OPTIAL WEB SERVICE COMPOSITION USING HYBRID GA-TABU SEARCH

[1] Sondos Bahadori, [2] Somayeh Kafi, [3] Kamran Zamani far, [4] Mohammad Reza Khayyambashi

[1] Islamic Azad University Ilam Branch, Ilam Azad University, Daneshjo Bolvar, Ilam, Iran
[2] Islamic Azad University Deylam Branch, Deylam Azad University, North Shariaty Street,Boshehr, Iran
[3] Department of Computing, University of Isfahan, Isfahan,Iran
[4] Department of Computing, University of Isfahan, Isfahan,Iran
E-mail: sondos_bahadori@ilam-iau.ac.ir, kafi@ce.sharif.edu, zamanifar@ eng.ui.ac.ir,m.r.khayyambash@ eng.ui.ac.ir

ABSTRACT

Web services are rapidly changing the landscape of software engineering. One of the most interesting challenges introduced by web services is represented by Quality of Service (QoS) _aware composition and late binding. This allows to bind, at run-time, a service oriented system with a set of services that, among those providing the required features, meet some nonfunctional constraints, and optimize criteria such as the overall cost or response time. In other words, QoS_aware composition can be modeled as an optimization problem. We propose to adopt Genetic Algorithms to this aim. In order to increase the performance of genetic algorithm we use Tabu search. By using Tabu search ultimately a kind of composition is selected which is in close connection with user wants.

Keywords: *Web Service Composition, Genetic Algorithm, Tabu Search*

1. INTRODUCTION

Web services are autonomous software systems identified by URIs which can be advertised, located, and accessed through messages encoded according to XMLbased standards (e.g., SOAP, WSDL, and UDDI [16]) and transmitted using Internet protocols [2]. Web services encapsulate application functionality and information resources and make them available through programmatic interfaces, as opposed to the interfaces provided by traditional Web applications which are intended for manual interactions. In addition, since they are intended to be discovered and used by other applications across the Web, Web services need to be described and understood both in terms of functional capabilities and Quality of Service (QoS) properties.

The emergence of Web services (e.g., for order procurement, finance, accounting, human resources, supply chain, and manufacturing) has created unprecedented opportunities for organizations to establish more agile and versatile collaborations with other organizations. Widely available and standardized Web services make it possible to realize Business-to-Business Interoperability (B2Bi) by inter-connecting Web services provided by multiple business partners according to some business process: a practice known as Web Services Composition [9], [5], [1], [8]. For example, an integrated financial management Web service can be created by composing more specialized Web services for payroll, tax preparation, and cash management.

A composite service has specific functions that can be divided into some component functions. These component functions can be accomplished by some component services respectively. An example of web service composition was shown in Fig. 3 of [15]. The dependencies among component functions were represented with the state charts in [14]. Some candidate services (concrete services) with same functions and different values of QoS attributes are discovered for every task (abstract services). Thus, there are various composite plans for each execution path of composite service. Moreover, since the number of concrete services with same functions and different values of QoS attributes is increasing with the proliferation of web service, the composite size should be larger and larger. For example, in one execution path, there are 10 component functions in this composite path,

and 15 candidate web services for each component function. In this kind of composition scenario, the composite size should be about 15^{10}. Since web service requesters always have both functional requirements and global QoS requirements, it is needed to select concrete services for a given task to get the best composite plan and maximize user satisfaction (here, the word ‘‘best” means that the composite plan has the optimum QoS property values). Web service selection with global QoS constraints plays an important role in web service composition [6]-[7].

The remainder of this paper is organized as follows. After a review of the literature of QoS-aware web services selection using QoS computation in Section 2, searching for a solution with Genetic Algorithms in section 3, Section 4 reports and discusses results obtained in the simulations. Finally, Section 5 concludes.

2. QUALITY COMPUTATION BASED SELECTION OF WEB SERVICES

According to Std. ISO 8402 [16] and ITU E.800 [12], QoS may include a number of non-functional properties such as price, response time, availability and reputation. Thus, QoS value of a composition service can be achieved by fair computation of QoS of every component service. In this section, some traditional optimization techniques [3], [14], [13] and Genetic Algorithm (GA) [9], [10], [15] are discussed in detail.

The QoS computation based on QoS matrix is a representative solution. In [13], web services were ranked by means of normalizing QoS matrix. However, it was only a local optimization algorithm but not a global one for service selection. Other works in the area of QoS computation include [13]-[14], which proposed local optimization and global planning. The local optimization approach could not take global QoS constraints into consideration. When the size of composite service is very large, for example 15^{10}, the overhead of global planning is quite enormous. Hereby, both had limitation to some extent.

The above ways are not able to effectively solve the issue of web service selection with global QoS constraints. This kind of issues is NP-hard [13].

GA is more suitable for solving these issues. But GA can play an important role only when the combinatorial size is very large. In [10], some numerical simulations show that Integer Programming outperforms GA if the combinatorial size is small. In their opinion, GAs should be preferred instead of Integer Programming in the case of widely used services, such as hotel booking, weather services or ecommerce services, etc. On the other hand, Integer Programming is to be preferred in the case of very specific (e.g., scientific computation) services. Two different GAs were proposed in [9], [15].

In [15], binary strings of chromosome were proposed for service selection. Every gene in chromosome represented a service candidate with values 0 and 1. Thereby, the more the number of service candidates or web service clusters was, the longer chromosome was. Since only single service candidate could be selected in each of web service clusters, only one gene was 1 and others were 0 in all genes of every cluster. When the number of component services and the number of candidate services of each component service are all very big, the length of genome will be very long. This kind of manner resulted in poor readability. Further, the authors only proposed the coding manner of chromosome for service selection with little further consideration of the rest parts of genetic algorithm, such as selection mechanism.

3. SEARCHING FOR A SOLUTION WITH GENETIC ALGORITHMS

Differently from other approaches proposed in literature, such as linear integer programming, GAs do not impose constraints on the linearity of the QoS composition operators (and thus of objective function and constraints). This Permits the use of our approach for all possible (even customized) QoS attributes, without the need for linearization.

To let the GA search for a solution of our problem, we first need to encode the problem with a suitable genome. In our case, the genome is represented by an integer array with a number of items equals to the number of distinct abstract services composing our service. Each item, in turn, contains an index to the array of the concrete services matching that abstract service. Figure 1 gives a better idea of how the genome is made.

The crossover operator is the standard two-point crossover, while the mutation operator randomly selects an abstract service (i.e., a position in the genome) and randomly replaces the corresponding concrete service with another one among those available.

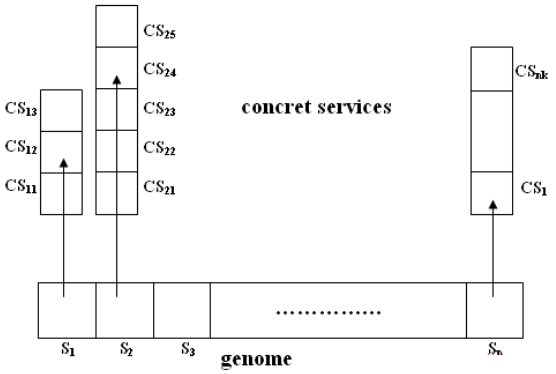


Figure 1: Genome Encoding

The problem can now be modeled by means of a fitness function and, eventually, some constraints. The fitness function needs to maximize some QoS attributes (e.g., reliability), while minimizing others (e.g., cost). When user {Denoted, domain} specific QoS attributes are used, the specification of the fitness function is left to the workflow designer.

3.1. Fitness Function

To select the best combination according with global QoS constraints, a fitness function was provided in [4]. The function can express end users' favoritism concerning QoS by means of weight factors. It ensured that the individual disobeying the constraints can be selected proportionally into the next generation population so as to maintain population diversity. The following is the definition of fitness function. Some QoS models and QoS computation formulas for composite service were available in [14], [9]. But, via comparison in experiments, the optimal solution based on the QoS computation formula in [9] is better than the one based on [14]. Consequently, the QoS computation formula in [9] is adopted, and an objective function is defined in formula (1):

$$f = \frac{\sum_{j}(Q_j \times w_j)}{\sum_{k}(Q_k \times w_k)} \qquad (1)$$

where w_j, $w_k \in [0, 1]$, and w_j, w_k are real positive weight factors, and represent the weight of criterion j and k for which end users show their favoritism concerning QoS by providing values respectively. The sum of them is 1. Q_j and Q_k denote the sum of QoS values of the individual j and k, respectively.

End users usually assert their function requirements as well as global constraints, e.g. Cost < 60, Time < 150. The individual whose QoS violates the constraints will be rejected in [9]. However, this approach belongs to absolute rejection. It influences population diversity seriously and always results in a local optimal solution. The individual disobeying the constraints should be selected proportionally into the next generation population on a basis of a certain technique. The most common method is penalty technique for constrained optimization problems. Hereby, a fitness function with penalty character is defined in formula (2):

$$Fit(g) = f - \lambda \sum_{j=1}^{n} \left(\frac{\Delta P_j}{R_{jMax} - R_{jMin}}\right)^2 \qquad (2)$$

Where R_{jMax}, R_{jMin} are the maximum value and the minimal value of the no.j quality constraint respectively, n is the number of quality constraints, k is a parameter used to adjust the scale of penalty value. P_j represents the calculation value of a Q_i or some Q_is. The following formula (3) is the definition of ΔP_j:

$$\Delta P_j = \begin{cases} P_j - R_{jMax} & if\ P_j > R_{jMax} \\ 0 & if\ R_{jMin} \le P_j \le R_{jMax} \\ R_{jMax} - P_j & if\ P_j < R_{jMin} \end{cases} \qquad (3)$$

3.2. Enhanced Initial Population Policy

During evolution, initial population is the predecessor of all successor populations. If all chromosomes in the initial population have very high fitness, the probability for child chromosomes to have high fitness will be high and global optimal solution will be possibly obtained and convergence time will be probably shortened. It is also possible to avoid prematurity situation. Therefore, it is the basis to obtain better convergence. It has an important influence on the final convergence. Hence, an optimized initial population can effectively overcome slow convergence. On the basis of the above, the value of every gene in the initial chromosome will be selected in order that every initial chromosome has high fitness. Meanwhile, every path should have its chromosomes in the initial population so as to have the ability to search every path and obtain the global optimal solution. The following is the proposed policy.

The value of every task in every chromosome is set according to a local optimized method. The value of every task is QoS value of selected candidate service. The larger QoS value of a candidate service is, the larger the probability to be selected is. The QoS model in [14] is used to calculate QoS value of every candidate service. The probability of one candidate to be selected is the result of its QoS value divided by the sum of QoS values of all candidates of same task. The ‘‘roulette wheel selection” is the mechanism to select candidates of every task. The chromosome will have high QoS value after every task has high QoS value. So, the chromosome will have high fitness. The result is that every chromosome has so high fitness value that entire population has very high fitness value. From the above, the special initial population becomes the basis to get higher fitness during the later evolution.

3.3. Hybrid Genetic Algorithm–Tabu Search Approach for Optimizing

Tabu search is a "higher level" heuristic procedure for solving optimization problems, designed to guide other methods (or their component processes) to escape the trap of local optimality. Tabu search has obtained optimal and near optimal solutions to a wide variety of classical and practical problems in applications ranging from scheduling to telecommunications and from character recognition to neural networks. It uses flexible structures memory (to permit search information to be exploited more thoroughly than by rigid memory systems or memory less systems), conditions for strategically constraining and freeing the search process (embodied in Tabu restrictions and aspiration criteria), and memory functions of varying time spans for intensifying and diversifying the search (reinforcing attributes historically found good and driving the search into new regions).Tabu search can be integrated with genetic algorithm and it has the ability to start with a simple implementation that can be upgraded over time to incorporate more advanced or specialized elements.

Using Tabu search, search space of genetic algorithm is increased. In this problem, after using crossover and mutation operator in each iteration of evolution, the chromosome with best fitness is selected as Tabu and added to Tabu list. Chromosomes in Tabu list have an aspiration time. In each iteration, aspiration time is decreased. If the value of aspiration time of each chromosome in Tabu list becomes zero, it is deleted from this list and added to aspiration list. After finding Tabu, chromosomes in current population that are similar to Tabu are replaced with the chromosome in aspiration list. If aspiration list was empty this chromosome replaced with chromosomes in last population with smallest fitness value. Using this replace, diversity of population increase. And genetic algorithm can search more space. Therefore, genetic algorithm survives from the trap of local optimality.

Hitherto, some important elements in the GA for QoS-driven web services selection have been proposed. Here, the structure of the GA is available in sequence as Fig. 2.

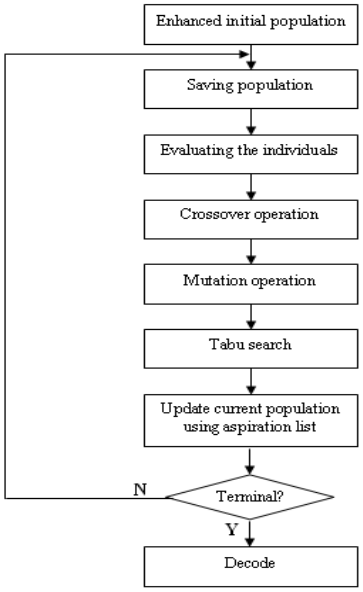
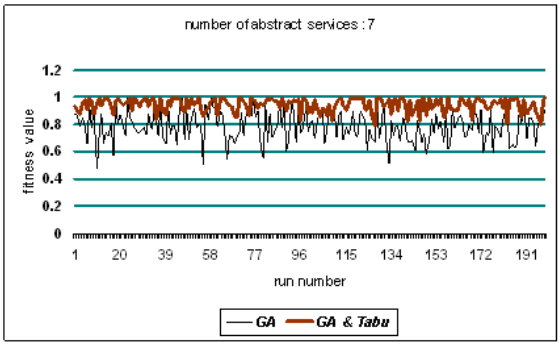


Figure 2: Hybrid GA_Tabu search flowchart

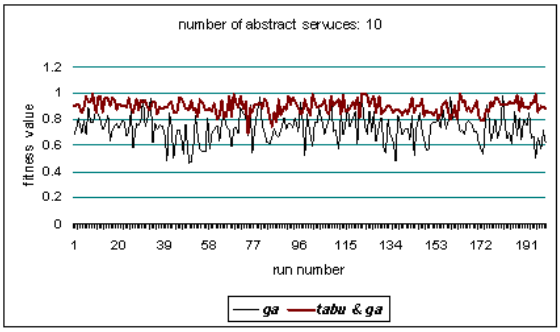
4. Comparing Hybrid GA_Tabu Search With GA

As a case study for comparison, we considered a workflow containing 7,10,15,20 distinct abstract services. For each abstract service, we considered 5 available concrete services. Then, we compared the performance of GAs and hybrid GA–Tabu search as follows: For both approach we compared the fitness of best chromosome for 200 times running of these approaches. In Fig. 3, population size was 200, crossover probability was 0.7, and mutation probability was 0.1. These two approaches were implemented in .net.

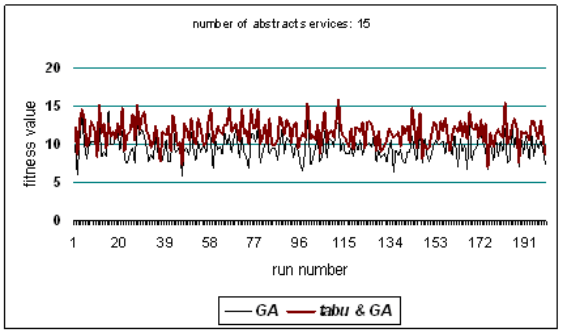
Both approaches were executed 200 times, and then the average values were computed.



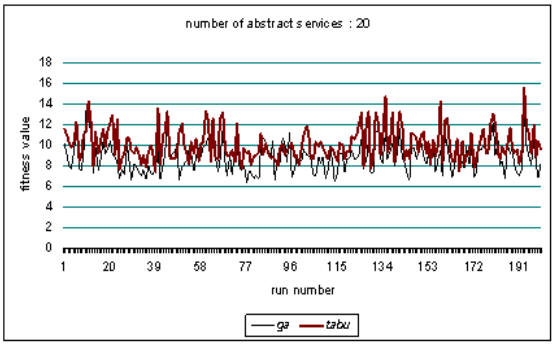
Algorithm	Average Value of Fitness
Proposed approach	0.94710825
SGA(standard GA)	0.789092379
best fitness	1



Algorithm	Average Value of Fitness
Proposed approach	0.900964
SGA(standard GA)	0.737892
best fitness	1



Algorithm	Average Value of Fitness
Proposed approach	11.60794
SGA(standard GA)	9.55836



Algorithm	Average Value of Fitness
Proposed approach	10.9129
SGA(standard GA)	8.745199

Figure 3: Evolution of fitness parameters

The lesson learned from our experimentation is basically that, using hybrid GA–Tabu search the composition with better fitness can be found. And global constraints are considered. So the selected composition is in close connection with user wants.

5. CONCLUSIONS

This paper proposed a hybrid GA–Tabu search approach for QoS_aware service composition, i.e., to determine a set of concrete services to be bound to abstract services contained in a composition to meet a set of constraints and to optimize fitness criterions on QoS attributes. Compared with GA, this approach increase diversity of population and cause more space of search domain is searched and escaped the trap of local optimality. Using this approach the composition with better fitness value is found.

REFERENCES:

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, "Web Services", *In Springer Verlag*, 2002.

[2] "Web Services Architecture Requirements Working Group", *http://www.w3.org/TR/wsa-reqs*, 2004.

[3] F. Casati , and M.-C. Shan, "Dynamic and Adaptive Composition of E-Services", *In Information Systems*, vol. 26, no. 3, May, 2001, pp. 143- 162.

[4] Z. Cheng-Wen, S. Sen, C. Jun-Liang, "Efficient population diversity handling genetic algorithm for QoSaware web service selection", In *ICCS* (International Conference on Computational Science), England, 2006, Part IV, LNCS 3994, Springer-Verlag, Berlin, pp. 104–111.

[5] B. Benatallah , M. Dumas , Q.Z. Sheng , and A.H. Ngu , "Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services", In *Proc. Int'l Conf. Data Eng*. (ICDE) , Feb, 2002, pp. 297-308.

[6] D.A Menasce´, "QoS issues in web services", In *IEEE Internet Computing* 6 (6) 72–75, 2002.

[7] D.A. Menasce´, Composing web services: a QoS view, In *IEEE Internet Computing* 8 (6) 88–90, 2004.

[8] B. Medjahed, A. Bouguettaya, and A.K. Elmagarmid, "Composing Web Services on the Semantic Web", In *the VLDB J*., vol. 12, no. 4, 2003, pp. 333-351.

[9] G. Canfora, M. Di Penta, R. Esposito, M.L. Villani, "A lightweight approach for QoS–aware service composition", *Proceedings of the 2nd International Conference on Service Oriented Computing* (ICSOC'04), New York, USA,pp. 36–47, 2004.

[10] G. Canfora, M. Di Penta, R. Esposito, et al., "An approach for QoS-aware service composition based on genetic algorithms", In *Genetic and Evolutionary Computation Conference* (GECCO), vol. 1, Washington DC, USA, pp.1069–1075, 2005.

[11] ISO 8402, "Quality management and quality assurance –vocabulary", 1994.

[12] ITU-T Recommendation E.800, "Terms and definitions related to quality of service and network performance including dependability", 1994.

[13] M. Srinivas, L.M. Patnaik, "Genetic algorithm: a survey", In *IEEE Computer* 27 (6) 17–26, 1994.

[14] Z. Liang-Zhao , B. Boualem , et al.,2004 . QoS-aware middleware for web services composition", In *IEEE Transactions on Software Engineering* 30 (5) 311–327.

[15] Z. Liang-Jie , L.Bing, C. Tian , et al, "On demand web services-based business process composition", In *IEEE International Conference on System, Man, and Cybernetics* (SMC'03), Washington, USA, pp. 4057–4064, 2003.

[16] F. Curbera, et al., "Unraveling the Web Services: An Introduction to SOAP, WSDL, and UDDI", In *IEEE Internet Computing, vol. 6, no. 2,* Mar./Apr, 2002.