World Wide Web
DOI 10.1007/s11280-009-0064-6

Graph Theoretic Topological Analysis of Web Service Networks

Hyunyoung Kil · Seog-Chan Oh · Ergin Elmacioglu · Wonhong Nam · Dongwon Lee

Received: 12 May 2008 / Revised: 25 May 2009 /
Accepted: 5 June 2009

Abstract Using graph theory, we analyze the topological landscape of web service networks formed by real-world data set, either downloaded from web service repositories or crawled by a search engine. We first propose a flexible framework to study syntactic web service matchmaking in a unified manner. Under the framework, then, the data set is analyzed from diverse perspectives and granularity. By and large, the data set is shown to exhibit small world network well and power-law-like distribution to some extent. Finally, using random graph theory, we demonstrate how to accurately estimate the size of the giant component of such web service networks.

Keywords web services · small world · power-law · giant component · matchmaking

A preliminary version [17] of this paper was presented at the SMR '06: VLDB Workshop on Semantic Matchmaking and Resource Retrieval.

H. Kil · W. Nam (✉) · D. Lee
The Pennsylvania State University, University Park, 16802 PA, USA
e-mail: wnam@psu.edu

H. Kil
e-mail: hykil@psu.edu

D. Lee
e-mail: dongwon@psu.edu

S.-C. Oh
General Motors Corporation, Detroit, MI, USA
e-mail: seog-chan.oh@gm.com

E. Elmacioglu
Yahoo, Inc., Sunnyvale, CA, USA
e-mail: ergin@yahoo-inc.com

Published online: 20 June 2009

1 Introduction

Web services are software systems designed to support machine-to-machine interoperation over the Internet. Many researches have been carried out for the web service standard, and these efforts have significantly improved flexible and dynamic functionality of the *service oriented architecture* in the current web services. Moreover, the recent realization—i.e., the core web service specifications of WSDL [33], SOAP [35] and UDDI [29] are *insufficient* to realize the goal of the *Semantic Web*—has developed a plethora of new means such as OWL [34], OWL-S [22], WSDL-S [1], and WSML [12] to annotate rich semantics to web services (ontology), enabling truly automatic data exchange on the Web. Therefore, it is expected that, in the near future, service vendors will publish their offerings as "semantic web services", i.e., semantically enriched web services. However, in practice, there has been little study as to how useful current public web services are. In particular, since web services that are specified in WSDL and published in UDDI form a network, one can use network analysis methods to study the characteristics of web service networks.

In general, the topological structure of a network affects the processes and behaviors occurring in the network. For instance, the topology of a social acquaintance network may affect the spread rate of gossip or disease. Similarly, the topology of the Internet is known to be correlated with the robustness of communication therein. Accordingly, understanding the structural properties of networks often help gain better insights and develop better algorithms [3, 28]. Therefore, in this paper, we study two directions: (1) *data* $\Rightarrow$ *model* (Sections 5.1 and 5.2): identifying models (i.e., properties) from web service networks, and reversely (2) *model* $\Rightarrow$ *data* (Section 5.3): generating web service networks from models. Since currently there are not enough semantic web services specified in OWL-S or WSDL-S yet, our study in this paper is limited to ones in WSDL.

To investigate properties of the current web services, we collect a number of web services from public repositories and by using a search engine (e.g., Google). Then, we construct three kinds of web service networks where a node is a web service (an operation, or a parameter resp.) while an edge is an invocation between web services. The connections of nodes, i.e., invocations between web services, may represent machine to machine interoperation which is the main goal of web services. Also, one can consider the edges as an information production flow—e.g., a cuisine and a zip code are able to produce the name and address of a restaurant by a web service `findRestaurant`, and the address can be used to obtain the map of the restaurant by a web service `findMap`. To identify edges, we focus on *parameter matchmaking* since the WSDL standard does not support pre/post conditions or world state changes. Note that since the descriptions of web services, operations, or parameters are in natural languages and optional, they cannot be used for formal matchmaking. Consequently, we use two web service matchmaking methods that allow a web service ws_1 to call another web service ws_2 if the output parameters of ws_1 matches the input parameters of ws_2 exactly (approximately resp.). Hence, our matchmaking between web services is beyond a strict syntactic matchmaking.

By analyzing the networks constructed, we show that the current web service networks have the small world and power-law-like properties. The finding that in most cases there exists a short path between pairs of nodes provides a valuable hint to web service composition techniques, which allows the techniques to focus

on short paths first. The observation that there exist a small number of hub nodes in web service networks is helpful to improve web service discovery and composition—one can complete these processes earlier by considering the hub nodes first among a number of candidates. For the other direction, we estimate the size of the giant component in each network we construct by using a theoretic formula. Then the calculated size is compared with the actual size of the giant component in the corresponding network. Through this comparison, we show that the estimated size is very closed to the actual size, which implies that a theoretical model can help to study the interoperable portion of web service networks even without measuring the actual networks.

Our main contributions are as follows:

- We propose a flexible web service matchmaking framework that allows to incorporate different "matching" in a unified manner. By using different matching functions, the framework can cover from exact to approximate (e.g., as in Cosine or WordNet) matching.
- We propose three dimensions of granularity to form web service networks from parameters to operations to web services. By looking into web service network using different glasses, we can understand the characteristics of the network better.
- We have conducted extensive experimentation under the proposed framework, and applied graph analysis techniques to study the distribution, average distance, diameter, and clustering coefficient of the network. Furthermore, we examine if the graph shows small world properties or power-law-like distribution.
- Reversely, given a model, being able to generate web service networks is important in many settings. Using random graph theory, therefore, we demonstrate how to accurately estimate the size of the giant component in our web service networks.

2 Background

2.1 Web services

A *web service* in a WSDL file can be viewed as a collection of *operations*, each of which in turn consists of input and output *parameters*. When an operation *op* has input parameters $IN = \{p_1, \cdots, p_n\}$ and output parameters $OUT = \{q_1, \cdots, q_m\}$, we denote the operation by $op(IN, OUT)$. Furthermore, each parameter is a pair of $(name, type)$. We denote the name and type of a parameter p by $p.name$ and $p.type$, respectively. To decide invocations between web service operations, we first should check the type of the operations. The WSDL specification lists four types of operations [33]: (1) *one-way*: the operation receives a message but will not return a response; (2) *request-response*: the operation receives a request and will return a response; (3) *solicit-response*: the operation sends a request and will wait for a response; and (4) *notification*: the operation sends a message but will not wait for a response. If an operation op_1 wants to invoke another operation op_2, op_1 should have a compatible type with op_2. Since this checking process can be done trivially,

from here forward, we assume that types of operations are all compatible, and focus on other issues instead.

Example 1 Consider the web service w:

```
<message name="findRestaurant_Request">
    <part name="zip" type="int"/>
    <part name="foodPref" type="string"/>
</message>
<message name="findRestaurant_Response">
    <part name="name" type="string"/>
    <part name="address" type="string"/>
</message>
<portType name="findRestaurantPortType">
    <operation name="findRestaurant">
        <input message="findRestaurant_Request"/>
        <output message="findRestaurant_Response"/>
    </operation>
</portType>
```

The web service, w, consists of one request-response type operation, `findRestaurant`, that takes two input parameters, $p_1 = (\texttt{zip}, \text{integer})$ and $p_2 = (\texttt{foodPref}, \text{string})$, and returns two output parameters, $q_1 = (\texttt{name}, \text{string})$ and $q_2 = (\texttt{address}, \text{string})$. Therefore, a client program wishing to get the name and address of a restaurant may invoke `findRestaurant` (16801, "Thai") to w.

2.2 Small world and power-laws

In general, a network is called the *small world network* if it shows the properties of both random and regular networks.

Definition 1 (Random Network) A random network consists of N nodes, where each pair of nodes is connected with the probability p. As a result, edges are randomly placed among a fixed set of nodes [37].

Definition 2 (Regular Network) A regular network consists of N nodes, where a node i is adjacent to nodes $[(i + j) \bmod N]$ and $[(i - j) \bmod N]$ for $1 \leq j \leq K$. If $K = N - 1$, it becomes the complete N-node graph, where every node is adjacent to every other $N - 1$ nodes [37].

Random networks are characterized by their short average distances among reachable nodes. On the other hand, in regular networks, each node has highly clustered neighbor nodes, such that the connectivity between neighboring nodes is very high. Consequently, small world networks show both (1) highly clustered structure and (2) small average shortest distance. To measure the connectivity and short average distances, the following metrics are often computed.

- L: The **average shortest distance** (i.e., number of hops) between reachable pairs of vertices. $L(p)$ is defined as L of Watts-Strogatz graph [38] with probability p. L_{random} is identical to $L(1)$.
- C: The **average clustering coefficient**. For a node i with v_i neighbors, $C_i = \frac{2E_i}{v_i(v_i-1)}$, where E_i is the number of edges between v_i neighbors of i. C is the average clustering coefficient C_i for a network. Again, C_p is defined as C of the Watts-Strogatz graph with the probability p. C_{random} is identical to $C(1)$.
- $Index_{SN}$: The **small world network index** is defined as: $Index_{SN} = \frac{|C_{actual} - C_{random}|}{|L_{actual} - L_{random}|}$, where C_{actual} and L_{actual} represent C and L of the measured network, respectively, and C_{random} and L_{random} represent C and L of the random graph with the same number of nodes and the average number of edges per node as the measured network.

If a network has the small world properties, then its L and C shows: $C_{actual} \gg C_{random}$ and $L_{actual} \gtrsim L_{random}$. That is, the average clustering coefficient is much larger than that of a random network, while the average shortest distance is slightly larger than or similar to that of a random network. Therefore, the more distinct the small world properties of a network are, the bigger $Index_{SN}$ of the network becomes.

A power-law distribution often occurs in complex systems where a majority of nodes have very few connections, while a few nodes have a high degree of connections. Typical power-law function has the form of $y = Cx^{-\alpha}$, and is captured as a straight line in log-log plots. The existence of power-law distribution has been observed in many real and artificial networks such as power grid, WWW, or collaboration network, and believed to be one of signs of mature and robust networks.

2.3 Giant component

Many properties of the random network model are shown to be exactly solvable in the limit of large network size [25, 26, 28]. In particular, in this paper, we are interested in the size of giant component of random graphs. Informally,

Definition 3 (Giant Component) The giant component is a connected component of a graph larger than a size threshold $\theta(n)$.

Typically, the giant component consists of the majority of nodes in the graph. Here, we use the theoretical framework derived in [28] to estimate the giant component size in web service networks by using the generating functions [39]. The idea is, instead of dealing with the degree distribution directly, a generating function $G_0(x)$ that encapsulates all the information in the degree distribution p_k is used as follows: $G_0(x) = \sum_{k=0}^{\infty} p_k x^k$, where k represents the degree and p_k is the desired degree distribution of the network. This form is easier to work on rather than working with the complexities of the distribution function. For instance, we can represent the average degree as $z = \sum_k k p_k = G_0'(1)$ for any degree distribution by using this generating function.

One important point towards the analysis of component sizes, we need to be able to represent the degree distribution of the node we reach by following a randomly chosen edge. It is not the same as p_k since there are k edges that arrive at a node

with degree k. Hence, it is proportional to kp_k, and can be generated by the following normalized function:

$$\frac{\sum_k kp_k x^k}{\sum_k kp_k} = x\frac{G_0'(x)}{G_0'(x)} \tag{1}$$

If we start from a randomly chosen node and follow its edges to reach its neighbors, we need to subtract the edge connecting the node to the neighbor in order to find the degree distribution of remaining outgoing edges of the neighbors of that node. Thus we can define the generating function for the degree distribution of a randomly chosen node's neighbors $G_1(x)$ as:

$$G_1(x) = \frac{G_0'(x)}{G_0'(1)} = \frac{1}{z}\,G_0'(x) \tag{2}$$

The average size of components to which a randomly chosen node belongs when there is no giant component formed in a network is derived in [28] as:

$$\langle s\rangle = 1 + \frac{G_0'(1)}{1 - G_1'(1)} = 1 + \frac{z_1^2}{z_1 - z_2} \tag{3}$$

where $z_1 = z$ is the average number of neighbors of a node and z_2 is the average number of second neighbors. The expression diverges when $G_1'(1) = 1$ which is the point to mark a phase transition at which a giant component first appears. By rewriting the conditions, we can say that a giant component exists in a network if

$$\sum_k k(k-2)p_k \geq 0 \tag{4}$$

The size of a giant component, if there exists, can be calculated from the following simple heuristic argument. Let u be the probability that a node chosen uniformly at random from the network is not in the giant component. In order words, this value is the fraction of all nodes outside the giant component. Then this probability is equal to the probability that none of the node's neighbors belong to the giant component which is just u^k if the node has degree k. If we average this over the probability distribution [28], we obtain the self consisting function, $u = G_1(u)$. Then for the smallest non-negative real solution of u, the following equation gives the size of the giant component, S:

$$S = 1 - G_0(u) \tag{5}$$

2.4 Related work

Many empirical networks are well modeled by complex networks including the scale free and small world networks. The small world networks are generated by Watts-Strogatz model [38]. Albert et al. [4] proposed a set of different models for generating scale free networks, based on the growing process of the Internet and other empirical complex networks. Denning [13] surveyed various network laws with a focus on the power-law distribution and the scale free networks. In this paper, we apply the developed techniques to examine web services networks.

Many web service matchmaking solutions are based on the keyword matching supported by the category browsing of UDDI. However, the keyword based matching cannot fully capture real functions of web services. To address this limitation, many researchers have developed various methods to assess the similarity of web services for matchmaking. Wu [40] suggested a matchmaking process based on a lightweight semantic comparison of signature specifications in WSDL by means of several assessment methods. Wang and Stroulia [36] assessed the similarity of the requirement description of the desired service with the available services via the semantic information retrieval method and a structure matching approach. Maedche and Staab [20] provided multiple phase cross evaluation to assess the similarity between two different ontology. Blake and Nowlan [9] manually analyzed real, fully-operational web services currently available on the Internet and, discovered insights into how real web service messages are defined. Finally, they used the insights for web service discovery. In addition, Cibran et al. [11] proposed a new layer, the *Web Services Management Layer (WSML)* between the application and the web services. WSML decouples web services from client applications and enables swapping between semantically equivalent web services based on availability. In this paper, we do not propose a particular matchmaking method. Instead, we advocate a flexible framework that can accommodate a plethora of matchmaking schemes in a unified manner. Due to the availability, we choose Cosine [8] and WordNet based matching schemes in this paper [16] (to be elaborated in Section 4). However, note that it is also possible to incorporate the aforementioned matching approaches such as [36, 40] in our framework, if the implementation is available.

Recently, a number of researches have been carried out for matchmaking with semantic web service descriptions. Paolucci et al. [30] proposed a semantic matchmaking algorithm by matching of inputs and outputs of the service profiles. Bellur and Kulkarni [6] proposed a more exhaustive matchmaking algorithm, based on the concept of matching bipartite graphs, to overcome the problems faced with Paolucci et al's work [30]. Akkiraju et al. [2] presented an algorithm to compose web services in the presence of semantic ambiguity by combining semantic matching and AI planning algorithms. They used cues from domain-independent and domain-specific ontologies to compute an overall semantic similarity score between ambiguous terms. Bianchini et al. [7] proposed an ontology-based hybrid approach in which several matchmaking techniques were combined to provide an adaptive service discovery environment. Lecue et al. [18] presented a web service composition technique by adding annotation and exploiting matchmaking between input/output parameters of web services. Their approach extends existing methods (Exact, Plug-in, Subsume, Intersection and Fail) with concept abduction to provide explanations of misconnections between web services. Dong et al. [14] suggested a web service search engine, Woogle, which has the web service similarity search capability. Woogle first clusters parameter names into semantically meaningful concepts, which are then used to compute the similarity between parameter or operation names.

Li and Horrocks [19] used DAML-S to design a service matchmaker. DAML-S is used to represent knowledge for promoting service capability matching. A description logic (DL) based on DAML-S is used to implement matchmaking details. Medjahed et al. [24] proposed an ontology-based framework for the automatic composition of web services. From the AI planning perspective to the automatic web service composition, Sirin et al. [31] demonstrated how an OWL reasoner can be

integrated with an AI planner to overcome automatic web service composition problems. They identify the challenges of writing service descriptions and reasoning when an expressive language such as OWL is used. Martin et al. [23] described OWL-S for more complete specifications of the capabilities and behavior of Web Services, so as to support automation of service-related activities, discovery and composition. In addition, they described selected tools, technologies and research directions based upon OWL-S. As an independent research branch, Zhang and Zhang [41] considered the problem of web service quality. It is unlikely for an organization to dynamically select a partner merely based upon the information in UDDI. Therefore, the test of quality and trustworthiness of web services is critical for the success of web service paradigm. Recently, Cai [10] proposed a mechanism to help the communities to grow into scale-free networks, and extended WSDL into Scale-Free Web Services using Web Services Resource Framework.

3 The matchmaking framework

3.1 Flexible matching

A network consists of nodes and edges between them. In our framework, different entities can be used as nodes and edges in a unified manner. First, as nodes, we consider three kinds—*parameters*, *operations*, and *web services*—from finer to coarser granularity. Second, as edges, we use the notion of *parameter matching* and *operation invocation*. When the "meanings" of two parameters, p_1 and p_2, are interchangeable, in general, they are said to be "matching" each other. The simplest way to check this is if two parameters have the *same* name and type: $(p_1.name = p_2.name) \wedge (p_1.type = p_2.type)$. Since web services are designed and created in isolation, however, this naive matching is often too rigid and thus misses cases like p_1 = ("password", string) and p_2 = ("passwd", string). On the other hand, if web services are annotated with rich semantics (e.g., using RDF [32] or WSDL-S [1]), then the so-called "semantic" matching can be easily reasoned out. However, in practice, the majority of public web services does not have annotated semantics yet. In order to cover all the spectrum of matching, therefore, we propose a generic Boolean function, `match`(p_1, p_2), that determines if two parameters p_1 and p_2 are matching. Formally,

Definition 4 (Type-match) A boolean function, `type-match`$(p_1.type, p_2.type)$, returns True if: (1) $p_1.type = p_2.type$, or (2) $p_1.type$ is derived from $p_2.type$ in a type hierarchy.

Definition 5 (Name-match) A boolean function, `name-match`$(p_1.name, p_2.name, S, \theta)$, returns True if the similarity between $p_1.name$ and $p_2.name$ by a similarity metric S is over the given threshold θ: i.e., $S(p_1.name, p_2.name) \leq \theta$.

For instance, `name-match`("password", "passwd", =, 1) represents the exact matching, and would return False since "password" $\neq$ "passwd".[1] Similarly, by

[1]For the equality checking (i.e. =), the threshold value other than 1 is not meaningful.

using string matching functions such as cosine metric or WordNet similarity, one can express the approximate name matching. For instance, `name-match`("flight", "plane", WordNet, 0.9) would return True since two names of parameters have the similarity of 1 (> 0.9) as WordNet. In our experimentation, we use three metric functions: = (i.e., literal equality), cosine similarity with TF/IDF weights [5], and WordNet [16] based approximate similarity. Based on two boolean functions above, we define the generic parameter match function as follows:

Definition 6 (Match) A boolean function, $\texttt{match}(p_1, p_2, S, \theta)$, returns True if: (1) $\texttt{type-match}(p_1.type, p_2.type) = \text{True}$, and (2) $\texttt{name-match}(p_1.name, p_2.name, S, \theta) = \text{True}$.

Definition 7 (Parameter Matching) When a boolean function, $\texttt{match}(p_1, p_2, S, \theta)$, returns True, it is said that a parameter p_1 matches a parameter p_2: "$p_1 \sim p_2$".

In order to invoke an operation op_1 with input parameters $IN = \{p_1, \cdots, p_n\}$, one may need to provide values for input parameters. When one can provide all input parameters, op_1 is said "fully invocable". When one can provide at least one input parameter, op_1 is said "partially invocable". For instance, Google API has a search operation with several input parameters, but it can be invoked by *null* values for many of parameters. Similarly, consider a client program wishing to invoke an operation op_1 first and then have op_1 invoke another operation op_2 directly (i.e., a case of web service composition). In this case, if the output parameters of op_1 satisfy all input parameters of op_2, then op_1 "fully" invokes op_2, and if output parameters of op_1 satisfy some input parameters of op_2, then op_1 "partially" invokes op_2. Formally,

Definition 8 (Operation Invocation) For two operations, $op_1(IN_1, OUT_1)$ and $op_2(IN_2, OUT_2)$,

- op_1 fully invokes op_2, denoted by "$op_1 \mapsto op_2$", if for every input parameter $p \in IN_2$, there exists an output parameter $q \in OUT_1$ such that $q \sim p$.
- op_1 partially invokes op_2, denoted by "$op_1 \dashrightarrow op_2$", if there exists an input parameter $p \in IN_2$ and an output parameter $q \in OUT_1$ such that $q \sim p$.

We denote each invocation by **full-** and **partial-invocation**, respectively.

3.2 Web service network model

In this section, using the notions of nodes and edges defined in Section 3.1, we propose a flexible web service network model as follows.

Definition 9 (Web Service Network Model) A web service network is generated by a 4-tuple model $\mathcal{M} = (T, S, \theta, I)$, where:

- T is the type of nodes and can be either "`p`" for parameters, "`op`" for operations, or "`ws`" for web services.
- S (and θ) is the similarity metric to be used in parameter matching (and its threshold resp.).

– I is the type of operation invocation and can be either "FI" for full invocation (denoted as $\mathcal{M}^f$) or "PI" for partial invocation (denoted as $\mathcal{M}^p$).

Example 2 A model $\mathcal{M}_1 = (\texttt{op}, \text{cosine}, 0.75, \texttt{FI})$ generates an operation node network where parameter matching is done using cosine metric with a threshold 0.75. Furthermore, an edge from an operation op_1 to op_2 is added only when $op_1 \mapsto op_2$. On the other hand, another model $\mathcal{M}_2 = (\texttt{ws}, \text{word-net}, 0.9, \texttt{PI})$ generates a web service node network where intra-parameter matching is done using WordNet based approximate similarity metric with a threshold 0.9. Furthermore, an edge from a web service ws_1 to ws_2 is added if $op_1 \dashrightarrow op_2$ for $op_1(\in ws_1)$ and $op_2(\in ws_1)$—that is, partial invocation among operations.

Consider Figure 1 as an example. Here, a web service network is formed by a set of web services, each of which consists of a set of operations. An operation is invoked with a set of input parameters and produces a set of output parameters. There are three kinds of nodes representing web services (e.g., ws_1 and ws_2), operations (e.g., op_{11}, op_{12} and op_{21}) or parameters (e.g., $p_1, \cdots, p_7$). Each web service node containing a set of operation nodes is connected to parameter nodes

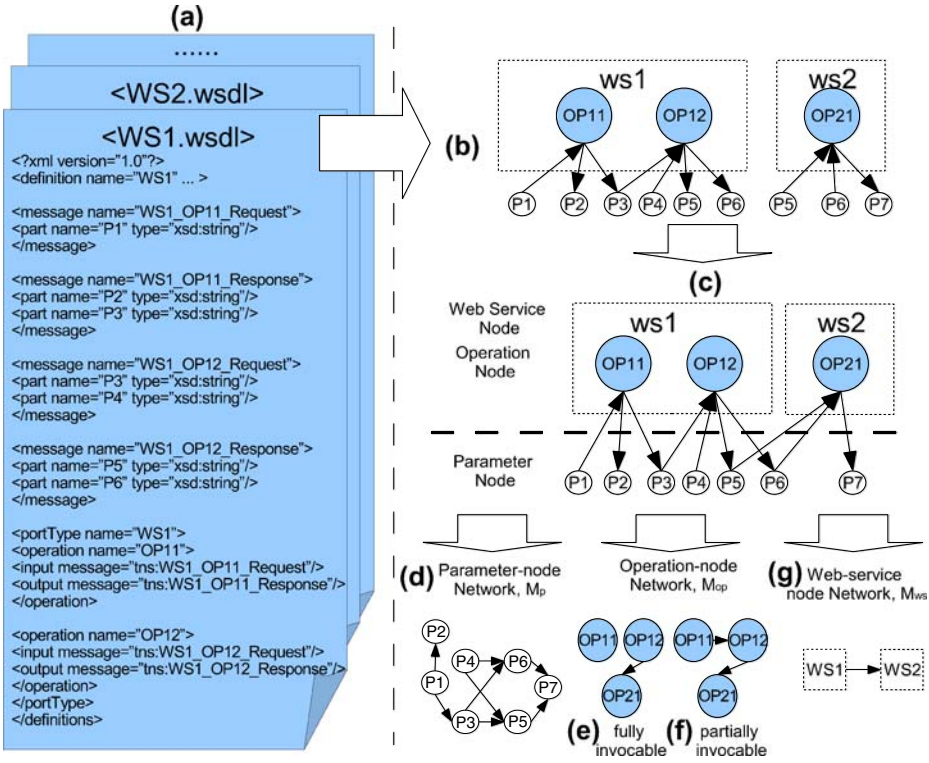


Figure 1 Web service networks: **a** WSDLs, **b** conceptual networks, **c** networks from diverse models, **d** $\mathcal{M}_p$, **e** $\mathcal{M}_{op}^f$, **f** $\mathcal{M}_{op}^p$, and **g** $\mathcal{M}_{ws}$.

with "directed" edges. These edges show the flow of the parameters as inputs or outputs of operations. An edge from a parameter node p to an operation node op indicates that the parameter p is used as one of inputs for the operation op. On the other hand, an edge from an operation node op to a parameter node p represents that the operation op produces the parameter p as an output. For example, p_1 is one of inputs of op_{11} in ws_1. Similarly, p_5 is not only an output of op_{12} but also an input of op_{21} in ws_2.

In order to study subtle differences among node types, one can "project out" the aforementioned web service network into three kinds as follows:

- A *parameter node network*, i.e., $\mathcal{M}_p = (\mathtt{p}, S, \theta, I)$, consists of parameter nodes and edges representing operations that have an input parameter as the source node and an output parameter as the target node. For instance, if $op_1(IN_1, OUT_1)$ exists, then we create edges from each parameter $p \in IN_1$ to every output parameter $q \in OUT_1$.
- An *operation node network*, i.e., $\mathcal{M}_{op} = (\mathtt{op}, S, \theta, I)$, consists of nodes representing operations and edges representing invocations in-between. If an operation op_1 can (either partially or fully based on I) invoke an operation op_2, there is an edge from op_1 to op_2 in $\mathcal{M}_{op}$.
- A *web service node network*, i.e., $\mathcal{M}_{ws} = (\mathtt{ws}, S, \theta, I)$, consists of web service nodes and edges representing the existence of invocable operations between web services.

Figure 1d describes a parameter node network $\mathcal{M}_p$ based on Figure 1b. For instance, p_1 can be transformed to p_2 by op_{11}. For operation node example, op_{21} can be invoked with two input parameters both of which are produced by op_{12} in Figure 1c. Therefore, there exists an edge op_{12} to op_{21} in both operation node networks (Figure 1e and Figure 1f). On the other hand, p_3, one of outputs for op_{11}, is used as one of inputs for op_{12}. However, op_{12} has another input parameter p_4 not produced by op_{11}. In the partial invocation mode, therefore, an edge from op_{11} to op_{12} is added. On the contrary, in the full invocation mode, no edge is added. Note that in the example of Figure 1, we obtain the same web service node network whether we use partial or full invocation mode. However, in general, different networks will be formed depending on the choice of invocation mode.

Table 1 shows the summary of symbols.

Table 1 Summary of symbols.

Symbol	Meaning
$p_1, q_1, ...$	Parameters
$op_1, op_2, ...$	Operations
$ws_1, ws_2, ...$	Web services
$\mathcal{M}_p$	Parameter node network
$\mathcal{M}_{op}^p$	Partially invocable operation node network
$\mathcal{M}_{op}^f$	Fully invocable operation node network
$\mathcal{M}_{ws}^p$	Partially invocable web service node network
$\mathcal{M}_{ws}^f$	Fully invocable web service node network

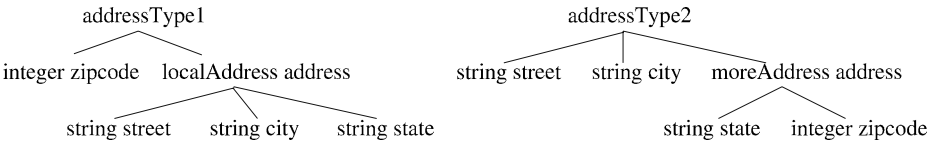


Figure 2 Compatible types with different structures.

4 Set-up

4.1 Data set

From web service repository and by using Google, we have downloaded a total of 2,100 publicly available web services (as WSDL files). We refer to those data sets as **PUB**. The pre-processing of **PUB** data set consisted of four steps as follows:

(1) *Data Gathering*: First, 1,554 files were taken from Fan et al. [15] who downloaded the files from public repositories such as `XMethods.org` or `BindingPoint.com`. Second, out of top-1,000 ranked WSDL files from Google for the query string "wsdl filetype:wsdl", 546 were downloaded using Google API (the rest were un-downloadable). Note that although we could have downloaded more than the first 1,000 WSDL files from Google, we decided to disregard the rest since their qualities degraded rather quickly (e.g., many WSDL files after top-1,000 were for testing as in "Hello World").

(2) *Validation & De-duplication*: According to WSDL standards, 740 invalid WSDL files were removed, and 1,360 files are left out. Then, 376 duplicate WSDL files at operation level were removed, yielding 984 valid WSDL files at the end.

(3) *Type Flattening*: For matching parameters, we use both name and type of parameters. However, since WSDL files are designed by different people in isolation, using only atomic types of XML Schema can be too rigid. For instance, consider two parameters: p_1 (address, addressType1) and p_2 (MyAddress, addressType2) of Figure 2. Although names of two parameters are similar, their types are user-defined and different. However, the types are in fact "compatible". Therefore, if we *flatten* these types into $p_1.type = \{$integer zipcode, string street, string city, string state$\}$ and $p_2.type = \{$string street, string city, string state, integer zipcode$\}$, then the parameters can be matched. We call this process type flattening.[2] After type flattening, each atomic type and name are compared using type hierarchy of XML Schema and flexible matching scheme (e.g., exact or approximate), respectively. The detailed statistics of type flattening is in Table 2. As the table, after type flattening we still have 7.41% of input parameters and 9.16% of output parameters as complex types, since the complex type information is missing in the WSDL files we downloaded and thus we are not able to flatten them.

(4) *Data Cleaning*: The final step is to clean data to improve the quality of parameters. For instance, a substantial number of output parameters (16%)

[2] An alternative to our type flattening is to use more expensive metric such as tree edit distance. For its simplicity, we used type flattening in this paper.

Table 2 Type distribution.

Type	Before flattening		After flattening	
	Input (num)	Output (num)	Input (num)	Output (num)
anyType	0.07% (7)	0.09% (5)	0.2% (35)	0.2% (51)
simpleType	65.52% (6,751)	32.37% (1,792)	92.43% (19,809)	90.63% (22,946)
string	53.75% (5,538)	22.00% (1,218)	65.06% (13,943)	54.74% (13,859)
number	7.79% (803)	7.12% (394)	17.69% (3,791)	22.89% (5,796)
time	0.90% (93)	0.22% (12)	1.85% (396)	2.93% (743)
boolean	3.74% (385)	2.46% (136)	7.16% (1,535)	9.14% (2,314)
complexType	34.41% (3,545)	67.54% (3,739)	7.41% (1,588)	9.16% (2,320)
Total number	10,303	5,536	21,432	25,317

were named "return", "result", or "response" which are too ambiguous for clients. However, often, their more precise underline meaning can be derived from contexts. For instance, if the output parameter named "result" belongs to the operation named "getAddress", then the "result" is in fact "Address". In addition, often, naming follows apparent pattern such as `getFooFromBar` or `searchFooByBar`. Therefore, to replace names of parameters or operations by more meaningful ones, we removed spam tokens like "get" or "by" as much as we could.

4.2 Similarity functions

We use three similarity functions for the name matching, `name-match`:

- *Exact Matching*: $p_1.name$ and $p_2.name$ match if ($p_1.name = p_2.name$).
- *Cosine Similarity*: The similarity of $p_1.name$ and $p_2.name$ is measured as the cosine value of the angle between two tf/idf vectors, v_1 and v_2, made from the tokens of $p_1.name$ and $p_2.name$: $S_{cos} = \cos(\theta) = \frac{v_1 \cdot v_2}{||v_1|| \cdot ||v_2||}$.
- *WordNet-based Similarity*: Since WordNet is a network of English words labeled with semantic classes (e.g., synonyms), it carries various semantic relations among words. Many researchers have proposed various ways to measure the approximate similarity of two words in WordNet. Among them, we employ Lin similarity function [16] that scales the information content of the least common subsumer by the sum of the information content of two vocabularies. Formally,

$$Similarity_{Lin}(c_1, c_2) = \frac{2 \log p_M(c_1, c_2)}{\log p(c_1) + \log p(c_2)}$$

where $p(c)$ is the probability of encountering an instance of concept c for any concept $c \in C$, and $p_M(c_1, c_2) = \text{MIN}_{c \in Sub(c_1, c_2)} p(c)$ where $Sub(c_1, c_2)$ is the set of concepts that subsume c_1 and c_2. Finally, our similarity function with WordNet is $S_{WordNet}(p_1.name, p_2.name) = Similarity_{Lin}(p_1.name, p_2.name)$.

Both cosine and WordNet metrics return a similarity value from 0 to 1, and use thresholds to determine matchmaking (i.e., if the calculated similarity is over the threshold, we consider that two parameter names match). The following example illustrates the Cosine similarity.

Example 3 Consider two parameters p_1 = "endRoamDate" and p_2 = "EndDate". Let us assume that two parameters are the same type. First, parameter names are tokenized to sets of lowercase tokens, i.e., "endRoamDate" $\rightarrow$ {"end", "roam", "date"} and "EndDate" $\rightarrow$ {"end", "date"}. Then, for each token, a tf/idf weight, w, is calculated based on the entire **PUB** data set as the corpora, e.g., $w(\text{date}) = 0.57$, $w(\text{end}) = 0.57$, and $w(\text{roam}) = 0.7$. With tf/idf weights, p_1 and p_2 are transformed to 3-dimensional vectors v_1 and v_2, respectively: $v_1 = [0.57, 0.57, 0.7]$ and $v_2 = [0.57, 0.57, 0]$. Finally, the similarity between "endRoamDate" and "EndDate" is $S_{cos}(p_1, p_2) = 0.816$.

At the end, we have generated a total of 25 (= 5 network types $\times$ 5 similarity metrics) web service networks: (1) three kinds of networks with "full invocation"—$\mathcal{M}_p$, $\mathcal{M}_{op}^f$, and $\mathcal{M}_{ws}^f$, and two of their counterparts of "partial invocation"—$\mathcal{M}_{op}^p$, and $\mathcal{M}_{ws}^p$; (2) five similarity variations—Exact, Cosine (0.75), Cosine (0.95), WordNet (0.75), and WordNet (0.95).

Table 3 Statistics of **PUB**—A: the number of nodes in each network, B: the number of nodes in a giant component, C: the percentage of the giant component, D: the average degree, and l: the network diameter.

Scheme	Model	A	B	C (%)	D	ℓ
Exact matching	$\mathcal{M}_p$	11,301	8,494	75.1	18.66	21
	$\mathcal{M}_{op}^p$	5,180	2,993	57.7	108.01	10
	$\mathcal{M}_{op}^f$	5,180	1,538	29.6	28.25	11
	$\mathcal{M}_{ws}^p$	984	608	61.7	52.41	6
	$\mathcal{M}_{ws}^f$	984	431	43.8	17.25	7
Cosine (0.95)	$\mathcal{M}_p$	10,952	8,385	76.5	18.13	21
	$\mathcal{M}_{op}^p$	5,180	3,232	62.3	110.43	10
	$\mathcal{M}_{op}^f$	5,180	1,667	32.1	28.89	11
	$\mathcal{M}_{ws}^p$	984	656	66.6	54.42	7
	$\mathcal{M}_{ws}^f$	984	474	48.1	17.73	9
Cosine (0.75)	$\mathcal{M}_p$	10,568	8,197	77.5	17.50	9
	$\mathcal{M}_{op}^p$	5,180	3,375	65.1	110.43	9
	$\mathcal{M}_{op}^f$	5,180	1,696	32.7	28.92	11
	$\mathcal{M}_{ws}^p$	984	676	68.6	54.04	7
	$\mathcal{M}_{ws}^f$	984	485	49.2	17.69	9
WordNet (0.95)	$\mathcal{M}_p$	8,870	7,077	79.7	18.82	18
	$\mathcal{M}_{op}^p$	5,180	3,742	72.2	214.51	7
	$\mathcal{M}_{op}^f$	5,180	2,045	39.4	39.31	9
	$\mathcal{M}_{ws}^p$	984	765	77.7	86.23	6
	$\mathcal{M}_{ws}^f$	984	558	56.7	28.77	6
WordNet (0.75)	$\mathcal{M}_p$	7,042	5,590	79.3	16.87	10
	$\mathcal{M}_{op}^p$	5,180	3,967	76.5	498.01	7
	$\mathcal{M}_{op}^f$	5,180	2,574	49.6	110.58	8
	$\mathcal{M}_{ws}^p$	984	807	82.0	178.93	5
	$\mathcal{M}_{ws}^f$	984	667	67.7	66.25	5

5 Main results

In this section, we present the main results of our study. We first present topological landscape of web service networks with respect to the small world property and power-law distribution (i.e., *data* $\Rightarrow$ *model*), then reversely demonstrate how to estimate the size of the giant component of networks (i.e., *model* $\Rightarrow$ *data*).

5.1 Small world

First, we analyze the **PUB** data set to see whether it exhibits characteristics of the small world network. Similar to many studies on the small world network [38], we restrict our attention to the giant connected component. The details of the giant component of each web service network are summarized in Table 3: the number of nodes in each network, the number of nodes in a giant component, the percentage of the giant component, the average degree, and the network diameter. Note that the percentage of the giant component is, in most cases, biggest in the WordNet (0.75) among other matching schemes. It indicates that the usage of the approximate matching scheme reduces the number of isolated web services in web service networks.

Table 4 Small world properties of **PUB**.

Scheme	Model	L_{actual}	L_{random}	C_{actual}	C_{random}
Exact matching	$\mathcal{M}_p$	4.3185	3.4244	0.2229	0.0021
	$\mathcal{M}_{op}^{p}$	2.8590	1.9830	0.3056	0.0362
	$\mathcal{M}_{op}^{f}$	3.7605	2.5628	0.2147	0.0180
	$\mathcal{M}_{ws}^{p}$	2.2710	1.9222	0.4809	0.0874
	$\mathcal{M}_{ws}^{f}$	2.9659	2.4250	0.2610	0.0405
Cosine (0.95)	$\mathcal{M}_p$	4.1760	3.4442	0.2324	0.0022
	$\mathcal{M}_{op}^{p}$	2.8651	1.9881	0.3125	0.0340
	$\mathcal{M}_{op}^{f}$	3.7538	2.5787	0.2001	0.0173
	$\mathcal{M}_{ws}^{p}$	2.2847	1.9254	0.4925	0.0833
	$\mathcal{M}_{ws}^{f}$	3.0046	2.4803	0.2499	0.0359
Cosine (0.75)	$\mathcal{M}_p$	4.1981	3.4730	0.2397	0.0020
	$\mathcal{M}_{op}^{p}$	2.8671	1.9925	0.3190	0.0326
	$\mathcal{M}_{op}^{f}$	3.7392	2.5910	0.1990	0.0172
	$\mathcal{M}_{ws}^{p}$	2.2923	1.9307	0.4822	0.0801
	$\mathcal{M}_{ws}^{f}$	2.9990	2.4394	0.2392	0.0375
WordNet (0.95)	$\mathcal{M}_p$	3.6088	3.3282	0.2612	0.0027
	$\mathcal{M}_{op}^{p}$	2.4234	1.9425	0.3251	0.0574
	$\mathcal{M}_{op}^{f}$	3.4165	2.4440	0.1493	0.0190
	$\mathcal{M}_{ws}^{p}$	2.1222	1.8865	0.5290	0.1138
	$\mathcal{M}_{ws}^{f}$	2.6215	2.1839	0.2527	0.0510
WordNet (0.75)	$\mathcal{M}_p$	3.2656	3.3665	0.3118	0.0030
	$\mathcal{M}_{op}^{p}$	2.0546	1.8743	0.4818	0.1256
	$\mathcal{M}_{op}^{f}$	2.6506	1.9657	0.1842	0.0429
	$\mathcal{M}_{ws}^{p}$	1.8484	1.7790	0.6697	0.2214
	$\mathcal{M}_{ws}^{f}$	2.2226	1.9023	0.3487	0.0993

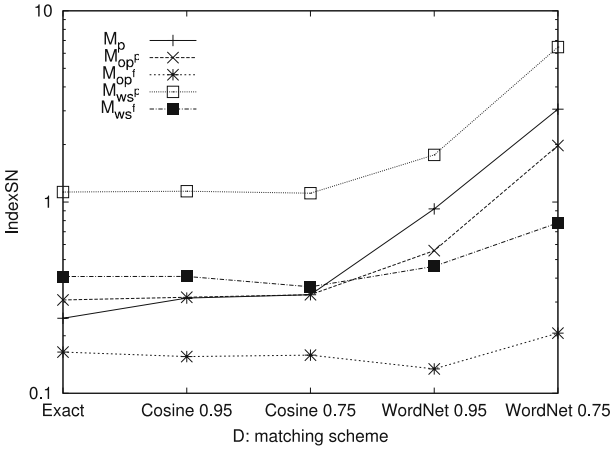


Figure 3 The small world index, $Index_{SN}$.

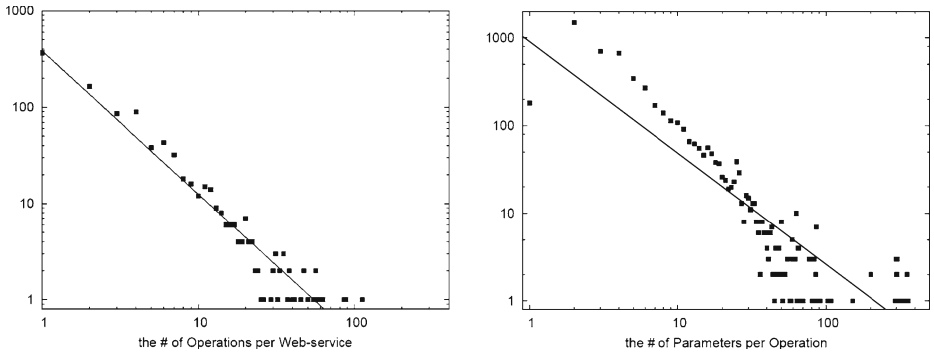
Table 4 shows the average shortest path, L, and clustering coefficient, C, for 25 web service networks, compared to random graphs with the same number of nodes and same average number of edges per node. It shows that all web service networks generated from the **PUB** data set are small world networks—$L_{actual} \gtrsim L_{random}$ and $C_{actual} \gg C_{random}$. This suggests that the real-world web services have short-cuts that connect nodes of networks.

Each matching scheme has a different level of small world properties. For example, in the parameter node network $\mathcal{M}_p$, $L_{actual}(= 3.2656)$ of the WordNet (0.75) is smaller than $L_{actual}(= 4.3185)$ of the exact matching, while $C_{actual}(= 0.3118)$ of the WordNet (0.75) is bigger than $C_{actual}(= 0.2229)$ of the exact matching. For better understanding, Figure 3 illustrates the value of $Index_{SN}$ for different matching schemes. Intuitively, $Index_{SN}$ represents how strong small world property a network has. In the figure, we can see that the usage of approximate matching scheme, WordNet (0.75), shows more clearly small world properties than the exact and cosine similarity matching schemes. In the case of $\mathcal{M}_p$ with the WordNet (0.75) matching, $Index_{SN}$ is ten times bigger than one of the exact matching case. This result implies that the approximate matching scheme makes the network denser with the increased number of edges. For the web service composition perspective, this result suggests that approximate matching scheme can facilitate more productive and effective service compositions than when only exact syntactic matching is used. Moreover, the finding that in most cases there exists a short path between pairs of nodes provides a valuable hint to web service composition techniques, which allows the techniques to focus on short paths first.

5.2 Power-laws

First, we examine how *complex* web services are by measuring how many operations (parameters resp.) are involved in each web service (operation resp.) [15]. These results are shown in Figure 4, fitted to a power-law function $y = Cx^{-\alpha}$ (in log-log plot). They show *near*[3] power-law distributions with the exponent of 1.49 and

[3] A recent study [27] reported that most of real-world power-law distributions exhibited an exponent of $2 \leq \alpha \leq 3$.

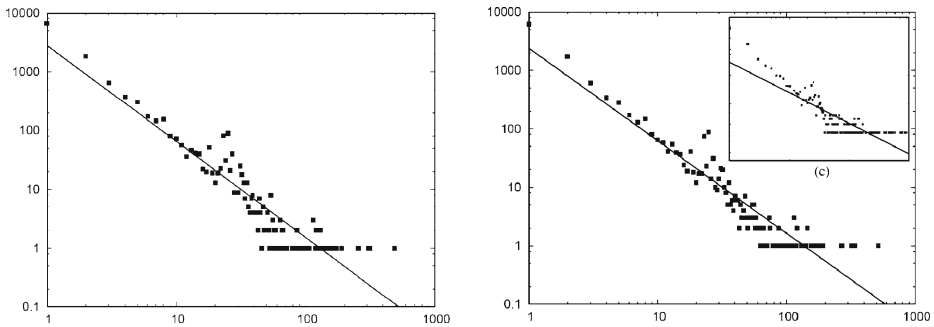


(a) $\alpha = 1.49$ (b) $\alpha = 1.27$

Figure 4 The complexity of web services and operations. Y-axis is the number of samples of web services and operations, respectively (**a**, **b**).

1.27, respectively. In Figure 4a, 37% of the web services have just one operations and 71% of the web services have less than 5 operations. On the other hand, the largest web service has over 110 operations. For the operation complexity, 181 operations among 5,180 operations have only one input/output parameter (see the leftmost point in Figure 4b). Considering a power-low distribution, this number is very small. The main reason is that it is natural for operations to have at least one input parameter and one output parameter. In fact, the number of operations with 2 parameters (see the second point) is quite large enough—over 1,000. Consequently, we can ignore the leftmost point in this figure. In addition, the whole distribution also shows a power-law-like property. In more detail, even after type flattening, around 65% of operations have less than 6 parameters while the maximum number of the parameters in an operation is 360.

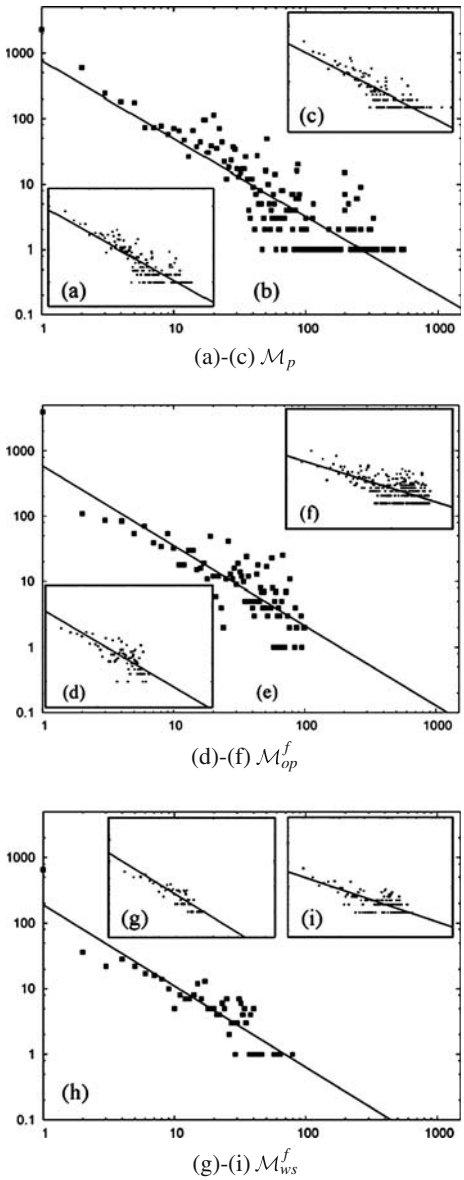
Second, in Figure 5, we examine the *popularity* of parameter names to see if some parameter names occur more often than others. X-axis is the frequency of parameter names while Y-axis is the number of samples. That is, (10, 100) indicates that there are 100 parameters that occur 10 times. Again, all of them show near



(a) $\alpha = 1.63$ (b) $\alpha = 1.58$ & (c) $\alpha = 1.06$

Figure 5 The popularity of parameter names. X-axis is the frequency of parameter names while Y-axis is the number of samples. **a** Exact matching, **b** Cosine (0.75), and **c** WordNet (0.75) (*inset*).

power-law distributions. From these distributions, we can observe a very small number of popular "hub" parameter names. For instance, the parameter ("name", string) appears most frequently in our collection. In addition, as the matching method gets more flexible, from (a) to (c) in Figure 5, more parameters are involved in matching. Since the matched parameters are considered as a single parameter, the number of distinct parameters decreases but the frequency of parameter names increases. For example, while p_1 appears 490 times in Exact matching among 11,301 distinct parameter names, p_1 appears 3,278 times in WordNet (0.75) matching among



(a)-(c) $\mathcal{M}_p$

(d)-(f) $\mathcal{M}_{op}^f$

(g)-(i) $\mathcal{M}_{ws}^f$

Figure 6 Out-degree distribution of three web service networks, $\mathcal{M}_p$ for (**a**)–(**c**), $\mathcal{M}_{op}^f$ for (**d**)–(**f**) and $\mathcal{M}_{ws}^f$ for (**g**)–(**i**): **a** Exact matching: $\alpha = 1.15$, **b** Cosine (0.75): $\alpha = 1.19$, **c** WordNet (0.75): $\alpha = 1.04$, **d** Exact matching: $\alpha = 1.18$, **e** Cosine (0.75): $\alpha = 1.12$, **f** WordNet (0.75): $\alpha = 0.64$, **g** Exact matching: $\alpha = 1.25$, **h** Cosine (0.75): $\alpha = 1.24$, and **i** WordNet (0.75): $\alpha = 0.68$.

7,042 distinct parameter names, where similar parameters like p_1 ("name", string) and p_2 ("identification", string) are considered to be a "match" and consolidated. Therefore, Figure 5c (inset) has the smallest α and the largest tail among these three.

In Figure 6, we examine the out-degree distributions of three web service networks of three matching schemes. Compared with previous figures, by and large, out-degree distributions show weaker power-law distributions. Nevertheless, it is evident to observe the existence of hub parameters, operations, or web services with huge number of out-degrees while majority has only a few.

In the parameter node networks of $\mathcal{M}_p$ of Figure 6a–c, due to flexible matching, the number of nodes decreases from (a) to (c): 11,301 for Exact matching, 10,568 for Cosine (0.75), and 7,042 for WordNet (0.75). However, even though Figure 6c has the smallest number of nodes of 7,042, it has the largest number of edges and biggest out-degree. For example, a parameter node including ("name", string) has 317 out-degrees (to 2.7% of nodes) in Exact matching, 306 out-degrees (to 2.9% of nodes) in Cosine (0.75), and 1,378 out-degrees (to 19.57% of nodes) in WordNet (0.75).

In operation node networks (see Figure 6d–f), every $\mathcal{M}_{op}^f$ corresponding to three matching schemes has the same number (5,180) of operation nodes since they are not merged. From exact to approximate matching, the total number of edges increases from 22,253 to 25,174 to 184,429. The fact that approximate matching schemes such as Cosine or WordNet have larger out-degrees indicates that information can flow more flexibly among web service operations through many edges. In Figure 6g–i, the web service node networks of $\mathcal{M}_{ws}^f$ also have a fixed number of 984 nodes, and the out-degree increases from (g) to (i). While some edges in operation node networks correspond to intra-connections within the same web service, the edges of $\mathcal{M}_{ws}^f$ are amount to pure inter-connection among different web services. As parameter node networks and operation node networks, more cooperation with different web services can be expected as we have more flexible approximate matching. In addition, the observation that there exist a small number of hub nodes in web service networks is helpful to improve web service discovery and composition—one can complete these processes earlier by considering the hub nodes first among a number of candidates.

5.3 Estimating the size of giant component

In this section, reversely from Sections 5.1 and 5.2, we study how to apply the graph model to generate data set (i.e., web service networks) accurately (*model* $\Rightarrow$ *data*). In particular, we focus on estimating the size of the giant component in a network. When a new web service w is created, it is natural for w to be connected to majority of nodes by being a part of the giant component in a network. Therefore, many real networks are reported to have giant components with a substantial size and density (e.g., [27, 28]).

The giant component size is calculated according to (5) in Section 2.3, and the results are shown in Figure 7. For each matching scheme, the theoretical value of the giant component size is very close to the measured one except for $\mathcal{M}_p$. This implies that this simple model can be helpful to estimate the inter-operable portion of such networks even without analyzing the actual network beyond its degree distribution. Of course, it is not a perfect model since there is still a gap for the giant component of $\mathcal{M}_p$. The theory is based on randomness between connections. If the actual networks

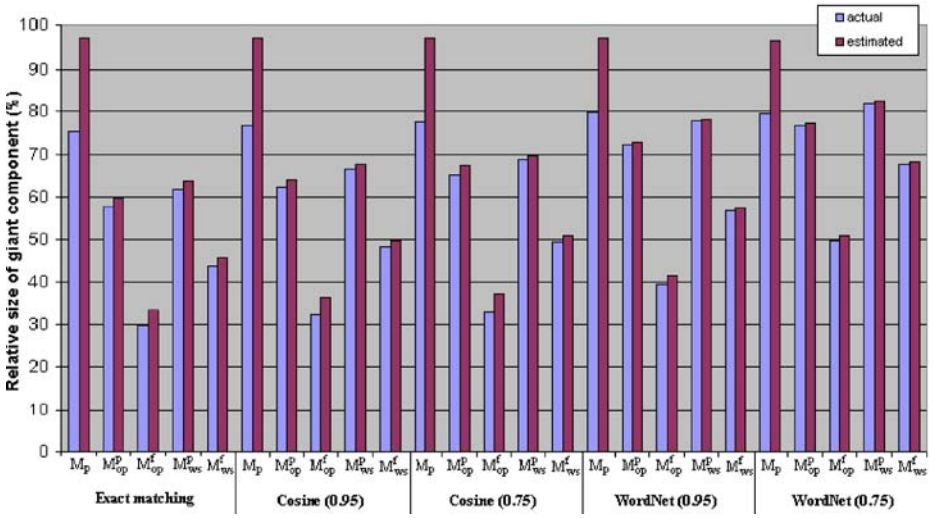


Figure 7 Actual vs. estimated size of giant components.

were random enough then one would expect the model to agree perfectly with real-world measurements. The fact that in some cases the results are not consistent to actual networks indicates lack of randomness in web service networks.

6 Observations and limitations

In this section, we discuss three important observations from (and limitations of) our experimentation.

Semantic Web The fact that Sections 5.1 and 5.2 show the small world and power-law-like properties more clearly as the matchmaking scheme becomes more *flexible* (from Exact to Cosine to WordNet) can be interpreted as an evidence of the needs of more semantically-rich web service networks. Many real networks (that are known to be robust and error-tolerant) have shown both small world and power-law properties. Therefore, we may argue that as web service networks become more mature and robust, they will show both small world and power-law-like properties better. The ultimate case of flexible matching is to use the semantics of parameters or operations. For instance, in the WordNet method of our experimentation, both "play" and "start" can be treated as synonym, meaning "begin". However, "play" may be used as a noun with the meaning "a theatrical performance" rather than a verb "begin". Therefore, this kind of mis-matching would cause a problem in discovering and composing web services (semi-)automatically. We envision that this kind of problems be solved only when the semantic web (services) are fully adopted in future.

Graph Fitting Graphs in Figure 6 among graphs in Section 5.2 follow weaker power-law-like patterns. This is due to outliers which can happen for various reasons.

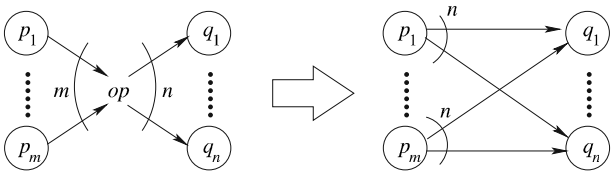


Figure 8 The effect of the operation complexity to the out-degree distribution.

First, operations with a number of parameters can significantly affect the number of outgoing edges on $\mathcal{M}_p$. For instance, in Figure 8, consider an operation with m input parameters and n output parameters. After transforming the operation and its parameters to $\mathcal{M}_p$, Then in parameter node networks, all of m input parameters of the operation have the same outgoing edges to all n output parameters, which produces m parameters with n out-degrees. If m is large, then the point (n, m) would sit far above the fitting line, becoming an outlier. Second, in $\mathcal{M}_{op}$, web services with many operations make similar results with $\mathcal{M}_p$. Since operations in a web service use similar parameters in names and types in general, the operations can invoke each other and have the same set of outgoing edges. As a result, operations within a web service can have the same out-degrees. Finally, we believe that outliers can happen due to the fact that we ignore service domains. Services from different domains are often not appropriate to invoke each other (i.e., ws_1 in the airline domain may not be meant to be used for ws_2 in the insurance domain), although they could have their inputs/outputs matched. We expect that as the size of web services increases and service domains are taken into account for matching, fitting becomes smoother.

Generative Model The agreement between the web service networks and theory in Section 5.3 suggests that there be no statistical difference between a web service network and an equivalent random network in terms of the giant component size. However, at the same time, the existence of disagreeing points in the analysis also suggests that one needs better models or tools (than the simple random network model) to be able to fully explain the properties and behaviors of the web service networks. Needless to say, more investigation is needed to fully untangle this problem.

7 Conclusion

In this paper, we have studied topological aspects of the real-world data set. Under the uniform framework of matchmaking, we have constructed various web service networks—some are based on exact matching while others employ more flexible approximate matching. Our findings are the following. First, regardless of the matching schemes and network types, all of web service networks show small world properties well and power-law-like distribution to some extent. Second, as more flexible matchmaking is employed, the resulting network shows the properties of small world and power-law network more clearly. Since many "robust" real-world complex systems also show the similar properties of small world and power-law network, we argue that the usage of more flexible approximate matching as in "semantic" web services would make the network more robust and useful.

Future Work Ample directions are ahead. First, we may try more extensive similarity/distance metrics (e.g., Edit distance, *n*-gram) and matching schemes (e.g., [21, 36, 40]) under our matchmaking framework. We expect to see the similar pattern (with different exponent) emerging from these cases as well. Second, toward true measurement of the semantic web services, we may use WSDL-S or OWL-S files, in addition to WSDL files, downloaded from search engines. As people annotate more semantics to their web services, we expect to have richer data set. Third, we may evolve our matching scheme using reasoning technologies, since the operation invocation may be decided by not only input/output parameter matching, but also pre/post conditions or state changes. Forth, we may aim at devising novel web service discovery and composition algorithms that can take advantage of the learned knowledge about network topology and properties.

References

1. Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M.-T., Sheth, A., Verma, K.: Web service semantics—WSDL-S. W3C Member submission (2005)
2. Akkiraju, R., Srivastava, B., Ivan, A.-A., Goodwin, R., Syeda-Mahmood, T.: SEMAPLAN: combining planning with semantic matching to achieve web service composition. In: Internatioal Conference on Web Services (ICWS), pp. 37–44 (2006)
3. Albert, R., Barabasi, A.-L.: Topology of evolving networks: local events and universality. Phys. Rev. Lett. **85**, 5234–5237 (2000)
4. Albert, R., Jeong, H., Barabasi, A.-L.: The diameter of the world wide web. Nature **401**, 130–131 (1999)
5. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley, Reading (1999)
6. Bellur, U., Kulkarni, R.: Improved matchmaking algorithm for semantic web services based on bipartite graph matching. In: Internatioal Conference on Web Services (ICWS), pp. 86–93 (2007)
7. Bianchini, D., Antonellis, V.D., Melchiori, M.: Flexible semantic-based service matchmaking and discovery. World Wide Web **11**(2), 227–251 (2008)
8. Bilenko, M., Cohen, W., Fienberg, S., Mooney, J., Ravikumar, R.: Adaptive name matching in information integration. IEEE Intell. Syst. **18**(5), 16–23 (2003)
9. Blake, M.B., Nowlan, M.F.: A web service recommender system using enhanced syntactical matching. In: Internatioal Conference on Web Services (ICWS), pp. 575–582 (2007)
10. Cai, H.: Scale-free web services. In: Internatioal Conference on Web Services (ICWS), pp. 288–295 (2007)
11. Cibrán, M.A., Verheecke, B., Vanderperren, W., Suvée, D., Jonckers, V.: Aspect-oriented programming for dynamic web service selection, integration and management. World Wide Web **10**(3), 211–242 (2007)
12. De Bruijn, J., Fensel, D., Keller, U., Kifer, M., Lausen, H., Krummenacher, R., Polleres, A., Predoiu, L.: Web service modeling language (WSML). W3C Member submission (2005)
13. Denning, P.J.: Network laws. Commun. ACM **47**(11), 15–20 (2004)
14. Dong, X., Halevy, A., Madhavan, J., Nemes, E., Zhang, J.: Similarity search for web services. In: International Conference on Very Large Data Bases (VLDB), pp. 372–383 (2004)
15. Fan, J., Kambhampati, S.: A snapshot of public web services. SIGMOD Rec. **34**(1), 24–32 (2005)
16. Fellbaum, C. (ed.): WordNet: An Electronic Lexical Database. MIT, Cambridge (1998)
17. Kil, H., Oh, S.-C., Lee, D.: On the topological landscape of web services matchmaking. In: VLDB Workshop on Semantic Matchmaking and Resource Retrieval. CEUR, vol. 178, pp. 19–34 (2006)
18. Lécué, F., Delteil, A., Léger, A.: Applying abduction in semantic web service composition. In: Internatioal Conference on Web Services (ICWS), pp. 94–101 (2007)
19. Li, L., Horrocks, I.: A software framework for matchmaking based on semantic web technology. In: International World Wide Web Conference (WWW), pp. 331–339 (2003)
20. Maedche, A., Staab, S.: Measuring similarity between ontologies. In: European Conference on Knowledge Acquisition and Management (EKAW), pp. 251–263 (2002)

21. Maguitman, A.G., Menczer, F., Erdinc, F., Roinestad, H., Vespignani, A.: Algorithmic computation and approximation of semantic similarity. World Wide Web **9**(4), 431–456 (2006)
22. Martin, D., Burstein, M., Hobbs, E., Lassila, O., Mcdermott, D., Mcilraith, S., Narayanan, S., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K.: OWL-S: semantic markup for web services (2004)
23. Martin, D.L., Burstein, M. H., McDermott, D.V., McIlraith, S.A., Paolucci, M., Sycara, K.P., McGuinness, D.L., Sirin, E., Srinivasan, N.: Bringing semantics to web services with OWL-S. World Wide Web **10**(3), 243–277 (2007)
24. Medjahed, B., Bouguettaya, A., Elmagarmid, A.: Composing web services on the semantic web. VLDB J. **12**, 333–351 (2003)
25. Molloy, M., Reed, B.: A critical point for random graphs with a given degree sequence. Random Struct. Algorithms **6**(2/3), 161–180 (1995)
26. Molloy, M., Reed, B.: The size of the largest component of a random graph on a fixed degree sequence. Comb. Probab. Comput. **7**(3), 295–305 (1998)
27. Newman, M.E.J.: Power laws, pareto distributions and Zipf's law. Contemp. Phys. **46**, 323–351 (2005)
28. Newman, M.E.J., Strogatz, S.H., Watts, D.J.: Random graphs with arbitrary degree distributions and their applications. Phys. Rev., E **64**(2), 026118 (2001)
29. OASIS: UDDI—online community for the universal description, discovery, and integration (2008)
30. Paolucci, M., Kawmura, T., Payne, T., Sycara, K.: Semantic matching of web services capabilities. In: International Semantic Web Conference (ISWC), pp. 333–347 (2002)
31. Sirin, E., Parsia, B., Hendler, J.: Template-based composition of semantic web services. In: AAAI Fall Symposium on Agents and the Semantic Web (2005)
32. W3C: Resource description framework. http://www.w3.org/RDF/ (2009)
33. W3C: Web services description language (WSDL) version 2.0 part 1: core language. http://www.w3.org/TR/wsdl20/ (2007)
34. W3C: The web ontology language. http://www.w3.org/TR/owl-features/ (2004)
35. W3C: SOAP—simple object access protocol. http://www.w3.org/TR/soap12-part1/ (2007)
36. Wang, Y., Stroulia, E.: Semantic structure matching for assessing web service similarity. In: IEEE International Conference on Services Computing (SCC), pp. 194–207 (2003)
37. Watts, D.J.: The Dynamics of Networks between Order and Radnomness. Princeton University Press, Princeton (1999)
38. Watts, D.J., Strogatz, S. H.: Collective dynamics of 'small-world' networks. Nature **393**(4), 440–442 (1998)
39. Wilf, H.S.: Generatingfunctionology, 2nd edition. Academic, London (1994)
40. Wu, J., Wu, Z.: Similarity-based web service matchmaking. In: IEEE International Conference on Services Computing (SCC), pp. 287–294 (2005)
41. Zhang, J., Zhang, L.-J.: Web services quality testing. Int. J. Web Serv. Res. **2**(2), 1–4 (2005)