

QoS-aware Selection of Web Service Composition Based on Harmony Search Algorithm

Nastaran Jafarpour*, Mohammad Reza Khayyambashi*

* *Department of Computer, Faculty of Engineering, University of Isfahan, Isfahan, Iran*

jafarpour@eng.ui.ac.ir, m.r.khayyambashi@eng.ui.ac.ir

Abstract— Designing of the composite services with desired quality is an interesting challenge of the web service environments. In a QoS-aware web service composition, appropriate services with acceptable quality are selected among several function-equivalent candidate services. The selection is performed in such a way that creates a composition with optimal quality which can satisfy user's quality constraints.

In this paper, the problem of QoS-aware selection of web service composition is modeled as an optimization problem. Then the Harmony Search algorithm is adopted to find an optimal or near-optimal composition which can satisfy local and global user's constraints on quality attributes. The proposed method is a rapid and lightweight approach which can be applied to large service compositions with many service candidates.

Keywords— Web service composition, Quality-of-Service, Harmony Search algorithm, Global optimization, Workflow

I. INTRODUCTION

A web service is an XML-based software application that enables program to program interactions. Web services work independently from their location, platform, and implementation. They are described, published, offered, and invoked via standard protocols such as WSDL, SOAP, and UDDI. Each atomic web service performs a specific task or a set of simple tasks. But the necessity of implementing workflows has lead to combine single services and build composite web services. A composite web service is a partially-ordered collection of atomic web services that executes a structured workflow. Each web service within the composition executes a task within the workflow. Indeed, standard protocols adaptation and location independency has enabled web services to integrate together and produce a value-added service called composite web service.

Web service composition management can be summarized in the following steps: service composition designing, service discovery, composition selection, and composition execution monitoring. In service composition designing step, a workflow is mapped to a combination of abstract services. An abstract service is a semantic description of specific functionality that can be matched to concrete services published on the Web. These concrete services are found

during the service discovery phase. There may be several offered services with the same functionality but different values of quality attributes. These services are offered by different service providers. It is the role of composition selection phase to select one concrete service for replacing each abstract service in the composition. In the last phase of composition management, the system keeps tracks of composition executions and monitors the quality. The information collected in this phase will be used in refining the composition if necessary. Considering the availability of workflows and a set of discovered services, this paper mainly focuses on the selection of composition plans respecting the resulting quality of service (QoS).

The quality of service is a key factor in discriminating candidate web services with identical functionality. It can be expressed by several attributes. QoS attributes quantify the non-functional aspects of a system. Major investigated QoS attributes of web services are: Response time (the duration between sending a request and receiving the results, measured in milliseconds), Execution price (the fee that a service requester has to pay), Reliability (the probability of responding a request correctly within the maximum expected time), Availability (the probability of readiness of a service for immediate use), Reputation (the degree of the trustworthiness of a service), and Fidelity (a measure of how well a service is being rendered) [1, 2]. Essential QoS attributes are usually advertised by service providers or can be measured by a third party acting as a service execution broker.

As explained before, a composite web service consists of multiple abstract services and each abstract service has one or more candidate services with the same functionality and different quality characteristics. The problem of QoS-aware selection of composite services refers to selecting a concrete service for each abstract service in the composition while obtaining optimal (or near-optimal) 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, e.g. "The price of a single web service must be 5\$ at most". In contrast, global constraints present limitations on the whole composite service execution, e.g. "The total execution time must be less than 3 seconds" [3].

QoS-aware selection of composite web services introduces a global optimization problem with multiple constraints. The

goal of this optimization is to maximize the total quality of the composite service while meeting users' constraints. Finding an optimal combination of concrete services among a large number of possible solutions takes significant computation efforts. This QoS-aware selection problem has been modeled as a special case of the knapsack problem, so that its complexity is NP-hard [4]. Different researches have adapted Integer programming, Genetic Algorithms (GA), and some heuristic methods to solve this problem.

In this paper, an evolutionary optimization algorithm called Harmony Search algorithm is applied to find compositions with optimal quality while meeting local and global constraints. Experimental results show that the proposed method performs the selection of optimal web 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 in comparison with other approaches even if the number of atomic services in the composition increases, it is suitable for Web environments.

The rest of this paper is organized as follows. Some literature is reviewed in section 2. The model of problem and the process of computing quality of web service compositions are presented in section 3. Section 4 describes Harmony Search algorithm and subsequently applies it in QoS-aware selection. Numerical simulations of the proposed method are explained in section 5. The work is finally concluded in section 6.

II. RELATED WORKS

Recent years, web service composition management has been studied extensively and some methods based on applied mathematics and artificial intelligence have been proposed to select QoS-driven web service compositions. In order to select QoS-driven composition, it is necessary to calculate the QoS of a composite web service based on QoS attributes of atomic services. For computing the aggregated QoS, a mathematical model is presented in which reduction rules are applied on the workflow iteratively until a single task remains in the flow [1]. This task expresses the total QoS of the workflow as a result of using aggregation formulas while reducing the workflow structure. Although this model does not discuss about searching optimal composition, its reduction rules and aggregation formulas have been deployed extensively in subsequent researches.

Since the problem of composition selection with multiple QoS constraints is mapped to the multi-choice multi-dimensions 0-1 knapsack problem in [4], its complexity is NP-hard. Also by modeling the set of candidate services to a directed acyclic graph, the problem can be described as the multi-constraint optimal path problem.

A middleware platform for QoS-driven selection of web services has been developed in [2]. This approach has two options: one is based on local optimization, the other addresses global optimization by using linear programming. Applying 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 address these disadvantages of linear programming, Genetic Algorithm (GA) has been applied to select the (near)-optimal composition in [5].

The impact of different parameters of GA, e.g. mutation rate and fitness function is investigated in [6]. Experimental results of that work show that genetic algorithm offers a good performance in comparison with some other heuristics; though experiments are based on a single global constraint. Handling of population diversity in GA is proposed in [7] by using simulated annealing in order to prevent prematurity of GA. In this literature, the relation matrix coding schema is adopted to 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 [8] 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. But this method is not suitable for large scale web service compositions.

A hybrid meta-heuristic method which combines Tabu Search and simulated annealing is proposed by [9] 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.

In this paper, the reduction rules proposed by [1] are used for QoS aggregation. Then the proposed method based on Harmony Search algorithm is applied to find compositions with optimal aggregated QoS which are meeting local and global constraints. In order to evaluate the validity and performance of the proposed method, its results have been compared with the results of Genetic Algorithm through numerical simulations.

III. QOS OF SERVICE COMPOSITIONS

This work assumes a combination of n abstract services, $S = \{s_1, s_2, \dots, s_n\}$, which are combined through composition patterns i.e. the control flow between single tasks within the composition. For service s_i ($1 \leq i \leq n$), there are m_i ($m_i \geq 1$) discovered candidate services which are indexed in vector C_i . Each element of this vector dedicates a vector of quality attributes. This model considers four quality attributes (including response time, cost, availability, and reliability), however can flexibly cover any other attribute.

A. QoS normalization

The scopes of quantified attributes are too different from each other to be compared in a fairly manner without being processed. 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, some attributes (like availability and reliability, called positive attributes) have direct proportion with the quality i.e. the more the value of attribute, the upper the quality. Others (like response time and cost, called negative attributes) are in reverse proportion with the quality i.e. the less the value of attribute, the upper the quality. 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 positive attributes and (2) in for normalizing negative attributes [2].

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

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

In these equations, values of QoS attributes are shown by $q_{i,j}$ where i determines i th abstract service in the composition and j refers to j th candidate service in vector C_i (vector of candidate services corresponding to abstract service i).

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 $q_j^{\min}$ and $q_j^{\max}$ in (2); then the value of response time of each service in this vector, $q_{i,j}$, is normalized to $n_{i,j}$.

B. 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. Some QoS aggregation formulas, for basic composition patterns and major QoS attributes are collected in table 1, proposed by [1] and [5].

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. In column 4, k is the number of loop iterations estimated by the workflow designer and will be improved by monitoring composite service executions [5].

Solutions of composition selection algorithm are expressed as sequences of possible assignments of candidate services to abstract services, called compositions plans. The overall QoS of a composition plan is calculated by applying QoS aggregation formulas to normalized QoS values of its

component services. The goal of QoS-aware selection algorithm is to find a composition plan which leads to the best quality of service composition.

The QoS attributes are scaled to range of [0, 1] by performing normalization. But they are separated from each other. The Simple Additive Weighting (SAW) is used to consider QoS attributes altogether at the same time. In this approach, the overall QoS of a composition plan is defined by (3),

$$QoS(composite) = \sum_{i=1}^4 w_i \cdot N_i(composite) \quad (3)$$

Where $N_i(composite)$ is the aggregated QoS of a composition for attribute i and the weight of each attribute (w_i) is its importance degree defined by user preferences and $\sum_{i=1}^4 w_i = 1$.

The purpose of QoS-aware selection is maximizing the overall quality of service composition expressed by (3), so it is called the objective function (or fitness function) from the perspective of optimization algorithms.

C. 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 positive attribute of $s_i \geq$ | a negative 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 positive aggregated attribute of composition plan $\geq$ | a negative 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.

IV. APPROACH DESCRIPTION

In this section, Harmony Search is introduced at first. Then it is customized to find a solution for QoS-aware selection problem.

A. Harmony Search algorithm

Harmony Search (HS) is a meta-heuristic evolutionary optimization algorithm, recently developed by Geem et al. [10]. It imitates musical process of searching for a perfect state of harmony.

Table 1. QoS Aggregation Formulas Per Composition Pattern and QoS Attribute

QoS attribute	Sequential	Parallel	Conditional	Loop
Time (T)	$\sum_{i=1}^n T(s_i)$	$\text{Max} \{T(s_i)_{i \in \{1..n\}}\}$	$\sum_{i=1}^n p_i \cdot T(s_i)$	$k \cdot T(s)$
Cost (C)	$\sum_{i=1}^n C(s_i)$	$\sum_{i=1}^n C(s_i)$	$\sum_{i=1}^n p_i \cdot C(s_i)$	$k \cdot C(s)$
Availability (A)	$\prod_{i=1}^n A(s_i)$	$\prod_{i=1}^n A(s_i)$	$\sum_{i=1}^n p_i \cdot A(s_i)$	$A(s)^k$
Reliability (R)	$\prod_{i=1}^n R(s_i)$	$\prod_{i=1}^n R(s_i)$	$\sum_{i=1}^n p_i \cdot R(s_i)$	$R(s)^k$

The Harmony Search algorithm is simple in concept and few in parameters. It imposes a few mathematical requirements and can easily be implemented. 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.

The Harmony Search steps are as follows [11]:

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 $x = (x_1, x_2, \dots, x_N)$ 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 $(1 - \text{HMCR})$ 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.

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.

Harmony Search algorithm does not require setting initial values for decision variables, thus it may escape local optima. Since this combinatorial optimization algorithm can handle discrete variables too, it can be used for optimization of the QoS of web service compositions.

B. QoS-aware selection of composite services with Harmony Search

As described in section 3.2, the objective function of quality optimization problem can be defined as (3). Since $N_i(\text{composite})$ 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, $\{s_1, s_2, \dots, s_n\}$. 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 ($1 \leq i \leq n$) 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 $1 - \text{HMCR}$).

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 the evaluation [11]. However it is a risky 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.

V. EXPERIMENTAL RESULTS

In this section, the evolution of the proposed method is performed through numerical simulations.

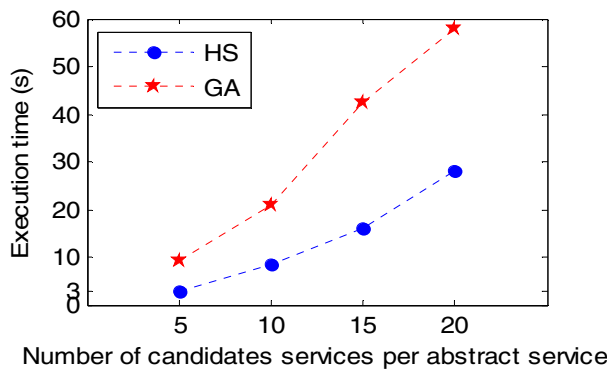


Figure 1. Harmony Search vs. Genetic Algorithm: Execution time comparison

In these simulations, a web service composition schema with 10 abstract services is considered. This schema is constructed by applying sequential, parallel, and conditional composition patterns.

In current simulation, Harmony Search parameters are set up as follows: HMCR = 0.7; Maximum number of iterations = 1000; HMS = 10. For the performance evaluation of the proposed method, it is compared with Genetic Algorithm in

terms of execution time. For this comparison, the genome of GA is encoded like [12]. The fitness function is assumed to be the same as objective function of HS (3).

GA settings are: Two-point 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 1 plots average execution time of 20 executions of Harmony Search and Genetic Algorithm when the number of concrete services per abstract service varies between 5 and 20 in steps of 5. This simulation expresses that proposed method for QoS-aware selection based on Harmony Search works significantly faster than the method based on Genetic Algorithm.

The optimality of Harmony Search results (i.e. the QoS of selected composition plan) is compared with Genetic Algorithm results in figure 2. QoS-ratio of each experiment denotes what percentage of the best solution is obtained by that experiment. The simulation considers having 5, 10, 15, and 20 candidate services corresponding to each abstract service. The experiments are repeated 20 times per each assumption.

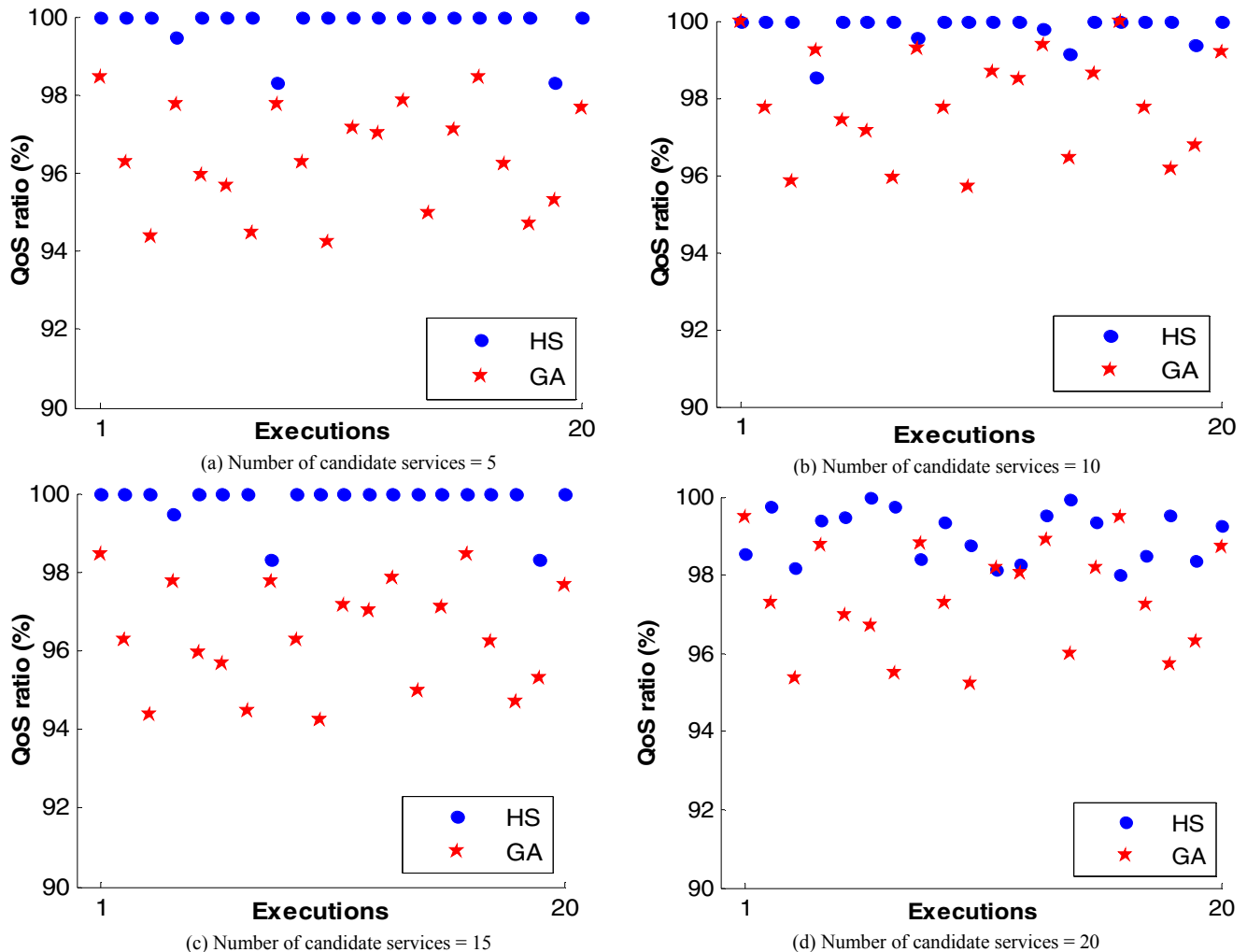


Figure 2. Harmony Search vs. Genetic Algorithm: Optimality of results

As figure 2 shows, the method based on Harmony Search reaches to above %98 of the best QoS in every simulation while Genetic Algorithm leads to above 94%. It is concluded that the proposed method is able to find a composition plan with optimal quality correctly and quickly.

VI. CONCLUSION

By introduction of web services, the Web has become a place for sharing a wide variety of services. Simple atomic web services can be combined together to create value-added service compositions. In order to fulfill users' non-functional requirements, it is necessary to be able to evaluate the quality of services. 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 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 methods, 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 by applying little modifications.

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] J. Cardoso, J. Miller, A. Sheth, and J. Arnold, "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, 2004.
- [2] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-Aware Middleware for Web Services Composition," *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, pp. 311-327, 2004.
- [3] D. Ardagna and B. Pernici, "Global and local QoS guarantee in web service selection," *LECTURE NOTES IN COMPUTER SCIENCE*, vol. 3812, p. 32, 2006.
- [4] T. Yu and K. Lin, "Service Selection Algorithms for Composing Complex Services with Multiple QoS Constraints," *LECTURE NOTES IN COMPUTER SCIENCE*, vol. 3826, p. 130, 2005.
- [5] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani, "A lightweight approach for QoS-aware service composition," in *Genetic and Evolutionary Computation Conference (GECCO 2005)*, Washington, DC, USA, 2005.
- [6] M. C. Jaeger and G. Muhl, "QoS-based selection of services: The implementation of a genetic algorithm," in *KiVS 2007 Workshop: Service-Oriented Architectures und Service-Oriented Computing (SOA/SOC)*, Bern, Switzerland, 2007, pp. 359-370.
- [7] C. Zhang, S. Su, and J. Chen, "DiGA: Population diversity handling genetic algorithm for QoS-aware web services selection," *Computer Communications*, vol. 30, pp. 1082-1090, 2007.
- [8] C. Jin, M. Wu, T. Jiang, and J. Ying, "Combine automatic and manual process on web service selection and composition to support QoS," 2008, pp. 459-464.
- [9] J. M. Ko, C. O. Kim, and I. H. Kwon, "Quality-of-service oriented web service composition algorithm and planning architecture," *The Journal of Systems & Software*, 2008.