# A Performance Evaluation Study for QoS-aware Web Services Composition Using Genetic Algorithms

**Pedro F. do Prado, Luis H. V. Nakamura, Julio C. Estrella, Marcos J. Santana, Regina H. C. Santana**

Institute of Mathematics and Computer Science – University of São Paulo (USP)
São Carlos – SP, Brazil, 13560 – 970

{pfprado,nakamura,jcezar,mjs,rcs}@icmc.usp.br

***Abstract.*** *In this paper four different Genetic Algorithms (GAs) are proposed to solve the QoS-aware Web Services Composition (QWSC) problem in six different search-space sizes and a realistic deadline (a point not covered in many related works). The results obtained are compared with some results from an Exhaustive Search (ES) algorithm that always guarantees the global optima. Differently from some related works, statistical techniques are adopted in this paper in order to ensure more precise results from the four GAs. The results obtained showed that the design of experiments and the performance evaluation can be used to determine which genetic operators are better suited for the QWSC problem.*

## 1. Introduction

Nowadays, QoS-aware Web Services Composition (QWSC) is one of the most interesting research issues on Service-Oriented Architecture (SOA). Actually, it is not a new research issue and a Genetic Algorithm (GA) was proposed to solve this problem in 2005 by [Canfora et. al. 2005]. In that paper, an empirical study compared a GA with Integer Programming (IP) based algorithm. The results proved that the GA was better suited to the QWSC problem. Other works also compared IP-based algorithms with GAs concluding that a GA is a better alternative [Ko et. al. 2008]. Furthermore, recent works also used GAs or hybrid algorithms (GA combined with another technique) to solve this problem [Batouche et. al. 2010; Fanjiang et. al. 2010].

QWSC is a combinatorial NP-hard problem so it is a complex problem to solve. The number of possible composition plans (the size of the search-space) grows exponentially according to the size of the composite plan. Thus, the use of ES algorithms or numerical method algorithms is limited to only very small search-space sizes. When the growing proliferation of the use of Web Services (WS) is considered, these algorithms become more and more obsolete.

Another important characteristic of the QWSC is the fact of it is a soft real-time problem. However, many papers found in the open literature do not explicit approach this characteristic. Only in some more recent papers this characteristic is mentioned [Fanjiang et. al. 2010; Liu et. al. 2010]. According to [Liu et. al. 2010], due to this fact it is necessary to obtain a solution within the established deadline , even if the solution found is only approximate to the optimal one.

According to [Cheng 2002], a real-time system is one where time requirements of the system are related to external events (the environment). For this, it must be ensured that each request is answered before the deadline (a time constraint corresponding to the maximum time delay within a request must be completed). For a conventional (not real time) computer system be considered correct, it should provide a proper output to an input. For a real-time system, the correct output must be provided according to the existing deadline.

E-commerce constitutes an important Internet application that is soft real-time and can benefit from the use of WS [Bravetti et. al. 2001]. This happens because in complex e-commerce applications, different companies interact and, obviously, they could have different platforms and languages for their systems. It is also important that e-commerce applications guarantee QoS, avoiding that dissatisfied customers leave the site and do not come back, that would generate monetary losses [Menascé et. al. 2001].

Another important aspect of the QWSC problem is related to the number of concrete WS per abstract WS. In some related papers, this number it is not realistic and the experiments were conducted with a too small search-space size. In [Liu et. al. 2010] were available: two, three, two, five and eight concrete WS for each abstract WS in the WS composition flow. This is much different than the hundreds of concrete WS per abstract WS that can be found in real-world Universal Description Discovery and Integration (UDDI) repositories [Ko et. al. 2008]. In a small search-space size such as adopted in [Liu et. al. 2010], the global optima could be always found just using an ES algorithm.

In this paper, it is defined five QoS attributes that are important to e-commerce applications: availability, cost, response time, reputation and confidentiality. Furthermore, four different GAs were developed to deal with the QWSC problem and some of the results were compared with an ES algorithm (that always finds the global optima). It was defined a deadline to the algorithms and six different search-space sizes. Also a well-planned performance evaluation experiment was realized, to analyze how it contributes to the optimization of GAs for the QWSC problem.

This paper is organized as follows. In Section 2 the concepts related with WS and QoS are presented. Section 3 contains the concepts of Genetic Algorithms. In Section 4, the testing environment where the experiments were executed is described. The experiment design, which includes fixed and variable factors during experiments, is also presented in this section. In Section 5, the results are analyzed according to the response time and QoS obtained. Finally in Section 6, the conclusions are presented and possible future works are discussed.

## 2. WEB SERVICES AND QOS

According to [Alonso et. al. 2004] the definitions of WS range from very generic and all-inclusive to very specific and restrictive. Often, a WS is seen as an application accessible by other applications over the Web. This is a very open definition, under which just about anything that has an URL would be a WS. A more precise definition is provided by the World-Wide Web Consortium (W3C), which characterizes WS as: "A WS is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards" [W3C 2004].The second definition is adopted in this paper because is more detailed, placing the emphasis on the need for being compliant with Internet standards.

According to [Wang et. al. 2010] the increasing popularity of employing WS for distributed systems contributes to the significance of service discovery. However, duplicated and similar functional features existing among services require service consumers to include additional aspects to evaluate the services. How to select the best composite WS plan among available service (WS) candidates is an interesting practical issue.

QoS could be referenced as a set of non-functional properties of WS, such as performance, reliability, availability and security. With the increasing number of WS with similar functionality, service quality measures are used to differentiate the existing services [Kalepu et. al. 2003]. Some of the QoS attributes found in the related works are:

**Availability:** The quality availability is the probability that the service can be accessed and used. It means that this quality is obtained by the number of times the service answers a request divided by the number of total requests [Claro et. al., 2005].

**Cost:** the amount of money charged by the service provider in order to access the service.

**Response time:** it is the time spent between the time when the request is made and the time the client receives the response.

**Reputation:** it is a measure of the client satisfaction by using the service.

**Confidentiality:** determines that only the receiver and the sender must be able to understand the content of the transmitted message.

Given that each WS has its own QoS attributes, to calculate the QoS of the composition plan as a whole, it is necessary to use aggregate functions [Ko et. al. 2008]. For example, the Table 1, adapted from [Ko et. al. 2008], shows an example of aggregation of these attributes:

**Table 1. QoS attributes. Adapted from [Ko et. al. 2008].**

| QoS attribute | Aggregation function |
|---|---|
| Availability | $\prod_{i=1}^{i=n} availability\ (WSi)$ |
| Cost | $\sum_{i=1}^{i=n} cost\ (WSi)$ |
| Response time | $\sum_{i=1}^{i=n} response\_time\ (WSi)$ |
| Reputation | $\sum_{i=1}^{i=n} reputation\ (WSi) * 1/n$ |
| Confidentiality | $\sum_{i=1}^{i=n} confidentiality\ (WSi) * 1/n$ |

In Table 1 the QoS attributes of cost and response time composition are just the sum of individual values for each WS. The availability attribute is given by multiplying the values of each WS. Finally, the attributes of reputation and confidentiality represent an average value of each WS.

These aggregate functions are used when the execution flow of the composition is sequential and, for different types of flows, there are others aggregate functions. But, according to [Luo et. al. 2011] any type of flow could be transformed into the one on sequence. Without loss of generality, it is assumed in this paper the sequential WS composition just as proposed by [Luo et. al. 2011].

The WS composition plan could be described as a sequence of tasks (abstract WS) with an initial and a final task. For any abstract WS, it could have some candidate services (concrete WS) with same or similar functionality but different QoS attributes. Thus, there are various composition plans for each execution path of composite service. For example, if there are one execution path, with 10 abstract WS and 15 concrete WS per abstract, then the number of composition plans should be about $15^{10}$ [Zhang and Ma 2009].

In [Fanjiang et. al. 2010] was mentioned that QWSC could be divided in two aspects: QoS-aware selection and orchestration creation. This paper is focused on QoS-aware selection and does not cover the use of Business Process Execution Language (BPEL) for the creation of the execution flow. This division can be observed in Figure 1.

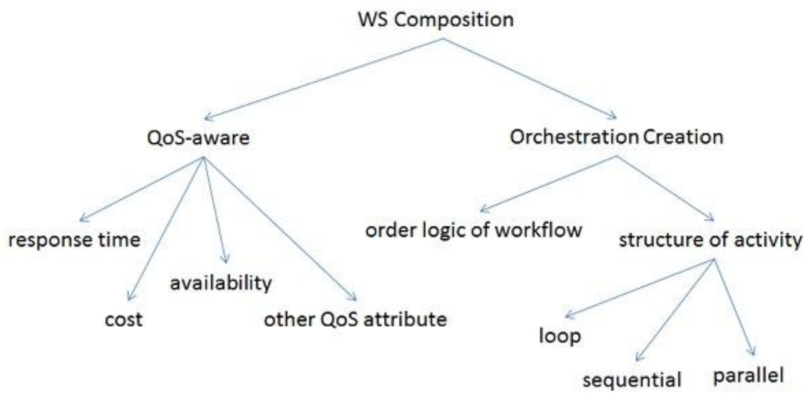

**Figure 1. Aspects of WS Composition. Adapted from [Fanjiang et. al. 2010].**

## 3. GENETIC ALGORITHMS

Genetic Algorithms (GAs) are stochastic optimization methods that work on the principle of evolution via natural selection, employing a population of individuals that undergo selection in the presence of variation-inducing operators such as mutation and crossover. A fitness function is used to evaluate individuals and reproductive success varies with fitness. The steps of a typical GA are as follows [Ai and Tang 2011]:

**1.** Create an initial population;

**2.** Generate new offspring by applying genetic operators, namely selection, crossover and mutation, one after the other;

**3.** Evaluate the fitness value of each individual in the population;

**4.** The process is finished if a desired solution was found or the execution time was expired. Otherwise, the process repeats steps 2 and 3.

In this paper it is adopted the same genetic encoding scheme of [Tang and Ai 2010], so, an individual in the population represents a WS selection plan and it is encoded in an array of *n* integers $X_1, X_2 \ldots, X_n$, where n is the total number of abstract WS in the workflow of the composite WS. In the genetic encoding scheme, each gene represents an abstract WS in the composite WS and a value of the gene represents a concrete WS of the abstract WS.

## 4. PERFORMANCE EVALUATION OF GENETIC ALGORITHMS

### 4.1. Environment Configuration

The main goal of this study is to evaluate different genetic operators to determine which combination is better to solve the QWSC problem. Thus, the test environment is composed of three machines: one representing a client, another a service provider and a third one executes a MySQL server with the data about the QoS attributes of the Web services. In the considered environment, the three machines are in the same network and are linked by a gigabit network switch. Figure 2 illustrates the environment test, were the machines used are heterogeneous and their configuration is presented in Table 2.

Some tools used for developing the testing environment are:

• **Apache Tomcat:** Web container used for deploying the WS of the service provider.

• **MySQL server:** used to store data about the QoS attributes of the WS.

• **Apache Axis2:** is a WS / SOAP / WSDL engine, used to host the WS responsible for fulfilling the requests of the client. The WS receives the request and then executes the proper GA to each request and sends the response to the client.

Table 2. Hardware information of the testing environment.

| Machine | CPU | Clock | Cache | RAM |
|---|---|---|---|---|
| Service provider | Intel (R) Core (TM) 2 Quad | 2.66 GHz | 3 MB | 8 GB |
| MySQL server | Intel (R) Core (TM) i3 | 3.10 GHz | 3 MB | 4 GB |
| Client | Intel (R) Core (TM) 2 Quad | 2.4 GHz | 4 MB | 4 GB |

The experiments were performed using the default configuration of all tools used. The interaction among the machines is as follows: the client requests a WS composition plan to the service provider, the service provider search the MySQL server to get data about the QoS attributes of the WS and then executes one of the GAs and responds to the client witch was the WS composition plan selected and their Normalized Composition Aggregated QoS (NCAQ).

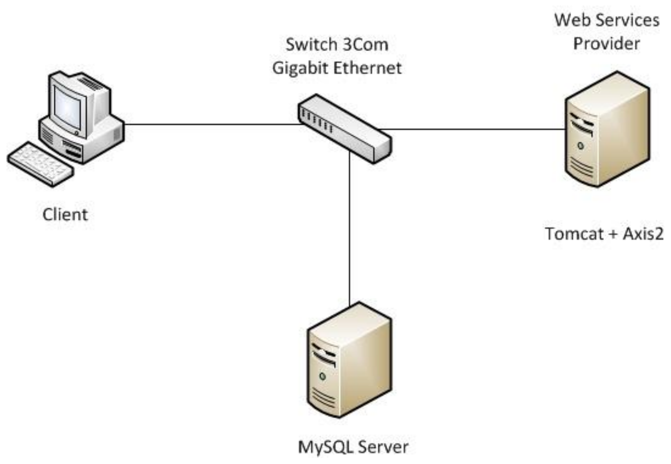

Figure 2. The testing environment.

All experiments have a time that represents the exchange of messages between the three machines in the test environment and the time required to normalize the QoS attributes obtained from the machine running the MySQL server. This time was recorded in ten repetitions and was calculated the average, standard-deviation and confidence interval (with confidence level of 95%). This can be seen in Table 3. This time is already considered in all experiments, so the average response time of the experiments showed in Figure 3 includes it and the time expended to run the GA.

Table 3. The environment time in milliseconds.

| Average | Standard-deviation | Confidence Interval |
|---|---|---|
| 209 | 7 | 5 |

## 4.2. Experiment Design

The experiments were conducted varying three factors (algorithm, number of abstract WS and number of concrete WS per abstract WS) in order to verify the influence of the GA operators in the NCAQ of the obtained WS composition plan. The different combinations of the GA operators can be observed in Table 4. The parameterization of these factors can be observed in Table 5.

**Table 4. Different GAs evaluated.**

| Name | Selection Operator | Elitism |
|---|---|---|
| GA 1 | Roulette wheel | Deactivated |
| GA 2 | Tournament with 16 players | Activated |
| GA 3 | Roulette wheel | Activated |
| GA 4 | Tournament with 16 players | Deactivated |

**Table 5. Experiments parameterization.**

| | |
|---|---|
| Algorithm | GA 1, GA 2, GA 3 and GA 4 |
| Number of abstract WS | 4, 6 and 12 |
| Number of concrete WS per abstract WS | 100 and 200 |

Another key point to be analyzed in the experiment design is the definition of fixed parameters that should be considered in the application parameterization. These factors were: number of generations – 5; deadline – 1,500 ms; mutation operator – deactivated and crossover operator – one-point crossover.

Thus, for each experiment, the client starts a request for the service provider, sending the parameters which search-space to search, i.e. the search-space of the composition flow that has four abstract WS and one hundred concrete WS per abstract; and which GA to use through a Java client program that calls the WS in the service provider responsible for running the selected GA algorithm and gets the data about QoS in the MySQL server. The MySQL table contains the QoS data about the WS, only 10% of the data stored is about the concrete WS that the algorithms should examine; i.e., in the "4 – 100" search-space mentioned above, there is a total of four thousands concrete WS QoS data. In fact, there are many different ways to store the QoS data and some approaches uses semantic web and other technologies such as web ontology. In the work considered in this paper, it is not considered the impact that different approaches could cause in the final response time, although this time could be relevant under certain circumstances. The WS in the service provider, after getting the QoS data, normalizes it using the formulas presented in [Su et. al. 2007]. The WS also starts the selected GA and then sends this normalized QoS data to the GA. After running five generations, the GA selects the best chromosome and sends this information to the WS that responds to the client with the WS composition plan selected (obtained from the chromosome) and the NCAQ of the composition (fitness of the chromosome). For each experiment, this process was repeated thirty times and the average response time, the average NCAQ obtained, the standard-deviation and a confidence interval with 95% of confidence degree, based on t-Student table values, are all calculated. All this was done to give a higher reliability for the result analysis.

Six groups of experiments, representing the six search-space sizes evaluated were defined and can be seen in Table 6. The population size of all experiments also was defined, in a way that they accomplish the fixed parameters, as shown in Table 7.

**Table 6. Groups of experiments.**

| # | Abstract WS | Concrete WS | Algorithm |
|---|---|---|---|
| 1 | 4 | 100 | GA 1, GA 2, GA 3, GA 4 and ES. |
| 2 | 6 | 100 | GA 1, GA 2, GA 3, GA 4 and ES. |
| 3 | 12 | 100 | GA 1, GA 2, GA 3 and GA 4. |
| 4 | 4 | 200 | GA 1, GA 2, GA 3, GA 4 and ES. |
| 5 | 6 | 200 | GA 1, GA 2, GA 3 and GA 4. |
| 6 | 12 | 200 | GA 1, GA 2, GA 3 and GA 4. |

**Table 7. Population of all experiments.**

| Abstract WS | Concrete WS | Population (thousands) | | | |
|---|---|---|---|---|---|
| | | GA 1 | GA 2 | GA 3 | GA 4 |
| 4 | 100 | 2.6 | 35 | 2.5 | 40 |
| 4 | 200 | 2.6 | 35 | 2.5 | 40 |
| 6 | 100 | 2.6 | 35 | 2.5 | 40 |
| 6 | 200 | 2.6 | 35 | 2.5 | 40 |
| 12 | 100 | 2.6 | 32 | 2.5 | 35 |
| 12 | 200 | 2.6 | 32 | 2.5 | 35 |

Table 1 presents the aggregate functions of QoS attributes considered in this article. However, it is also need a way to assess the QoS of the composition as a whole, taking into account the QoS attributes defined. In [Canfora et. al., 2005] was presented a function that achieves this goal, it can be seen in Equation 3. Equation 4 shows the function to be optimized in the experiments in this article. It has no weights as Equation 3, as defined in this article that all QoS attributes have the same weight. Given that the QoS attributes were normalized in a way that 0 is the worst result and 1 is the best result possible, simply add up all the attributes of QoS, no matter whether they should be minimized or maximized.

**Equation 3. Function to be optimized in [Canfora et. al., 2005].**

$$F(x) = \frac{Availability_{w1} + Reliability_{w2}}{Cost_{w3} + Response\_time_{w4}}$$

**Equation 4. Function to be maximized in all experiments.**

$$F(x) = Availability + Cost + RespTime + Reputation + Confidentiality$$

**Table 8. Example of Calculating the Normalized Composition Aggregated QoS (NCAQ).**

| **Name** | **Cost** | **Resptime** | **Availability** | **Reputation** | **Confidentiality** |
|---|---|---|---|---|---|
| WSc1 | 1 | 1 | 1 | 1 | 1 |
| WSc2 | 1 | 1 | 1 | 1 | 1 |
| WSc3 | 1 | 1 | 1 | 1 | 1 |
| WSc4 | 1 | 1 | 1 | 1 | 1 |
| Aggregated QoS | 4 | 4 | 1 | 1 | 1 |
| Composition Aggregated QoS | (4 + 4 + 1 + 1 + 1) = 11 | | | | |
| Normalized Composition Aggregated QoS | 11 / 11 = 1 | | | | |

First, for each QoS attribute, the aggregated QoS is calculated using the formulas presented in Table 1. Thereafter, the composition aggregated QoS is computed using the formula shown in Equation 4. Finally, this number is normalized between 0 and 1 and called Normalized Composition Aggregated QoS (NCAQ). This is the attribute cited in Figure 4.

## 5. RESULTS ANALYSIS

In this section, the response times of the five algorithms and NCAQ obtained are analyzed. These two metrics were compared and the influence of factors was calculated for the four GAs. In Figure 4, the horizontal axis represents the six different search-space sizes evaluated, i.e. 4 -100 means a composition with four abstract WS and one hundred concrete WS per abstract. The vertical axis represents the average NCAQ obtained; normalized between 0 and 1 (the higher is the better). The algorithm called Exhaustive Search (ES) always guarantees the global optima.

### 5.1 Time analysis

A deadline of 1,500 milliseconds was defined for the algorithms. This deadline is realistic because, as showed in Section 1, e-commerce applications are soft real-time systems and the users will not accept long delays in order to conclude their purchases.
The ES algorithm does not attend this deadline in any of the six search-spaces evaluated, but it was evaluated in some cases to see how long it takes and also to

found the global optima. Figure 3 shows the response times of the experiments. The vertical axis represents the average response time in milliseconds (30 repetitions) and the horizontal axis represents the six different search-spaces sizes evaluated. There is also the confidence interval (with 95% of confidence degree) in the lines inside the columns. For any search-space size, the four GAs has their confidence interval overlapped which means that they are statistically equal.

The ES algorithm was tested in the experiment groups one, two and four. In experiment group one it gets really close to the established deadline, with an average response time of 1,768 milliseconds. In experiment two it takes about forty hours, clearly unacceptable for an e-commerce application. In experiment group four takes about 171 seconds, another unacceptable response time for the experiments. With these results, the conclusion is that even for small search-space sizes, an ES algorithm does not comprise a good choice.

The four GAs have the same statistical average response time and follow the 1,500 milliseconds established deadline. This is an important point because if they had different response times, it could benefit the slower one with a better NCAQ obtained.

## 5.2 QoS analysis

Figure 4 shows the average NCAQ obtained by the five algorithms analyzed. In experiment group one, the GA 2 is clearly better than the other three GAs, with the average NCAQ of 0.783. The GA 4 is the one that is closest to GA 2, but the confidence intervals of both are not overlapped and so the GA 2 is statistically superior. The difference between the average NCAQ of GA 2 and ES is 0.018 indicating that GA 2 is near to the global optima. GA 1 and GA 3 are statistically equals because their confidence intervals are overlapped and both are worse than GA 2 and GA 4.

In experiment group two both GA 2 and GA 4 are statistically equals, because their confidence interval are overlapped. In the same way that experiment group one, GA 2 and GA 4 are better than GA 1 and GA 3. Again, GA 1 and GA 3 are statistically equals. In these experiments, the difference between GA 2 and ES was more significant with 0.06.

The experiment group three shows that GA 2 and GA 4 are statistically equals. GA 1 and GA 3 are also statistically equals. It was not possible to execute the ES to found the global optima because it would probably take thousands of years.

For the experiment group four, GA 2 is better than GA 4 because their confidence intervals are not overlapped. GA 1 and GA 3 are statistically equals. The difference between the ES and GA 2 was 0.05.

Experiment group five shows that GA 2 and GA 4 are statistically equals. GA 1 and GA 3 are also statistically equals. The ES was not tested in this group of experiments because it would probably take too long.

In experiment group six, GA 2 and GA 4 are statistically equals. GA 3 was a little better than GA 1. The ES was not tested in this group of experiments too.

Considering the six groups of experiments the GA 2 was better in two search-space sizes (4-100 and 4-200) and statistically equal to the GA 4 in the others four search-space sizes. This is already enough to point the GA 2 as the better one. But, another desirable characteristic of the GAs is the stability of their results. In fact, the results in Figure 4 shows that in all experiments GA 2 has a smaller confidence interval (excluding the ES for sure) indicating that is also the most stable of the four GAs.

## 5.3 Influence of Factors

The influence of factors was also calculated: selection operator and elitism operator. This was calculated only for the four GAs and it is related to the average NCAQ obtained. The results can be seen in Table 9. In all six groups of experiments the higher influence was from the selection operator, pointing the tournament with sixteen players much better than the roulette wheel. The elitism operator has much less influence, probably because it increases the time needed to execute the algorithm. As the deadline is fixed, in counterparty was necessary to decrease the population size. This decrease in the population size probably decreased the benefit of the elitism operator.

**Table 9. Influence of factors.**

| Influence of Factors (%) | | | | |
|---|---|---|---|---|
| **#** | **A – Selector Operator** | **B – Elitism** | **AB** | **Total** |
| 1 | 97.149 | 2.844 | 0.007 | 100 |
| 2 | 99.856 | 0.140 | 0.004 | 100 |
| 3 | 99.814 | 0.003 | 0.183 | 100 |
| 4 | 96.205 | 2.252 | 1.542 | 100 |
| 5 | 99.952 | 0.043 | 0.005 | 100 |
| 6 | 98.035 | 0.588 | 1.377 | 100 |

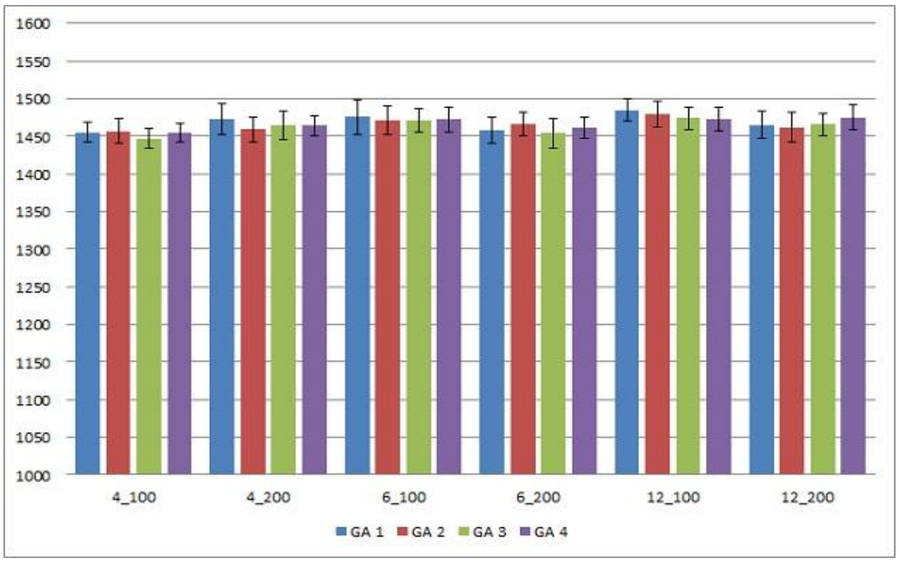

**Figure 3. The average response time of the four GA algorithms.**

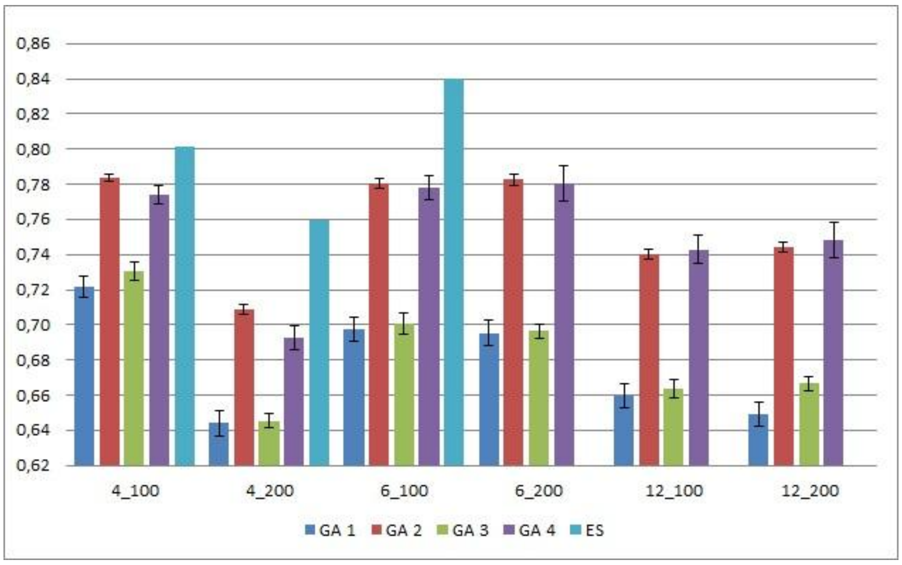

**Figure 4. The average NCAQ of all five algorithms.**

## 6. Conclusions and Future Works

In this paper five different algorithms were presented and evaluated: an exhaustive search (ES) and four different GAs. Such algorithms were evaluated considering the response time, the accomplishment of the established deadline and the NCAQ obtained. A testing environment was configured to enable the implementation of the experiments. As mentioned in Section 1, the deadline is important because e-commerce applications are soft real-time systems and it is necessary to find good solutions within the established deadline.

Another interesting result shows that is possible to use the design of experiments and performance evaluation to improve GAs focused on QWSC. In the experiments considered in this paper, the selector operator and the elitism operator were evaluated. Therefore, future works can consider a more complete performance evaluation including more GA operators, such as: crossover operator, mutation operator, mutation rate, crossover rate and so on.

Furthermore, different deadlines can be established and by doing that, it is possible to discover that for a given deadline parameter one GA is better. For other ones, an ES algorithm or a heuristic algorithm could also be better. Thus, it is also possible to improve the GAs and create different hybrid algorithms and use the design of experiments and performance evaluation to evaluate them.

In future works is planned to analyze the influence of factors: number of abstract WS in the composition flow and number of concrete WS per abstract. By doing that, could be possible to determine which algorithm is the best one for each search-space size; i.e. for a "2 – 100" search-space size the ES algorithm is probably the better choice, because guarantees the global optima and comprise the deadline; for a "4 – 100" search-space size GA 2 is the better choice. Combining the information of the influence of the deadline and the influence of the search-space size, it could be possible to dynamically determine which algorithm is the best (or at least, a good choice) for each requisition to a WS Composition Engine, that will execute the appropriate QWSC algorithm.

It is also planned in the next steps to use the design of experiments and performance evaluation to develop new hybrid algorithms that probably should have a better performance than some of the GAs present in the related works found in the literature.

## References

G. Canfora, M. Di Penta, R. Esposito, M. L. Villani. (2005) "An approach for QoS-aware Service Composition based on Genetic Algorithms." In: Proceedings of ACM Genetic and Evolutionary Computation Conference. GECCO ´05.

J. M. Ko, C. O. Kim, I. Kwon. (2008) "Quality-of-service oriented web service composition algorithm and planning architecture." The Journal of Systems and Software (JSS), vol. 81, pp. 2079-2090, Elsevier.

B. Batouche, Y. Naudet, F. Guinand. (2010) "Semantic Web Services Composition Optimized by Multi-Objective Evolutionary Algorithms." In: Proceedings of the Fifth International Conference on Internet and Web Applications and Services. IEEE Computer Society.

Y. Fanjiang, Y. Syu, C. Wu, J. Kuo, S. Ma. (2010) "Genetic Algorithm for QoS-aware Dynamic Web Services Composition." In: Proceedings of the International Conference on Machine Learning and Cybernetics. IEEE Computer Society.

H. Liu, F. Zhong, B. Ouyang, J. Wu. (2010) "An approach for QoS-aware Web Service Composition based on Improved Genetic Algorithm." In: Proceedings of the International Conference on Web Information Systems and Data Mining. IEEE Computer Society.

M. Bravetti, R. Lucchi, R. Zavattaro, G. Gorrieri. (2001) "Web Services for e-commerce: guaranteeing security access and quality of service." In: Proceedings of the ACM Conference on e-commerce.

D. A. Menascé, D. Barbará, R. Dodge. (2001) “Preserving QoS of e-commerce sites through self-tuning: A Performance Modelo Approach.” In: Proceedings of the ACM Conference on e-commerce.

L. Ai, M. Tang, C. Fidge. (2011) “Partitioning composite web services for decentralized execution using a genetic algorithm.” Journal of Future Generation Computer Systems (FGCS), vol. 27, pp. 157-172, Elsevier.

R. Jain. (1991) “The Art of Computer Systems Performance Analysis: techniques for experimental design, measurement, simulation and modeling.” Wiley.

G. Alonso, F. Casati, H. Kuno, V. Machiraju. (2004) “Web Services: Concepts, Architecture and Applications.” Springer-Verlag, 1st edition.

P. Wang, K. Chao, C. Lo. (2010) “On optimal decision for QoS-aware composite service selection. Expert Systems with Applications.” Vol. 37. pp. 440 – 449. Elsevier.

S. Kalepu, S. Krishnaswamy, S. W. Loke. (2003) “Verity: A QoS metric for selecting web services and providers.” In Proceedings of the Fourth International Conference on Web Information Systems Engineering Workshops (WISEW ’03). IEEE CS.

Y. Luo, Y. Qi, D. Hou, L. Shen, Y. Chen, X. Zhang. (2011) “A novel heuristic algorithm for QoS-aware end-to-end service composition.” Journal of Computer Communications, vol. 34, pp. 1137-1144, Elsevier.

C. Zhang, Y. Ma. (2009) “Dynamic Genetic Algorithm for Search in Web Service Compositions based on Global QoS Evaluations.” In: Proceedings of the International Conference on Scalable Computing and Communications. IEEE Computer Society.

E. E. Silva. (2001) “Optimization of reinforced concrete structures using genetic algorithms.” (master thesis) University of São Paulo.

B. L. Miller, D. E. Goldberg. (1995) “Genetic Algorithms, Tournament Selection , and the Effects of Noise.” Technical Report number 95006, University of Illinois at Urbana-Champaign.

R. Linden. (2008) “Genetic algorithms an important tool of computational intelligence.” 2nd edition, Brasport.

M. Tang, L. Ai. (2010) “A Hybrid Genetic Algorithm for the Optimal Constrained Web Service Selection Problem in Web Service Composition.” In: Proceedings of the World Congress on Computational Intelligence, Barcelona, Spain.

S. Su, C. Zhang, J. Chen. (2007) “An improved genetic algorithm for web services selection.” In: Proceedings of the International Conference on Distributed applications and interoperable systems. ACM.

A. M. K. Cheng. (2002) “Real-time systems: Scheduling, analysis and verification.” 1 ed. Wiley.

W3C. (2004) “Web Services Architecture.” Available at: http://www.w3.org/TR/ws-arch/#whatis. Last access: 26/02/2012.