosen to encode each specific concrete service.
2. Initialize the mating pool: generate the initial population of antibodies with valid genes randomly.
3. Perform the Immune selection operation. First, select the best antibody, that is the one with the highest fitness, and place it in the mating pool. Next, the best antibody is replicated by a ratio (elite ratio) that indicates the proportion of the amount of best antibodies in the mating pool. At last, all the other antibodies are proliferated and suppressed according to the following formula and selected into the mating pool by the means of roulette.
$$f_i' = f_i / C_i \qquad (3)$$
Where f_i denotes the fitness of the antibody; f_i' is the fitness after proliferation and suppression, and the concentration is C_i.
4. Decide on the genes to be mutated. First, select the N_s antibodies which have high fitness (N_s is set as a parameter to the algorithm); Second, find the common genes in each of the "fit" antibodies – these are potential vaccines V_{gen}. Third, select the number N_s of antibodies which have low fitness, and again find the common genes in each of antibodies – these are non-vaccines. Finally, try to dispose antibodies matching the non-vaccines from the potential vaccines. We now have a set of vaccines.
5. Perform the clonal selection operation. Basically, clonal selection can enhance the local search by enlarging the search scope. In our work, we modified the clonal selection algorithm introduced in [10] by including inoculation based on the vaccines identified in step 4. This means that in the clonal selection operation, the mutation action is only performed on the genes which are not vaccines.

During the evolution, the fitness of every antibody is normalized, to keep it in a computable range.

5. A Case Study

To demonstrate the efficiency of our method, a real scenario has been assumed. In this scenario, firstly, we use seven abstract services with a simple sequential control flow. Among these services, there are different numbers of concrete service candidates with different QoS attributes, such as service cost, service response time, service availability and service reliability. In our example, the number of concrete services corresponding to the abstract services is two, three, five, two, five, five and eight respectively. The QoS attributes of concrete services are illustrated in Table 5.1, while Fig 5.1 represents the complete search space.

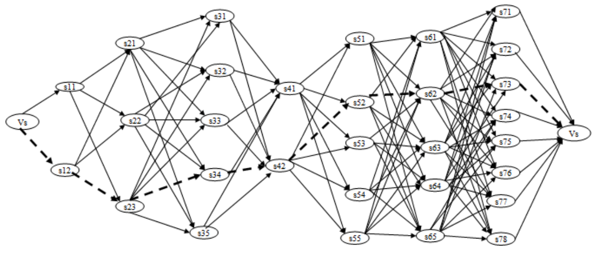


Fig 5.1. Example of composite service process.

Moreover, we encode every concrete service into numeric bits of binary data as part of the antibody. In this example, the number of bits of every gene is three. So we can get the whole valid genes space (Fig. 5.2) and combine these binary characters as genes to the antibodies.

						111
						110
						101
		100		100	100	100
		011		011	011	011
	010	010		010	010	010
001	001	001	001	001	001	001
000	000	000	000	000	000	000
s1	s2	s3	s4	s5	s6	s7

Fig 5.2. The encoding space of antibodies

Abstract-Service	Concrete-Service-Candidates	Service-cost	Service-Response-Time	Service-Availability	Service-Reliability
S1	S11	112.0	204.0	3.0	24.0
	S12	1.0	1.0	38.0	177.0
S2	S21	212.0	12.0	12.0	12.0
	S22	22.0	22.0	22.0	22.0
	S23	3.0	3.0	342.0	342.0
S3	S31	221.0	12.0	12.0	14.0
	S32	32.3	32.3	33.3	34.43
	S33	34.0	34.0	18.0	34.0
	S34	42.0	42.0	42.0	42.0
	S35	3.0	3.0	323.0	323.0
S4	S41	24.3	45.3	43.3	44.3
	S42	3.0	3.0	210.0	273.0
S5	S51	46.3	36.3	6.3	64.3
	S52	4.0	3.0	349.0	374.0
	S53	44.0	44.0	44.0	44.0
	S54	55.0	57.0	55.0	55.0
	S55	22.0	22.0	22.0	22.0
S6	S61	46.3	36.3	6.3	64.3
	S62	4.0	3.0	349.0	374.0
	S63	44.0	44.0	44.0	44.0
	S64	55.0	57.0	55.0	55.0
	S65	22.0	22.0	22.0	22.0
S7	S71	22.0	22.0	22.0	24.0
	S72	33.0	33.0	33.0	33.0
	S73	3.0	3.0	377.0	394.0
	S74	65.0	65.0	65.0	65.0
	S75	33.0	33.0	33.0	33.0
	S76	44.3	44.3	44.3	44.3
	S77	22.0	22.0	22.0	22.0
	S78	50.0	70.0	70.0	10.0

Table 5.1. QoS attributes of concrete services

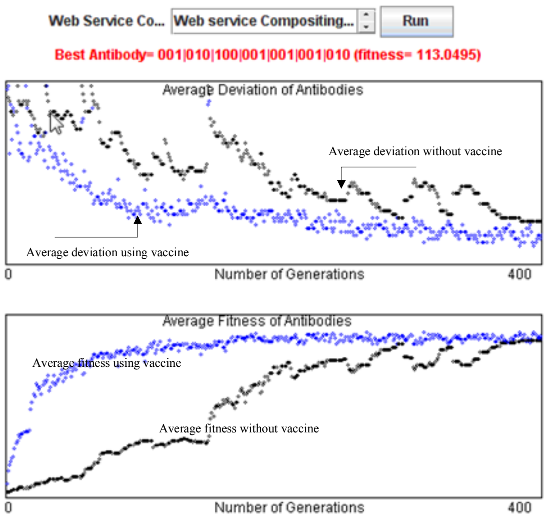
Finally, using the heuristic Immune algorithm in section 4, the evolution converges towards the best antibody: 001|010|100|001|001|001|010 with fitness of 113.0495. This means the best composite web services is $S_{12}S_{23}S_{34}S_{42}S_{52}S_{62}S_{73}$.

6. Experimentation Result and Analysis

In this paper, we have conducted the experiment with the data as follows: first, we tested a simple situation with a sequential workflow with seven abstract web services, whereby different numbers of matching concrete web services exist as discussed in the previous section. The QoS attributes are the service costs, the response times, and the availability and service reliability stored within four matrices. The data of some additional parameters that we used is as follows: the population of antibodies for Immune algorithm of mating pool is about 60 antibodies, the elite ratio with 20%, clone rate with 60%, the crossover probability with 70% and the mutation probability with 60%. The affinity threshold is 95%.

Many experiments with different initial antibodies population, generated at random, have been tested. In the experiment, two parameters, namely the average deviation of antibodies and the average fitness of antibodies, have been drawn at random. One typical aggregated experimental result is shown in Fig. 6.1. The figure has two parts: the upper part is the average deviation of antibodies, while the lower part shows the average fitness of antibodies. In the average deviation of antibodies, the upper curve represents the normal Immune algorithm without heuristic knowledge, the other shows the results using the heuristic Immune algorithm. In the average fitness of antibodies part, the curve achieving highest fitness first is the heuristic Immune algorithm, while the other is the normal Immune algorithm.

As our experiment shows, when the random initial population of antibodies is far from the solution, the Immune algorithm with heuristic information can improve the speed of evolution significantly.


Web Service Co...
Web service Compositing...
Run
Average Deviation of Antibodies
Average deviation without vaccine
Average deviation using vaccine
0
Number of Generations
400
Average Fitness of Antibodies
Average fitness using vaccine
Average fitness without vaccine
0
Number of Generations
400

Best Antibody= 001|010|100|001|001|001|010 (fitness= 113.0495)

Fig 6. 1 The results of the experiment with Ns=10.

7. Related Work

Many related work has focused on how to select web services with high quality of service from amongst those with the same functionality. All of this work can be divided into two groups: those methods which handle service selection using semantic web services ideas, and those that focus on selecting services which differ in their quality of service but provide the same functionality [3, 4, 6, 7, 13-17].

The first paper using Immune algorithm to address QoS-aware web service selection is [8]. In this paper, the Immune Algorithm has been adopted, however the algorithm in that paper is much like the clonal selection operation compared with our algorithm. The improvements that our method provides over theirs are: the encoding method developed in our work contains only valid antibodies, no invalid antibody will be included in the antibody population during the whole evolution (that is any antibody would provide a solution, albeit not an optimal one). Second, our method uses the Immune selection operation to maintain the antibody population diversity, which will avoid the premature convergence during evolution by maintaining the population diversity. Experimental evidence shows that this two step Immune algorithm can obtain the optimal or near-optimal solution. Most notably, we have added the use of heuristic information to the algorithm and our experiment demonstrates that this indeed helps to achieve solutions much quicker than using the traditional Immune algorithm.

8. Conclusion and Future Work

In this paper, a heuristic Immune optimization algorithm has been proposed for QoS-aware Web services selection. In the Immune algorithm, the evolution is confined in the valid antibody space, not only in the initial antibody population, but also in the antibodies generated by the mutation operation. With this encoding method, the search space will be reduced.

QoS-aware Web Service composition, which is essentially an NP-complete problem, has led many researchers to propose the use of natural evolution computing methods. In these natural computing methods, the initial population is usually generated at random; when a good initial population is selected convergence is fast. However, generally, a good initial population cannot be guaranteed. The use of heuristic information speeds up the convergence of evolution, especially when the initial population of antibodies is not good.

While our experiments showed that the algorithm provided the expected solutions, we will explore whether the solution achieved by that heuristic Immune algorithm is similar to that of the Immune algorithm without heuristics in theory.

Acknowledgments. This research work is supported by funding from the China Overseas Scholarship Council. This research work has been undertaken during Jiuyun Xu's visit at the University of Leicester.

References

[1] Medjahed, B., A. Bouguettaya, and A.K. Elmagarmid, *Composing Web services on the Semantic Web.* The VLDB Journal, 2003. **12**: p. 333-351.

[2] BPEL, http://www.ibm.com/developerworks/library/ws-bpel/.

[3] Yu, T., Y. Zhang, and K.-j. Lin, *Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints.* ACM Transactions on the Web, 2007. **1**(1): p. 1-22.

[4] Zeng, L., et al., *QoS-Aware Middleware for Web Services Composition.* IEEE Transactions on Software Engineering, 2004. **30**(5): p. 311-327.

[5] Arpinar, I.B., et al., *Ontology-Driven Web services composition platform.* Information Systems and E-Business Management, 2005. **3**(2): p. 175-199.

[6] Zhang, C., S. Su, and J. Chen, *DiGA: Population diversity handling genetic algorithm for QoS-aware web services selection.* Computer Communications, 2007. **30**: p. 1082-1090.

[7] Canfora, G., et al. *An Approach for QoS-aware Service Composition based on Genetic Algorithms*. in *proceedings of the Genetic and Computation Conference(GECCO 2005)*. 2005. Washington, DC: ACM Press.

[8] Yan, G., et al., *Immune Algorithm for Selecting Optimum Services in Web Services Composition.* Wuhan University Journal of Natural Sciences, 2006. **11**(1): p. 221-225.

[9] Timmis, J., *Aritificial immune systems - today and tomorrow.* Natural Computing, 2007. **6**(1): p. 1-18.

[10] Castro, L.N.d. and J. Timmis. *An Artificial Immune Network for Multimodal Function Optimization*. in *Proceedings of IEEE Congress on Evolutionary Computation (CEC'02)*. 2002.

[11] Zhang, Z., *Immune optimization algorithm for constrained nonlinear multiobjective optimization problems.* Applied Soft Computing, 2007. **7**: p. 840-857.

[12] Cao, P. and R. Xiao, *Assembly planning using a novel immune approach.* International Journal of Advanced Manufacturing Technology, 2007. **31**(7-8): p. 770-782.

[13] Berbner, R., et al. *Heuristic for QoS-aware Web Service Composition*. in *IEEE International Conference on Web Services (ICWS'06)*. 2006.

[14] Canfora, G., et al. *QoS-Aware Re-planning of Composite Web Services*. in *Proceedings of the IEEE International Conference on Web Services (ICWS'05)*. 2005.

[15] Tian, M., et al. *Efficient Selection and Monitoring of QoS-Aware Web Services with the WS-QoS Framework*. in *Proceedings of IEEE/ACM International Conference on Web Intelligence 2004*. 2004.

[16] Bilgin, A.S. and M.P. Singh. *A DAML-Based Repository for QoS-Aware Semantic Web Service Selection*. in *ICWS 2004*. 2004.

[17] Cardoso, J., et al., *Quality of Service for Workflows and Web Service Processes.* Web Semantics: Science, Services and Agents on the World Wide Web, 2004.