SCIENTIFIC RESEARCH CENTER

International Journal of Research in Science and Engineering

Volume 1, Issue 2

QoS-Based Web Services Composition using an Improved Genetic Algorithm

Mahyar Taheri[*], Babak Shirazi[**], Hamed Fazlollahtabar [***]

* *Department of Information Technology Engineering, Mazandaran University of Science and Technology, Babol, Iran*
**Department of Industrial Engineering, Mazandaran University of Science and Technology, Babol, Iran
****Faculty of Industrial Engineering, Iran University of Science and Technology, Tehran, Iran*

*Corresponding Author: hfazl@iust.ac.ir

Abstract

Web Services Composition is one of the most important issues in service oriented architectures. For answering to majority of complicated business process, it might not have been just a single service, so several services must have combined to reach a suitable one. Composite service will generate by combining single web services. Each web service may have different implementations with similar functions, but something which makes it different from other similar services is the quality of service. In this study, QoS based web service composition is under study and architecture for automated web service composition is proposed. In this architecture, at first users enter their functional and non-functional requirements into the system by users interface, then an improved genetic algorithm (IGA)is developed for optimal composition of web services in order to indulge users' requirements in a suitable time. Proposed approach his implemented and evaluated by C# language. The evaluation results have shown that the IGA outperforms is better than simple GA when the number of abstract services are large.

Keywords: Web Service; Web Service Composition; Web Service Selection; Quality of service; Improved Genetic Algorithm; Meta-Heuristic Algorithm

1. Introduction

Today, because of increase in flow of information in inside and outside of organization and its management, organizations don't have any choice except using the advantages of Information Technology and information systems. Service Oriented Architecture offers new model in implementation of informational system and lets the system developers to focus most of their attentions and realizing characters which the organization needs them (Griffiths and Chao 2010). There are various definitions for Service Oriented Architecture which are often based on the usage of it. In (MacKenzie 2006(accessed 2008-02-14)) a definition of Service Oriented Architecture is given by OASIS: It is a model for organizations and the usage of their distributed abilities is under control of different people. Based on the given definition Service Oriented Architecture is a model. In other word SOA is an approach or a meditative method, so we can't say that is a tool, which we can buy (Josuttis 2007). Most advantages of using SOA include: Reusability of services, its ability in decreasing the expenses, improvement organization's agility and business (Svensson and Wallen 2006). The other strong point of it is in leveling the operations between heterogeneous information systems. It would be used for integration and connection between informational systems of web services, but in most cases a single service can't offer all complicated requirements of customers alone, because of that answering to these requirements should be used by multiple services. So the ability of the organizations and companies in selection and composition of web services in developing software and informational services would become very important in order to use the functions and variety abilities of services could offer the customers' requirements and their complex requirements (Ma and He 2009).

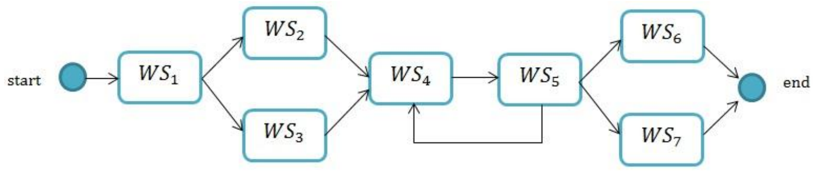


Figure 1: Service composition in workflow

When an organization requests service composition, at first should design a work flow of activities that each of them performs with a Web service. Fig.1 shows workflows of organization's activities that include 7 web services (W_1, W_2, W_3…, W_7). Any of these web services can have different implementation but similar function. Users who order web services usually express their non-functional requirement with quality criterion. The quality of service defines the abilities of a product or a service for facing with non-functional requirements of a user and these criteria can be used as a benchmark for differentiate and distinction between services and the service provider (Sha, Shaozhong et al. 2009). There might be services between the similar ones that based on quality criteria would be more efficient for users, therefore there is a period of time which is given to perform activities existed several services with similar functions, then the service would be for a user choose based on requirements and quality of a service. In this study we considered 5 qualities of services which are: execution time, latency, reliability, availability and success rate which are more popular than others however the proposed approach will develop in a way that the new quality criteria can be added easily or omit one of them. Discovering and composition of services on the run time of that also was the most important part of progressive challenge in service oriented application. If we have all number of activities in workflow (k) and for performing each of them there is a web service with a similar function (n), in this situation n^k causes different composition of web services for performing work processes. Therefore achievement of an optimal composition from different existed compositions would be very complicated and searching all cases is very expensive and time consuming. On the other hand the number of produced composition by considering the quality criteria of services will increase(Wang, Tong et al. 2007, Zhang 2011). In this study we use the Improved Genetic Algorithm for optimization of selection and composition of web services. Improved Genetic Algorithm is adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection and genetic. On the beginning evolutionary approaches include single series that these parts will develop based on rules. In our method evolution will happen by using Genetic Operators. In this model for determining whether the produced composition of web services is a good one or not a kind of fitness function is used.

The remainder of the paper is organized as follows. After a review of the literature of web services selection in Section 2, in Section 3 we explain the way how to make formula and un-scaling the quality of services. In the Section 4, the way to operate the fitness function for composing web service is told. In Section 5, we present the proposed architecture and in the Section 6the approach will be evaluated and finally conclusion and the future work would be mentioned in Section 7.

2. Related work

These days' composition and selection of web services changed into a very important subject in academic issues based on quality of service. The approach of selection and ranking web services is provided by Yin and You's quality of services. In this approach to supervise services' QoS and facilitate the service evaluation, we assume that all users' concerned services' QoS aspects are controlled and reported to the service directory (SD), which saves services' QoS information along with their registrations. This information will be applied by the service evaluator to rank services and help users select services when there are more than one registered services satisfying users' functionality requirements (Yau and Yin 2011).

The framework of Dynamic Cos is an automatic mechanism for composition of services in execution time. This architecture supports all the phases and stakeholders of the dynamic service composition life-cycle, as automatic service discovery, selection and composition and with the CLM, the composition algorithm has to find a composition of services that fulfill the service request (Goncalves da Silva, Ferreira Pires et al. 2009).

SODIUM (Service-Oriented Development in a Unified fraMework) was an international project, involving research, technological and industrial partners, dedicated to tackling interoperability challenges that companies face at the data, services and business levels. The project has developed a Generic Service Model, containing the common concepts of heterogeneous services from multiple points of view. The special characteristics of individual service technologies (such as Web services, Grid services or P2P services) are then dealt with as extensions to the core. The SODIUM methodology adopts a model-driven and iterative approach for service composition. The approach utilized OMG's QoS profile to represent collections of QoS properties (Topouzidou 2007).

One of approaches of selecting web services based on quality criteria takes place by using a 2- dimensional Boolean array which is called selection matrix. Row of matrix represents web services and the columns show quality of services. When the value of matrix's cell equals 1, it means that the user's requirements of quality of service are the same as published value for service and otherwise the value equal 0. Then when the matrix row has maximum number of 1 we can conclude that service can be the best web service for meeting the user's quality of service requirements (Chaari, Badr et al. 2008).

Multi Objective Genetic Algorithm (MOGA) is presented for optimal selection of web service based on quality of services. In this algorithm at first a mechanism is designed for possible combinations and feasible web services according to constraints of services composition (like dependency constrains and conflict between web services) and more genetic operators (selection operator, cross over and mutation) and strategies for distribution of diverse population that these populations are the same various composition of web services is introduced (Wang and Hou 2008).

A flexible approach is presented by Yang and Chun Hung for composing services based on genetic algorithm which can do the composition of web services based on users' functional and non-functional requirements. In the presented approach for composition of workflows, like sequential, conditional, fork, is considered. Procedures in the presented approach is as follows that the users send their requirements in the algorithm and then the algorithm of the best composition for meeting the requirements of the users among existing services in UDDI is provided (Fanjiang, Syu et al. 2010).

A flexible architecture is presented by Bounhamdi and Jarir for dynamic web service composition at run time. In this firstly a general framework for ranking and selection of services with characteristics of presented WSSR-Q which is based on service description model. Then an algorithm will be presented for ranking of service in order to calculation and normalize service quality values for all web services. Afterwards a quality updating mechanism would be presented to update the quality which is based on users' feedback values (Zou, Xiang et al. 2009).

Another approach to select and compose efficiently of web services is using the Particle Swarm Optimization which has been presented by Zhongjun Liang. In this composition approach some constraint for choosing web services is presented. Constraints of the qualitative requirements of laws designed. These constraints intended between services on the implementation approach. In this approach at first a list of impossible composition is mentioned and then Particle Swarm Optimization algorithm is used among feasible combinations (Liang, Zou et al. 2012).

Selection of a suitable service according to functional and non-functional requirements of users is one of the most important goals of selection and composition of web services, some approaches, which suggest the selection of web services based on quality of service, have been addressed. Then we will mention some disadvantages and shortcomings of available algorithm. In some presented approaches optimal selection of web service considered dependent from others and there is no constraint for combining services, so in these ways when candidate service for selecting, dependent the other services, can't have good selection. Some presented approaches of Quality of Service matrix (QoS matrix) are using qualitative features for calculating. The basis of some of presented methods of matrix are for computing properties of quality of service and while the number of service quality or web service increase, the matrix becomes very big and the number of calculations increase and become complicated so using these methods are not always suitable.

Some of existing methods don't support multiple quality criteria of users. In these approaches the user can just one quality of service for selecting a web service. Also in some other available approaches although they support different quality of service but the user can't prioritize for each of quality of service by weighting.

But in this study an approach is presented for the automatic and dynamic composition of Web services, this approach use a meta-heuristic algorithm in order to increase the scale of problem, which gives a suitable composition to users in a reasonable time and also can prioritize each quality by weighting of each quality of services which were considered before. The details are given later.

3. Modelling

In this study, when we talk about an abstract web service we refer to a web service in a workflow and while we say concrete web service it is the implementation of an abstract web service.

WS_i: i is an abstract web service in a workflow.

$WS = (WS_1, WS_2, \dots WS_i, \dots, WS_n)$

On top $(1 \le i \le n)$

$S_{i,j}$: A concrete web service of j is in an abstract web service i.

On top $(1 \le i \le n)$ and $(1 \le j \le K_i)$ and also K_i is the total number of concrete web services for each of the WS_i abstract web services.

$q_{i,j}^{l}$:The Lth QoS for concrete web services of j which on the ith abstract service

Q= $(q_{i,j}^{1}, q_{i,j}^{2}, q_{i,j}^{3}, \dots q_{i,j}^{m})$

On top m is total number of quality service and $(1 \le i \le n)$ and $(1 \le j \le K_i)$ and $(1 \le l \le m)$.

C^l: Each of the criteria for quality of service, to assign a weight due to its importance for the user.

C= $(c^1, c^2, c^3, c^4, \dots, c^m)$

On top m is the total number of quality of service and $(1 \le l \le m)$ also sum of total QoS criteria is equal to 1 ($c^1 + c^2 + c^3 + c^4 + \dots + c^m = 1$).

Each of the presented of the quality of service in this research has different feature and sometimes they are probably conflict each other, therefore in order to compare different Criteria, at first we should un-scale this QoS by a mechanism. In this un-scaling method, for Comparable value of these data of QoS to each other, by at least (q_{min}^{l}) and maximum (q_{max}^{l}) of these which are found in services by research, we transfer all the data of quality of service in 0 and 1. We use equation 1 for the QoS that have direct relationship with the criteria of quality and equation 2 for the QoS which have vice versa relationship.

$$Q_{i,j}^{l} = \begin{cases} \dfrac{q_{i,j}^{l} - q_{min}^{l}}{q_{max}^{l} - q_{min}^{l}} & if(q_{max}^{l} != q_{min}^{l}) \\ 1 & if(q_{max}^{l} = q_{min}^{l}) \end{cases}$$ **Equation1**

$$Q_{i,j}^{l}=\begin{cases}\frac{q_{max}^{l}-q_{i,j}^{l}}{q_{max}^{l}-q_{min}^{l}} & if(q_{max}^{l}!=q_{min}^{l})\\ 1 & if(q_{max}^{l}=q_{min}^{l})\end{cases}$$ **Equation2**

4. Fitness function

Genetic algorithm uses a fitness function to evaluate each element of the current solution. This function computes fitness of a composition services with different structures, according to Quality of Service and user weighting. Each of the single services can be combined with various structures includes: sequential, fork, conditional and loop. In this study the dynamic service composition is provided and according to requirement of user, single services can be combined to each other. Therefore calculated fitness of a composite service, according to the structures of used is different and after analyzing the process model, it is calculated by the specified user.

Table1: Aggregation method of QoS properties

	sequential	**fork**	**conditional**	**loop**
response time	$\sum_{i=1}^{n} T(S_i)$	$Max(T(S_1)\ldots T(S_n))$	$Max(T(S_1)\ldots T(S_n))$	$k*T(S)$
latency	$\sum_{i=1}^{n} L(S_i)$	$Max(L(S_1)\ldots l(S_n))$	$Max(L(S_1)\ldots L(S_n))$	$k*L(S)$
availability	$\prod_{i=1}^{n} Av(S_i)$	$\prod_{i=1}^{n} Av(S_i)$	$Min(Av(S_1)\ldots Av(S_n))$	$(Av(S))^k$
reliability	$\prod_{i=1}^{n} Re(S_i)$	$\prod_{i=1}^{n} Re(S_i)$	$Min(Re(S_1)\ldots Re(S_n))$	$(Re(S))^k$
success rate	$\prod_{i=1}^{n} Se(S_i)$	$\prod_{i=1}^{n} Se(S_i)$	$Min(Se(S_1)\ldots Se(S_n))$	$(Se(S))^k$

In table1 there is used a list of quality criteria in this research and along with the way of computing, aggregate the QoS property values in different structures are mentioned, in order to computing fitness function, one complicated Service at first step, the value of each quality of service criteria is in table1 according to the different structures and therefore aggregate the QoS property values in complicated service is computed. At 2nd step, aggregate the QoS property values is mentioned in the previous step to un-scaling and after doing that, weights which are determined by user are added to each till the fitness function for a complicated service is computed.

As it is shown in Fig.2 in computing the aggregate the QoS property values, services which in connection to each other similar structures are determined as virtual service and it can be connected by the other services. As an example at Fig.2, two services s2 and s3 are combined to each other by sequential structure and virtual service (Sequential (s2, s3)) are made and this virtual service is as a fork with service s4 and create virtual service(Fork (Sequential (S_2,S_3),S_4)).And the structures and the way to connect to other abstract service would be determined (Ko, Kim et al. 2008). As the composition of service which is presented on the approach is dynamic and each user requests his specific composition, therefore in order to compute fitness function, at first structural model should be determined for different QoS criteria.

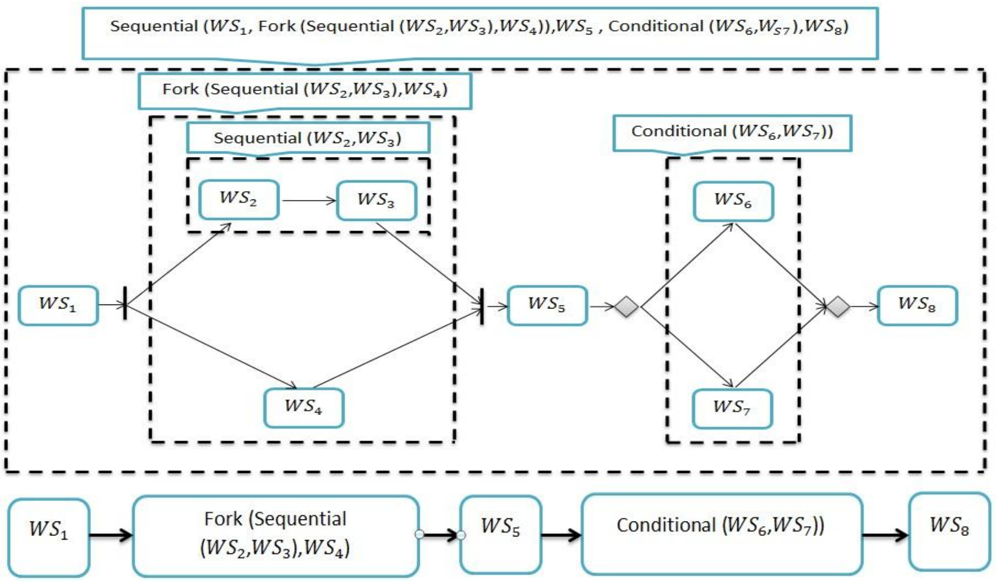

Sequential (WS_1, Fork (Sequential (WS_2,WS_3),WS_4)),WS_5 , Conditional (WS_6,W_{S7}),WS_8)
Fork (Sequential (WS_2,WS_3),WS_4)
Sequential (WS_2,WS_3)
Conditional (WS_6,WS_7))
WS_1
WS_2
WS_3
WS_4
WS_5
WS_6
WS_7
WS_8
WS_1
Fork (Sequential (WS_2,WS_3),WS_4)
WS_5
Conditional (WS_6,WS_7))
WS_8

Figure 2: Aggregation process example of QoS properties

For the process model of above mentioned, after analysis on the model, as it is clear at Fig.2, this structural model for each of QoS criteria (q^l) is acquired of equation 3.

SM_{q^l}=Sequential ($q^l(S_1)$, Fork (Sequential ($q^l(S_2)$,$q^l(S_3)$)),$q^l(S_4)$)),$q^l(S_5)$,Conditional ($q^l(S_6)$,$q^l(S_7)$)),$q^l(S_8)$) **Equation3**

By replacing the value of QoS in the acquired structural model, the value of criteria is computing in composition of services. In continue in order to computing fitness function of a complicated service, it is needed to use the equations1 and 2 which were explained before and for q^lwhich have inverse relation by quality criteria, such as response time and latency, $SM_{q^l}$is in equation 2 and for the others q^l, $SM_{q^l}$is in equation 1. Therefore fitness function of the made complicated service at Fig.2 is acquired by this formula.

Equation4

$$F = \sum_{l=1}^{2} C^l * \left(\frac{SM_{q^l_{max}} - SM_{q^l}}{SM_{q^l_{max}} - SM_{q^l_{min}}}\right) + \sum_{l=3}^{5} C^l * \left(\frac{SM_{q^l} - SM_{q^l_{min}}}{SM_{q^l_{max}} - SM_{q^l_{min}}}\right)$$

5. Proposed Approach

In this section, we introduce proposed approach and the architecture. Our proposed architecture for efficient Web services composition is shown in Fig.3. Afterward every one of these parts is explained separately.

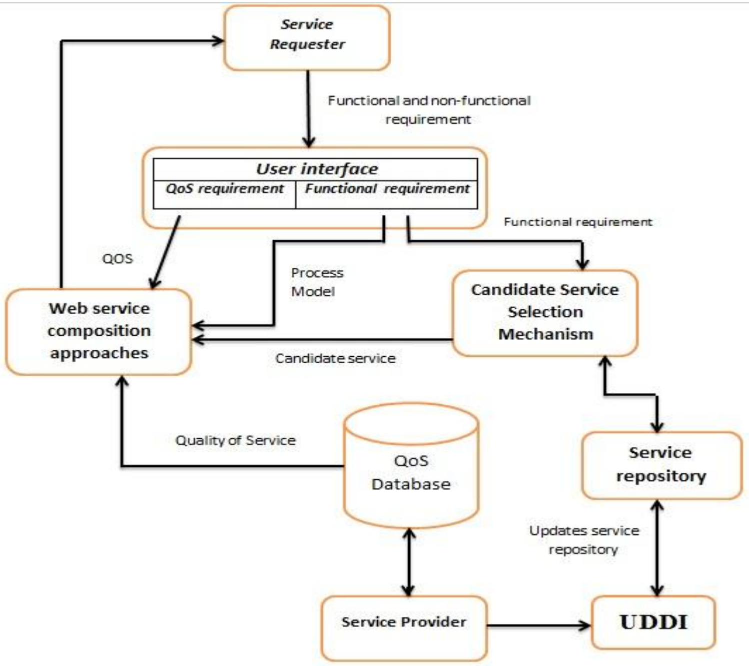


Figure 3: Suggested architecture for QoS-Based web service composition

User interface: in this architecture the functional and non-functional requirements of requester is received by a user interface, and changed into a process model by Business Process Execution Language.

Candidate service selection: when users request a complicated service instead of to searching all services in the repository, at first, a number of services that meet the functional requirements of the service requester are selected as the candidate service

QoS Database: it's a Data base in which the values of QoS criteria of all existing web services in the service repository are stored.

Service repository: in this part, there are different web services to support the different requirements of users and after that new service is registered in UDDI by providers, repository service will be updated and web services' quality will be held in service repository.

Web service composition approach: in this step, all candidate service and quality requirements of users which were achieved before, go into service composition approach. Now by using one Meta heuristic approach we search among all candidate service according to the QoS criteria till select the best composition between different situations. Finally after finding the most suitable composition, according to the requirement of requester, this selected service present to the user. Our proposed approach his based on Genetic in nature. In this approach a **chromosome** (also sometimes called a **genome**) is a set of parameters which define a proposed solution to the problem that the genetic algorithm is attempting to solve.

2.1. *Improved Genetic Algorithm*

Different from other approaches proposed in the literature, such as linear integer programming, GAs do not impose constraints on the linearity of the QoS composition operators. This permits the use of our approach for all possible (even customized) QoS attributes, without the need for linearization. This section elaborates our improved genetic algorithm. This improved genetic algorithm uses a local search to improve the individuals in the population and utilizes a knowledge based crossover operator.

```
Procedure: Genetic Algorithm
Initializing population
While (not termination coordination) do:
    Computing fitness values of all Composition Services
    Using crossover operation to build a new composition service
    Mutation operation
    Local search
    Selecting Composition Services based on fitness values
    Building new population
End.
```

Figure 4: A hybrid genetic algorithm for the web service selection problem

2.1.1. *Genetic Encoding*

An individual in the population of our Improved Genetic Algorithm represents a web service selection plan and it is encoded in an array of *n* integersx_1, x_2, ... , x_n, where *n* is the total number of abstract web services in the workflow of the composite web service. In the genetic encoding scheme, each gene represents an abstract web service in the composite web service and a value of the gene represents a concrete web service of the abstract web service. Fig. 5 illustrates the encoding scheme.

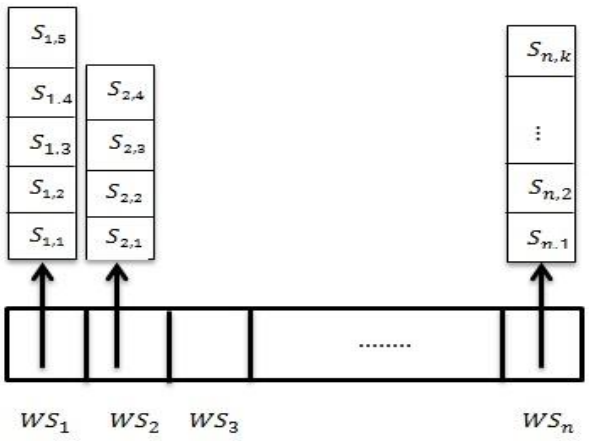
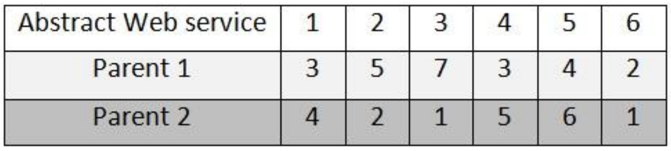


Figure 5: Genetic Encoding scheme

2.1.2. *Genetic Crossover*

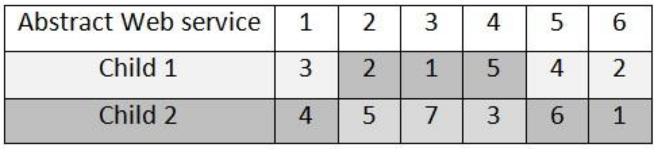
Different from the crossover operator used in the Genetic Algorithm, the crossover operator used in the IGA is a two-point crossover. Two-point crossover operator randomly selects two crossover points within a chromosome then interchanges the two parent chromosomes between these points to produce two new child.



Abstract Web service	1	2	3	4	5	6
Parent 1	3	5	7	3	4	2
Parent 2	4	2	1	5	6	1

Figure 6: Consider the two parents selected for crossover

Interchanging the parents chromosomes between the crossover points.



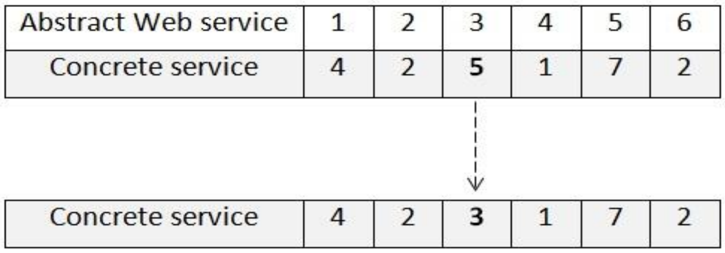
Abstract Web service	1	2	3	4	5	6
Child 1	3	2	1	5	4	2
Child 2	4	5	7	3	6	1

Figure 7: The Childs produced

2.1.3. *Mutation*

After a crossover is performed, mutation takes place. Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of chromosomes to the next. Mutation occurs during evolution according to a user-definable mutation probability, usually set to fairly low value, say **0.01** a good first choice.

The mutation operator in this study randomly selects a concrete web service selection for an abstract web service and replaces the concrete web service selection with an alternative concrete web service of the abstract web service. Fig. 7 illustrates the mutation operator.



Abstract Web service	1	2	3	4	5	6
Concrete service	4	2	**5**	1	7	2

Concrete service	4	2	**3**	1	7	2

Figure 8: mutation operator

2.1.3. *Local search*

Genetic local search (GLS) is a population based iterative search scheme for combinatorial optimization problems. Roughly, it consists of the application of genetic operators to a population of local optima produced by a local search procedure. The process is iterated until either a solution is generated or a maximal number of generations are reached.

Therefore maintain a solution, known as current solution (X_C), and search its neighborhood (Local_Search (X_C)) for a better one. If a better solution S ϵ (Local_Search (X_C)) is found, S becomes the new X_C and the neighborhood search starts again. If no further improvement can be made, i.e. l. S ϵ Local_Search (X_C) such as S improves X_C, then, a local or global optimum has been found.

```
Function Local_Search()
for i ← i = 1 to COUNT_ITER do
            Temp ← $X_C$
    for j ← 1 to COUNT_POINT do
                Abs_Point ← Random (0, n)
                Change_Bit (Temp [Abs_Point])
            endfor
            If (Temp.Fitness>Chromosome.Fitness) then
                $X_C$← Temp
                Temp=null
            endif
    endfor
    return$X_C$
endfunction
```

Figure 9: Local Search Algorithm

To determine the termination condition of GAs, we used time measure as termination criterion. If there is no change in 2 seconds in the obtained fitness solution, the algorithm is stopped and we consider the maximum obtained fitness.

6. Evaluation

The computation time and quality of the results produced by the GAs depend on the size and the complexity of the web service selection problem. The size of the problem is dependent on two parameters. 1) The number of abstract web services in the workflow. 2) The number of concrete web services for each of the abstract web services.

The first set of test problems included 10 test problems with different numbers of abstract web services. The number of abstract web services ranged from 5 to 50 with an increment of 5. The number of concrete web services for each of the abstract web services was fixed to 20. The second set of test problems also included 5 test problems. The number of concrete web services for each of the abstract web services ranged from 10 to 50 with an increment of 10. The number of abstract web services was fixed to 10.

In all test cases, service composition is generated randomly by different structure include sequential, fork, conditional and loop. Also values of QoS criteria for abstract web service uses a real Dataset (Al-Masri and Mahmoud 2007, Al-Masri and Mahmoud 2007). QWS is a real dataset and represents 2500 real web services that exist on the Web. In this paper in order to test the performance of IGA and simple GA, they were implemented in Microsoft Visual C# 2008 and all the experiments were conducted in a PC with a 2.2 GHz Intel Core 2 Duo CPU and a 2 GB RAM.

Firstly, IGA and simple GA were used to solve each of the test problems in the first test set. Considering the stochastic nature of the GAs, each of the experiments was repeated 10 times and then calculated the average fitness value and average execution time for each of the experiments for each of the GAs.

Table 2: Fitness of IGA and simple GA by increasing the number of abstract services

Abstract Service	IGA	SGA
5	0.33304	0.32273
10	0.33242	0.32219
15	0.32896	0.32832
20	0.32907	0.32841
25	0.32849	0.32789
30	0.32898	0.32717
35	0.32615	0.32521
40	0.32663	0.32523
45	0.32648	0.32473
50	0.32696	0.32458

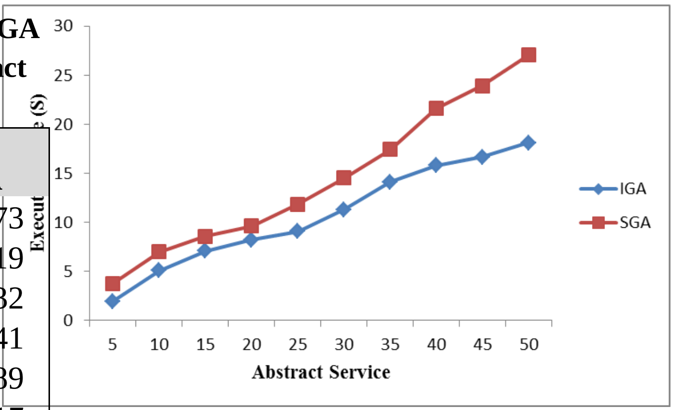


Figure 10: Execution time of IGA and simple GA by increasing the number of abstract services

Table 2 shows the fitness of IGA and simple GA for first test set and it includes 10 test case and Fig. 10 shows that by increasing these abstract services from 5 to 50, what kind of changes will happen in the execution time.

As it was clear from Table 2, IGA in 10 test cases had better performance and had better fitness than the simple GA. Also according to Fig. 10,Improved Genetic Algorithm by increasing the number of abstract services from 5 to 50 in all test cases has less execution time than simple GA. Therefore above results shows better performance of IGA than simple GA in terms of fitness and execution time.

In continue IGA and simple GA was used to solve each of the test problems in the second test set. Considering the stochastic nature of the GAs, each of the experiments was repeated 10 times and then calculated the average fitness value and average computation time for each of the experiments for each of the GAs.

Table 3 shows average fitness of IGA and simple GA for second test set and Fig. 11, shows increase of average execution time for IGA and simple GA when the number of concrete web services ranged from 10 to 50

Table 3: Fitness of IGA and simple GA by increasing the number of concrete services

Abstract Service	IGA	SGA
10	0.33235	0.32225
20	0.33237	0.32228
30	0.33218	0.32216
40	0.33219	0.32203
50	0.33212	0.32211

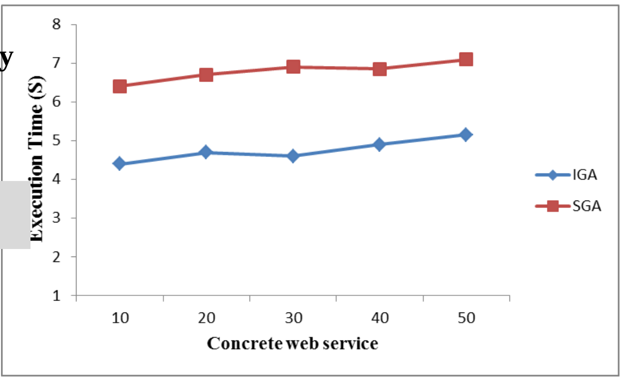


Figure 11: Execution time of IGA and simple GA by increasing the number of abstract services

According to Table 3 in terms of the fitness by increasing the number of concrete services IGA has better performance than the simple GA inmost test cases and as it was clear, in the Fig. 11, IGA and simple GA had no correlation with the number of concrete web services per abstract web service. IGA has the execution time less than 6 seconds for all test cases and it has better execution time than simple GA.

7. Conclusion and future work

In this research QoS-Based web service selection and composition with the IGA meta-heuristic approach is presented to find an optimal solution in a suitable time but the most important goal that should be paid attention in service composition is the scalability of presented approach, because the weak scalability, when the problem size is increasing, makes the approach unusable.

This goal is successfully provided by on IGA approach. All test cases on pervious section are choose in a way to observe the execution time of IGA approach with the increase in number of abstract and concrete web services and we studied the performance and scalability of the presented approach. The obtained results of simulation show good performance and scalability of approaches in large scale of tested cases.

In Section 5 the general architecture for automated web service selection and composition is presented. In this research only implementation and evaluation of one part of the architecture, which it was service composition approach is studied, but other components of the architecture are described only briefly. Therefore, the next step of this research will considered the implementation of various components of the proposed architecture. As another future work we can increase performance of IGA by combining it with another Meta heuristic approach for example to generate initial population in IGA we can use another Meta heuristic approach such as particle swarm optimization (PSO).

References

Al-Masri, E. and Q. H. Mahmoud (2007). Discovering the best web service. Proceedings of the 16th international conference on World Wide Web, ACM.

Al-Masri, E. and Q. H. Mahmoud (2007). Qos-based discovery and ranking of web services. Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on, IEEE.

Chaari, S., et al. (2008). Enhancing web service selection by QoS-based ontology and WS-policy. SAC '08 Proceedings of the 2008 ACM symposium on Applied computing, New York, NY, USA, ACM.

Fanjiang, Y. Y., et al. (2010). Genetic algorithm for QoS-aware dynamic web services composition Machine Learning and Cybernetics (ICMLC), 2010 International Conference on, Qingdao, IEEE.

Goncalves da Silva, E., et al. (2009). Supporting dynamic service composition at runtime based on end-user requirements. 1st International Workshop on User-generated Services, Stockholm, Sweden, CEUR Workshop Proceedings.

Griffiths, N. and K. M. Chao (2010). Agent-based service-oriented computing, Springer-Verlag New York Inc.

Josuttis, N. (2007). SOA in practice-Art of Distributed System Design. 2007, O'Reilly & Assoc.

Ko, J. M., et al. (2008). "Quality-of-service oriented web service composition algorithm and planning architecture." Journal of Systems and Software **81**(11): 2079

Liang, Z., et al. (2012). "A hybrid approach for the multi-constraint Web service selection problem in Web service composition." Journal of Information & Computational Science **9**(13): 3771-3781.

Ma, C. and Y. He (2009). An Approach for Visualization and Formalization of Web Service Composition. International Conference on Web Information Systems and Mining, 2009. WISM 2009. , Shanghai, IEEE.

MacKenzie, C. M. (2006(accessed 2008-02-14)). Reference model for service oriented architecture. Public Review Draft 2.

Sha, L., et al. (2009). A QoS based web service selection model. Information Technology and Applications, 2009. IFITA '09. International Forum on, Chengdu, IEEE.

Svensson, C. and L. Wallen (2006). "SOA and M & A-Relationships between Service Oriented Architectures (SOA) and Mergers and Acquisitions (M & A)." Department of Informatics, Lund University, Lund, Sweden.

Topouzidou, S. (2007). "SODIUM, service-oriented development in a unified framework." Final report ISTFP6-004559. http://www. atc. gr/sodium.

Wang, H., et al. (2007). QoS-Based Web Services Selection. e-Business Engineering, 2007. ICEBE 2007. IEEE International Conference on, Hong Kong, IEEE.

Wang, J. and Y. Hou (2008). Optimal Web Service Selection based on Multi-Objective Genetic Algorithm. Computational Intelligence and Design, 2008. ISCID '08. International Symposium on, Wuhan, IEEE.

Yau, S. S. and Y. Yin (2011). QoS-Based Service Ranking and Selection for Service-Based Systems. Services Computing (SCC), 2011 IEEE International Conference on, Washington, DC, IEEE.

Zhang, C. (2011). Adaptive Genetic Algorithm for QoS-aware Service Selection. Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on, Biopolis, IEEE.

Zou, G., et al. (2009). An agent-based web service selection and ranking framework with QoS. Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on, Beijing, IEEE.