# QoS-aware Selection of Web Service Compositions using Harmony Search Algorithm

Nastaran Jafarpour, Mohammad Reza Khayyambashi
Department of Computer Engineering
Faculty of Engineering
University of Isfahan
Isfahan, Iran
jafarpour@eng.ui.ac.ir
m.r.khayyambashi@eng.ui.ac.ir

*Journal of Digital Information Management*

***ABSTRACT:*** *Extensive use of web services has lead to introduce web service compositions to execute business workflows. Since several functionequivalent services, provided by different service providers, may be available in the Web, the problem of selection of web service compositions arises. QoSaware selection means considering the quality of the service composition in choosing constituent services. So a method is needed for computing the quality of a composition and determining the composition with the (almost) best quality. The resulting composite has optimal quality and satisfies user's quality constraints. In this paper, the problem of QoSaware selection of web service composition is modeled as an optimization problem. Then a variation of Harmony Search algorithm is used to find a nearoptimal composition. Harmony Search is a recentlydeveloped optimization algorithm that mimics the improvisation process. The proposed method based on Harmony Search has been applied to a large variety of composition schemas, generated by a simulation software, and the results show that the proposed method works faster, compared with other approaches.*

**Categories and Subject Descriptors**
**H.3.5 [Online Information Services]:** Web-based services

**General Terms:** Measurement, Performance

**Keywords:** Web Service Composition, Quality-of-Service, Harmony Search Algorithm, Global Optimization, Workflow

**Received:** 11 November 2009, Revised 19 March 2010, Accepted 3 April 2010

## 1. Introduction

A web service is an XML-based software application that enables program-to-program interactions. Each web service performs a specific task or a set of simple tasks. It can be advertised by service providers and invoked by service requesters through the Web, using a set of standard protocols such as WSDL, SOAP, and UDDI. Web services work independently from their location, platform, and implementation. Standard protocols adaptation and location independency have enabled web services to integrate together and produce a value-added larger service called composite web service. A web service composition is a partially-ordered collection of atomic web services that executes a structured workflow. A web service composition is specified by a composition schema. A composition schema is a high level service process model which consists of atomic processes and control flows. An atomic process is a service operation that can be implemented by a concrete web service. At run time, each atomic process within the composition schema must be bound to an available concrete web service offered by a service provider on the Web. A composition plan is a realized composition schema where a concrete web service is assigned to each atomic process.

The development of web service compositions has lead to focus on its management aspects. Web service composition management can be categorized in following steps: service design, service discovery, service selection, and service execution monitoring. In service design phase, a business workflow is mapped to a combination of abstract services and a composition schema is created. A composition schema consists of several abstract services. An abstract service is a semantic description of a particular functionality. Each abstract service matches to one or more concrete services published on the Web. Available concrete services are found during service discovery phase. The role of service selection phase is binding a single concrete service to each abstract service and creating a composition plan. By monitoring the service execution during the last phase of composition management, the system is able to refine the composition plan in future executions if it is necessary.

A composite web service consists of multiple abstract services. Matching each abstract service, there may be found several services with the same functionality offered by different service providers. The quality of service (QoS) is a key factor in distinguishing function-equivalent services. The problem of QoS-aware selection of web service compositions denotes selecting a concrete service for binding to each abstract service while obtaining the highest possible QoS of the composition and satisfying local and global constraints defined by the user. Local constraints restrict QoS attributes of a single service within the composition while global constraints define an upper or lower bound for aggregated QoS values of a composite service. Indeed, QoS-aware selection enforces some given quality criteria and also tries to maximize the quality of composition during service selection phase. Considering the availability of a composition schema and the corresponding set of discovered services, this paper mainly focuses on selection of composition plan while optimizing the resulting QoS. In this paper, an evolutionary optimization algorithm called Harmony Search algorithm is applied to find compositions with optimal quality. Experimental results show that the proposed method performs the selection of optimal service composition so faster and can gain higher QoS in comparison with the method based on GA. Since the proposed method requires fewer computational

efforts, it is suitable for Web environments, even if the number of atomic services in the composition increases.

The rest of this paper is organized as follows. Some related literature is reviewed in section 2. In section 3, the definitions of QoS attributes and constraints, the model of problem, and the process of computing quality of web service compositions are explained. Section 4 describes Harmony Search algorithm and subsequently applies it for QoS-aware selection, then expresses the determination of Harmony Search parameters. The simulation software used in this wor k is introduced in section 5. Finally, the numerical simulations and conclusion are presented in section 6 and 7 respectively.

## 2. Related work

QoS-aware selection of composite web services introduces a global optimization problem with multiple constraints. Finding an optimal combination of concrete services among a large number of possible solutions takes significant computation efforts. The QoS-aware selection problem has been modeled as a Multi-choice Multi-dimension 0-1 Knapsack Problem (MMKP) (Yu and Lin, 2005). In MMKP, there are $K$ groups with $n_i (1 \le i \le K)$ items, each item has a profit $p_{ij}$ and requires resource $r_{ij} = (r_{ij1}, ..., r_{ijm})$. The total available resources in the knapsack are $R = (R_1, ..., R_m)$. The goal of MMKP is to select one item from each group to be included in the knapsack while maximizing the total profit considering resource constraints. The MMKP problem is NP-hard.

Different methods based on applied mathematics and artificial intelligence have been proposed to solve this problem. For computing the aggregated QoS based on QoS of atomic services, a mathematical model is presented in which reduction rules and aggregation formulas are applied on the workflow iteratively until a single task remains in the flow (Cardoso et al., 2004). This task will express the total QoS of the whole workflow. Also a middleware platform has been developed for QoS-driven selection of web services based on linear programming (Zeng et al., 2004). Using linear programming imposes linearization of the constraints. Moreover, linear increment in the number of abstract services and candidate services leads to exponential growth of algorithm's computational time. To overcome these disadvantages, Genetic Algorithm (GA) has been applied to select the (near)-optimal composition (Canfora et al., 2004).

The impact of different parameters of GA, e.g. mutation rate and fitness function is investigated in (Jaeger and Muhl, 2007). Experimental results of that work show that Genetic Algorithm offers a good performance at computational effort in comparison with some other heuristics; however experiments are based on a single global constraint. Handling of population diversity in GA is proposed in (Zhang et al., 2007) by using simulated annealing in order to prevent prematurity of GA. In this literature, the relation matrix coding schema is adopted for coding genomes. This new coding schema is able to represent all paths of the service selection, re-planning, and cyclic paths simultaneously. However it leads to slower convergence to optimal solution and reduces the overall performance of the selection algorithm. In (Jin et al., 2008) a combination of genetic algorithm and Ant Colony algorithm is proposed for composition selection. This approach can put feedback information to original GA and prevent large redundant repeats. However it is not suitable for large scale web service compositions.

A hybrid meta-heuristic method which combines Tabu Search and Simulated Annealing is proposed by (Ko et al., 2008) to search for a high quality constraint-compliant service composition plan. Applying tabu list and probabilistic move to inferior plans aims this approach to find solutions very faster.

The proposed method of this paper based on Harmony Search algorithm selects optimal service compositions faster in comparison with other approaches while it uses less computational efforts and does not impose linearization of the constraints. The proposed approach can take several users' quality constraints into account while optimizing the total quality of service composition plan.

## 3. Quality of service

### 3.1 QoS attributes

The quality of services can be characterized according to various dimensions called QoS attributes or QoS classes. QoS attributes can be divided into two types, i.e. cost and benefit (Xiong et al., 2007). Execution cost and response time are examples of cost attributes. The higher the value of cost attributes, the less the quality of service. Benefit attributes are like availability, reliability, and reputation. The higher the value of benefit attributes, the better the quality. In some literature, cost attributes are called negative attributes or attributes with decreasing direction while benefit attributes are positive attributes with increasing direction (Zeng et al., 2004, Jaeger and Rojec-Goldmann, 2006, Jaeger and Ladner, 2006).

This work has chosen four following QoS attributes for doing experiments however it can flexibly take any other attribute into account.

- **Response time:** The delay time in milliseconds between sending a request and receiving the results. This duration consists of the processing time of the service and data transmission time.
- **Execution price:** The fee that a service requester has to pay for invoking a service. It is directly advertised by the web service providers.
- **Reliability:** The probability of correctly responding to a request within the expected time indicated in the web service description. Its value is computed according to data of past invocations using the expression $q_{reliability}(s)$= (*number of times of successfully delivering the service*) / (*total number of invocations*).
- **Availability:** The probability that a service is accessible. Its value is computed by $q_{availability}(s)$= *(total time in which the service is available within the time frame*) / (*the time frame*) (Zeng et al., 2004).

The QoS information can be gained from service providers e.g. cost, or from service requesters' feedback e.g. response time, or from a third party or broker e.g. service reputation.

Different users may give different importance to QoS attributes. So the QoS-aware selection must be able to take the preferences of users into account by using weighted attributes (Xiong et al., 2007).

### 3.2 QoS of service compositions

This work assumes a composition schema *S* including *n* abstract services, $S = \{s_1, s_2, ..., s_n\}$. For service $s_i$ $(1 \le i \le n)$ within the composition, there are $m_i (mi \ge 1)$ discovered candidate services which are indexed in vector $C_i$. Each service is shown as a vector of quality attributes i.e. response time, cost, reliability, and availability.

#### 3.2.1 QoS normalization

The scopes of quantified attributes are too different from each other to be compared in a fairly manner. For example, availability is a probability ratio and varies between 0 and 1 while response time is expressed in milliseconds by a positive

number. Moreover, cost attributes have decreasing direction while benefit attributes have increasing direction. A possible solution to overcome these inconsistencies and perform a fair QoS estimation is normalizing values of QoS attributes in the range of (0, 1). Therefore values near zero indicate lower quality meanwhile values near one suggest upper quality. Equation (1) is used to normalize benefit attributes and equation (2) is for normalizing cost attributes (Zeng et al., 2004).

$$n_{i,j} = \begin{cases} \dfrac{q_{i,j} - q_i^{\min}}{q_i^{\max} - q_i^{\min}} & if \quad q_i^{\max} \neq q_i^{\min} \\ 1 & if \quad q_i^{\max} = q_i^{\min} \end{cases} \qquad (1)$$

$$n_{i,j} = \begin{cases} \dfrac{q_i^{\max} - q_{i,j}}{q_i^{\max} - q_i^{\min}} & if \quad q_i^{\max} \neq q_i^{\min} \\ 1 & if \quad q_i^{\max} = q_i^{\min} \end{cases} \qquad (2)$$

In these equations, $q_{i,j}$ is a QoS attribute of $j$ th candidate service of $i$ th abstract service in the composition schema. Normalizing equations are applied to every attributes of candidate services in each vector $C_i$ independently. As an example, the minimum and maximum values of response time of candidate services matching abstract service $i$ are found and set to $q_i^{min}$ and $q_i^{max}$ in (2); then the value of response time of each service in this vector, $q_{i,j}$, is normalized to $n_{i,j}$.

### 3.2.2 QoS aggregation

To optimize the quality of service composition, a method must be applied to estimate the QoS of a service composition from its constituent services. This estimation is called QoS aggregation. QoS aggregation formulas are defined for each pair of QoS attribute and composition pattern and applied during the reduction of workflows. Some QoS aggregation formulas, for basic composition patterns and major QoS attributes are collected in table 1, proposed by (Cardoso et al., 2004) and (Canfora et al., 2004). In these aggregation formulas, $n$ is the number of services in the composition. In third column, $p_i$ denotes the probability of taking $i$th branch of a conditional structure, like switch or if construct. The probability of taking branches is assumed to have uniform distribution for the first run. For next executions, it is estimated based on previous executions by the execution broker.

For computing the aggregated QoS of a workflow, it must be reduced by reduction rules. Reduction rules are repeatedly applied to the workflow until only one atomic process remains. The remaining process contains the aggregated values of QoS attributes of the whole workflow (Cardoso et al., 2004). As an example, the reduction rule for sequential patterns is as follows: Two sequential process $p_1 = (t_1, c_1, r_1, a_1)$ and $p_2 = (t_2, c_2, r_2, a_2)$ can be reduced to one process $p_{12}$ with response time of $t_1+t_2$, cost of $c_1+c_2$, reliability of $r_1*r_2$, and availability of $a_1*a_2$. Figure 1 illustrates the processes of the workflow reduction and QoS aggregation for a sample composition schema.

| QoS attribute | Sequential | Parallel | Conditional |
|---|---|---|---|
| Time (T) | $\sum_{i=1}^{n} T(S_i)$ | $Max\ \{T(S_i)^{i\in\{1\ldots n\}}\}$ | $\sum_{i=1}^{n} p_i \cdot T(S_i)$ |
| Cost (C) | $\sum_{i=1}^{n} C(S_i)$ | $\sum_{i=1}^{n} C(S_i)$ | $\sum_{i=1}^{n} p_i \cdot C(S_i)$ |
| Availability (A) | $\prod_{i=1}^{n} A(S_i)$ | $\sum_{i=1}^{n} C(S_i)$ | $\sum_{i=1}^{n} p_i \cdot A(S_i)$ |
| Reliability (R) | $\prod_{i=1}^{n} R(S_i)$ | $\prod_{i=1}^{n} A(S_i)$ | $\sum_{i=1}^{n} p_i \cdot R(S_i)$ |

Table 1. QoS aggregation formulas per composition pattern and QoS attribute

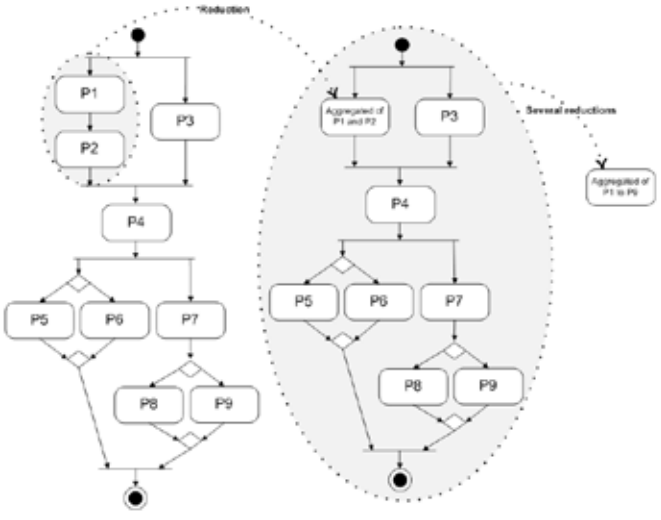


Figure 1. An example of workflow reduction and QoS aggregation for a composition schema

The objective of QoS-aware selection problem is to optimize multiple QoS attributes so it must consider them altogether. For this purpose, normalized values of QoS attributes are added using Simple Additive Weighting (SAW), then the result is optimized.

According to SAW, the overall QoS of a composition plan is defined by (3),

$$QoS(composite) = \sum_{i=1}^{4} w_i \,.N_i\ (composite) \qquad (3)$$

where $N_i$*(composite)* is the aggregated of normalized values of attribute $i$ and the weight of each attribute $(w_i)$ is its importance degree defined by user preferences and $\sum_{i=1}^{4} w_i = 1$ . QoS-aware selection tries to maximize the overall quality of service composition expressed by (3). Equation (3) is called the objective function or fitness function.

### 3.3 Constraints definition

As mentioned in section 1, the solution of QoS-aware selection algorithm must not only have the optimal QoS, but also meet the user's local and global constraints. The model of local constraints is defined as:

*{a benefit attribute of* $s_i \geq$ *| a cost attribute of* $s_i \leq$ *} (a given threshold).*

Similarly, global constraints define upper or lower bound of aggregated QoS values of a composite service:

*{a benefit aggregated attribute of composition plan* $\geq$ *| a cost aggregated attribute of composition plan* $\leq$ *} (a given threshold).*

Note that aggregated value of real values of QoS attributes must be examined in users' constraints, instead of normalized values.

## 4. Approach description

### 4.1 Harmony Search algorithm

Harmony Search (HS) is a meta-heuristic evolutionary optimization algorithm, developed in 2001 by Geem et al.

(Geem and Kim, 2001). It imitates musical process of searching for a perfect state of harmony. Harmony Search algorithm is simple in concept and few in parameters. It imposes few mathematical requirements and can easily be implemented. HS is a new heuristic optimization algorithm that can produce better solutions than other existing algorithms in less number of iterations. It does not require setting initial values for decision variables, thus it may escape local optima.

In Harmony Search algorithm, the harmony memory (HM) is a memory location where all current solution vectors (sets of decision variables) are stored. In each evolution of Harmony Search, if a solution vector with relatively good fitness is generated, it will be saved in harmony memory and might be used in next generations.

Harmony Search steps are as follows (Lee and Geem, 2005):

Step 1: Initialize the problem and algorithm parameters.

Step 2: Initialize the harmony memory.

Step 3: Improvise a new harmony. Step 4: Update the harmony memory.

Step 5: Repeat steps 3 and 4 until satisfying termination criterion.

In step 1, the objective function with N decision variables and the set of possible values for each decision variable are defined. The HS algorithm parameters are also specified in this step. HS parameters include: harmony memory size (HMS), harmony memory considering rate (HMCR), pitch adjusting rate (PAR), and the termination criterion (maximum number of searches).

In step 2, the HM matrix is filled with as many randomly generated solution vectors as the HMS.

In step 3, a new harmony vector like $x= (x_1, x_2, \ldots, x_N)$ is generated based on three rules: (1) memory consideration, (2) pitch adjustment and (3) random selection. In the memory consideration, the value of a decision variable for the new vector is chosen from any of the values for that decision variable in the specified HM. The HMCR, which varies between 0 and 1, is the rate of choosing one value from the historical values stored in the memory while (1-HMCR) is the rate of randomly selecting one value from the possible range of values. Every component obtained by the memory consideration is examined to determine whether it should be pitch-adjusted. In pitch adjustment, the current value of a decision variable is replaced with one of its neighboring value. The probability of pitch adjustment is specified by PAR parameter. During improvisation (generating a new harmony), HMCR and PAR are used to improve the solution vector globally and locally respectively. Memory consideration, pitch adjustment or random selection is applied to each variable of the new harmony vector in turn.

In step 4, if the new harmony vector is better than the worst harmony in the HM, judged in terms of the objective function value, the new harmony is included in the HM and the existing worst harmony is excluded from the HM.

In step 5, if the stopping criterion (maximum number of improvisations) is satisfied, evolution is terminated. Otherwise, Steps 3 and 4 are repeated. Finally, the best vector of memory in terms of objective function value is the (near)-optimal solution.

The Harmony Search incorporates the structure of existing heuristic methods. It preserves the history of past vectors in Harmony Memory similar to Tabu Search, and is able to vary the adaptation rate, i.e. harmony memory considering rate, from the beginning to the end of computation resembling Simulated Annealing, and manages several vectors simultaneously in a manner similar to Genetic Algorithm. However, the major difference between GA and HS is that HS makes a new vector from all the existing vectors, while GA makes the new vector only from two of the existing vectors as parents. Moreover, HS can independently consider each component variable in a vector while it generates a new vector, but GA cannot because it has to keep the structure of a gene (Geem and Kim, 2001).

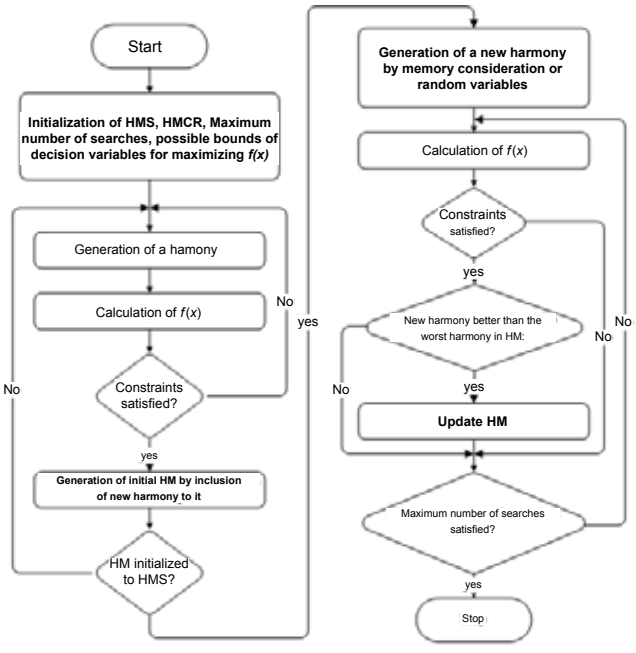


Figure 2. Harmony Search Algorithm

In addition to finding optimal solution of an objective function, Harmony Search algorithm is able to impose constraints on solution vectors. For this purpose, generated vectors in memory initialization phase and improvisation phase must be checked whether they satisfy the constraints and before entering the memory. Figure 2 depicts the detailed procedure of Harmony Search algorithm for a constrained optimization (Lee and Geem, 2005). In this figure, main steps of HS are bolded.

### 4.2 QoS-aware selection of composite services with Harmony Search

As described in section 3.2.2, the objective function of quality optimization problem can be defined as equation (3). Since $N_i$*(composite)* in this function shows the aggregated QoS attribute of the composition, it is a function of QoS attributes of single abstract services. Considering a service composition containing $n$ abstract services, the objective function has $n$ decision variables, $\{s_1, s_2, \ldots, s_n\}$. Each decision variable is a vector with four elements (QoS attribute values). Assuming solution vectors are indices of selected candidate services in composition plan, decision variable $s_i$ $(1 \le i \le n)$ can vary between 1 and $m_i$ (the number of discovered candidate services matching abstract service $s_i$).

Candidate services stored in each $C_i$ vector are semantically equivalent, but their QoS values are not relevant to each other. So pitch adjustment of a decision variable to its neighboring values cannot lead to improvement of the value of objective function. It works just like random selection in improvisation. Thus, pitch adjustment is omitted from improvisation phase of this application. So solution vector generation is performed by considering harmony memory (with probability HMCR) and random selection (with probability 1-HMCR).

When HS starts, the memory is initialized by randomly generated compositions. During evolution of HS, the memory is filled up by compositions with better quality; and at the end of

evolution, the best composition is introduced as the result. If several compositions with the same best quality are found, the final solution is randomly chosen among them.

In the case that the user defines quality constraints, candidate services are initially filtered by local constraints. In order to satisfy global constraints, each newly generated composition must meet two criteria before being stored in memory: 1) Its aggregated quality satisfies global constraints, 2) Its quality is better than the worst composition included in memory.

Harmony Search suggests to initialize the memory with the vectors that meet the constraints and then begin to evaluation (Lee and Geem, 2005). However it is a riskful initialization since it is possible that the number of constraints-compliant composition plans is less than HMS and initialization phase becomes an indefinite loop. If no constraints-compliant solution is found by the algorithm, it will be announced to the user to relax quality constraints.

### 4.3 Harmony Search parameters

Since appropriate values for HS parameters varies from one problem to another, several evaluations have been done in order to choose the values of HS parameters that lead to better performance of this algorithm in QoS-aware selection problem. In this section, the value of Harmony Memory Consideration Rate (HMCR), Harmony Memory Size (HMS), and the maximum number of searches are investigated. Three test cases have been used to determine these parameters while each test case has an arbitrary composition schema with 10, 15, or 20 abstract services. Each evaluation has been repeated for 20 times to avoid the effects of randomness.

#### 4.3.1 Harmony Memory Consideration Rate (HMCR)

HMCR is the rate of choosing values from the memory. An HMCR value of 1.0 or near 1.0 is not recommended by the algorithm's designer because the solution may be improved by values not stored in the memory (Lee and Geem, 2005). Indeed, HMCR works like mutation rate in the Genetic Algorithm to overcome local optimal solutions. The effect of different values of HMCR in QoS-aware selection is illustrated in figure 3. QoS ratio is the rate of QoS of the final solution to the QoS of the best solution ever has been found. The more the QoS ratio, the better the optimality of the solution. It is concluded that the larger values of HMCR lead to better QoS ratio. Because small values of HMCR do not care about the experience of previous searches and test new values. In next simulations, HMCR=0.7 is used.

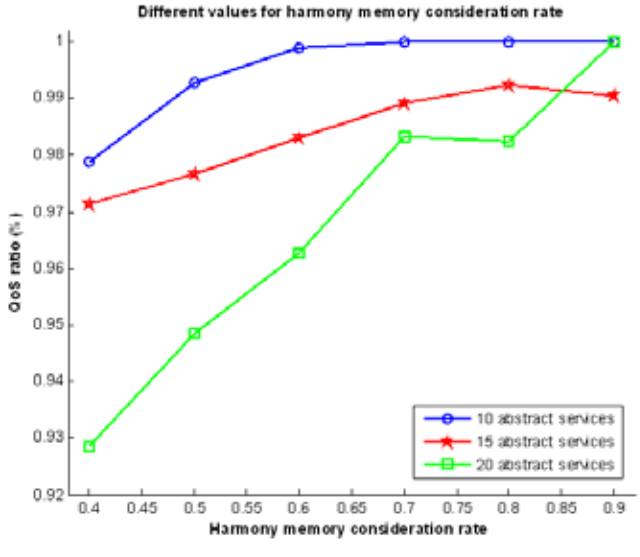


Figure 3. Different values for harmony memory consideration

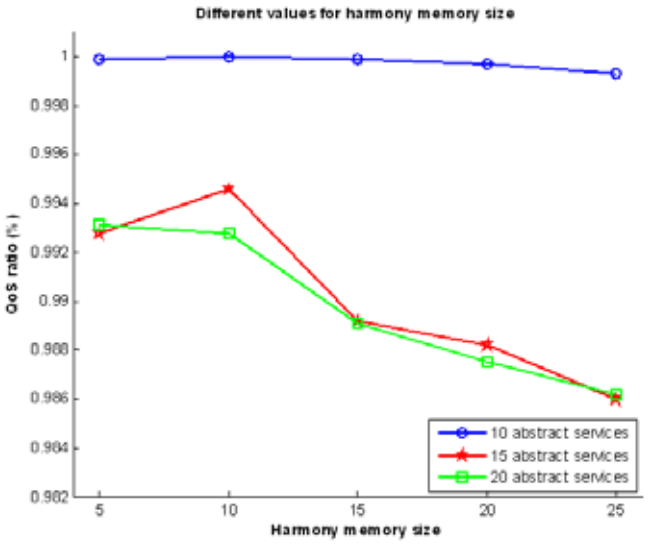


Figure 4. Different values for Harmony memory size rate

#### 4.3.2 Harmony Memory Size (HMS)

HMS is the number of randomly generated solutions in first execution and the number of good solutions that are kept from previous executions. It resembles the population size in the Genetic Algorithm. The impact of HMS is shown in figure 4. Small values of HMS prevent the diversity of the solutions and may lead to local optimal. The value of 10 is used for HMS in next simulations of this work.

#### 4.3.3 Maximum number of searches

Maximum number of searches is the number of improvisations a new harmony (in current problem, creating a new composition plan and testing its quality). The parameter has a direct relation with the execution time of the algorithm and the optimality of the solution. As it is shown in figure 5 and 6, the value of 1000 is a suitable value for maximum searches in order to quick execution of HS. For greater values, execution time grows without any considerable improvement in the optimality of the solution.

## 5. Evaluation tool

In this work, for the evaluation of the proposed method, a simulation software has been developed that can simulate

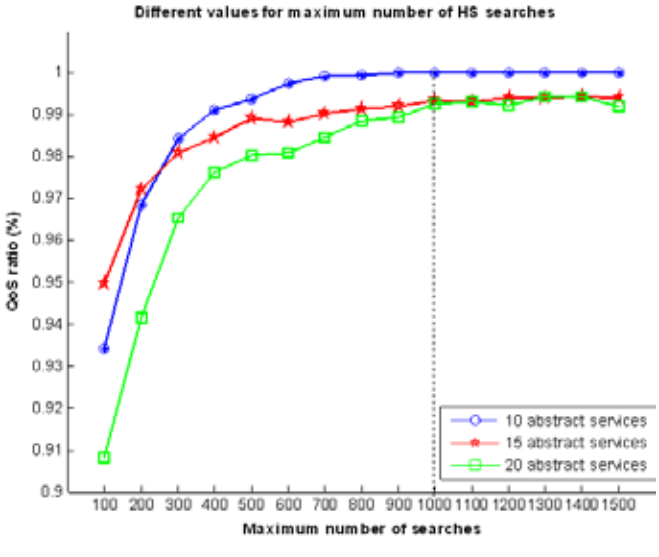


Figure 5. Different values for maximum searches searches

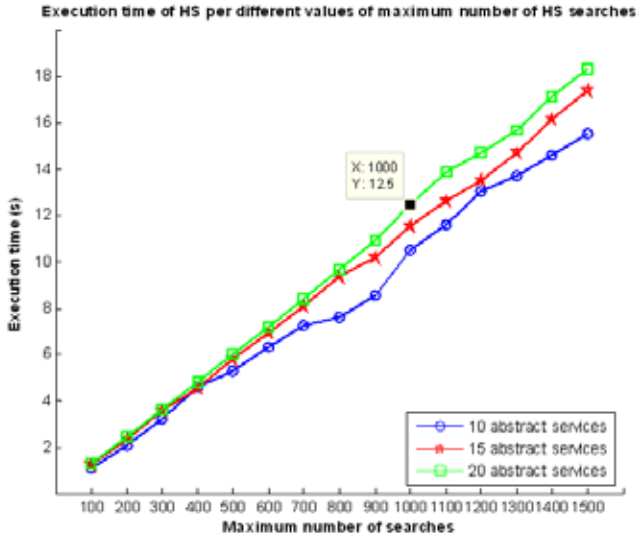


Figure 6. Execution time for Different values of maximum

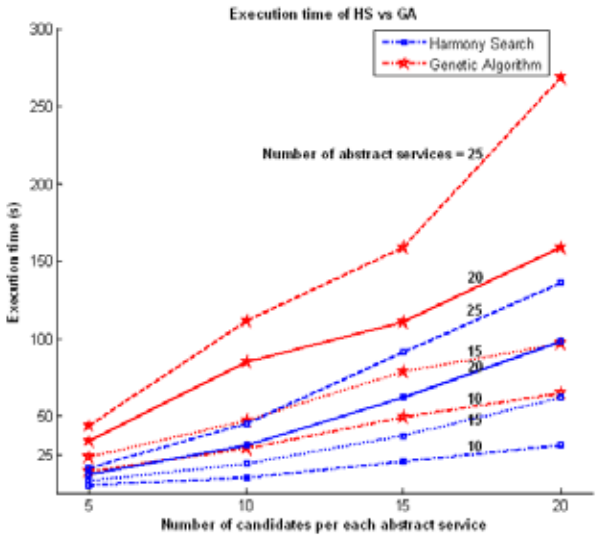


Figure 7. Execution time of Harmony Search vs. Genetic Algorithm

web service compositions and their QoS attributes. Then the proposed method based on Harmony Search and a method based on Genetic Algorithm are triggered to run and optimize the quality of generated compositions. The quality of results and the execution time of these two are compared in order to evaluate the performance of the proposed method. The steps of the simulation are as follows:

- Generation of the composition schema: Given *n*, the number of abstract services within a composition, the composition is randomly generated. The employed composition patterns are sequence, AND parallel, and XOR parallel pattern. In following simulations, *n*, the number of abstract services in a composition, varies between 10 and 25, based on discussions in (Jaeger and Rojec-Goldmann, 2006), and the number of candidate services for each abstract service is between 5 and 20.
- Generation of values of QoS attributes: The values of QoS attributes are randomly generated by simulation software. The range of QoS values are listed in table 2, based on assumption of (Jaeger and Rojec-Goldmann, 2006):

| QoS attribute | Value range |
|---|---|
| Response time (millisecond) | [150..2000] |
| Cost (dollar) | [1..1000] |
| Availability | [0.975..0.999] |
| Reliability | [0.950..0.999] |

Table 2. Considered ranges of QoS attributes

In random generation of QoS values, considering real services, it is assumed that the less the response time and the more the availability a service has, the more it costs.

## 6. Experimental results

In this section, the performance of the proposed method is compared with Genetic Algorithm. The simulations consider arbitrary composition schemas with 10, 15, 20, 25 abstract services. The number of candidate services per each abstract service is assumed 5, 10, 15, and 20 For each composition schema. The reported results are acquired using 20 rounds of experiments for two methods.

The HS parameters are set up as follows: HMCR = 0.7; number of iterations = 1000; HMS = 10. To set up the method based on Genetic Algorithm, the genome of GA is encoded like (Canfora et al., 2008). The fitness function is assumed to be the same as objective function of HS (eq. 3). GA settings are: Scattered crossover probability = 0.7; Mutation probability = 0.1; Population size = 50; roulette wheel selection as selection operator; and elitism of the two best individuals.

Figure 7 plots average execution time of two mentioned methods. It is obvious that the execution time arises while number of discovered candidate services increases. Also the results expresses that the proposed method based on HS works significantly faster than the method based on GA.

## 7. Conclusion

By introduction of web services, the Web has become a place for sharing a wide variety of services and service compositions. A challenging problem of service composition management is the selection of constituent services of a service composition in a way that the composition presents the optimal QoS to the users and satisfies their quality criteria.

In this paper, a method was proposed for QoS-aware selection of web service compositions which applies Harmony Search algorithm for optimizing the QoS. The proposed method introduces lower computational time in comparison with Genetic Algorithm-based methods and has proper convergence to optimal compositions. Like GA-based method, it does not need to linearize quality constraints. Although experiments are based on four major QoS attributes, the method can be extended to support more attributes.

This paper is a part of an ongoing research about web service compositions. Next steps concern covering more composition patterns of workflow models, and re-planning the composition plan due to occurring changes in available candidate services at run time.

## References

[1] Canfora, G., Di Penta, M., Esposito, R., Villani, M. L. (2004). A lightweight approach for QoS-aware service composition. *In:* Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC'04)-short papers. New York, USA.

[2] Canfora, G., Di Penta, M., Esposito, R., Villani, M. L. (2008). A framework for QoS-aware binding and re-binding of composite

web services. *The Journal of Systems & Software.*

[3] Cardoso, J., Miller, J., Sheth, A., Arnold, J. (2004). Modeling Quality of Service for Workflows and Web Service Processes. *Web Semantics Journal: Science, Services and Agents on the World Wide Web,* 1 (3) 281-308. Elsevier Inc, MA, USA.

[4] Geem, Z. W., Kim, J. H. (2001). A new heuristic optimization algorithm: harmony search. Simulation, 76, 60.

[5] Jaeger, M. C., Ladner, H. (2006). A Model for the Aggregation of QoS in WS Compositions Involving Redundant Services. *Journal of Digital Information Management,* 4, 44.

[6] Jaeger, M. C., Muhl, G. (2007). QoS-based selection of services: The implementation of a genetic algorithm. KiVS 2007 Workshop: *Service-Oriented Architectures and Service-Oriented Computing* (SOA/SOC). Bern, Switzerland.

[7] Jaeger, M. C., Rojec-Goldmann, G. (2006). SENECA-Simulation of Algorithms for the Selection of Web Services for Compositions. *Lecture Notes in Computer Science*, 3811, 84.

[8] Jin, C., Wu, M., Jiang, T., Ying, J. (2008). Combine automatic and manual process on web service selection and composition to support QoS.

[9] Ko, J. M., Kim, C. O., Kwon, I. H. (2008). Quality-of-service oriented web service composition algorithm and planning architecture. *The Journal of Systems & Software.*

[10] Lee, K. S., Geem, Z. W. (2005). A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Computer methods in applied mechanics and engineering, 194, 3902-3933.

[11] Xiong, P. C., Fan, Y. S., Zhou, M. C. (2007). Web Service Configuration under Multiple Quality-of-Service Attribute. *IEEE Trans. on Automation Science and Engineering.*

[12] Yu, T., Lin, K. (2005). Service Selection Algorithms for Composing Complex Services with Multiple QoS Constraints. *Lecture Notes in Computer Science,* 3826, 130.

[13] Zeng, L., Benatallah, B., Ngu, A. H. H., Dumas, M., Kalagnanam, J., Chang, H. (2004). QoS-Aware Middleware for Web Services Composition. *IEEE Transactions on Software Engineering,* 311-327.

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

**Authors Biographies**

**Nastaran Jafarpour** was born in 1984 in Tehran, Iran. She received her BSc. in Computer Engineering, Software (2006) in Azad University Central Tehran branch. She is currently a M.Sc. student in Computer Engineering in faculty of engineering, university of Isfahan. Her research interest includes Distributed systems, Software engineering, Networking, Parallel programming.

**Mohammad-Reza Khayyambashi** was born in Isfahan, Iran in 1961. He received the B.Sc. degree in Computer Hardware Engineering from Tehran University, Tehran, Iran in 1987. He received his M.Sc. in Computer Architecture from Sharif University of Technology (SUT), Tehran, Iran in 1990. He got his Ph.D. in Computer Engineering, Distributed Systems from University of Newcastle upon Tyne, Newcastle upon Tyne, England in 2006. He is now working as a lecturer at the Department of Computer, Faculty of Engineering, University of Isfahan, Isfahan, Iran. His research interests include Distributed Systems, Networking, Fault Tolerance and E-Commerce.