Semantic Web Service Composition Based on Ant Colony Optimization Method

Toktam Ghafarian
Department of Computer Engineering
Ferdowsi University of Mashhad
Khayam Higher Education Institute
Mashhad, Iran
ghafarian@stu-mail.um.ac.ir

Mohsen Kahani
Department of Computer Engineering
Ferdowsi University of Mashhad
Mashhad, Iran
kahani@um.ac.ir

Abstract

As web services proliferate, finding a service that can perform a given request becomes more difficult. In some cases, a composition of several services may be required. In semantic web service composition, a repository of services is given in which input and output parameters of each service are annotated with a concept from ontology. Given request is defined by a set of known input concepts and a set of wanted output concepts. Semantic composer should find a sequence of services, called composition that satisfies the wanted output concept.

In this paper, a semantic composer based on ant colony optimization method is proposed. This semantic composer is tested using the challenge set of Web Service Challenges (WSC) 2008. The proposed composer can find optimal composition length in each challenge set in a reasonable time. The result of the suggested composition has been compared with the best result reported in WSC 2008 and in most cases; it performs similar or better results than the other systems.

1. Introduction

The aim of research on semantic web services is the provision of a way for automated handling of web services. This is because web service stack doesn't supply sufficient means for describing services in a way that supports mechanisms for discovering, composing and executing web services.

Semantic web introduces a framework for semantic description of web services and related aspects. However, when a specific functionality has not been provided by a single service, a combination of services is required.

The composition of services produces a set of queried output parameters from a set of given input parameters. In the semantic web service composition problem, we have a repository of services. Each service has input and output parameters. However, instead of value of parameter, each parameter is annotated to a concept within an ontology defined for the services. A given task consists of a set of known input concepts and a set of desired output concepts. A semantic composer should find a sequence of services that provides desired output parameters.

The remainder of this paper is organized as follows: In Section 2 a review of the previous works has been presented. Section 3 defines the semantic web service composition problem. In Section 4 the semantic web service composition system based on ant colony optimization method is discussed. Section 5 presents the experimental results and finally in Section 6 a conclusion is drawn.

2. Literature Review

Many service management systems use semantic service discovery [1] [2] [3]. Also some efforts were done for semantic web service composition such as the works that described below:

Zhang, Raman, et al. [4] use hash table for mapping services to the their output and input parameters. The method is based on breath-first search and is used for forward and backward composition of services. A backwards search successively finds services that supply unknown parameters until all desired outputs are satisfied. Forward searching (like our proposed algorithm) in this context means that services that can be invoked with the known parameters are iteratively added to the composition until all of the desired outputs can be provided. They concluded backward search outperforms forward search and accept it in their final solution.

The Multiagent Web Service Composition Engine (SCE) [5] contains two architectural components: the Java Agent Development Framework (JADE) and a service description repository. In this system services and the given requests are represented by agents that communicate with each other and solve the request, cooperatively.

Weise, Bleul, et al. [6] proposes three different approaches: an uninformed search, an informed search, and a genetic algorithm approach for semantic web service composition. They concluded uninformed search is not suitable for large service repositories.

Informed Search which uses greedy search has superior performance than uninformed approach. The genetic algorithm is much slower but it can always find the correct result for all requests.

Blanco, Cardinale et al. [7] suggest two strategies to prune the space of possibilities for suitable composition while minimizing the evaluation cost. The first one, DP-BF used a combination of a best first strategy with a dynamic-programming technique and can find good web service compositions by pruning the search space to a small portion. The second one, PT-SAM, uses a Petri net unfolding algorithm and tries to find a desired marking from an initial state. They claim that the result of their algorithm is very close to the optimal solution by the exhaustive algorithm and the optimization time is close to the time required by SAM [8].

SWSDS is a composition system [11], [12] that is used both syntactic and semantic matching for service parameter by switching a so-called search index. SWSDS uses an uninformed composition algorithm but extended with the constraint that each service is only considered one time to be part of a composition. The approach outperforms the weak performance of the uninformed approach.

The INFRAWEBS approach [15] is a system that composes WSMO-based services [16]. This composition algorithm finds an appropriate service composition by run-time decomposition of the user goal into sub-goals and discovering the existing services capable of satisfying these sub-goals. It keeps compatibility of services by the consistent description of the composite goal template, prepared by the service application provider in the design-time. The run-time Composer can find a proper orchestration of services in the composition as well as to find the appropriate service substitutions when some of the services in the composition can not be executed. In this system the run-time Composer is implemented as two separate sub-modules: service composition finder and composition execution engine.

3. Semantic Web Service Composition Problem

Defining a semantic web service requires an ontology that defines all concepts and relations between them. Instead of using values for the input and output parameters of services, these parameters are annotated to the concepts defined in an ontology.

Suppose M is a set of all concepts defined in the ontology of web services. According to [6] if $A, B \in M$ are two concepts, operator *subsume* is defined between *A, B* if and only if *A* is a generalization of *B* (B is then a specialization of A). If neither *subsume (A, B)* nor *subsume (B, A)* holds, *A* is not related to *B*. If *subsume (A, B)* and *subsume (B, A)* holds, *A=B*

Suppose an input parameter *x* of a service is annotated to the concept *A* and value *y* is annotated to the concept *B*. Thus a service parameter *x* can be initialized by value y and service invoked if *subsume (A,B)* holds. Therefore, *y* supply more or equal information needed by *x*.

In the semantic web service composition system, we have a set of services defined in the service repository S, each service $s \in S$ explained by a set of input and output parameters annotated to the concepts of ontology in the form of $s.in, s.out \subseteq M$ respectively. We can invoke service s if we can supply all of its input parameter concepts.

A given request to the composition system consists of a set of known input concept $R.in \subseteq M$ and a set of wanted output concept $R.out \subseteq M$. A semantic composer should discover a set of services

$\{s_1, s_2, ..., s_n\}$ that satisfy "(1)", this equation imply all of input parameter concept of service s_1 should be provided by $R.in$. Also after invoking service s_1 both $R.in$ and $s_1.out$ (output parameter concept of s_1) are available for executing the next service s_2 and so on.

In fact, after adding each service in the valid composition, all of output parameter concepts of new service along with other output parameter concept of pervious services together with known input concept can be used for invoking the next service in the composition until the valid composition is generated. The valid composition should satisfy all of wanted output concept $R.out$. Therefore each valid composition should be hold

$$\begin{aligned}
&\forall A \in s_1.in \exists B \in R.in : subsumes(A,B) \wedge \\
&\forall A \in s_i.in, i \in \{2,...,n\} \\
&\exists B \in R.in \cup s_{i-1}.out \cup ... \cup s_1.out : subsumes(A,B) \wedge \qquad (1) \\
&\forall A \in R.out \exists B \in s_1.out \cup ... \cup s_n.out \cup R.in : \\
&subsumes(A,B)
\end{aligned}$$

A semantic composer should explore all of possible permutation of the services in the service repository that satisfy "(1)". If service repository is relatively large, an evolutionary meta-heuristics method is a candidate to search the feasible space and find optimal composition of services. Genetic algorithm successfully used to solve this problem [6]. In this paper, a composition algorithm based on ant colony optimization method is suggested. It is tested via all challenge set presented in the WSC challenge 2008. This system can be reached to the optimal sequence of services for each challenge set. Also it is compared with results of the best solution gained in this challenge. In the most cases the suggested system is able to find optimal composition length with similar or lower time than best system reported in this challenge.

4. Semantic Web Service Composition Based on Ant Colony Optimization Method

Ant colony optimization (ACO) is a population based metaheuristic that can be used to find optimal solution for difficult optimization problem.

In ACO, there are a set of software agents called ants that explore search space to find a good solution for a given optimization problem. Ants incrementally build the solution. Construction process is stochastic and usually a heuristic function guides the ants to select the next component of the solution at each step. This process is biased by a pheromone model. Several types of ACOs have been proposed in the literature. The most successful ones are, ant system [13], Ant Colony System [9] and Max-Min Ant system [14]. In this paper, the Ant Colony System (ACS) introduced by Dorigo and Gambardella (1997) [9] has been used to investigate the search space of semantic web service composition problem.

In the proposed semantic composer, at first, the service repository is become shorter by an algorithm. This algorithm chooses a subset of all services in the repository as candidate service (CS) set. For each service s in the candidate service set the following condition should be hold:

$$\begin{aligned}
&s \subset S \wedge s \in CS \\
&\forall A \in s.in, \exists B \in R.in : subsumes(A,B) \qquad (2)
\end{aligned}$$

In fact "(2)" implies that all of the input parameter concepts of the selected services in the candidate service set should be satisfied by the known input concept $R.in$ and subsume operator between these concepts should be hold. Therefore, all of services in the candidate service set $s \in CS$ can be invoked and selected as first service in the possible valid service composition $\{s_1, s_2, ..., s_n\}$.

At first, each ant initializes your composition with a random service $s \in CS$ in the candidate service set. Also every ant initializes two sets called known concept and wanted concept. Known concept set is initialized with known input concept $R.in$ and wanted concept set is initialized with wanted output concept $R.out$ of a given request. In the proposed system, each ant uses forward search. As explained before, it means that services that can be invoked with the known concepts are iteratively added to the composition until all wanted output concepts can be generated.

Every ant in each iteration of constructing your trail uses a simple heuristic function to select a feasible service from CS as defined in "(3)". Suppose s' is the last service in the current incomplete composition and $s \in CS$:

$$\begin{aligned}
&\forall A \in s'_i.out \\
&\exists B \in s.in : subsumes(B,A) \qquad (3) \\
&\therefore s \in FeasibleSet
\end{aligned}$$

Above equation describes that each service in the candidate service set has a chance to be selected as next service in the valid composition, but the chances are not equal. In fact, services that have more input concept that is hold subsumes operator with output concepts of the last service in the current incomplete composition have more chances compared to other services in the candidate service set. The more input concept holds subsumes operator with output concept of last service, the more chances to be selected as the next service by the ant. Services in the candidate service set holds "(3)" called feasible service set and can be invoked after the last service in the current incomplete composition.

If none of the services in the candidate service set satisfy "(3)", the feasible service set is empty. However, services in the candidate service set with more output concepts have more chances to be selected as the next service than the other services.

According to the simple heuristic described above for each service $s_i \in CS$ in the candidate service set value η_i in the ACS is computed.

In this problem, pheromone trail parameter τ_i is associated to each service s_i in the service repository. In the ACS optimization algorithm, decision rule used by each ant during the construction process is pseudorandom proportional rule: the probability for each ant to select a service from candidate service set depends on a random variable q uniformly distributed over [0, 1] and a parameter q_0; if $q <= q_0$ among the candidate services, the service that maximum the product $\tau_i.\eta_i^{\beta}$ is chosen, otherwise it makes a probabilistic decision at each service in the candidate service set.

In ACS optimization method, we have two pheromone update phase, local pheromone update performed by all ants after selecting each service in construction step. Each ant applies it only to the last service selected. The pheromone trail parameter τ_i of the selected service s_i in the service repository is changed according to the equation "(4)".

$$\tau_i = (1-\varphi).\tau_i + \varphi.\tau_0 \quad (4)$$

$\varphi \in (0,1]$ is the pheromone decay coefficient and τ_0 is the initial value of pheromone. Local pheromone update makes it less likely that several ants produce identical compositions during one iteration.

Meanwhile offline pheromone update is performed only by the best ant, and the pheromone trail parameters τ_i of the selected services by the global best ant are updated according to the "(5)".

$$\tau_i = (1-\rho).\tau_i + \rho.\Delta\tau_i^{best} \quad (5)$$

$\Delta\tau_i^{best} = 1/L_{best}$ if the best ant used service s_i in the composition and $\Delta\tau_i^{best} = 0$ otherwise. L_{best} is the length of best valid composition found until that generation.

After each ant selects a service and adds it to the current incomplete composition, the known concept set and the candidate service set for each ant is changed according to "(6)" and "(7)". Suppose an ant select service s_i at current iteration, therefore all of the concepts in the output parameter of service s_i should be inserted to the known concept set as defined in "(6)".Thus, after selection of a new service, the known concept set of each ant is expanded by the output parameter concepts of the selected service.

$$\text{Known concept set} = \text{known concept set} \cup s_i.out \quad (6)$$

Since known concept is changed after each service selection in the current incomplete composition, candidate service set is also changed. This set is changed according to the "(7)".

$$\begin{aligned} &\forall s_i \in S, \\ &\forall A \in s_i.in, \exists B \in KnownConcept : subsumes(A,B) \\ &\therefore CS = CS \cup s \end{aligned} \quad (7)$$

The above equation implies new services that belong to the service repository and can be invoked are added to the candidate service set at the end of one iteration. These new services are added because all of input parameter concepts of them are satisfied by the changed known concept set. Thus the search space of each ant expands after selecting a new service. Therefore, new service added to the current composition until all of wanted concept is satisfied and a valid service composition is generated. Finally the best ant provided optimal composition length the in reasonable time is reported.

5. Experimental Result

Test set are selected from Web Service Challenge 2008 [10]. The data are categorized as challenge sets. Each challenge set has a WSDL file containing a set of services along with annotations of their input and output parameters. The number of services changes from one challenge set to another. Every service has an arbitrary number of parameters. Beside the WSDL file, an OWL file has been provided, as well. The OWL file contains the taxonomy of concepts used in the challenge set in the OWL format. Also, a challenge request is specified using WSDL description.

The challenge document requests a service composition with the given input concepts and the requested output concepts. In WSC 08, there are some challenge sets and proposed semantic composer is tested with them. Minimum composition length and time elapsed to get optimal service composition is reported in Table 1.

As shown in Table 1, the proposed composition system is able to find service composition with optimal length as reported in WSC 2008. Therefore, the semantic composer based on ant colony optimization method are robust to find optimal solution for each challenge set for different number of services.

Also a comparison between proposed composition system and the winner of the WSC 2008 reported in Table 2 and Figure 1.

Values in each cell of Table 2 indicate min composition length/min execution time(ms) that proposed system and other systems can obtain for 3 challenge sets 04,05,06.

The proposed composition system (PCS) can find the optimal composition length the same as the best ones in this challenge. The composition length in challenge set 04, challenge set 05 and challenge set 06 are short, medium and long, respectively. In challenge set 04, 05 the elapsed time is lower than Pennsylvania State University (PSU). In the challenge set 06, the suggested composition system can find optimal composition but the time elapsed to reach this is greater than that of Tsinghua University (TU) and University of Groningen (UOG) systems. The bigger time achieved in this experiment is because of using evolutionary algorithm. The evolutionary algorithm run via some generation and usually requires more time in comparison with other greedy algorithms. However, the composition length of the proposed system is the optimal composition length reported on this challenge. It should be noted that the University of Kassel (UOK) and the Pennsylvania State University systems could not gain optimal service composition for challenge set 6.

Ant colony parameter used in the proposed composition system is defined as follows:

$$\varphi = 0.1, \rho = 0.1, Q_0 = 0.8, \beta = 2$$

Time reported in this paper is measured on Quad CPU 9400, 2.67 GHz and 3.25 GB of RAM.

Table 1: result of proposed composition system via each challenge set of WSC 08

Challenge set	Number of services	Number of concepts	Number of given input concept	Number of given output concept	Minimum composition Length (Proposed Algorithm)	Min Execution Time (ms)	Optimal composition length (reported in WSC 2008)
ChallengeSet01	100	5000	3	2	10	32	10
ChallengeSet02	500	5000	4	1	5	78	5
ChallengeSet03	500	5000	3	1	40	985	40
ChallengeSet04	1000	10000	6	4	10	1578	10
ChallengeSet05	1000	10000	2	3	20	70953	20
ChallengeSet06	2000	40000	9	4	40	1807219	40
ChallengeSet07	4000	10000	8	1	20	4446047	20
ChallengeSet08	8000	40000	5	4	30	1892828	30

Table 2: comparison between proposed composition system and best system introduced in WSC08

	PCS	TU	UOG	PSU	UOK
Challenge set 4	10/1578	10/312	10/219	10/28078	10/828
Challenge set 5	20/70953	20/250	20/14734	20/726078	21/300219
Challenge set 6	40/1807219	46/406	37/241672	error	no result

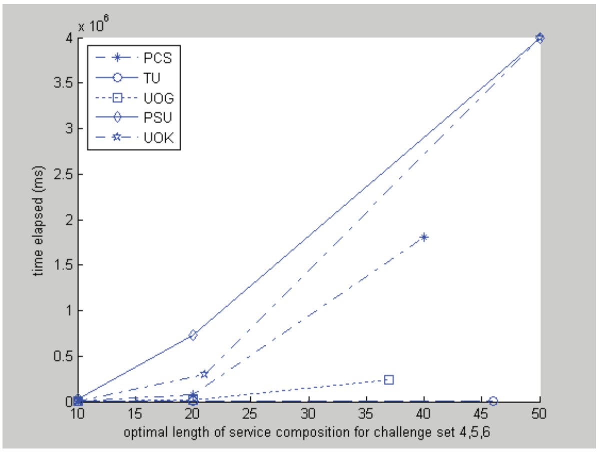

Figure 1. A comparison between the proposed system and the best results of WSC 2008

6. Conclusion

In this paper a semantic web service composition system is suggested. The application of metaheuristic optimization method such as genetic algorithm to solve this problem has been considered in [6] but in this paper ant colony optimization method was used.

It was shown that the proposed system can find the optimal length of web service composition with the different challenge set for different number of services. The algorithm was shown to be robust for finding the optimal solution. However, the time needed to reach to the optimal composition in some cases is longer that other methods, such as methods based on greedy search method (using either breath first or depth first searches).

7. References

[1] A. Gomez-Perez, R. Gonzalez-Cabero, and M. Lama, "Ode sws: A framework for designing and composing semantic web services" , *IEEE Intelligent Systems*, vol. 19, July –Augest 2004 , pp. 24–31.

[2] S. Bleul, T. Weise, and K. Geihs, "An ontology for quality-aware service discovery", *Computer Systems Science Engineering*, vol. 21, no. 4, July 2006.

[3] A. Gomez-Perez, R. Gonzalez-Cabero, and M. Lama, "A framework for designing and composing semantic web services", Proceedings of 2004 AAAI Spring Symposium Series, March 2004.

[4] Yue Zhang, Krishna Raman, Mark Panahi, and Kwei-Jay Lin, "Heuristic based service composition for business processes with branching and merging" , In Proceedings of CEC/EEE 2007, pp. 525–528.

[5] Paul A. Buhler, Dominic Greenwood, and George Weichhart , "A multiagent web service composition engine ", In Proceedings of CEC/EEE 2007 , pp. 529–532.

[6] Thomas Weise, Steffen Bleul, Diana Comes, Kurt Geihs , "Different Approaches to Semantic Web Service Composition", in proceeding of ICIW 2008 , pp. 90-96.

[7] Eduardo Blanco ,Yudith Cardinale , Mariıa-Esther Vidal ,"Techniques to Produce Optimal Web Service Compositions" , in procedding of Services 2008, pp. 553-558

[8] A. Brogi, S. Corfini, and R. Popescu , "Composition oriented service discovery" , In Proceeding of Software Composition' 05, LNCS, vol. 3628, 2005, pp. 15–30.

[9] M. Dorigo, L. M. Gambardella, "Ant colony system :a cooperative learning approach to the traveling salesman problem ", *IEEE Transactions on Evolutionary Computation*, vol.1 , no. 1, 1997.

[10] Proceedings of IEEE CEC/EEE 2008, July 21-24, 2008.

[11] B. Xu, T. Li, Z. Gu, and G. Wu, "SWSDS: Quick web service discovery and composition in SEWSIP", in IEEE Joint Conference CEC/EEE 2006, pp. 429–451

[12] Z. Gu, B. Xu, and J. Li, "Inheritance-aware document-driven service composition" ,in IEEE Joint Conference CEC/EEE 2007, pp. 513–516

[13] M. Dorigo,V. Maniezzo and A. Colorni , "Ant Sysstem: optimization by a colony of cooperating agents" , *IEEE Transaction on systems , Man , and Cybernetics* –part B , vol. 26, no. 1 , 1996,pp. 29-41.

[14] T. Stuzel and H. H. Hoos , "MAX-MIN Ant System" , *Future Generation Computer Systems* , vol. 16 , no. 8, 2000,pp. 889-914

[15] Agre, Marinova , "An INFRAWEBS Approach to Dynamic Composition of Semantic Web Services" , Cybernetics and Information Technologies , vol. 7, no. 1,2007,pp. 45-61.

[16] Roman, D., U. Keller, H. Lausen (Eds.) , Web Service Modeling Ontology, 2006, WSMO Final Draft.