AAAI Technical Report SS-12-04
Intelligent Web Services Meet Social Computing

Optimizing Service Composition Network from Social Network Analysis and User Historical Composite Services

Yuanbin Han, Shizhan Chen, Zhiyong Feng
School of Computer Science and Technology
Tianjin University
Tianjin, 300072, China
ybhan09@gmail.com, shizhan@tju.edu.cn, zyfeng@tju.edu.cn

Abstract

Service composition, which achieves the goal of value-added services, has been considered as the core technique of Service-oriented Computing (SOC). To cope with the challenge of ever-increasing number of web services, graph-based web service network has emerged as a potential solution to the state of art SOC. In such a way, composite services are constructed by applying searching algorithms to the built graph, and proved to achieve outstanding performance in complexity. However, web service network suffers two crucial disadvantages: poor connectivity and negative links, and both of them have crucial negative impact on service composition. To cope with the problems, we propose two methods in this paper. Firstly, leveraging social network analysis, we focus on enriching web service network by adding valuable services, which will play positive roles in solving poor connective problem. Secondly, we show a serious status that numerous negative links contained in the underlying networks, and then we propose to identify and remove the negative links based on users' historical composite services.

Introduction

Service-oriented Computing (SOC) is gaining increasing interest by researchers, especially as it links up with semantic techniques (Martin & Domingue 2007). Composing existing web services is the key technique of SOC tasks, since it is the primary way to satisfy users' complex requirements, as well as to achieve the goal of "value-added" services. Current service composition approaches are essentially based on sophisticated AI-Planning techniques (Sirin et al. 2004; Peer 2005; Oh, Lee & Kumara 2006). Particularly, to cope with the challenge of ever-increasing number of web services on the Internet, **graph-based web service network** has been considered as a potential solution to the state of art SOC techniques.

Copyright © 2012, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

In general, a web service network consists of nodes and links, where each node represents a web service and each link indicates the interaction between services (e.g. in a simple paradigm, if service A has an output a, and service B has an input a, then B can call A for composing, thus a directed link is associated with the two services ($A \rightarrow B$)). Due to the nature of web service network, the ties of the network imply the interoperability among services. Hence, composite services can be automatically computed by searching algorithms to the network. (Hashemian & Mavaddat 2005, 2006) for example, proposed modeling available web services as well as their inputs-outputs information into a dependency graph. Therefore, a composite service could be computed by using searching algorithms among connectable services. Similarly, in the work by (Shin, Lee & Suda 2009), the authors proposed a two-layer service relation graph including enriching the functional semantics of services, as well as corresponding composing method, which performed significant influence in time complexity as well as accuracy. Both of the works (Hashemian & Mavaddat 2005, 2006; Shin, Lee & Suda 2009) pointed out that graph-search based service composing methods yielded low time complexity, compared with most of other AI-based approaches.

However, graph-search based service composing techniques still suffer high complexity and low efficiency bottlenecks, since they essentially rely on the connectivity and validity of the based web service network. For first, isolated services in a networked service graph cannot be involved in composing. In addition, in our study, we have observed that web service network contains a large number of negative links. E.g. the "*car price*" service returns price of cars for users who intend to buy a car, and "*coffeewithwhiskey price*" service returns a

coffeewithwhiskey for the given price. The two services are connected by a "*price*" link ("*car price*"→ "*coffeewithwhiskey price*") in the web service network. Clearly the link could be easily identified by human that it is incorrect, since the two connected services no relation in practical context. Obviously, negative links will lead to high time complexity while composing services, as well as results that deviate from users' intended requirements.

To facilitate graph-search based service composing, it is crucial that the problems both poor connectivity and negative links in the web service network have to be taken into account. The objective of this paper is to optimize web service network for solving the two above challenges. More concretely, our contributions are:

Enrich the connectivity for web service network. We first generate the original web service network from the services dataset, and show that a number of isolated services are intuitively observed from visualization of the network. Then we perform the social network analysis (SNA [1]) to find the underlying properties of the built graph. Moreover, we focus on how to add generated services to the built network for enriching its connectivity. The most crucial contribution behind is how to define valuable services that much-needed by the network. Experimental evaluations show that our findings and approach play a positive role to web service network.

Prune negative links using users' historical composite services. A simple yet effective method is proposed for extracting basic composition units from users' historical composite services, which can be mapped into links in the underlying networks. Therefore, by identifying basic composition units based on a learning mechanism, we can prune those negative links.

Dataset and Web Service Network

Description of the dataset

The dataset used in our work is OWL-S Service Retrieval Test Collection, version 3.0 (OWLS-TC3) [2], which was released in July, 06, 2009. In total, OWLS-TC3 contains 1007 semantic web services (i.e. 1007 .owls files) described by OWL-S language(OWL-S 2004) as well as divided into seven different domains, 23 different ontologies ((i.e. 23 .owl files)) for specifying the semantics of input/output (I/O) concepts of services, and others. It should be noted that there are two main restrictions in OWLS-TC services: services are directly described by only atomic process, and no precondition and effect parameter values are specified (i.e. they can be viewed as stateless services). We parsed the most popular 4 domain services (i.e. *communication*, *economy*, *education* & *travel*) in OWLS-TC as the dataset in our study, and extracted a sub-ontology from the 23 different ontologies, as summarized in Table 1.

# *parsed services*	867
Extracted sub-ontology size(# *classes*)	271
# *parsed services used in this work*	790
# *parameters*(both *inputs* and *outputs*)	250

Table 1: Dataset. We abandoned 77 *parsed services* since they either have no *inputs* or have no *outputs.*

Designing of original graph-based web services network

In this section, we form the dataset into two types of networks. Firstly, we define some basic notations:

Definition 1 (Parameters). $P = \{p_1, p, \dots, p_m\}$ denotes the *I/O* concepts of services.

Definition 2 (Ontology). Let $T = \{c_1, c_2, \dots, c_t\}$ be the ontology we extracted. More specially, to completely specify the semantics of P, we have $P \subseteq T$, as the dataset we comprised in Table1.

Definition 3 (Services). Let $S = \{s_1, s_2, \dots, s_n\}$ be the set of services in the dataset. Each service $s_i \in S$ is represented by a tuple $s_i = < I_i, O_i >$, where $I_i \subset P$ and $O_i \subset P$ are the associated *I/O* of services.

Definition 4 (Service-parameter network). A directed graph is formalized as $SPN = (\{S \cup P\}, E^{SPN})$, where each $e \in E^{SPN}$ includes four types of relations:

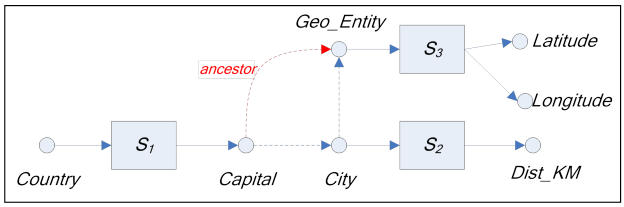
- $\Re^{s \to p}$: for a service $s_i = < I_i, O_i >$, if $p \in I_i$.
- $\Re^{p \to s}$: for a service $s_i = < I_i, O_i >$, if $p \in O_i$.
- $\Re^{pi \to pj}$: for any two parameters p_i, $p_j \in P$, if p_j is the parent or ancestor concept of p_i in T.
- $\Re^{pi \leftrightarrow pj}$: for any two parameters p_i, $p_j \in P$, if they are semantically equal to each other in T.

We emphasize that our *SPN* is similar to the *I/O dependency graph* presented by (Hashemian & Mavaddat 2005, 2006), and almost the same as *data dependency graph* (the bottom layer of their two-layered graph) in (Shin, Lee & Suda, 2009). The main difference between *I/O dependency graph* and *data dependency graph* was that services were also regard as nodes in *data dependency graph*, our *SPN* differs from *data dependency graph* that we enhance the semantics by including both parent–child and ancestor–child semantic relation in $\Re^{pi \to pj}$. The purpose of adding ancestor–child semantics is to evaluate more thorough importance of concepts by their degrees in *SPN* (the feature is useful in next section). For example in Figure 1(Similar to the example in (Shin, Lee & Suda, 2009)), the *Geo_Entity* concept maintains a degree of 3 in *SPN*, rather than a degree of 2 in *data dependency graph*. Since measuring node importance by its degree only

[1] Here we use term "SNA" since we consider that web services also capture interactions which are much like human activities.

[2] *semwebcentral.org*: http://projects.semwebcentral.org/. For more details, readers can refer to *OWLS-TC Version 3.0 User Manual* contained in the *package*.

reveals "local" features, adding ancestor–child semantics can reasonably catch more "global" features in semantics.



Service	Inputs	Outputs	Description
S_1	*Country*	*Capital*	Provide the capital of a country
S_2	*City*	*Dist_KM*	Calculate the distance between 2 cities
S_3	*Geo_Entity*	*Latitude, Longitude*	Inform latitude/longitude for a geographical entity

Figure 1: An illustration of *SPN*

Definition 5 (Service-service network). A directed graph is formalized as $SSN = (S, E^{SSN})$, where each $e \in E^{SSN}$ denotes the sequence relationship among services. It seems *SSN* is almost the same as the *action graph* in (Shin, Lee & Suda 2009). However, there is an essential difference between the two graphs that action concepts and services may hold one-to-many relationship in *action graph*.

The purpose of constructing *SSN* is to characterize more potential features of web service network for facilitating SOC, as studied later in this work. The process of constructing *SPN* and *SSN* are given in Algorithm1 and Algorithm2 respectively. Due to the limited space of paper, detailed descriptions are not presented, while necessary statements can refer to the definitions.

Algorithm1: *SPN* construction

Input: Services set S, an ontology T
Result: Graph *SPN*
//Most notations below can be found in the definitions;
begin
  $SPN \leftarrow \emptyset$
  for each *service* $s \in S$ **do**
    for each *concept* $co \in s.O$ **do**
      $SPN \leftarrow SPN \cup \{s \rightarrow co\}$ //$\Re^{s \rightarrow p}$
    for each *concept* $ci \in s.I$ **do**
      $SPN \leftarrow SPN \cup \{ci \rightarrow s\}$ //$\Re^{p \rightarrow s}$
  for each *concept* $a \in P$
    for each *concept* $b \in P$
      if $a \neq b$
        if a is the parent *or* ancestor of b in T
          $SPN \leftarrow SPN \cup \{b \rightarrow a\}$ // $\Re^{pi \rightarrow pj}$
          /*We do not compute whether there is an "$a \rightarrow b$" edge for avoiding duplicate computing; */
        if a is semantically equal to b in T
          $SPN \leftarrow SPN \cup \{a \leftrightarrow b\}$ //$\Re^{pi \leftrightarrow pj}$
  Return *SPN*

Algorithm2: *SSN* construction

Input: Services set S, an ontology T
Result: Graph *SSN*
/* Note that *SSN* can also be obtained by extracting paths only passing *I/O* concepts between services pairs in *SPN*. We use the following method because it has low complexity;*/
begin
  $SSN \leftarrow \emptyset$
  for each *service* $x \in S$ **do**
    for each *service* $y \in S$ **do**
      if $x \neq y$
        $x.O \leftarrow x.O \cup$ parent$(x.O) \cup$ ancestors$(x.O)$
        if $x.O \cap y.I \neq \Phi$
          $SSN \leftarrow SSN \cup \{x \rightarrow y\}$
      /* Similar to Algorithm1, We do not compute whether there is an "$y \rightarrow x$" edge for avoiding duplicate computing; */
  Return *SSN*

Network Optimization

Visualization and Basic SNA for Initial Networks

In this section, we track the two generated networks *SPN* and *SSN* by employing Pajek (de Nooy, Mrvar & Batagelj 2005), and use visualization to intuitively analyze the two networks, as shown in Figure 2. From Figure 2(right), the visualization of *SSN*, we can clearly witness a large body of isolated services. Some statistical findings about the two networks are summarized in Table 2.

Enrich graph-based web services network

Thus far we described how to construct original web service networks *SPN* and *SSN*, as well as some basic analysis about them. We now present how to enrich web service networks, and show its significance in practice.

SPN	# vertices—Total	1027
	# vertices—I/O concepts	237
	# vertices—Services	790
	#vertices—I/O concepts with 0 In-Degree	65
	# links	2504
SSN	# vertices	790
	# isolated vertices	97
	# links	13818

Table 2: Statistical findings of *SPN* & *SSN*. These findings are useful in optimizing networks.

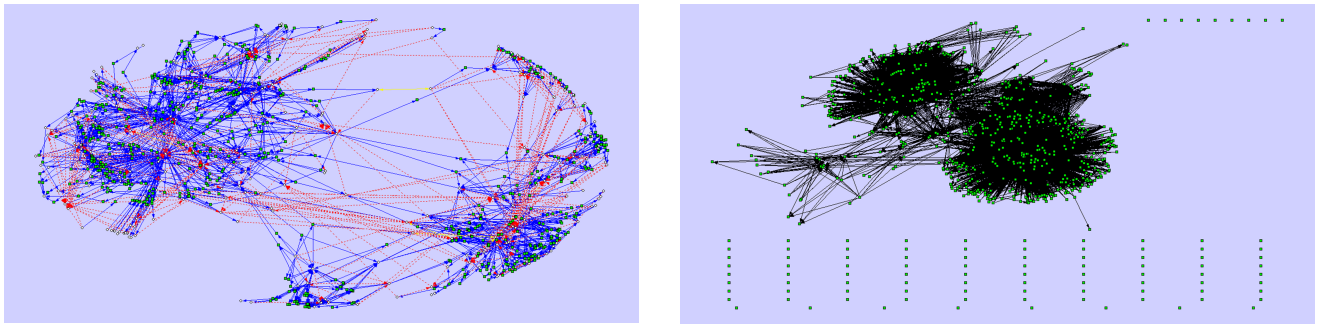


Figure 2: Visualizations of the two original networks: *SPN* (left), where green boxes denote services, white circles denote *I/O*s, blue directed lines indicate $\Re^{s \to p}$ and $\Re^{p \to s}$, red dotted directed lines indicate $\Re^{pi \to pj}$ and yellow bi-directed lines indicate $\Re^{pi \leftrightarrow pj}$; *SSN* (right), it contains only services and directed lines between services.

Empirically, we consider the problem of isolated web services in *SSN* was related to the degree of *I/O* concepts in *SPN*. Since *I/O* concepts maintaining a zero in-degree in *SPN* imply that there is no service provides them as outputs. Consequently, in *SSN*, there is no service directing to those services with only zero in-degree concepts as inputs in *SPN*. To verify above statement, we simply create a super service, wherein the outputs consist of all 65 zero in-degree concepts in *SPN*, and without loss of generality, the inputs are represented by a fair concept which both in-degree and out-degree are non-zero in *SPN*(here we use "*city*"). We perform some typical SNA on both original *SSN* and *SSN* by adding super service (*SSN*$^+$), as summarized in Table 3.

Moreover, we also evaluate *small world* effect about the two networks. (Watts & Strogatz, 1998) suggested that *small world* network should possess a small average distance, and a large clustering coefficient. In this paper, we follow the proximity ratio proposed by (Walsh, 1999) to measure *small world* effect:

$$\mu = \frac{C}{C_r} \frac{L_r}{L}$$

where C_r and L_r denote the average distance and average distance defined by a random network (same size) respectively. In service-based networks, roughly speaking, *small world* effect implies that the spread of dataflow among services is fast, and most pairs of services could be connected by a short path. This is significant for composing service, as we can make an estimate that composite services can be computed in limited steps.

From Table 3, there are two main improvements benefitting from adding the super service:

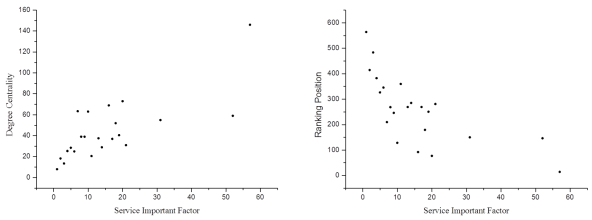
- Isolated services in *SSN* are connected in *SSN*$^+$;
- Resulting a better *small world* effect in *SSN*$^+$ than *SSN* (however, both of them struggle with weak *small world* effect, according the proximity ratio μ;

Therefore, adding such a super service can significantly enrich the web service network. However, it is impractical to construct such a super service. It is more reasonable in practice that web service network is dynamic and capturing a gradual increase with fine-grained functional services. We address this challenge by predicting valuable services which are much-needed to be added into network. To do so, we mentioned before that zero in-degree concepts in *SPN* might lead to isolated services in *SSN*. Moreover, when a zero in-degree concept in *SPN* also maintains a high out-degree, we say that the concept is crucial and much-needed. Thus, for every zero in-degree concept in *SPN*, we created a service, wherein the output is the corresponding zero in-degree concept, and the input is represented by a fair concept which both in-degree and out-degree are non-zero in *SPN* (here we still use "*city*"). To measure how important the new created services are, for each of them, we specify an *important factor* determined by the *out-degree* value of the corresponding zero *in-degree* concept in *SPN*. Then we add each new created service into *SSN* separately, thus we have 65 enriched *SSN*s (each has 791 vertices). We perform SNA on the 65 enriched *SSN*s to measure importance of the new added service.

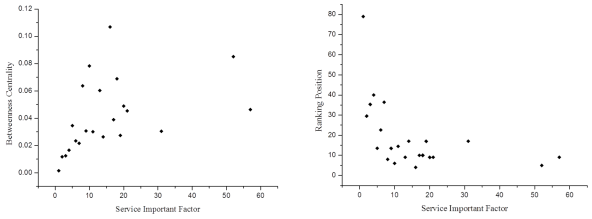
Three typical measures of centrality are studied: *degree centrality*, *betweenness centrality* and *closeness centrality*, as well as their ranking position in the corresponding networks, as shown in Figure 3. *Degree centrality* of a service in *SSN* is defined by its degree (For simplicity, degree value of nodes of *SSN* in this paper means total degree, i.e. both the in-degree and out-degree). Generally speaking, the higher of *degree centrality* a service maintains, it often holds a more active role and a more advantaged position. Rather than *degree centrality* which measures only local centrality of a service, services with higher *betweenness centrality* imply that they will more

Network	N	N^0	M	D	K	L	C	L_r	C_r	μ
SSN	790	97	13818	0.022	34.98	4.429	0.060	2.520	0.029	1.177
SSN$^+$	791	0	14528	0.023	36.73	5.122	0.102	2.618	0.024	2.172

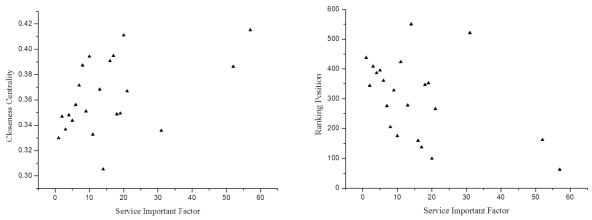
Table 3: SNA for *SSN* and *SSN*$^+$: ***N***—# vertices, ***N*^{0}**—# isolated vertices, ***M***—# arcs, ***D***—density, ***K***—average degree, ***L***—average distance, ***C***—clustering coefficient, ***L*$_r$**—average distance in random networks, ***C*$_r$**—average distance in random networks.



(a) *Degree centrality* (left) and its ranking position (right).



(b) *Betweenness centrality* (left) and its ranking position (right).



(c) *Closeness centrality* (left) and its ranking position (right).

Figure 3: *Social Centrality* vs. *Important Factor* of the new added services. Note that there are total 22 points in each figure, since some new added services maintain the same *important factor*, and we take the average result for them. Note that in the ranking position figures, the lower of its position value, the more important position it holds.

frequently reside on shortest paths between any other two services (shortest paths mostly represent portions of composite services, or even they are themselves composite services). *Closeness centrality* measures services' ability that how many other services it can reach through relatively shorter paths[3]. Figure 3 shows that new added services with a higher important factor will capture higher centrality, as well as rank a topper position in *SSN*s.

Discussion: In practice, web service networks usually possess dynamic evolutional features, old services are dying, and new services have been created by providers. We have presented how to predict crucial services which can significantly enrich the service-based network. In so doing, it can give advice to service providers for creating much-needed web services, which has been a great benefit to both web service networks and providers. We only study the *degree* aspect in *SPN*, as future work, we will focus on how to enrich the poor connective between components by analyzing more "global" measures.

Prune negative links using users' historical behavior.

As the example we showed in section 1, a crucial issue comes from the fact that there are a lot of negative links in web service network. In this section we describe how to identify and thus prune negative links based on users' historical composite services. The idea is briefly followed as: since composite services generated by graph-search based service composing can be specified in form of chains or sub-graphs. We decompose composite services into pieces as corresponding to links in *SSN*. Hence, we can learn positive links from correct composite services, and negative links from composite services abandoned by users (those failed to satisfy users' requirements). Concrete description of the proposed methods is presented below.

We first state a requirement that all the composite services here are generated based on graph-based service network. Let *CS* be a composite service. To represent a *CS*, we define two operators: ("•") denotes two services are sequentially composed, and ("+") denotes two services are composed in parallel. Also, the two operators qualify the following properties for any given three services *A*, *B* and *C*: $A+B=B+A$; $A•B\neq B•A$; $(A+B)•C=A•C+B•C$. We emphasize that any *CS* (neglecting the involved *I/O*s) can be represented by joining the involved services with ("•") and ("+") [4]. Note the two operators defined here slightly differ from the four operators in (Hashemian & Mavaddat 2006) due to the underlying nature of the based graphs.

We denote a smallest composite unit as an atomic composite service(*ACS*), which consists of only two single services and sequentially connected by ("•"). Apparently, an *ACS* is corresponding to a link in *SSN* (e.g. $A•B$ means $A\rightarrow B$ in *SSN*). To decompose a given *CS*, we firstly transform all its $(A+...+B)•C$ parts into $A•C+...+B•C$, thus we could obtain all *ACS*s by splitting *CS* with ("+").

Next, based on users' historical *CS*s, we show how to identify positive and negative *ACS*s. Four sets are used:

- A candidate positive *ACS* set (*CPACS*) for representing the candidate positive *ACS*s,
- A positive *ACS* set (*PACS*) for representing *ACS*s which have been identified as positive from *CPACS* by a positive threshold ($threshold_p$);
- A candidate negative *ACS* set (*CNACS*) for representing the candidate negative *ACS*s,
- A negative *ACS* set (*NACS*) for representing *ACS*s which have been identified as negative from *CNACS* by a negative threshold ($threshold_n$).

For each *ACS*, it is associated with a counter to identify that an *ACS* can be viewed as positive or negative, if its counter exceeds certain thresholds (thresholds are derived

[3]There are various explanations about the two centrality measures in different area. Methods of computing centrality in our work can refer to *Pajek* (de Nooy, Mrvar & Batagelj 2005).

[4]Interested readers can verify the two operators in the composite example of (Shin, Lee & Suda, 2009) by neglecting parameters.

from experience). The process of identifying *ACS*s is given in Algorithm 3[5]. By identifying *ACS*s, we can prune the negative links based on *NACS*. We argue that in practice this process should be dynamic and viewed from an evolutionary perspective.

Algorithm3: Identifying and pruning negative links from users' historical *CS*

Input: Graph *SSN, CS*, *CPACS*, *PACS*, *CNACS*, *NACS*
Result: Graph *SSN, CPACS*, *PACS*, *CNACS*, *NACS*

```
1  begin
2    ACS ← decompose(CS)
3    if CS is a successfully used composite service by user
4      for each ACS_i ∈ ACS do
5        if ACS_i ∈ CPACS
6          ACS_i.counter + 1;
7          if ACS_i.counter > threshold_p;
8            CPAC←CPACS-{ACS_i};
9            PACS ←PACS ∪ {ACS_i};
10       else
11         ACS_i.counter = 1;
12         CPACS← CPACS ∪ {ACS_i};
13   else
14     CNACS ←CNACS-( CNACS∩(PACS∪NACS);
       /*Update CNACS timely for somewhat avoiding
         innocent casualties; */
15     Tmp_ACS ←ACS - (ACS∩(PACS∪NACS))
16     for each ACS_i ∈Tmp_ACS do
17       if ACS_i ∈ CNACS
18         ACS_i.counter + 1;
19         if ACS_i.counter > threshold_n;
20           CNACS←CNACS -{ACS_i};
21           NACS← NACS ∪ {ACS_i};
22           Pruning ACS_i in SSN;
23       else
24         ACS_i.counter = 1;
25         CNACS← CNACS ∪ {ACS_i};
26   Return Graph SSN, CPACS, PACS, CNACS, NACS
```

To evaluate our methods, we have manually identified some *CS*s generated from results in our previous work (Guo, Chen & Feng 2011). We found the fact that some of the generated *CS*s could not meet users' real intent, though they were deemed as correct by the composing program. However, it required massive manual tasks and huge time consuming for generating synthetically user requirements scenarios as well as identifying *CS*s results by human, and it went far enough for the quantities covering the whole network for simulating users' historical data. Nevertheless, we designed an alternative investigation plan to evaluate the negative links in the underlying networks. Since identifying of a composite service will ultimately depend on its *ACS*s, i.e. each link in the network, we concocted a questionnaire with four colleagues to identify the 13818 links in *SSN*. The results demonstrate a serious situation that approximately 44% of the links are negative, as summarized in Figure 4, yet we find negative links also exist in real-life dataset.

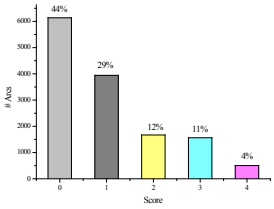


Figure 4: How bad is the situation? Each link in the questionnaire was represented by the begin service, destination service, the text description of the two services parsed from OWL-TC3, as well as parameters residing on the link. The score here represents the No. of participators who identify current link as positive. The result shows 44% links are deemed as negative by all four participators.

Discussion: As mentioned above, we have shown the seriousness of negative links in web service network, and presented a reasonable method to prune them. We emphasis that the serious status of negative links can still be avoid to a certain extent through advances in semantic techniques, such as combining the other parts of stateful semantic web services, e.g. the precondition and effect in the profile of OWL-S. However, there still challenges remain in the state of art semantic web services, and so far now, it remains a tough challenge for including stateful semantic web services in service-based networks. Our methods could be viewed an alternate solution to the problem by leveraging wisdom of the crowd.

In addition, real sense of "social" service networks can be extracted from the collected users' historical data (such networks are constructed in a "top-down" manner), thus interesting social features can be discovered by SNA for facilitating the state of art, such as community structure, clique structure. A relevant such "top-down" effort was studied in (Tan, Zhang & Foster 2010).

Related Work

There is a significant number of related works in the area of SOC. As mentioned above, we have detailed some typical graph-search based automatic service composing methods (Hashemian & Mavaddat 2005, 2006; Shin, Lee & Suda 2009). In this section, we briefly present literatures related to network analysis of web service network.

Due to advances in network analysis techniques, analysis of web service networks has gained a growing interest for facilitating SOC. (Cai 2007) proposed to form avatars (users) and WSDL-based web services within the community into a scale-free network. In (Kil, Oh & Lee 2006; Kil et al. 2009), the authors performed topological analysis on real-world web services networks, and

[5]We simply categorize abandoned *CS*s as incorrect, innocent links may also be pruned due to the fact that user may choose the best one by its QoS value among candidate correct *CSs*. Hence, a stricter $threshold_n$ should be determined. Also, we consider the QoS aspect as a future work.

demonstrated that their networks exhibited *small world* effect well and somewhat *power-law-like distribution*. However, further efforts were needed for strengthening services' semantics in matchmaking. Moreover, they also developed services discovery and composition methods in diverse and large-scale service-based networks (Oh, Lee & Kumara 2008).

In terms of SNA for reusing users' historical composite services, (Tan, Zhang & Foster 2010) studied SNA for network based on life-science workflows, and proposed a workflow reuse framework. The essential difference compared to our work is that we focus more on the sequence invoking relationships in composite services, rather than the collaborating features in (Tan, Zhang & Foster 2010). In addition, we consider reusing of *ACS*s of users' historical composite services, not reusing all-in-one-piece. Another work worth mentioned here is that (Yu & Woodard 2008) performed some remarkable network analysis on mashups ecosystem collected from Programmableweb.com. Mashups somewhat bear a resemblance to composite services(APIs), and can be viewed as abundant resources by sharing the wisdom of crowds in terms of software reuse, are gaining momentum by academia as well as industrial.

Conclusion and Future Work

In this paper we addressed the issue of optimizing web service network. To cope with the challenges of poor connectivity and negative links in web service network, we have proposed two methods: enrich connectivity by leveraging SNA techniques, as well as pruning negative links based on users' historical composite services. As discussed earlier, we argue that both the two methods have practical significance.

One limitation goes that only atomic web services were considered in this work, our ongoing work is focusing on real-world dataset. As future directions, from the lessons learned, we found that in *SPN*, services tend to gather into several central clusters consist of *I/O*s with high similarity, we will extend our network model into a more abstract level to enhance the scalability. Moreover, we will extend our network model into weighted graph through learning from users' historical data (based on the counter of *ACS*). This is very useful especially when designing interactive and dynamic composition.

Acknowledgements

We gratefully thank members of the IKSE team for their helps in this work: Hao, Y.; Lan, B.; Liang, Q.; Wan, P.; and Zhang, Z. We also thank semwebcentral.org for open access to their data. This work was supported by NSFC grant 61173155 and National High-Tech Research and Development Program of China grant 2007AA01Z130.

References

Cai, H. 2007. Scale-free Web Services. In *ICWS 2007*, 288–295.

de Nooy, W.; Mrvar, A.; and Batagelj, V. 2005. *Exploratory Social Network Analysis with Pajek*, Cambridge University Press.

Guo, Y.; Chen, S.; Feng, Z. 2011. Composition Oriented Web Service Semantic Relations Research. *In International Joint Conference on Service Sciences*, 69-73.

Hashemian, S.V.; and Mavaddat, F. 2005. A Graph-Based Approach to Web Services Composition. In *SAINT 2005*, 183-189.

Hashemian, S.V.; and Mavaddat, F. 2006. A Graph-Based Framework for Composition of Stateless Web Services. In *ECOWS 2006*, 75-86.

Kil, H.; Oh, S.; and Lee, D. 2006. On the Topological Landscape of Web Services Matchmaking. In *VLDB Workshop on Semantic Matchmaking and Resource Retrieval. CEUR*, 178, 19-34.

Kil, H.; Oh, S.; Elmacioglu, E.; Nam, W.; and Lee, D. 2009. Graph Theoretic Topological Analysis of Web Service Networks. *World Wide Web* 12(3): 321-343.

Martin, D.; and Domingue, J. 2007. Semantic Web Services: Past, Present and Possible Futures (Systems Trends and Controversies). *IEEE Intelligent Systems* 22(6).

Oh, S.; Lee, D.; and Kumara, S.R.T. 2006. A Comparative Illustration of AI Planning-based Web Services Composition. *ACM SIGecom Exchanges* 5(5): 1-10.

Oh, S.; Lee, D.; and Kumara, S.R.T. 2008. Effective Web Service Composition in Diverse and Large-Scale Service Networks. *IEEE Transactions on Services Computing* 1(1): 15-32.

OWL-S. 2004. Semantic Markup for Web Services. W3C member submission 22 November 2004.

Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F. 2007. Service-Oriented Computing: State of the Art and Research Challenges. *IEEE Computer* 40(11): 38-45.

Peer, J. 2005. Web Service Composition as AI Planning-a Survey. Technical report, University of St. Gallen, Switzerland.

Shin, D.; Lee, K.; Suda, T. 2009. Automated Generation of Composite Web Services Based on Functional Semantics. *Journal of Web Semantics* 7(4):332-343.

Sirin, E.; Parsia, B.; Wu, D.; Hendler, J.A.; and Nau, D.S. 2004. HTN Planning for Web Service Composition Using SHOP2. *Journal of Web Semantics* 1(4):377-396.

Tan, W.; Zhang, J.; and Foster, I. 2010. Network Analysis of Scientific Workflows: a Gateway to Reuse. *IEEE Computer* 43(9): 54-61.

Walsh, T. 1999. Search in a Small World. In *IJCAI '99*, 1172-1177.

Watts, D.J.; and Strogatz, S. H. 1998. Collective Dynamics of 'Small-World' Networks. *Nature* 393(4): 440–442.

Yu, S.; and Woodard, C.J. 2008. Innovation in the Programmable Web: Characterizing the Mashup Ecosystem. In *ICSOC Workshops*, 136-147.