On the Topological Landscape of Web Services Matchmaking

Hyunyoung Kil, Seog-Chan Oh, and Dongwon Lee*

The Pennsylvania State University
University Park, PA, USA
{hkil,seogchan,dongwon}@psu.edu

Abstract. Despite the rapid development in web services technologies, there has been little study on the public web services from network analysis perspective. To address this, in this paper, we have performed a topological study on various web service networks using real-world data sets. We first propose a flexible framework by which we can study semantic web services network "matchmaking" in a unified manner. Under the matching framework, then, we have conducted extensive experiments to study the characteristics of various web services networks. By and large, publicly available web services networks exhibit the typical characteristics of small world networks well and power law like distributions to some extent.

1 Introduction

The recent realization—the core web services specifications of WSDL [7], SOAP [17], and UDDI [8] are not sufficient to realize the goal of the *Semantic Web*—has developed a plethora of new means such as OWL [9], OWL–S [14], WSDL–S [16], and WSML [5] to annotate rich semantics to web services (ontology) so that truly automatic data exchange on the web is possible. Therefore, it is expected that, in the near future, service vendors will publish their offerings as "semantic web services" — 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 the web services network. 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. Therefore, in this paper, we aim at studying the web services network using network analysis techniques.

* The author is partially supported by Microsoft Scientific Data Intensive Computing (SciData) Award, 2005.

The focus of our study is on the web services specified in WSDL since currently there are not enough "semantic web services" specified in OWL–S or WSDL–S yet.Nevertheless, our matchmaking between web services is beyond syntactic matchmaking. By incorporating flexible matchmaking framework, we study the effect of semantic web services over real-world web services in WSDL. Our contributions of this paper are as follows:

- We propose a flexible web services 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 to semantic matching.
- We propose three dimensions of granularity to form web services networks from parameter to operation to web services. By looking into web services network using different glasses, we can understand the characteristics of the network better.
- We have conducted extensive experimentation under the proposed framework, and applied network analysis techniques to study the distribution, average distance, diameter, and clustering coefficient of the network.

1.1 Background

For the efficient exposition of the paper, in this section, we present a simplified abstraction of web services specified in WSDL.

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 $IN = \{p_1, ..., p_n\}$ and output parameters $OUT = \{q_1, ..., q_m\}$, 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. The WSDL specification lists four types of operations [7]: (1) *one-way*: the operation can receive a message but will not return a response; (2) *request-response*: the operation can receive a request and will return a response; (3) *solicit-response*: the operation can send a request and will wait for a response; and (4) *notification*: the operation can send a message but will not wait for a response. In order to decide if an operation op_1 can invoke an operation op_2, one needs to check if the types of op_1 and op_2 are compatible or not. Since this can be done trivially, from here forward, we assume that types of operations are all compatible, and focus on other issues instead.

For instance, consider the following web service w:

```
<message name='findRestaurant_Request'>
  <part name='zip' type='xs:integer'>
  <part name='foodPref' type='xs:string'>
</message>
<message name='findRestaurant_Response'>
  <part name='name' type='xs:string'>
  <part name='phone' type='xs:integer'>
</message>
```

```
<portType name="allRestaurant">
  <operation name="findRestaurant">
    <input message="findRestaurant_Request"/>
    <output message="findRestaurant_Response"/>
  </operation>
</portType>
```

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

1.2 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 the classical Watts-Strogatz network model [23]. Albert, Barabasi and Jeong [1] have 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 [10] 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 the semantic web services networks.

The problem of matching descriptions of two services is related with the classical problem of the *schema matching* in Database. For instance, [20] presents an array of latest techniques on the schema matching, and some of them could be used in our study. Note that we do not propose a particular matchmaking method in this paper. Instead, we advocate a flexible framework that can accommodate a plethora of matchmaking schemes in a unified manner. Then, based on the framework, we studied various topological properties of web services network. Due to the availability, for instance, we happen to use Cosine and WordNet based matching schemes. However, it is also possible to incorporate the aforementioned matching approaches such as [24, 22] or schema matching techniques in [20] in our framework.

Current web service matchmaking solutions are based on the keyword matching supported by the category-browsing of UDDI. However, the keyword-based matching ignores the real functions of web services. To address this limitation, researchers have developed a set of methods which assess the similarity of web services to achieve matchmaking. Wu [24] suggested a matchmaking process based on a lightweight semantic comparison of signature specifications in WSDL by means of several assessment methods. Wang and Stroulia [22] assessed the similarity of the requirement description of the desired service with the available services via the semantic information-retrieval (IR) method and a structure-matching approach. Maedche and Staab [13] provide multiple-phase cross-evaluation to assess the similarity between two different ontology. An excellent survey of modern matchmaking algorithms and their applications to the web

service matching field is available in [24]. Finally, some of the recent ontology-based service matchmaking works such as [2, 15, 3] are relevant to our research too. We leave the extension of our framework to accommodate those recent developments as future work.

2 The Matchmaking Framework

In this Section, we formalize the notion of matchmaking on web services.

2.1 Flexible Matching

Network consists of nodes and edges. 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, two parameters 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 [4] or WSDL-S [16]), then the so-called "semantic" matching can be easily reasoned out. However, in practice, majority of public web services do 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 or not. Formally,

Definition 1 (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 2 (name-match) *A boolean function,* `name-match`*$(p_1.name, p_2.name, \mathcal{D}, \theta)$, returns True if the distance between $p_1.name$ and $p_2.name$ by a distance metric $\mathcal{D}$ is below the given threshold θ: i.e., $\mathcal{D}(p_1.name, p_2.name) \leq \theta$.* □

For instance, `name-match`("password", "passwd", =, 1) represents the exact matching, and would return False since "password" $\neq$ "passwd"[1]. Similarly, by using string matching functions such as Edit distance [25] or cosine metric, one can express the approximate name matching. For instance, `name-match`("password", "passwd", edit-distance, 3) would return True since two names of parameters have the edit distance of 2 (< 3). In the experimentation, we use three metric functions: = (i.e., literal equality), cosine distance with

[1] For the exact matching, the threshold value other than 1 is not useful.

TF/IDF weights [21], and WordNet [12] based semantic distance. However, note that it is also possible to incorporate the aforementioned matching approaches such as [22] and [24] in our framework, if the implementation is available. Based on two boolean functions, now, we define a generic parameter match function as follows:

Definition 3 (match) *A boolean function,* `match`$(p_1, p_2, \mathcal{D}, \theta)$*, returns True if: (1)* `name-match`$(p_1.name, p_2.name, \mathcal{D}, \theta) = True$*, and (2)* `type-match`$(p_1.type, p_2.type) = True$*.* □

Definition 4 (Parameter Matching) *When a boolean function,* `match`$(p_1, p_2, \mathcal{D}, \theta)$*, returns True, it is said that a parameter* p_1 **matches** *a parameter* p_2 *(i.e.,* p_1 *can be used in the place of* p_2*), and written as "*$p_1 \sim p_2$*".* □

In order to invoke an operation op_1 with input parameters $IN=\{p_1, ..., p_n\}$, one may need to provide values for all required (i.e., mandatory) input parameters. When one can provide all required input parameters, op_1 is said "fully invocable". When one can provide at least one required input parameter, op_1 is said "partially invocable", so partially invocable operations includes a set of fully invocable operations. For instance, Google API has a search operation with several input parameters, but only part of them are mandatory. 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 services composition). In this case, if the output parameters of op_1 can satisfy all required input parameters of op_2, then op_1 "fully" invokes op_2, and if there exists any output parameter of op_1 satisfying any input parameter of op_2, then op_1 "partially" invokes op_2. Formally,

Definition 5 (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 "*$op_1 \rightarrowtail op_2$*", if for every mandatory 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 a mandatory 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.* □

2.2 Web Service Network Model

In this Section, using the notions of *nodes* and *edges* defined in Section 2.1, we propose a flexible web service network model as follows.

Definition 6 (Web Service Network Model) *A web service network is generated by a 4-tuple model* $\mathcal{M}=(\mathcal{T}, \mathcal{D}, \theta, \mathcal{I})$*, where:*

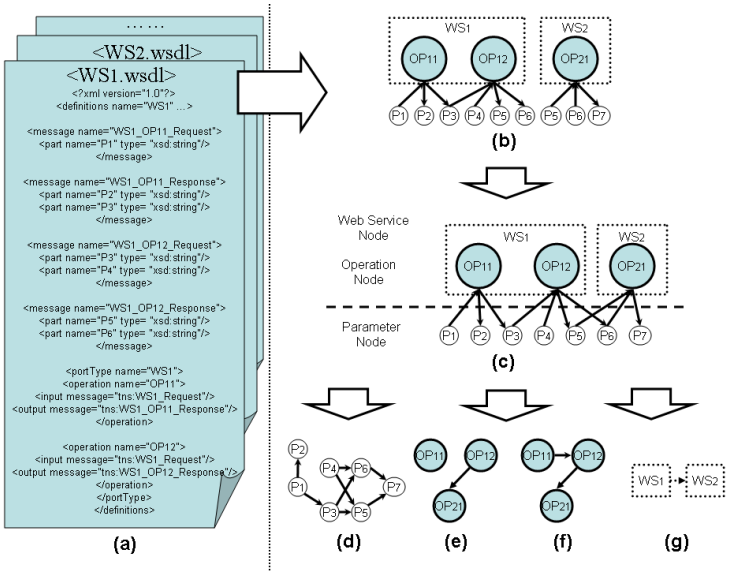


Fig. 1. Web service networks: (a) WSDLs, (b) Conceptual web service network, (c) Web service networks from diverse models, (d) Parameter node network, $\mathcal{M}_p$, (e) Fully invocable operation node network, $\mathcal{M}^f_{op}$, (f) Partially invocable operation node network, $\mathcal{M}^p_{op}$, and (g) Web service node network, $\mathcal{M}_{ws}$

- $\mathcal{T}$ *is the type of node and can be either* `p` *for parameter,* `op` *for operation, or* `ws` *for web service*
- $\mathcal{D}$ *(and* θ *resp.) is the distance metric to be used in parameter matching (and its threshold resp.)*
- $\mathcal{I}$ *is the type of operation invocation and can be either* `FI` *for full invocation or* `PI` *for partial invocation* □

Example 1. A model $\mathcal{M}_1$=(op, cosine, 0.75, 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 $op_1 \rightarrowtail op_2$. □

Example 2. A model $\mathcal{M}_2$=(ws, word-net, 0.9, PI) generates a web service node network where intra-parameter matching is done using WordNet based semantic distance 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 ($\in ws_2$) – 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 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 of the operation op. On the other hand, an edge from an operation node op to a parameter node p indicates 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 difference among node types, one can project out the aforementioned web service network into three kinds as follows:

- A *parameter node network*, i.e., $\mathcal{M}_p$=(p, $\mathcal{D}$, θ, $\mathcal{I}$), consists of parameter nodes and edges representing operations that has the source node as an input parameter and the target node as an output parameter. For instance, if $op_1(IN_1, OUT_1)$ exists, then we create edges from each parameter p of IN_1 to every output parameter q of OUT_1. When approximate matching or semantic matching is used for intra-parameter matching, each node in $\mathcal{M}_p$ is in fact a set of similar nodes. That is, suppose there is an edge from p_1(password, string) to p_3(firstName, string) in $\mathcal{M}_p$. This indicates that one can retrieve "firstName" information by providing "password" information. Now, if an approximate matching method (e.g., edit distance) or semantic matching method determines that a parameter p_1(password, string) matches a parameter p_2(passwd, string), then p_2.name is merged with p_1.name to form an edge: {password, passwd} $\rightarrow$ firstName. That is, the source node is a set of two nodes.
- An *operation node network*, i.e., $\mathcal{M}_{op}$=(op, $\mathcal{D}$, θ, $\mathcal{I}$), consists of nodes representing operations and edges representing the invocation in-between. That is, if an operation op_1 can (either partially or fully based on $\mathcal{I}$) invoke an operation op_2, there is an edge $op_1 \rightarrow op_2$ in $\mathcal{M}_{op}$.
- A *web service node network*, i.e., $\mathcal{M}_{ws}$=(ws, $\mathcal{D}$, θ, $\mathcal{I}$), consists of all web service nodes and directed edges representing the existence of invocable operations between web services.

The Figure 1(d) describes a parameter node network $\mathcal{M}_p$ based on Figure 1(b). 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 1(c). Therefore, there exists an edge $op_{12} \rightarrow op_{21}$ in both operation node networks (Figure 1(e) and Figure 1(f)). On the other hand, p_3, one of outputs of op_{11}, is used as one of input of op_{12}. However, op_{12} requires another input parameter data p_4 not produced by op_{11}. In the partial invocation mode, therefore, an edge $op_{11} \rightarrow op_{12}$ is added. On the contrary, in the full invocation mode, no edge is added. Note that in the example of Figure 1, we get 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.

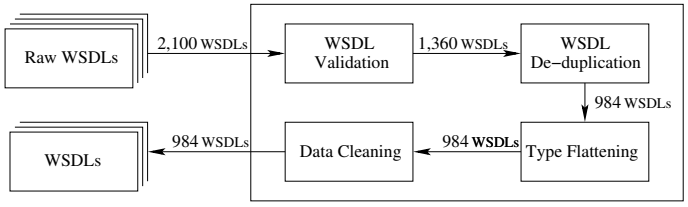


Fig. 2. Overview of pre-processing.

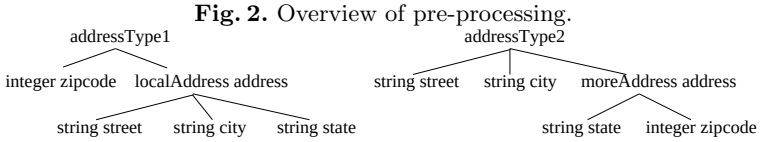


Fig. 3. Two compatible types with different structures.

3 Topological Landscape

3.1 Set-Up

The pre-processing for the experimentation consisted of five steps:

1. *Data Gathering*: A total of 2,100 real-world WSDL files were gathered from two sources. First, 1,554 files were taken from Fan et al. [11] who downloaded the files from public repositories such as `XMethods.org` or `BindingPoint.com`. Second, out of top-1000 ranked WSDL files from Google for the query string "`wsdl filetype:wsdl`", 546 were downloaded using Google API (the rest were un-downloadable).
2. *WSDL Validation*: According to WSDL standard, 740 invalid WSDL files were removed, and 1,360 files are left out.
3. *WSDL De-duplication*: 376 duplicate WSDL files at operation level were removed, yielding 984 valid WSDL files.
4. *Type Flattening*: In 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 3. Although names of two parameters are similar, their types are user-defined and different. However, two types are in fact "compatible". Therefore, if we *flatten* two 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 two parameters can be matched. We call this process as type flattening[2]. After type flattening, then each atomic type and name are compared using type hierarchy of XML Schema and flexible matching scheme (e.g., exact or approximate), respectively. The resulting statistics of type flattening is shown in Table 1.

[2] 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 1. Type distribution.

Type	Before Flattening		After Flattening	
	input(num)	output(num)	input(num)	output(num)
`anyType`	6.79% (7)	9.03% (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

5. *Data Cleaning*: The final step is cleaning data to improve the quality of parameters. For instance, substantial number of output parameters (16%) were named "return", "result", or "response" which is 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.

For flexible matching, we used: (1) *Exact Matching*; (2) *Cosine Distance*. For instance, when we compare two parameters p_1("BirthDay", string) and p_2("B-day", string), names are segmented to a set of tokens with TF/IDF weights like p_1({Birth, Day}, string) and p_2({B, Day}, string). Then, by converting both sets into vectors of v and w, their distance is measured as $\cos(\theta) = \frac{v \cdot w}{\|v\| \cdot \|w\|}$; (3) *WordNet-based Distance*. Since WordNet is a network of English words labeled with semantic classes (e.g., synonyms), it carries various semantic relations among words. People have proposed various ways to measure the "semantic distance" of two words in WordNet [6]. We use the one by Lin [12] that scales the information content of the least common subsumer by the sum of the information content of two vocabularies. Both cosine and WordNet metrics used threshold to determine the matchmaking (i.e., if the calculated distance is above threshold, new edge is added into the network).

At the end, we have generated a total of 25 (= 5 network types × 5 distance metrics) web services networks: (1) three kinds of networks – $\mathcal{M}_p$, $\mathcal{M}^f_{op}$, and $\mathcal{M}^f_{ws}$, and two of their counterparts of "partial invocation" – $\mathcal{M}^p_{op}$ and $\mathcal{M}^p_{ws}$; (2) five distance variations – Exact, Cosine (0.75), Cosine (0.95), WordNet(0.75), and WordNet(0.95).

Table 2. Small world properties of giant components in public web services.

Matching Scheme	Network	L_{actual}	L_{random}	C_{actual}	C_{random}
Exact matching	$\mathcal{M}_p$	4.31852	3.42441	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

3.2 Small World

First, we examine if web services network exhibit small world properties. In general, networks or graphs are called *small world* networks if they show the properties of both *random* and *regular* networks.

Definition 7 (Random Network) *$G_{(N,p)}$ is a random network of N nodes, if each pair of nodes is connected with the probability p. As a result, edges are randomly placed among a fixed set of nodes.* □

Definition 1 (Regular Network). *$Rg_{(N,k)}$ is defined as the regular network on N nodes, if node i is adjacent to nodes $[(i+j)\ mod\ N]$ and $[(i-j)\ mod\ N]$ for $1 \leq j \leq k$, where k is the number of valid edge of each node. If $k = N-1$, $R_{(N,k)}$ becomes the complete N-nodes graph, where every node is adjacent to all the other $N-1$ nodes.* *None*

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

are very high. Consequently, small world networks show both (1) highly clustered structure and (2) small shortest distance.

- L: The average shortest distance (i.e., number of hops) between reachable pairs of vertices. $L(p)$ is the L of Watts-Strogatz graph [23] with probability p. L_{random} is identical to $L(1)$.
- C: The average clustering coefficient. For a node i with v_i neighboring nodes, $C_i = \frac{2E_i}{v_i(v_i-1)}$, 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:

$$Index_{SN} = \frac{|C_{actual} - C_{random}|}{|L_{actual} - L_{random}|} \quad (1)$$

  where C_{actual} and L_{actual} represent C and L of the measured network, respectively, and both C_{random} and L_{random} represent C and L of the random graph with the same number of nodes and average number of edges per node as the measured network.

If a network has the small world properties, then its L and C shows: $C \gg C_{random}$ and $L \gtrsim L_{random}$. Therefore, the more distinct the small world properties of a network are, the bigger $Index_{SN}$ of the network becomes.

Table 2 shows the average shortest path, L and clustering coefficient, C for giant components extracted from each of the 25 web service networks, compared to random graphs with the same number of nodes and average number of edges per node. Note that we treat all edges of each network as undirected and unweighted[3]. It is interesting that all web services networks are the small world network by showing $L \gtrsim L_{random}$ and $C \gg C_{random}$. This suggests that the real web services have *short-cuts* that connect nodes of networks. Otherwise, nodes of networks would be much farther apart than L_{random}.

There are distinct changes between different matching schemes in terms of the small world properties. For example, in the parameter node network $\mathcal{M}_p$, L(=3.2656) of the WordNet(0.75) is much smaller than L(=4.318) of the exact matching, while C(=0.3118) of the WordNet(0.75) is much bigger than C(=0.222) of the exact matching. For better understanding, Figure 4 illustrates the changes of $Index_{SN}$ for different matching schemes. Recall that $Index_{SN}$ tends to increase as C increases or L decreases. In the Figure, we can see that the usage of semantic matching scheme, WordNet(0.75) generates more distinct small world properties than the exact and cosine distance matching schemes. In the case of $\mathcal{M}_p$ with the WordNet(0.75) matching applied, $Index_{SN}$ is ten

[3] This crude approximation is often acceptable since our goal is to study the network topology, not the flow of materials on edges (e.g., information, electricity). Similar assumption was made by Watts and Strogatz in [23]. In Section 3.3, directed network cases are analyzed further.

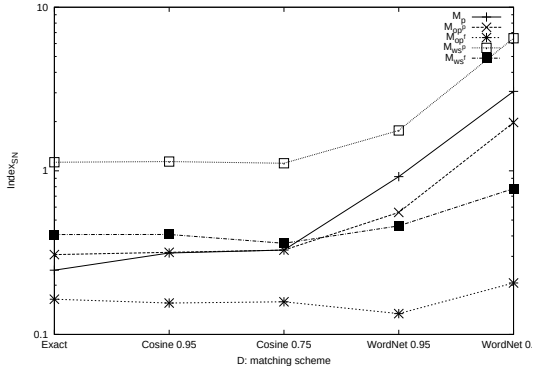


Fig. 4. $Index_{SN}$ changes.

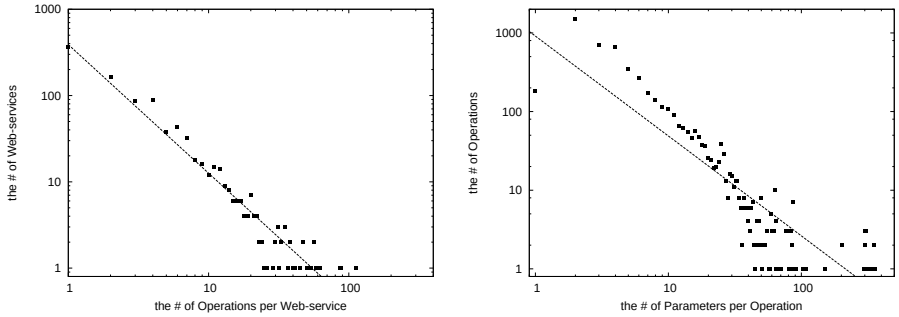
times bigger than that of the exact matching case. This result suggests that the semantic matching scheme makes the network more dense with the increased number of edges, so that L decreases while C increases. From the web service composition perspective, this result suggests that semantic matching scheme can facilitate more productive and effective service compositions than when only exact syntactic matching is used.

3.3 Power Laws

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 degrees. 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 [10, 19]. In this section, we examine if semantic web services network follows power law distribution or not.

First, we examine how *complex* web services are in general. One way to measure the complexity of web services is to measure how many operations (parameters resp.) are involved in each web service (operation resp.) [11]. These results are shown in Figure 5, fitted to a power-law function $y = Cx^{-\alpha}$ (in log-log plot). They show *near*[4] power law distribution with the exponent of 1.49 and 1.27, respectively. In Figure 5(a), 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 services have over 110 operations. Similarly, in Figure 5(b), 181 operations (among 5,180 operations) have less than two input/output parameters. In addition, 194 operations have no input parameter and 255 operations have no output parameter. However, the whole distribution also shows a power law like property. After type flattening, around 65% of operations have less than

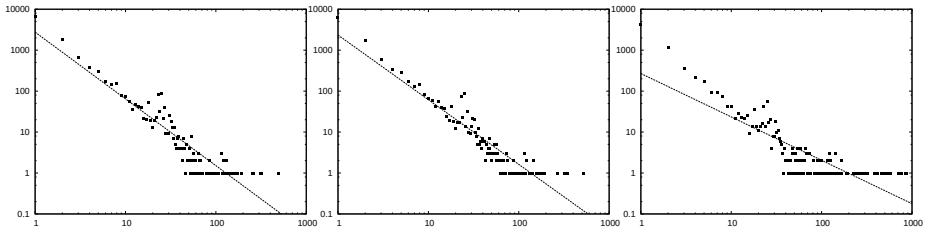
[4] A recent study [18] reported that most of real-world power law distributions exhibited an exponent of $2 \leq \alpha \leq 3$.



(a) # of operations per web service ($\alpha = 1.49$)

(b) # of parameters per operation ($\alpha = 1.27$)

Fig. 5. The complexity of web services and operations.



(a) Exact Matching α=1.63

(b) Cosine (0.75) α=1.58

(c) WordNet (0.75) α=1.06

Fig. 6. The popularity of parameter names. X-axis is the frequency of parameter names while Y-axis is the number of samples.

6 parameters while the maximum number of the parameters in an operation is 360.

Second, we examine the *popularity* of parameter names – 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, a coordinate (10, 100) indicates that there are 100 number of parameter names that occur 10 times. Again, all of them show near power law distributions. From these distributions, we can observe a very small number of popular "hub" parameter names. For instance, a parameter p_1 ("*name*", *string*) appears most frequently in our collection. In addition, as the matching method gets more flexible (from left to right in Figure 6), more parameters get to match each other. Since the matched parameters are considered as a single parameter, the number of distinct parameters gets smaller but the frequency of parameter names gets bigger. For example, while p_1 appears 490 times in Exact matching with 11,301 distinct parameter names, p_1 appears 3,278 times in WordNet (0.75) matching with 7,042 distinct parameter names, where similar parameters like p_2("*name*", *string*) or p_3("*identification*", *string*)

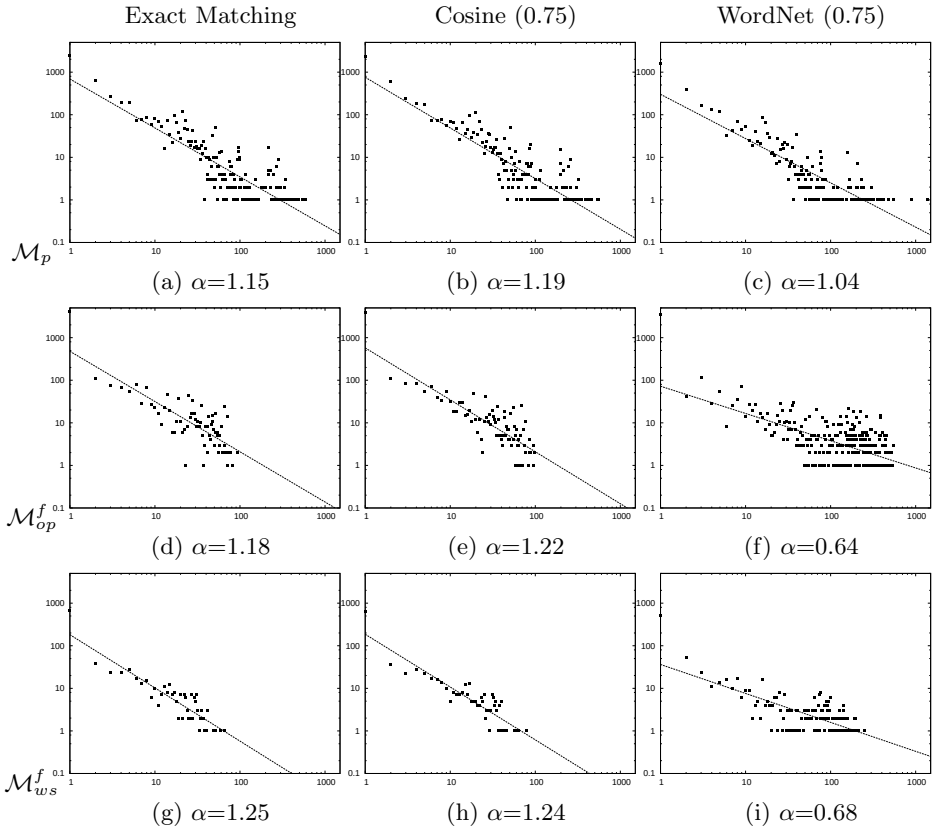


Fig. 7. Out-degree distribution of three web service networks (rows) of three matching schemes (columns). X-axis is out-degree, and Y-axis of the first, second, and third rows is the number of parameter, operation, and web service nodes, respectively.

are considered to be a "match" and consolidated. Therefore, Figure 6(c) has the the smallest α and the largest tail among three.

In Figure 7, next, we examine the out-degree distributions of three web service networks of three matching schemes. Unlike previous figures, by and large, out-degree distributions no longer follow power law distributions well (although we still fit them with power law functions). Nevertheless, it is evident to see 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 $\mathcal{M}_p$ (first row of Figure 7), due to flexible matching, the number of nodes decrease from left to right: 11,301 for Exact matching, 10,568 for Cosine (0.75), and 7,042 for WordNet (0.75). However, even though Figure 7(c) 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

Consine (0.75), and 1,378 out-degrees (to 19.57% of nodes) in WordNet (0.75). The distributions between Figure 7(a) and Figure 7(b) show little difference with respect to the number of nodes.

Unlike parameter node networks of $\mathcal{M}_p$, all operation node networks of $\mathcal{M}_{op}^f$ (second row of Figure 7) have 5,180 fixed operation nodes, but, from left to right, the total number of edges becomes bigger from 22,253 to 25,174 to 184,429. The fact that approximate matching scheme such as Cosine or semantic matching scheme such as WordNet have more out-degrees indicates that information can flow more flexibly among web services operations through many edges. In the last row of Figure 7, the web services node networks of $\mathcal{M}_{ws}^f$ also have a fixed number of 984 nodes, and out-degree increases from left to right. While some edges in operation node networks include intra-connections within the same web service, the edges of $\mathcal{M}_{ws}^f$ are amount to pure inter-connection among different web services. Consequently, more cooperation with different web services can be expected as we have more flexible semantic matching. Since one can invoke more number of operations, in general, it gets easier to combine and compose multiple web services [19].

Finally, we also examined the difference between partial vs. full invocation in invoking operations. Due to space constraint, we do not present the results. The finding is consistent with previous cases in that partial invocation based network gets more edges for its more flexible matching that full invocation based one.

4 Conclusion

In this paper, we have studied the topology of web services network using real-world data sets. To find out topological properties of the networks, we performed extensive experiments on various node networks under a flexible matchmaking framework that covers exact, approximate, and semantic matching. According to our experiments, we have learned that: (1) Public web service networks show small world network property well and follow a power law like distribution pattern to some degree. However they do not fully show the maturity seen in other complex networks yet; and (2) Semantic matching among web services generates networks with higher degrees than exact or approximate matching does. This suggests that more flexible web services composition can be achieved with the increased semantics in web services matchmaking.

References

[1] R. Albert and A.-L. Barabasi. "Topology of Evolving Networks: Local Events and University". *Phys. Rev. Lett.*, 85:5234–5237, 2000.

[2] V. De Antonellis, M. Melchiori, L. De Santis, M. Mecella, E. Mussi, B. Pernici, and P. Plebani. "A Layered Architecture for Flexible Web Service Invocation". *Software: Practice and Experience*, 36(2):191–223, 2005.

[3] D. Bianchini, V. De Antonellis, B. Pernici, and P. Plebani. "Ontology-based Methodology for E-service Discovery". *Information Systems*, 31(4):361–380, 2006.

[4] D. Brickley and R. V. Guha (Eds). "Resource Description Framework (RDF) Schema Specification 1.0". W3C Recommendation, Mar. 2000. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.
[5] J. Bruijn (Ed.). "The Web Service Modeling Language WSML", 2005. http://www.wsmo.org/wsml/wsml-syntax.
[6] A. Budanitsky and G. Hirst. "Evaluating WordNet-based Measures of Semantic Distance". *Computational Linguistics*, 32(1):13–47, 2006.
[7] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. "Web Services Description Language (WSDL) 1.1". W3C Recommendation, 2001. http://www.w3.org/TR/2001/NOTE-wsdl-20010315.
[8] L. Clement et al. (Ed.). "UDDI Version 3.0.2", 2004. http://uddi.org/pubs/uddi-v3.0.2-20041019.htm.
[9] M. Dean and G. Schreiber (Ed.). "OWL Web Ontology Language Reference", 2004. http://www.w3.org/TR/owl-ref/.
[10] P. J. Denning. "Network Laws". *Comm. ACM*, 47(11):15–20, 2004.
[11] J. Fan and S. Kambhampati. "A Snapshot of Public Web Services". *SIGMOD Record*, 34(1):24–32, 2005.
[12] C. Fellbaum (Eds). *"WordNet: An Electronic Lexical Database"*. The MIT Press, 1998.
[13] A. Maedche and S. Staab. "Measuring Similarity between Ontologies". In *European Conf. on Knowledge Acquisition and Management (EKAW)*, Siguenza, Spain, 2002.
[14] D. Martin (Ed.). "OWL-S 1.1 Release", 2005. http://www.daml.org/services/owl-s/1.1/.
[15] B. Medjahed and A. Bouguettaya. "A Multilevel Composability Model for Semantic Web Services". *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, 17(7):954–968, 2005.
[16] J. Miller et al. "WSDL-S: Adding Semantics to WSDL - White Paper", 2004. http://lsdis.cs.uga.edu/library/download/wsdl-s.pdf.
[17] N. Mitra (Ed.). "SOAP Version 1.2 Part 0: Primer". W3C Recommendation, 2003. http://www.w3.org/TR/2003/REC-soap12-part0-20030624/.
[18] M. E. J. Newman. "Power laws, Pareto distributions and Zipf's law". *Contemporary Physics*, 46:323–351, 2005.
[19] S.-C. Oh, D. Lee, and S. Kumara. "A Comparative Illustration of AI Planning-based Web Services Composition". *ACM SIGecom Exchanges*, 5(5):1–10, Dec. 2005.
[20] E. Rahm and P. Bernstein. "A Survey of Approaches to Automatic Schema Matching". *VLDB J.*, 10(4):334–350, 2001.
[21] G. Salton and M.J. McGill. *"Introduction to Modern Information Retrieval"*. McGraw–Hill, 1983.
[22] Y. Wang and E. Stroulia. "Semantic Structure Matching for Assessing Web Service Similarity". In *Int'l Conf. on Service Oriented Computing*, Trento, Italy, 2003.
[23] D. J. Watts and S. H. Strogatz. "Collective Dynamics of 'Small-World' Networks". *Nature*, 393(4):440–442, 1998.
[24] J. Wu and Z. Wu. "Similarity-based Web Service Matchmaking". In *IEEE Int'l Conf. on Service Computing (SCC)*, Orlando, FL, USA, 2005.
[25] K. Zhang and D. Shasha. "Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems". *SIAM J. Comput.*, 18(6):1245–1262, Dec. 1989.