Fault Tolerance in Automatic Semantic Web Service Composition based on QoS-awareness Using BTSC-DFS Algorithm

Azam Kargar  
Department of Computer Engineering  
Islamic Azad University, Yazd Branch  
Yazd, Iran  
kargar.azam@gmail.com

Sima Emadi  
Department of Computer Engineering  
Islamic Azad University, Yazd Branch  
Yazd, Iran  
emadi@iauyazd.ac.ir

Abstract—Quality-of-Service (QoS)-aware web service composition is very important in integrating individual services with respect to the functional and nonfunctional requirements. Despite the large number of candidate services, automation of the combination is essential in order to provide a good combination of service. Although many of the existing methods offer a solution that is optimal, most of them have little flexibility. In some cases, the compound service components fail, so the combination algorithm has to run again to find another optimal solution. Also in many situations, users prefer to have several alternative solutions. Therefore, providing a top-\(k\) service composition according to their QoS is more desirable. Because web services are unreliable, also since providing the transactional support in execution of a service composition is an important design requirement; so in this research, a fault management procedure is implemented to ensure the transaction execution of the service combinations. This procedure in the event of service failure undo the impact of this service by calling the equivalent service.

The proposed method encounters these three issues: 1) the semantic selection of services; 2) QoS-aware web service composition with the top-\(k\) solution; and 3) fault-handling/recovery procedure. In the proposed method, ontology concept ranking algorithm is used in service selection, and top-\(k\) method is employed to solve service combination. Error handling procedure will then be reviewed and designed to ensure the transactional execution of the service composition. The evaluation results show that the proposed method not only finds optimal solutions but also can provide alternative solutions with optimal QoS.

Keywords—Semantic Web Service Composition, Ontology Concept Ranking, Semantic Matching Relaxation, Top-\(k\).

I. INTRODUCTION

By increasing the number of services, it is necessary to find the services that provide the user functional requirements. Therefore, the service discovery process is required to be effective. Most of the methods in the service discovery process compare only the input and output of the services, which has a low accuracy and therefore cannot properly satisfy the needs of the user. In order to solve this problem in the proposed method and in the service selection phase, in addition to considering the input and output, other parameters such as concepts and semantics are considered, which leads to the discovery service process in more detail and precision. This algorithm offers a close replacement for user's requests in cases where it cannot meet the user's needs.

Also by increasing the number of services, the dynamic service composition in a dynamic environment must be compatible; furthermore, many of the alternative deal with scalability which require for appropriate combination solutions, which has been neglected in previous studies. In addition, in previous research, fault recovery/management procedures have not been considered in the implementation of the service combination, so in this research, a method is proposed to manage the failure of services during the running of a composite service. The proposed method shows these three solution:1) semantic selection of services; 2) QoS service-composition with K-solution; 3) fault-handling/recovery procedure. In the service selection, the ranking algorithm for ontology concepts is used to determine similar concepts based on different degrees of semantic matching. The service composition consists of three steps: (1) preprocessing, which involves creating a Rule Repository from similar semantic services, which is the output of the service selection phase, (2) filtering the services based on the user's request and step and (3) finding the \(K\) combination solution based on QoS parameters. This approach employs a combination of recursive search and first-depth search. So in this research, the service composition process is transformed into a graph searching problem, and each composition solution is represented in a directed acyclic graph.
The fault tolerance in service composition is usually supported by the exception handling routines in languages such as WS_BPEL. However, it was observed that putting the fault-handling/recovery logic and the business logic together in a same orchestration usually results in a large and unmanageable workflow, particularly when the fault-handling logic is generated automatically as proposed in some previous works. In this paper, the fault-handling/recovery logic has been generated as a separate module invoked from the main orchestration iteratively.

The present research includes five sections. The next section briefly discusses the related works about service composition. In section 3, the proposed method is presented. Section 4 analyzes and evaluates the proposed method. Finally, the conclusion is expressed.

II. RELATED WORK

In order to compare viewpoints with the proposed method of the research, the methods of combining services are categorized into methods based on the artificial intelligence programming based on graph theory, petri nets, and service-quality-aware composition. Further, the papers used in the research are described in detail by classification.

A. Rafiee et al. [1] have provided an integrated approach for automating the semantic combination using artificial intelligence programming techniques. Their proposed method is responsible for transferring the process, increasing semantic efficiency, choosing a service based on quality parameters, replacing services, and displaying results. Yuqian Lu et al. [2] proposed a model for combining the knowledge-based services and adaptive resource planning in a cloud computing environment. The goal of developing an integrated network environment is the rapid authorization of resource allocation for a specified service request, in accordance with governance policies, access policies, resource access information, and so on. Adami Pinto et al. [3] have evaluated the efficiency of an automated service combination. In the proposed approach, by examining the integration of the two tools, the automatic service combination system and the extensible framework, it attempts to access semantic services with the aim of evaluating the efficiency of the automated combination of services. In respect with service composition methods based on the theory of graph, Wang et al. [4] focus on the subject of the service combination due to its qualitative limitations. Initially, the problem is described regarding a number of real-world business scenarios. Then, the service quality and formal terms are defined. In the end, a graph-based algorithm and two different preprocessing methods are presented for improvement. Puttonen et al. [6] have proposed a service-based framework that automates the process of combining services based on semantic descriptions. The results indicate that their proposed method reduces the workload of semantic web service descriptions. Although the development of semantic descriptions will produce significant workloads, this method enables production systems to fully achieve new goals without additional efforts from domain experts. F. Karatas [7] has provided a method for identifying the quality of the service combination in terms of security objectives, which uses the concept of structural analysis and estimates the impact of a qualitative feature on the security goal. They claimed that since every security objective is modeled with a fitness function, the model is solved similar to the multi-objective optimization problem. D. Darling et al. [8] have designed a conceptual approach to combining services based on functional and nonfunctional requirements. They used an ontology-based method to identify the concept of the services and their functionality. Service composition techniques based on artificial intelligence algorithms create an optimal service composition with a simple and fast design, but these algorithms do not provide accurate and optimal responses. On the other hand, the graph-based algorithms provide a precise response, but as the number of services increases, computations and the graph traversal get complicated and require complex computational operations.

III. PROPOSED METHOD

The proposed method is shown in Fig. 1, which consists of three phases. In the service selection phase, semantic analysis methods of services are used to determine similar concepts based on different degrees of semantic matching. Service composition phase consists of three steps; (1) preprocessing, which involves creating a rules repository from similar semantic services, (2) semantic services filtering based on user requests and (3) finding the K combination solution QoS-based service composition in response to user’s query. The third phase is an orchestration pattern for building a transactional service composition in which the failure-recovery logic is imposed on the composition workflow as a separate module. This module is generated to schedule the execution of services within the composition based on the composition transactional logic and properties such that upon a service failure, a recovery path with the lowest cost is followed.

A. Service selection algorithm

In the semantic service selection, concept ranking algorithm is applied over this hierarchy in order to determine similar concepts based on different degrees of semantic matching relaxation. In this algorithm, using semantic analysis methods and hierarchical relationships between concepts, a list of similar semantic concepts is presented based on the user-defined threshold.

There are four hierarchical relationships between the two concepts A and B in the ontology [9]:

Exact (A, B): The two concepts must have the same URI or equivalent.

Plugin (A, B): Concept A must be derived from concept B.

Subsume (A, B): concept B must be derived from concept A.
Sibling (A, B): The two concepts A and B of the super class are not necessarily direct.

Two methods have been proposed to determine the semantic distance between two concepts in the ontology [1]: the edge-counting method, which is the distance between the two concepts in the shortest path between those concepts in the ontology, and the Upward Cotopic distance, which is the concept of superclass including itself.

As mentioned, in the semantic service selection, the concept ranking algorithm is used. The input to this algorithm includes Query concept q, Hierarchical relationship H, and Distance threshold D, and its output is a list of similar concepts to q. The output list of the algorithm is calculated on the basis of two measurement methods separately, and the values of the list of similar concepts that satisfy the user-defined threshold are chosen as semantic concepts similar to the concept in query.

B. Automatic Semantic Web Service Composition based on QoS-awareness Using BTSC-DFS algorithm

The first step is to preprocess the raw data of dataset and transform the services set to a rule repository. Rule repository is a kind of in-memory data structure, which can be accessed fast and efficiently when responding users’ requests. The second step is semantic service filtering: upon receiving the user request, the services are fetched from the service repository and the services that are not possible to be present in the final result of the combination in accordance with the user's request and the rules in the repository are filtered.

Filtered services are layered in order for fast processing in the next step. The final step, which is the core of the whole architecture, is responsible for finding top-k service composition result. The top-k service composition should find from dataset (such as UDDI). In the proposed method, the composition is mapped into multiple tasks and the results from all the tasks can form the final results. For each task, a solution sub-graph is generated and the resulted sub-graphs are combined to generate the final top-k results. The target of BTSC-DFS is to generate a solution sub-graph for a concept.

For each concept, the rule repository is searched and services that generate the output of the concept are obtained [5].

In each iteration of backtracking, instead of choosing all these services to continue to backtrack, BTSC-DFS only chooses parts of them (may be one or multiple). If services that provide the concept require the same inputs, they will be regarded as one common service chosen to backtrack. This is why the nodes in solution sub graphs may be a set of services. BTSC-DFS consists of two steps. First, find all possible critical paths that output the required concept. Then, add noncritical paths to each critical path to form a solution graph such that all the inputs of services are provided by either the user or other services in the solution graph.

C. Proposed model of recovery logic / error management routines

Since the logic of handling service failures might become very complex in some applications, this concern should be separated from the normal flow of the composition workflow based on the separation of concerns principle. To achieve this, in this study we have proposed an orchestration pattern for building a transactional service composition in which the failure-recovery logic is imposed on the composition workflow as a separate module. This module is generated to schedule the execution of services within the composition based on the composition transactional logic and properties such that upon a service failure, a recovery path with the lowest cost is followed. The method proposed for implementing the service combination includes two parts, namely the orchestration pattern and the recovery logic routine. The orchestration pattern consists of services and equivalent compensation services, which is based on business logic. In the recovery logic routine, the rollback cost and failure probability of the service are calculated first, and after finding the paths with the lowest cost, the combination is made and the result of the combination is placed in the tree. The list of required services is then sent to the orchestration via the tracker engine. The Controller web service is the interface between the tracking engine and the orchestration. The Web Service Controller takes the list of services from the tracker engine and sends it to the
orchestration. The orchestration sends the current status of the services through the controller web service to the tracker engine, and the tracker engine updates the tree based on the status of the services.

IV. EVALUATION OF THE PROPOSED METHOD

In this study, Automatic Semantic Web Service Composition Based On QoS-awareness using BTSC-DFS algorithm is evaluated in a variety of ways. The execution environment is: Intel (R) Core(TM) i3-4160 CPU @ 3.60 GHz with 4GB RAM, Windows 7 Ultimate SP1 32-bit. In the following, we provide the details of our experiments. The efficiency and precision of the algorithm are evaluated by the OWL-S-TC2 Dataset Semantic Web Services Version 1.1. This dataset contains 1007 services in seven different domains and 27 queries related to 10 to 75 services. The algorithm presented in this study by comparing the meanings of input and output parameters of the service, rating services and their ranking, has taken a step towards increasing the degree of accuracy in the discovery of services. The implementation time of the ranking algorithm for ontology concepts in the selection phase of the service is investigated first. Then, the combination algorithm is evaluated in terms of runtime and the amount of memory space consumed with the number of service variables. Details of the proposed method diagrams are described in more detail below.

A. Evaluation of the efficiency of the service selection algorithm

In Fig. 2, an evaluation is shown based on the ontology concepts ranking algorithm in domains with the number of services varying from 50 to 1000. The response time increases with the increase in the number of services in the investigated range.

B. Evaluation of Service Combination Algorithm

Evaluation of the accuracy of the combination algorithm: As shown in Table 1, the proposed algorithm at Recall has a higher precision value. This means that with an increase in the number of related services discovered, the number of unrelated services, as compared to the combination algorithm without semantic analysis, has decreased significantly.

The effect of semantic analysis on the combined algorithm with the number of service variables: In Fig. 3, the runtime of the combination algorithm is compared in a case using semantic analysis of services with a case without it, with the number of service varying from 100 to 1000. The use of semantics in the combination algorithm results in a shorter response time than the combination algorithm without semantic analysis.

C. Analysis and evaluation of the effectiveness of recovery routine

The effect of increasing the number of services on the overall implementation time of the combination: In Figs. 4 and 5, the overall execution time of the combined algorithm is compared with the use of recovery procedures without using it, with the number of services varying from 15 to 35. Experiments with a failure rate of 0% and 40% were performed. By increasing the number of services in the combination, the growth of Web service overhead in the proposed method is insignificant, using recovery procedures, in comparison with the top-k method, without the use of recovery procedures.

In Fig. 6, the average cost of service rollback is shown differently. According to Fig. 6, the average cost of rollback in the proposed method has significantly decreased.
Automatic Semantic Web Service composition based on QoS-awareness using BTSC-DFS algorithm. In the final step, error management procedures were explored and designed to ensure the transactional execution of the service combination. Experimental results show that the proposed method is able to provide alternative solutions with optimal QoS value. Also, the response time of the service selection algorithm grows up with an increase in the number of services. Because the ranking algorithm has high complexity in the Upward Cotopic distance calculation between the two concepts. In addition, according to the obtained graphs, the average cost of rollback in the proposed method has significantly decreased. Because services that have rollback dependency on another service cannot be called up until the successful completion of this service. Finally, the memory consumption is calculated according to the number of nodes of the solution graph. But since the number of graph nodes in the different datasets with the same number of services is not constant, it is not possible to compare the consumption of memory in terms of the number of services.

In this research, the response time of the proposed algorithm has been partially affected by similarity detection methods, which will be solved in our future work study. A solution can also be found for saving and reusing existing service combinations.

### REFERENCES