Impacts of Various Network Properties on the Performance of Web Service Composition

John Jung-Woon Yoo, Ali Rahmati and Luis Font
Department of Industrial & Manufacturing Engineering & Technology
Bradley University, Peoria, IL

Abstract

This paper discusses various properties in web service networks and analyzes the impact of them on the performance of web service composition. Mainly six factors characterize web service networks. They are the number of web services, the number of distinct parameters, the number of input parameters per web service, the number of output parameters per web service, the number of initial parameters and the number of goal parameters. Among these six factors, the relationship between two factors of the number of web services and the number of distinct parameters with respect to the solution time and the web service composition solution is discussed in detail in this paper.

Keywords
Web Service Composition, Network Properties, Performance Evaluation, Experiments

1. Introduction

Web services is the state-of-the-art technology that provides a variety of real-time services in the World Wide Web (WWW) environment. Web-based service providers publish their services through service repository, called Universal Description, Discovery and Integration (UDDI), so that users can invoke the published web services. Each web service is described in a standard language, such as Web Service Description Language (WSDL) [1]. The WSDL of a web service contains detailed information on the input/output data types, Uniform Resource Locator (URL) and transport protocol required to invoke the web service. Figure 1 shows an illustrative example of a WSDL description for getting contact information of students.

In order to obtain output information from a web service, all the required input information must be provided to the web service. If the seeking output information is simple, then a single web service can bring the output information. For example, finding a direction from City A to City B can be answered by invoking a single web service, such as Google Maps. However, if the required output information is complicated, there might not exist a single web service that provides the required output information. Instead, a set of web services is required, and the web services should be invoked in a specific sequence.

The task of sequencing multiple web services with initial input information to obtain specified goal output information is called web service composition (WSC). Suppose that there is a complicated service request of finding the nearest Japanese restaurant to a particular hotel in San Diego along with a driving direction from the hotel to the restaurant. A web service that answers this complex query rarely exists, and multiple web services, such as finding a restaurant, finding a hotel, and getting a direction, should be invoked in a particular sequence. The sequence can also include branches and merges of web services. In other words, some web services should be invoked in parallel, and some web services should be invoked in serial. Figure 2 illustrates a serial invocation of web services and a parallel invocation.
of web services. In serial invocation case, using GetHotel web service a user obtains hotel location information and then using GetRestaurant web service with the hotel location information, the user can find the location of a restaurant, such as the nearest restaurant from the hotel. In parallel invocation case, once the user gets hotel location information using GetHotel, it searches a restaurant and a theater using GetRestaurant and GetTheater, and then using the restaurant and theater location information, it can find out a driving direction using GetDirection web service.

As a variety of new service requests appear in dynamic business environments, automatic web service composition becomes an essential feature of commercial web services. This paper investigates the performance of web service composition by analyzing major factors that influence the structure of web service network. The rest of this paper is structured as follows: Section 2 reviews literature that addresses the performance of web service composition problems. Section 3 briefly explains the mathematical model for web service composition used in this paper for experiments. The experimental design and the analysis of experimental results are discussed in detail in Section 4. Finally, Section 5 highlights the contribution of this paper and future research topics.

2. Research Background

Bylander [2] proves that the complexity of planning problems is NP-complete, and Oh et al. [3] extends the proof to the web service composition problem. The web service composition problem has been formulated as the Artificial Intelligence (AI) planning problem and a variety of techniques has been used to solve the problem, such as planning-
graph techniques, propositional satisfiability techniques, constraints satisfaction techniques, situational calculus, rule-based planning, theorem proving, mathematical programming, among others [4-6].

McIlraith et al. [7] utilized the ConGolog language [8], a high-level logic programming language based on situational calculus, for an automatic web service composition. Rao et al. [5] applied a linear logic theorem proving approach to the web service composition problem, considering both functional requirements and non-functional attributes in composing web services. One limitation of their work is that they assume that core services are already selected by the user. Oh et al. [9] have defined the web service composition problem in description logic and solved it by using their flexible parameter matching algorithm. Ponnekanti and Fox [10] have developed SWORD, a rule-based set of tools for the composition of web services.

Mathematical programming-based approaches also have been used for web service composition. Zeng et al. [11] and Kritikos and Plexousakis [12] have proposed web service composition methods at a web service level without considering input and output parameters of web services. On the other hand, Yoo [13] and Cui et al. [14] have proposed an binary integer mathematical programming model for parameter-level web service composition. Cui et al. [14]’s model considered multiple objectives in their optimization model and made a brief experiment on the performance of their model.

In the literature review, we find that most of research has focused on methodology development for web service composition. However, there has been few experimental analysis on how major factors that impact the performance of solution methodologies. Therefore, this paper investigates the performance of web service composition. To analyze the performance, six factors that characterize the structure of web service network are identified. In this paper, two major factors are investigated through comprehensive experiments. The next section describes the mathematical programming formulation that has used for web service composition during the experiments, and in the following section the experimental design and results are discussed in detail.

3. Mathematical Model

Figure 3 outlines the binary integer programming formulation for the web service composition used in this paper for experiments. The definition of domain is as follows: $W$ is the set of web services; $P$ is the set of all parameters of web services in $W$; $P_{in} \subseteq P$, is the set of parameters that are used as input in web services in $W$; $P_{out} \subseteq P$, is the set of parameters that are used as output in web services in $W$; $P_{initial} \subseteq P$, is the set of parameters that are initially given or known; $P_{goal} \subseteq P$, is the set of parameters that need to find; $W^p_{input} \subseteq W \forall p \in P$, is the set of web services that have parameter $p$ as input; $W^p_{output} \subseteq W \forall p \in P$, is the set of web services that have parameter $p$ as output; Stage (s): $1 \leq s \leq S$, where $S$ is the maximum number of stages for web service composition.

A brief explanation on the formulation is as follows: (1) ~ (4) define decision variables; (5) is the objective function, where $c_w$ is the cost associated with web service $w$; (6) and (7) are initial constraints; (8) is goal constraints; (9) ~ (12) are web service invocation constraints; (13) and (14) are parameter usage constraints; (15) is sequence constraints; and (16) ~ (17) are the domains of variables. Basically, all decision variables are binary and the mathematical model decides which web service is invoked in which stage. For example, if web service $w_1$ is called in stage 2, the value of $y_{w_1,2} = 1$. In addition, the model decides which parameters are used in which purpose, input, output or known-unused in which stage. For example, if parameter $p_1$ is used as an input, $x_{p_1,2}^{input} = 1$. A simple objective function, minimization of the number of web services, is used in this paper to focus on the performance of web service composition by setting all $c_w$ to 1. Initial constraints specify all known parameters in the formulation. Goal constraints do all parameters that need to find. Web service invocation constraints requires all the required input parameters of a web service to be known before invoking it. Parameter usage constraints prevents any contradictory usage of parameters in each stage. For example, if a parameter is used as an input or an output, then the parameter should not be considered to be known but unused. Sequence constraints make sure that once a parameter is known, then it can be used. A more detailed explanation on the formulation can be found in [13].
\[ y_{w,s} = \begin{cases} 1 & \text{if web service } w \text{ is invoked in stage } s, \\ 0 & \text{otherwise.} \end{cases} \] (1)

\[ x_{\text{known-unused}}^{\text{input}} = \begin{cases} 1 & \text{if parameter } p \text{ is known but not used in stage } s, \\ 0 & \text{otherwise.} \end{cases} \] (2)

\[ x_{\text{input}}^{\text{output}} = \begin{cases} 1 & \text{if web service } w \text{ is invoked in stage } s \text{ such that } w \in W_p^{\text{input}}, \\ 0 & \text{otherwise.} \end{cases} \] (3)

\[ x_{\text{output}}^{\text{input}} = \begin{cases} 1 & \text{if web service } w \text{ is invoked in stage } s \text{ such that } w \notin W_p^{\text{input}} \land w \in W_p^{\text{output}}, \\ 0 & \text{otherwise.} \end{cases} \] (4)

Minimize \[ \sum_{w \in W} \sum_{s \in S} c_{w,s} y_{w,s} \] (5)

Subject to:

\[ x_{\text{input}}^{\text{output}} + x_{\text{output}}^{\text{input}} + x_{\text{known-unused}}^{\text{input}} = 0 \quad \forall \, p \in P_{\text{initial}} \] (6)

\[ x_{\text{output}}^{\text{input}} + x_{\text{known-unused}}^{\text{input}} = 0 \quad \forall \, p \notin P_{\text{initial}} \] (7)

\[ x_{\text{input}}^{\text{output}} + x_{\text{known-unused}}^{\text{input}} + x_{\text{output}}^{\text{input}} \geq 1 \quad \forall \, p \in P_{\text{goal}} \] (8)

\[ \sum_{w \in W : p \in W_p^{\text{output}}} y_{w,s} \geq x_{\text{input}}^{\text{output}} \quad \forall \, p \in P, s \in 1,...,S \] (9)

\[ y_{w,s} \leq x_{\text{output}}^{\text{input}} \quad \forall \, w \in W_p^{\text{output}} \setminus W_p^{\text{input}}, \forall \, p \in P, s \in 1,...,S \] (10)

\[ \sum_{w \in W : p \in W_p^{\text{input}}} y_{w,s} \geq x_{\text{output}}^{\text{input}} \quad \forall \, p \in P, s \in 1,...,S \] (11)

\[ y_{w,s} \leq x_{\text{output}}^{\text{input}} \quad \forall \, w \in W_p^{\text{input}}, \forall \, p \in P, s \in 1,...,S \] (12)

\[ x_{\text{output}}^{\text{input}} + x_{\text{known-unused}}^{\text{input}} \leq 1 \quad \forall \, p \in P, s \in 1,...,S \] (13)

\[ y_{w,s} \leq x_{\text{output}}^{\text{input}} + x_{\text{known-unused}}^{\text{input}} \quad \forall \, p \in P, s \in 1,...,S \] (14)

\[ x_{\text{output}}^{\text{input}} + x_{\text{known-unused}}^{\text{input}} \leq x_{\text{input}}^{\text{output}} + x_{\text{output}}^{\text{input}} + x_{\text{known-unused}}^{\text{input}} \quad \forall \, p \in P, s \in 1,...,S \] (15)

\[ y_{w,s} \in [0,1] \quad \forall \, w \in W, s \in 1,...,S \] (16)

\[ y_{w,s} \in [0,1] \quad \forall \, w \in W, s \in 1,...,S \] (17)

Figure 3: Binary integer programming formulation for web service composition

4. Experiments

Experiments are conducted to explore how the structure of web service network affects the performance of web service composition. Web service network is a network with nodes and directed arcs, in which web services are nodes and parameters are arcs as shown in Figure 4, where rectangles denote web services and arrows do parameters. There are two types of arrows. Arrows pointing to rectangles are input parameters while those pointing out of rectangles are output parameters, which indicates that web service network is a directed network. A web service can have multiple input and output parameters. For example, GetHotel web service has two input parameters, Hotel Name and ZIP code, and it has two output parameters, Hotel Address and Phone Number. When a web service composition is requested, a query that consists of initial parameters and goal parameters is given. Hotel Name and ZIP code in GetHotel web service are initial parameters while Direction in GetDirection is goal parameters. In web service network, two web
services are connected to each other if an output parameter of a web service is used as an input parameter of another web service. For example, Address is an output parameter of GetHotel while it is an input parameter of GetRestaurant. Thus, GetHotel and GetRestaurant web services are connected. The composition of web services is to find a series of web services that obtain the goal parameters starting from the initial parameters by the invocation of web services.

<table>
<thead>
<tr>
<th>Factors</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factor 1</td>
<td>The number of web services</td>
</tr>
<tr>
<td>Factor 2</td>
<td>The number of distinct parameters</td>
</tr>
<tr>
<td>Factor 3</td>
<td>The number of input parameters per web service</td>
</tr>
<tr>
<td>Factor 4</td>
<td>The number of output parameters per web service</td>
</tr>
<tr>
<td>Factor 5</td>
<td>The number of initial parameters</td>
</tr>
<tr>
<td>Factor 6</td>
<td>The number of goal parameters</td>
</tr>
</tbody>
</table>

Six factors that are considered in this experiment are listed in Table 1, which characterize the structure of web service network and are expected to significantly impact the performance of web service composition. They are (1) the number of web services, (2) the number of distinct parameters, (3) the number of input parameters per web service, (4) the number of output parameters per web service, (5) the number of initial parameters, and (6) the number of goal parameters. In this paper, experiments have been conducted to evaluate the effect of the first two factors on the performance of web service composition in terms of solution time and objective function value, where the objective function value is defined as the number of composed web services. The experiments were performed on a machine with Intel Core i5 CPU with 3.20 GHz and 8 GB RAM. ILOG CPLEX Version 12 was used to solve binary integer programming formulations for web service composition.

4.1 Experimental Design

To understand the performance of web service composition in different network environments, web service composition problems are generated randomly. The ranges of values that each factor has in this experiment are summarized in Table 2. This paper focuses on the first two factors listed in Table 1. Thus, the values of the two factors are varied in the experiments while the other four factors are fixed.

<table>
<thead>
<tr>
<th>Experiments</th>
<th># of Web Services</th>
<th># of Distinct Parameters</th>
<th># of Input Parameters</th>
<th># of Output Parameters</th>
<th># of Initial Parameters</th>
<th># of Goal Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>400–6,000 (interval 400)</td>
<td>20</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Case 2</td>
<td>600</td>
<td>20–100 (interval 20)</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

(1) The number of web services in a network

To explore the effect of the number of web services in a network on solution time and optimal solution value, fifteen sets of web service networks with different number of web services, from 400 to 6,000 with an interval of 400, are generated. Each set has thirty replications. To minimize effect caused by the other factors, the levels of the other factors are fixed; twenty distinct parameters are used; each web service has two input and two output parameters; a fixed query is used which consists of two initial and four goal parameters.

(2) The number of distinct parameters in a network

To explore the effect of the number of distinct parameters in a network on solution time and optimal solution value, five sets of networks with different number of distinct parameters, from 20 to 100 with an interval of 20, are generated. Each set has fifty replications. To minimize effect caused by the other factors, the levels of the other factors are fixed; one six hundred web services are used in all of the sets; each web service has two input and two output parameters; a fixed query is used which consists of two initial and five goal parameters.
4.2 Experimental Results

(1) The number of web services in a network

A set of experiments was conducted to analyze the trend of solution time as the number of web services in web service network increases. The number of web services are varied from 400 to 6,000 with an interval of 400 while the rest of factors are fixed as specified in Table 2. Figure 5 shows an exponentially increasing trend of solution time as the number of web services increases. Figure 6 illustrates a decreasing trend of the number of composed web services as the number of web services in network increases. The exponentially increasing trend of solution time indicates that the number of web services in network significantly increases the complexity of web service composition. On the other hand, the decreasing trend of the number of composed web services implies that as more web services are available goal parameters can be found with less number of web services. In other words, there will be higher probability of having favorable web services that lead to goal parameters as more web services are available.
Another set of experiments was conducted to analyze the trend of solution time and the number of composed web services in web service composition as the number of distinct parameters in web service network increases. The number of distinct parameters are varied from 20 to 100 with an interval of 20 while the rest of factors are fixed as specified in Table 2. As more distinct parameters are in a web service network the connectivity among web services is likely to decrease. On the other hand, if there are many common parameters among web services in a web service network, web services are likely to be more connected with each other. Figure 7 shows that solution time increases as the number of distinct parameters increases. Figure 8 also shows that the number of composed web services in web service composition increases as the number of distinct parameters increases. As more distinctive parameters are in a web service network, the number of invokable web services are likely to decrease due to less connectivity. The decreased number of invokable web services makes it difficult for the solver to find web services that lead to goal parameters and in turn causes longer solution time. In addition, less connectivity causes more web services to be invoked in order for the solver to find goal parameters because there are less number of invokable web services in web service network and in turn less number of favorable web services that lead to goal parameters.

5. Conclusion

This paper analyzes the impact of six factors which characterizes web service network on the performance of web service composition. Out of six factors, the number of web services and the number of distinct parameters in web service network are investigated in detail. Extensive experiments are conducted with respect to various levels of the two factors while the other factors are fixed. The experimental results shows that as the number of web services in a web service network increases the solution time increases exponentially while the number of composed web services decreases. In addition, the experiments reveals that as the number of distinctive parameters in a web service network increases both the solution time and the number of composed web services increases. One interesting fact identified
in this experiment is that as more web services are available the composition solution includes less number of web services. In other words, with less number of invocation of web services the goal parameters can be found. Another interesting fact observed in this experiment is that as more distinct parameters exist in web service network the composition solution includes more number of web services. This result can be better explained by using network connectivity concept. Since more distinct parameters causes more sparsely connected web service network, possibility of finding favorable web services to reach goal parameters decreases. Consequently, it requires more web services to be invoked before finding goal parameters.

This paper leaves further research topics. First, the way how the other four factors affect the performance of web service composition should be investigated. In addition, the interactions among the major factors can be analyzed statistically with respect to the performance of web service composition, such as solution time or the number of composed web services.

**References**