Knowledge-Based Systems 62 (2014) 98–112

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Dynamic composition of Web services using efficient planners in large-scale service repository

Guobing Zou [a], Yanglan Gan [b,*], Yixin Chen [c], Bofeng Zhang [a]

[a] School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China
[b] School of Computer Science and Technology, Donghua University, Shanghai 201620, China
[c] Department of Computer Science and Engineering, Washington University in St. Louis, MO 63130, USA

ARTICLE INFO

Article history:
Received 7 July 2013
Received in revised form 16 February 2014
Accepted 1 March 2014
Available online 13 March 2014

Keywords:
Web service
Service composition
Automated planning
Deterministic planner
PDDL

ABSTRACT

Web services as independent software components are published by service providers over the Internet and invoked by service requesters for their desired functionalities. In many cases, however, there is no single service in a Web service repository satisfying a service request. So how to design an efficient method for composing a chain of connected services has become an important research issue. Recently, much research has been done into the search time reduction when finding a composite service. However, most methods take a long time for traversing all of the Web services in a service repository, thus it makes their response time significantly overrun a user's waiting patience. This paper develops an efficient approach for automatic composition of Web services using the state-of-the-art Artificial Intelligence (AI) planners, where a Web service composition (WSC) problem is regarded as a WSC planning problem. Unlike most traditional WSC methods that traverse a Web service repository many times, our approach converts a Web service repository into a planning domain in PDDL just once, which will only be regenerated when the Web service repository changes. This treatment substantially reduces the response time and improves the scalability of solving WSC problems. We have implemented a prototype system and conducted extensive experiments on large-scale Web service repositories. The experimental results demonstrate that our proposed approach outperforms the state-of-the-art.

1. Introduction

Web services are loosely coupled, self-descriptive, modular and Web-accessible distributed software components. They can be published in a Web service repository, discovered by software agents and composed as new value-added Web services. In most cases, the standard Web Service Description Language (WSDL[1]) is used to describe the input and output interface of a Web service for its functionality at the syntactical level. Meanwhile, Simple Object Access Protocol (SOAP) is commonly used for transferring messages and communications among Web services. Recently, Web service has become more and more important as it offers an extremely versatile and powerful tool to dynamically create distributed applications on demand. Its applications increase rapidly in many fields, such as electronic commerce, enterprise application integration and geographic information systems.

Web service discovery (WSD) aims at finding a service to fulfill a given service request. In many cases, however, no single service in a Web service repository is capable of fulfilling a service request. Therefore, it is mandatory to find a chain of services. They can be functionally connected together as a new composite service to satisfy the given service request. The problem of finding a group of connected and composable services from a Web service repository is called Web service composition (WSC). There are two ways in solving a WSC problem. The first way is to build a workflow business model [1] by domain experts with the help of GUI-based modeling software. However, it is impractical and error-prone when a Web service repository involves a large number of services. The other is to automatically and efficiently compose existing services from a large-scale Web service repository, which has become an important research issue in Web service community.

To efficiently solve the problem of dynamic composition of Web services, the main idea of this paper is to consider a WSC problem as a WSC planning problem, and then to use the state-of-the-art AI planners (e.g., Metric-FF [2] and SatPlan06 [3,4]) to find a composition

* Corresponding author. Address: School of Computer Science and Technology, 2999 North Renmin Road, Shanghai, China. Tel.: +86 21 67792291; fax: +86 21 67792106.
E-mail addresses: gbzou@shu.edu.cn (G. Zou), ylgan@dhu.edu.cn (Y. Gan), chen@cse.wustl.edu (Y. Chen), bfzhang@shu.edu.cn (B. Zhang).

[1] http://www.w3.org/TR/wsdl.

http://dx.doi.org/10.1016/j.knosys.2014.03.002
0950-7051/

plan for a composition request. More specifically, we first convert all of the available services in a large-scale Web service repository into a planning domain in Planning Domain Definition Language (PDDL[2]), and then translate a composition request into a planning problem in PDDL. Finally, a WSC planning problem (consisting of a PDDL domain and a PDDL problem) is fed into an efficient AI planner, which can automatically find a composition plan for the given composition request.

The most distinguishing characteristic of our method lies in its shorter response time compared to other AI planning based WSC methods [5–9]. From the perspective of applicability, the response time is crucial to a WSC problem. It determines whether a WSC method can quickly respond to a composition request within a short period of time. Our proposed method only needs to traverse a Web service repository once. By doing so, time spent on parsing the Web service repository, which is a major part of the response time, can be greatly saved in our method. On the contrary, despite the fact that other AI planning based WSC methods, such as search by heuristic function [7,8] and planning graph model [9], have taken many efforts on reducing search time when finding a composite service, they repeatedly traverse all services in a Web service repository whenever a user submits a composition request.

The second characteristic of our proposed method is its compatibility with the exploitation of multiple automatic planners for service requesters. As it can be used with classical planners that support PDDL, users can choose from available desired planners according to their personalized requirements. For instance, SatPlan06 planner [3,4] can be used to find a composition plan with the minimum number of parallel steps, while Metric-FF planner [2] can find a feasible composition plan more quickly. Conversely, other AI planning based WSC methods, such as [5–9], take fixed composition modes or search strategies that cannot be served for complex user requirements in terms of their personalized preferences.

The third characteristic of our WSC method is its powerful applicability in real-world applications. Although most of recent WSC approaches [10–14] have taken into account semantic Web services described either in DAML-S[3] or OWL-S,[4] both providers and requesters have to describe the services in terms of ontological concepts to avoid semantic heterogeneity. The requesters may have difficulty in framing a service request correctly because of strict semantic rules to specify service functionality. Moreover, the construction of domain ontology for each area is also a challenging task with the help of domain experts. Currently, the dynamic composition of Web services in semantic level is still hard in practice from the view of the semantic annotation by service providers and composition request specification by service requesters. In comparison with the existing planner-based semantic WSC methods, we investigate WSC approach with Web services described in WSDL. It can be technically supported at present by industrial community. However, as more domain ontologies are being constructed and applied in real-world applications, semantic composition of Web services has emerged as a mainstream research direction in WSC community.

As described above, the innovation of this work can be summarized in two aspects. First, we propose a novel approach that translates a WSC problem to a WSC planning problem. It is solved by advanced AI planning techniques that can effectively compose Web services with shorter response time. Second, our WSC approach is based on standard PDDL specifications. It provides those service requesters with the flexibility of utilizing multiple efficient automated planners for personalized composition requests. However, Despite these advantages, the drawback of our approach is its syntactical matchmaking between preconditions and effects among WSC actions. That is, we currently mainly focus on in purely syntactic way rather than semantic composition of Web services. Therefore, this kind of matching scheme could mismatch the scenarios in real world applications, where WSC actions might be highly matched in terms of semantic way by similarity computation with a given threshold. In this aspect, although the proposed approach can solve large instances in a few seconds and outperform the existing WSC approach with faster response time and better scalability, to improve the correctness of WSC, our future work plans to consider semantic similarity calculation between the effects and preconditions of two WSC actions using semantic Web services posted on the website of ICEBE09,[5] where the input and output parameters are described and annotated by the taxonomy of concepts in OWL.

The proposed WSC approach has been implemented as a prototype system. Extensive experiments have been conducted on 18 groups of large-scale Web service repositories involving 81,464 Web services. The experimental results demonstrate that our proposed WSC approach using deterministic planning can significantly outperform WSPR [7,8], which is one of the state-of-the-art Web service composition methods.

The rest of this paper is organized as follows. Section 2 reviews the related work of WSC. Section 3 presents a motivating example in a real-world application. Section 4 formulates the WSC problem and WSC planning problem. Section 5 presents our approach for dynamic composition of Web services using planning. Section 5.1 illustrates its mapping mechanism in the translation process. Algorithms in Sections 5.2 and 5.3 are designed to generate a planning domain and a planning problem. Section 5.4 analyzes computational complexity of WSC planning problem translation. Section 5.5 finds a composition plan. Section 6 presents the WSC system architecture. Section 7 shows and analyzes extensive experimental results. Section 8 concludes the paper and discusses the future work.

2. Related work

According to the applied theories and techniques [15,16], WSC methods can be classified as workflow-based methods, AI planning based methods, graph theory based methods, and program synthesis based methods. In addition, there are other approaches used to address WSC problem, including algebraic process, π-calculus, petri net, model checking and finite-state machine [17].

In the workflow based methods, they first build an abstract business process model that consists of a set of tasks, control and data flow [15]. Then, each task in the process model contains a query clause used to search a real atomic service from a Web service repository. The authors in [18] presented an aggregated reliability (AR) model to measure the probability that the given state will lead to successful execution in the context, where each service may execute with some failure probability. Based on AR computation, it can dynamically select Web services performed on a composite service with more reliable execution. However, an abstract business workflow for a predetermined composite service needs to be modeled before dynamic service selection. The authors in [1] proposed a scientific workflow based editor, which allows scientists to query and compose distributed data sources. As a result, this kind of WSC methods are based on a workflow model and belong to a static composition approach. Moreover, it needs to be manually deployed by domain experts and GUI-based modeling softwares. Therefore, its dynamicity and flexibility are obviously reduced so that it is inappropriate for dynamic composition of Web services in a large-scale Web service repository.

[2] http://cs-www.cs.yale.edu/homes/dvm/.
[3] http://www.daml.org/services/.
[4] http://www.w3.org/Submission/OWL-S/.
[5] http://ws-challenge.georgetown.edu/wsc09/.