A Bloom Filter-Powered Technique Supporting Scalable Semantic Service Discovery in Service Networks

[1]Jia Zhang, [1]Runyu Shi, [1]Weiyi Wang, [1]Shenggu Lu, [1]Yuanchen Bai, [1]Qihao Bao, [2]Tsengdar J. Lee, [3]Kiran Nagaraja, [3]Nimish Radia
[1]Carnegie Mellon University -Silicon Valley, USA
[2]Science Mission Directorate, NASA Headquarters, USA
[3]Ericsson Research, USA
{jia.zhang; runyu.shi; weiyi.wang; shenggu.lu; yuanchen.bai; qihao.bao}@sv.cmu.edu;
tsengdar.j.lee@nasa.gov; {kiran.nagaraja; nimish.radia}@ericsson.com

***Abstract*—As more and more reusable web services are published on the Internet, how to help users quickly identify appropriate candidate services has become an increasingly critical challenge. Most of the current research efforts on service discovery rely on syntax and semantics-based service matchmaking. In contrast, this paper presents a novel way of applying network routing mechanism to facilitate service discovery, featuring scalability and performance. Services annotated by Web Ontology Language for Services (OWL-S) are organized into a network based on semantic clustering. Virtual routers are created representing clusters, and Bloom Filters are generated for service routing. A service search request is thus transformed into a network routing problem to quickly locate semantic service cluster and in turn to candidate services. In addition, the deterministic annealing technique is applied to facilitate service classification in the network construction. Dynamic network adjustment is operated to ensure the search performance in the network. Empirical study over common testbed annotated in OWL-S is reported.**

Keywords—Service discovery, Bloom Filter, scalable service discovery, deterministic annealing

I. Introduction

The Services Computing techniques have initiated an evolution in Software Engineering and led to the era of the third-generation Software Engineering - Service Oriented Software Engineering (SOSE). On one hand, developers intend to identify reusable software services and leverage them as components to build more comprehensive systems, thus resulting in faster-to-market and higher system reliability. On the other hand, developers are trained to build software as a service, so that the established system will have higher and finer-grained reusability and interoperability.

Such software reusability and interoperability are extremely important in the scientific world for knowledge sharing. For example, NASA Earth scientists have developed many data analytics algorithms over the past 40 years. To avoid from recreating the wheels, researchers shall leverage these programs either as components unchanged or repurposed for new experiments. Services computing techniques allow researchers to wrap up algorithms with programmable APIs for other researchers to remotely invoke, which is why researchers claim that the science nowadays has entered the era of Service Oriented Science [1].

However, as more and more reusable data services are published on the Internet, how to help users quickly identify appropriate candidate services has become an increasingly critical challenge. Most of the current research efforts on service discovery rely on syntax and semantics-based service matchmaking [2] between the requirements and the descriptions of service candidates. However, one major issue of these approaches is the scalability. When a large amount of service candidates exist to compare, the computational cost at runtime will be significant [3].

As Web services become more mainstream, very soon service discovery performance will become the bottleneck to hinder the extensive adoption of SOSE. The need to ensure scalable service discovery requires methodologies that are beyond the current state-of-the-art in this field. Therefore, it is promising if researchers start to explore at present how to model and enhance service discovery, so that related techniques can be investigated in the right context. Instead, if we wait until the time that service discovery scalability becomes an obstacle, some techniques might have to be reexamined and accommodated, which wastes valuable human resources.

In our previous work, we have modeled software services and their relationships into a service-oriented network [4]. In this paper, we propose a novel way of applying network routing mechanism to facilitate service discovery in service networks, featuring scalability and performance. The Bloom Filter technique (BF) [5] is applied to enable rapid service discovery. A Bloom Filter refers to a space-efficient hash-based, probabilistic data structure, coined by Burton Howard Bloom to support membership queries in database applications since 1970s [5]. In recent years, Bloom Filters have been widely used in networking routing due to its unique space saving feature [6].

In our service network, services are organized into clusters, analogous to machines into local networks. The root node of a cluster is analogous to the router of a local network. Bloom Filters are generated for all service nodes, based on the information carried by the services. Virtual routers are created based on service clustering to expedite service discovery. BF addresses are assigned to virtual routers as well. Thus, a service discovery request is transformed into a network routing problem aiming for

978-1-5090-2675-3/16 $31.00 © 2016 IEEE
DOI 10.1109/ICWS.2016.111

quickly locating semantic service cluster and in turn to candidate services.

One core criterion to impact the efficiency of applying Bloom Filter while remaining endurable false positive is the encoding mechanism of BF addresses. In contrast to other researchers who use BF to carry non-functional attributes with limited numbers [7], we strive to encode semantic data of functional information into BF. Therefore, we adapt the deterministic annealing method [8] to support hierarchical service clustering as well as to support hierarchical BF encoding for higher performance.

Semantic service discovery heavily relies on formal semantics of service specification [9]. Web Ontology Language for Services (OWL-S, https://www.w3.org/Submission/OWL-S) is an *ad hoc* standard language endorsed by W3C. Thus, in our research, we study deterministic annealing classification and Bloom Filter encoding based on OWL-S annotated service specifications. In addition, semantic structure of OWL-S is leveraged to facilitate service clustering and enhance its precision.

We have designed and conducted a series of experiments to evaluate the effectiveness and efficiency of our approach. The major contributions of our paper are two-fold.

First, we propose a hybrid Bloom Filter technique to facilitate service discovery and recommendation in a service network. Dynamic network adjustment is also applied to ensure network search performance.

Second, we apply the deterministic annealing method and leverage the structural information of OWL-S to support the construction of a service network based on service classification, to save bits used to build Bloom Filter addresses.

The remainder of the paper is organized as follows. In Section 2, we discuss related work. In Section 3, we describe the hierarchical service network construction. In Section 4, we present Bloom Filter generation algorithm. In Section 5, we discuss experimental design and analysis. In Section 6, we draw conclusions.

II. Related Work

Our previous work has developed knowledge networks based on People, Data, Workflow, Service (PDWS) [10]. How to quickly find and recommend a service in this knowledge network directly motivates the reported research in this paper.

Bloom Filter has been extensively applied in networking routing because of its significant space saving feature [6]. Researchers have adopted Bloom Filter to identify nodes in multihop mobile ad-hoc network [11]. Chen et al. [7] leverage BF to encode non-functional attributes, in order to support service discovery inside of pre-classified service groups. In contrast to their work, we apply BF to encode functional keywords that poses significant challenges since the number of attributes to be encoded is much larger. We leverage deterministic annealing and inherent structure of OWL-S to eliminate the issue. In addition, we use BF to drive the clustering and construction of service network.

In recent years, researchers have started to leverage deterministic annealing technique in Physics to support data clustering and classification [8]. Zhou et al. [12] apply deterministic annealing to cluster websites based on their carrying tags. Liu et al. [13] apply deterministic annealing method to cluster services using their tags. Our work differs from their works in two aspects. First, we take into consideration of all terms in service descriptions instead of only tags. Since a deterministic annealing process mainly relies on statistics of term usages in documents, we believe data sparsity problem using more detailed information of services (i.e., rich description) that carries richer correlations among terms. Second, we apply deterministic annealing process iteratively to create a hierarchical network structure from services. Each iteration will yield a leading service representing the corresponding service cluster.

Numerous researchers have worked on syntax and semantics based service discovery. Example of syntax-based service discovery includes performing profile-based service signature (I/O) matching. Semantics-based OWLS-MX [9] and WSMO-MX [14] propose to combine logic-based reasoning and syntactic concept similarity computations in OWL-S. Sbodio et al. [15] propose to use SPARQL as a formal language to describe the pre- and post-conditions of services. Junghans et al. [16] propose a practical formalism to describe functionalities and service requests. In contrast, we leverage service functional profiles and OWL-S structures to enhance service discovery.

Researchers have studied various machine learning techniques to facilitate automatic service discovery. Cassar et al. [17] apply probabilistic machine learning techniques to extract latent factors from service descriptions to construct a uniform service model in a vector form. Li et al. [18] propose a probabilistic approach based on Latent Dirichlet Allocation (LDA) to support service discovery, which leverages latent topics to link services with terms extracted from WSDL documents. Chen et al. [19] integrate WSDL documents and user tagging data and propose a user tagging augmented LDA model for service clustering. Zhong et al. [20] extract service evolution patterns by exploiting LDA and time series prediction. Unsupervised LDA is applied to calculate the similarity between services. They present a time-aware service recommendation framework taking into consideration of temporal information, content description and historical mashup-service usage in an evolving service ecosystem. In contrast to their works, we apply deterministic annealing to facilitate service clustering for the purpose of BF-oriented network construction.

III. System Modeling

Semantic service discovery counts on formal semantics of service specification [9]. Web Ontology Language for Services (OWL-S) is widely considered as an *ad hoc* standard language, endorsed by W3C, to describe web

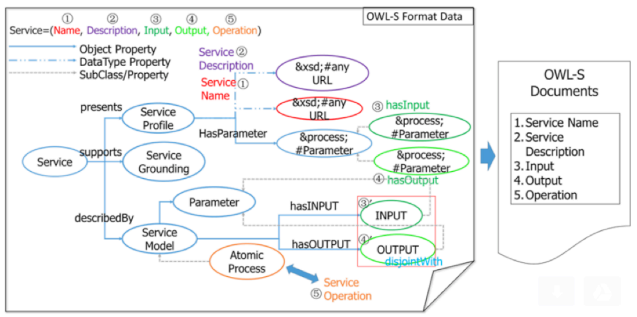


Fig. 1 OWL-S service document.

services. OWL-S annotates web services in order for them to be machine understandable and easily accessible in the Internet. OWL-S also enables effective reasoning of intelligent application based on ontology statement and makes complex tasks possible such as automatic web service discovery, invocation, composition and interoperation. We view OWL-S files as structured documents.

Definition 1: An OWL-S file is a document containing three segments: OWLS file = (ServiceProfile, ServiceGrounding, ServiceModel), where each segment comprises a collection of words, and ServiceProfile = (Name, Description, Parameters), ServiceModel = (AtomicProcess, Parameters), and Parameters = (input, output). Information of AtomicProcess can be used as service operations. The Parameters in ServiceModel are the same as those in ServiceProfile.

We closely study OWL-S structure and build service networks based upon OWL-S annotated services. As shown in Fig. 1, in ontology of services, OWL-S annotates three types of knowledge: "Service Profile" that indicates what one service provides for clients, "Service Model" that illustrates how one service is used and "service grounding" that tells clients how to interact with it. While an OWL-S model comprises ServiceProfile, ServiceGrounding, and ServiceModel, semantic information about services are mostly stored in the ServiceProfile section. As shown in Fig. 1, data in the "Service Grounding" and "Service Model" sections are mostly in the form of URI, which will be used in our future work to reason about service usage context. Major metadata includes service name, description, input and output parameters, and atomic process (i.e., service operation). We thus define a service node in a service network as follows.

Definition 2: An OWL-S powered web service document is a document containing six segments: Service = (Name, Description, Operation, Input, Output), where each segment comprises a collection of words.

A. *OWL-S Supported Service Similarity*

Based on the static semantics extracted from OWL-S descriptions, we study how to automatically divide services into clusters. Each service is modeled as a document. The content of a service is the service metadata used to define the profile of the service, conceptually modeled as a multidimensional feature vector. All words (normalized names) used in all service profiles establish the corpus ($\bigcup_{j=1}^{|S|} \forall w_i \in s_j$). The Porter stemming algorithm [21] is used for prefix and affix removal and Wordnet (http://www.wordnet.princeton.edu) for partially solving the synonym issue. Applying the term frequency–inverse document frequency (TF-IDF) algorithm [22], we calculate the weight $v_{j,i}$ of every word $w_{j,i}$ inside of the profile (sp) of every service s_i. It indicates the importance of the word representing the semantics of the service.

$$tf\_rdf(v_{j,i}) = tf_{j,i} \times idf_j = \frac{C_{j,i}}{\sum_k c_{k,i}} \times lg \frac{|S|}{1 + |\{s: w_{j,i} \in sp(s)\}|}$$

Where $tf_{j,i}$ is the term frequency (obtained by the number of occurrences of the word w_i in the profile of service i, divided by the size of the profile of the service to ensure that $tf_{j,i} \in [0,1]$); idf_j *is the* general importance of the word (obtained by dividing the total number of services by the number of services containing the word in their profiles, and taking the logarithm of the resulting quotient).

The concept relatedness between words is broader than that of similarity; we consider the latter for simplicity. Based on our previous work [23], we leverage the shortest path-based LCH algorithm [24] to measure the semantic similarity between two words based on the lexical database WordNet.

$$sim(w_i: ST, w_j: ST) = lch(w_i, w_j)$$

As aforementioned, one dimension of feature of a service is a collection of words. Similarity over a feature of two services could be turned into comparison of two word lists X and Y: $X = \{x_1, x_2, \ldots x_m\}, Y = \{y_1, y_2, \ldots y_n\}, m \geq 1, n \geq 1$. $\forall x_i \in X, y_j \in Y$ as modeled in a weighted complete bipartite graph $G = (V = (X, Y), E)$. An edge $(x_i, y_j) \in E$ exists, with a weight of $sim(x_i, y_j)$. Calculating the similarity between the two word lists is turned into finding an optional match, where the sum of the weights of the edges in the matching reaches a maximal value:

$$sim(sp_l(s_1), sp_l(s_2)) = \max \left(\sum_{k=1}^{\min(|X|,|Y|)} sim(x_{M1(k)}, y_{M2(k)}) \cdot tf\_idf(x_{M1(k)}) \cdot tf\_idf(y_{M2(k)}) \right)$$

where M1 and M2 are two mapping functions, each selecting the number of min(|X|,|Y|) elements from the collections X and Y, respectively:

$$M1: k \rightarrow x_{M1(k)}, \forall k1 \neq k2, M1(k1) \neq M1(k2);$$
$$M2: k \rightarrow y_{M2(k)}, \forall k1 \neq k2, M2(k1) \neq M2(k2);$$

In our project, we apply the Hungarian algorithm [25] to find the optimal match, with a cost of O(V^2E).

Based on the structure of a service profile inherited from

OWL-S structure, we developed a similarity computation algorithm for comparing the profiles of two services:

$$\begin{aligned} sim(s_i, s_j) = \alpha_1 \cdot sim(s_i.name{:}ST[], s_j.name{:}ST[]) \\ + \alpha_2 \cdot sim(s_i.desc{:}ST[], s_j.desc{:}ST[]) \\ + \alpha_3 \cdot sim(s_i.in{:}MSG, s_j.in{:}MSG) \\ + \alpha_4 \cdot sim(s_i.out{:}MSG, s_j.out{:}MSG) \\ + \alpha_5 \cdot sim(s_i.atomic{:}ST[], s_j.atomic{:}ST[]) \end{aligned}$$

where $\sum_{k=1}^{5} \alpha_k = 1$. The coefficients indicate that different weights may be assigned to different features of a service profile.

When similarity factor is calculated over pairs of services annotated in OWL-S, a similarity matrix is generated for all services. If the similarity between a pair satisfies $sim(s_1, s_2) \geq \gamma$, where γ is a preset value (e.g., 0.8), they can be put into the same service cluster.

B. Deterministic Annealing for Service Clustering

After service similarity comparison algorithm is decided, the next step is to classify services and build service networks. Although being simple, bottom-up clustering methods such as the hierarchical clustering approach focuses on local patterns without initially taking into account the global distribution. Thus, we adopt a top-down deterministic annealing [8] method to escape local minima of the given cost function and to build the network topology of the service network. In statistical physics, annealing refers to a process of finding the most probable set of element cluster representatives (with minimum free energy combined), by heating to above the re-crystallization temperature and gradually cooling down. In other words, deterministic annealing optimization could be viewed as a process of minimizing a predefined objective function at isothermal, stochastic equilibrium. Recent machine learning researchers have applied deterministic annealing as a clustering technique [8]. In our context, we apply the concept of deterministic annealing to build hierarchical networks among services.

Applying deterministic annealing in our context, the purpose is to gradually find a stable state when services are grouped into clusters. When two services are similar, they tend to be put into the same cluster to lower the free energy that can be defined as inverse proportion to their similarity. As described in Section A, services are modeled as documents each carrying multiple sections that reflect multidimensional feature vectors. We build objective function over word usages based on both of their statistical and structural analysis.

As explained in Section A, the feature vectors of services are built on top of their associated OWL-S files, the comprising words carry structural semantics. For example, a word "trip" extracted from the service name, description, and atomic service (i.e., operation) sections imply different structural meanings. We thus amend a probability calculation according the OWL-S features. A five-tuple O = (N, Q, IN, OUT, OP) represents all OWL-S keywords, O = $(o_1, o_{2,\ldots,} o_n)$, o_i represents OWL-S keywords of service s_i.

N: names of the OWL-S services. Set N consists of $(n_1, n_{2,\ldots,} n_n)$, n_i represents the name of OWL-S service s_i.

Q: descriptions of the OWL-S services. Set D consists of $(q_1, q_{2,\ldots,} q_n)$, q_i represents the description of OWL-S service s_i.

IN: input parameters of the OWL-S services. Set IN consists of $(in_1, in_{2,\ldots,} in_n)$, in_i represents the input parameters of OWL-S service s_i.

OUT: output parameters of the OWL-S services. Set OUT consists of $(out_1, out_{2,\ldots,} out_n)$, out_i represents the output parameters of OWL-S service s_i.

OP: operations provided by the OWL-S services. Set OP consists of $(op_1, op_{2,\ldots,} op_n)$, op_i represents the operations provided by OWL-S service s_i.

We introduce the following four conditional probability variables to be used in the deterministic annealing process. A_{ij} refers to the probability of one OWL-S keyword being classified into a given cluster; B_{ij} refers to the probability of cluster containing a given OWL-S keyword; X_{ij} refers to the probability of one OWL-S keyword appears in a given service; Y_{ij} refers to the probability of a service containing a given OWL-S keyword.

$$A_{ij} = p(o_i|c_j) = \sum_{w_k}^{W} \sum_{o_i}^{O} (w_k * p(o_i|c_j))$$

$$B_{ij} = p(c_i|o_j) = \sum_{w_k}^{W} \sum_{o_i}^{O} (w_k * p(c_i|o_j))$$

$$X_{ij} = p(o_i|s_j) = \sum_{w_k}^{W} \sum_{o_i}^{O} (w_k * p(o_i|s_j))$$

$$Y_{ij} = p(s_i|o_j) = \sum_{w_k}^{W} \sum_{o_i}^{O} (w_k * p(s_i|o_j))$$

where W = $(w_n, w_q, w_{IN}, w_{OUT}, w_{OP})$ represents the feature weight of name, description, input parameters, output parameters, operations of OWL-S service, which can be initialized as $(0.3, 0.3, 0.1, 0.1, 0.2)$.

We define an objective function as follows:

$$F = D - TH$$

where free energy F and entropy H are two terms in the physical annealing theory; temperature T controls entropy H in different scales during the minimization of F.

The goal of the annealing process is to minimize F, the free energy.

$$D = \sum_{i=1}^{N} \sum_{j=1}^{C} p(c_j|s_i) * d(s_i, c_j)$$

where conditional probability $p(c_j|s_i)$ measures the relativity between service s_i and cluster c_j ; $d(s_i, c_j)$ refers to the distance of service s_i to cluster c_j, which can be calculated by adapted KL-divergence distance algorithm

[26].

$$d(s_i, c_j) = \sum_{i=1}^{M} Y_{ij} * \log\left(\frac{Y_{ij}}{p(s_i|c_j)}\right)$$

where s_i refers to the ith service.

H is a measure of the level of randomness given below:

$$H = -\sum_{i=1}^{N}\sum_{j=1}^{C} B_{ij} * \log\,[B_{ij}]$$

The method recursively applies a split annealing process to each word group until termination conditions are satisfied. The termination condition for this algorithm is when a critical temperature is reached and all the clusters become effective clusters. Critical temperature is determined as follows:

$$T = 2\lambda_{max}$$

where $\lambda_{max}$is the largest eigenvalue of $C_{c|o}$, which is the covariance matrix of the posterior distribution of the cluster corresponding to O:

$$C_{c|o} = \sum_{x} B_{ij}\,(c_j - n_i)(c_j - n_i)^t$$

For each cluster, the goal is find the split until it is considered an effective cluster. The criterion of judging an effective cluster is through a coverage function as below:

$$cov(o_i, c_j) = \sum_{k=1}^{N} A_{ij} * b_{i,k}$$

where $b_{i,k} \in \{0,1\}$ indicates where there exists a service comprises both keywords o_i and o_j. The Bayesian Theorem can be used to calculate $p(c_i|o_j)$. The higher the cov value means the keyword o_i covers many other keywords in cluster c_j.

In the clustering process,

$$E(c_j) = max\; cov(o_i, c_j)$$

where $E(c_j)$ measures whether c_j is an effective cluster. Once $E(c_j)$ reaches one highest value, it means that the leading keyword set o_i has emerged, then the cluster is considered an effective cluster.

Alg. 1. Deterministic Annealing for Service Clustering
Input: A set of services annotated with OWL-S
Output: The probability of cluster distribution over services

```
1. Initialize C, T, $F^{(0)}$
2. loop
3.   repeat
4.     $k \leftarrow k + 1$
5.     Calculate ${B_{ij}}^{(k)}$ with ${B_{ij}}^{(k-1)}$
6.     Calculate $F^{(k)}$
7.   until $|F^{(k)} - F^{(k-1)}| < \varepsilon$
8.   record ${B_{ij}}^{(k)}$
9. if isEffectiveCluster(all clusters)
10.   return ${B_{ij}}^{(k)}$
11. end if
12. if Critical Temperature for cluster $c_j$ is reached
13.   ${B_{i+1j}}^{(k)} \leftarrow \frac{{B_{1j}}^{(k)}}{2} + \delta$ , ${B_{ij}}^{(k)} \leftarrow \frac{{B_{1j}}^{(k)}}{2} - \delta$ , C←C+1
14. else
15.   $B_{ij} \leftarrow {B_{ij}}^{(k)}$
16. end if
17.   $T = \alpha T\ (0 < \alpha < 1)$
```

Expectation-Maximum (EM) algorithm is utilized to minimize the free energy F. The pseudo code is illustrated as Alg. 1.

The deterministic annealing equation ( ) can be recast as:

$$F = \sum_{i=1}^{N}\sum_{j=1}^{C} B_{ij} * \left(\sum_{l=1}^{M} Y_{lj} * \log\left(\frac{Y_{lj}}{B_{lj}}\right) + T * \log\,(B_{ij})\right)$$

For the kth iteration of $p^{(k)}(c_j|s_i)$,

$${B_{ij}}^{(k)} = \frac{\exp\left(-\frac{d^{(k)}(o_i, c_j)}{T}\right) * p^{(k)}(c_i)}{\sum_{l=1}^{C} \exp\left(-\frac{d^{(k)}(o_i, c_j)}{T}\right) * p^{(k)}(c_j)}$$

where:

$$p^{(k)}(c_j) = \sum_{j=1}^{N} {B_{ij}}^{(k-1)} * p(o_i)$$

$${A_{ij}}^{(k)} = \frac{\sum_{j=1}^{N} {A_{ij}}^{(k-1)} * p(o_j) * Y_{ij}}{p^{(k)}(c_j)}$$

$$d^{(k)}(o_i, c_j) = \sum_{l=1}^{M} Y_{lj} \log\left(\frac{Y_{ii}}{p^{(k)}(s_l|c_i)}\right)$$

where $p(c)$ denotes the probability that the cluster is assigned to keyword o; $p(o)$ denotes that the probability that keyword o occurs in a service. EM will iterate until F converges to a minimum.

IV. Hierarchical Bloom Filter-Based Service Discovery

A. Modeling Services in Bloom Filter

A Bloom Filter is a space-efficient hash-based, probabilistic data structure to support membership queries in database applications [5]. Bloom Filters allow false positives but the space savings often outweigh the drawback when the probability of an error is controlled.

A Bloom Filter represents a set

$$S = \{s_1, s_2, .., s_n\}$$

which contains n elements that are encoded into an array of m-bit network addresses, resulting from k independent hash functions $H_1, H_2, \ldots, H_k$.

The hash functions map each element into a unique random number uniform over the m bits. For example, the hash function $H_1(s_1)$ sets the bits $h_1(s_1)$ to 1 for element s_1. Afterwards, in order to query whether element y is in set S, all the $h_i(y)$ bits will be checked. If all corresponding bits are set to 1, it indicates that y is in set S. The process is illustrated by an example described below.

From our service OWL-S data, we select a bank branch search example to illustrate our method. Earlier section explains how to cluster all the services into clusters according the OWL-S documents. There is a Citi Bank finder service, which is put in a sub-cluster under the "Bank Finding" cluster. For this specific Citi bank finding service cluster, its keywords set, extracted by Deterministic Annealing is as follows:

$$s_l = \{Bank, ATM, Citi, Branch\}$$

As shown in Fig. 2(a), its address bits are first initialized as all zeros. Hashing with Bloom Filter, the corresponding bits of the four keywords in the bit array are marked as "1" as shown in Fig. 2(b). For instance, adding word *"Bank"* to Bloom Filter, its Bloom Filter address is encoded as *"01001000010000."* Upon receiving a query like: *"Find a Citibank service,"* a keywords set will be extracted as:

$$s_y = \{Citi, Bank\}$$

The two elements, Citi and Bank, will be hashed with $H_i(s_y)$ to certain bits to check if those bits are "1" (Fig. 2(c)).

As shown in the example, BF uses a dense hash-table-like data structures for storing and testing set membership. Rich service semantic information can be encoded and embedded into hashed address bits. Applying BF techniques, service discovery can be released from highly time-consuming semantic match-making at runtime. Instead, Bloom Filters can significantly reduce the bandwidth requirements of epidemic protocols, which become the basis of our proposed solution for service discovery.

B. Modeling Service Network in Bloom Filter

We have adopted the directory tree structure and multi-scale Bloom Filter into our model. More formally, we have defined a hierarchical Bloom Filter model as a 4-tuple: a set of hierarchical keywords, a set of hash functions, the capacity and error rate.

$$HBF = < W, c, e, H >$$

Keywords: W represents the keywords which are extracted from OWL-S data. The OWL-S data contains structure information, for each type of information, after the clustering, each service will be added into the Bloom Filter network with unique address.

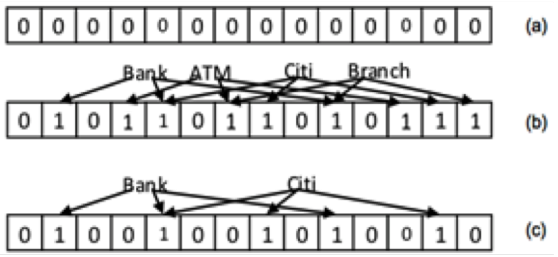


Fig. 2 OWL-S service profiling.

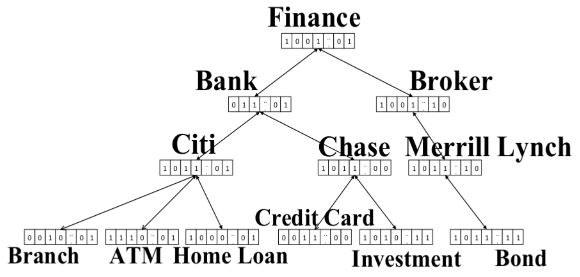


Fig. 3 Bloom Filter-powered service network.

Capacity: capacity c is calculated based on the number of the extracted data from OWL-S and used to decide bit array length.

Error rate: Along with capacity, error rate e is used to define the bit array length of Bloom Filter. In our model, we use stand error rate as initial of 0.1%.

Hash functions: The set of hash functions H is selected from stand hash function library, decided by the c and e. Note that there is a tradeoff between the size of Bloom Filters and the probability of false positive. Increasing the size of Bloom Filter could lower the probability of false positive, with the price of storage and computation.

Thus, the decision of the initial address bit number (M) can be calculated as follows, based on the number of slices as k, bit per slices as m, error rate as P, and capacity as k.

$$k = log2(1/P)$$
$$n \cong (k * m) * (ln(2)^2 / abs(ln(P)))$$
$$m \cong n * abs(ln(P)) / (k * ln(2)^2)$$
$$M \cong (ln(2)^2) / abs(ln(P))/n)$$

We thus apply the Bloom Filter technique to encode the addresses of a service network. Each service is modeled as a node in the network, whose address is represented by a bloom filter address. As shown in Fig. 3, services are organized into clusters, analogous to machines into local networks. Meanwhile, the root node of a cluster is analogous to the router of a local network. Note that such a root node may be either a virtual node if using common hierarchical clustering methods, or be a leading service if using the deterministic annealing approach as discussed in the previous section. Fig. 3 illustrates an example of BF-powered service network including the services described in Fig. 2. The BF address of the Citi node in Fig. 3 can be found using the strategy described in Fig. 2.

As the first step, services are organized as a tree-like structure. In different layers, the numbers of BF address bits are different. For each layer, the decision is based on how many bits are sufficient to differentiate all nodes. In case a cluster contains many leaf nodes, we choose not to use BF since the number of BF bits will be highly significant. Instead, semantic match-making methods will be adopted. As shown in Fig. 3, searching a service is transformed into finding an address in the network.

C. Network Service Search and Network Scaling

After a service network is constructed, every service in the network is encoded into a BF address. An incoming service query is also translated into a BF address. Service discovery is turned into a process of finding matching BF address for the requested BF address. Starting from the root, in each layer of the network, when the most matched BF address is found and the search will move down to its sub-network, until finding the exact matched node or reaching the leaf level. After reaching a leaf level, all the leaf nodes under the selected cluster will be checked until the top matching services are selected as recommendation candidates.

Once there are incoming new services, the number of leaf nodes will be increased, which may effect search performance. Leveraging the Continuum theory [27], our model predicts scaling functions and then fits the predictions into connectivity distribution to describe the growth of the service network. Once the connectivity is larger than a predefined threshold value, to remain search performance, the whole BF network will be re-computed to generate a new network with more clusters and less leaf nodes. In the remainder of this section, we will discuss the condition when the re-computation should be conducted.

In the Continuum theory [27], the probability that a node i will increase its connectivity k depends only on k_i and quantities characterizing the whole network, such as the number of the nodes and links. In this section, we discuss the prediction of network scaling. When a new service (i.e., node) is added to the network, links will be added or rewired as a consequence.

Assume m new links are added into the service network with probability p, the change ratio of the connectivity is:

$$(\frac{\partial k_i}{\partial t}) = pA\frac{1}{N} + pA\frac{k_i + 1}{\sum_j (k_i + 1)}$$

$where\ N$ refers to the number of nodes. Since the total change in connectivity after a step is $\Delta k = 2m, A = m$.

Consider rewiring of m links with probability q, the change ratio of the connectivity is:

$$\left(\frac{\partial k_i}{\partial t}\right) = -pB\frac{1}{N} + pB\frac{k_i + 1}{\sum_j (k_i + 1)}$$

The total connectivity does not change during the rewiring process, but B can be calculated by separating the two processes, obtaining $B = m$.

While adding a new node with probability $1 - p - q$, the change ratio of the connectivity is:

$$\left(\frac{\partial k_i}{\partial t}\right) = (1 - p - q)C\frac{k_i + 1}{\sum_j (k_i + 1)}$$

The number of links connecting the new node to the existing nodes in the system is m, thus $C = m$.

The connectivity distribution $P(k)$ can be determined analytically. Defining the unit of time in the model as one growth/rewire/new link attempt, the probability density of t_i is $p_i(t_i) = \frac{1}{m_0+t}$, thus the probability of the connectivity of the *i*th cluster is smaller than *k* is:

$$p[k_i(t) < k] = 1 - (\frac{m + (p-q)(\frac{2m(1-q)}{1-p-q}) + 1}{k + (p-q)\left(\frac{2m(1-q)}{1-p-q}\right) + 1})^{\frac{2m(1-q)+1-p-q}{m}}$$

Using $\mathrm{p(k)} = \frac{\vartheta p[k_i(t)<k]}{\vartheta k}$, we can generate:

$$p(k) = \frac{t}{m_0 + t}\ [\mathrm{m} + \left[(p-q)\left(\frac{2m(1-q)}{1-p-q}\right) + 1\right]^{\frac{2m(1-q)+1-p-q}{m}} \times [\mathrm{k} + (p-q)\left(\frac{2m(1-q)}{1-p-q}\right) + 1]^{-1-(p-q)\left(\frac{2m(1-q)}{1-p-q}\right)}$$

Thus the connectivity distribution, the main result provided by the continuum theory, shows a generalized power-law form:

$$p(k) \propto [k + (p-q)(\frac{2m(1-q)}{1-p-q} + 1) + 1]^{-1-\frac{2m(1-q)+1-p-q}{m}}$$

Therefore, when a new service joins the network, if our predicted probability of connectivity reaches a threshold (assume $p(T) > \varphi$ when k reaches a connectivity threshold *T*), the BF network will be re-computed to generate a new network.

V. Experiments

A. Experimental Setup

Our test bed is the OWL-S service retrieve test collection version 4 (OWLS-TC4) (http://projects.semwebcentral.org/projects/owls-tc/). The service collection includes annotated OWL-S files of 1,083 web services, which were originally set up for evaluation of the performance of OWL-S-based semantic Web service matchmaking algorithms. The service set covers services of nine domains where 286 services come from the education domain, 73 from medical care, 34 from food, 197 from travel, 59 from communication, 395 from economy, 40 from weapon, 60 from geography, and 16 from simulation. While an OWL-S model comprises ServiceProfile, ServiceGrounding, and ServiceModel, semantic information about services are mostly stored in the ServiceProfile section. Data in the “Service Grounding” and “Service Model” sections are mostly in the form of URI, which will be used in our future work to reason about service usage context. Major metadata includes service name, description, input and output parameters, and atomic process (i.e., service operation). We parsed and analyzed the OWL-S files with Java programs utilizing XQuery. In our experiment, we leveraged the OWL-S structure to facilitate the construction of the layered BF network: names, descriptions, input and output messages, and atomic operations.

We have designed a collection of experiments to evaluate the effectiveness and efficiency of our Deterministic Annealing Dynamic Bloom Filter (DADBF) approach. All experiments were conducted on a Dell PowerEdge Server R530 with 8-core CPU and 96GB memory, on Ubuntu 14.04 system. The network bandwidth service is 100Mbps.

B. Performance Study

We evaluated the performance of our Deterministic Annealing-supported, dynamic Bloom Filter network (DADBF) approach supporting service discovery. The evaluation metrics that we used are Mean Average Precision (MAP) and Mean Average Recall (MAR).

Precision is the fraction of retrieved services that are relevant:

$$Precision = \frac{Relevant\ recommendation}{Total\ generated\ Recommendation} = \frac{Ture\ Positives}{Ture\ Positives + False\ Positives}$$

Recall is the fraction of relevant services that are actually retrieved:

$$Recall = \frac{Relevant\ recommendation}{Total\ existing\ Recommendation} = \frac{Ture\ Positives}{Ture\ Positives + False\ Negatives}$$

Assume that each ranked recommendation lists up to position n. The average precision for one user is:

$$ap@n = \sum_{i=1}^{n} P(k)/n$$

The mean average precision for N users at position n is the average of the average precision over all users:

$$MAP@n = \sum_{i=1}^{N} ap@n_i/N$$

Assume that each ranked recommendation lists up to position n. The average recall for one user is:

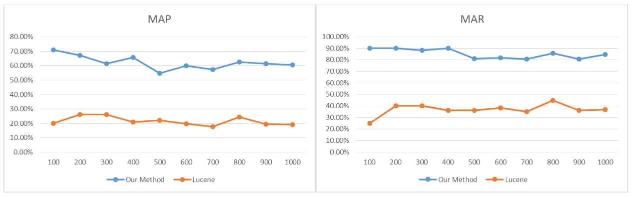
$$ar@n = \sum_{i=1}^{n} R(k)/n$$

The mean average recall for N users at position n is the average of the average recall over all users:

$$MAR@n = \sum_{i=1}^{N} ar@n_i/N$$

We compared our method with Lucene, the known open-source indexing and search algorithm. pylucene, a python wrapper for the Lucene used in Internet search engines and local searching, is compared in our experiment. The different OWL-S semantic layers are set as corresponding search fields in Lucene.

We split the dataset in 80:20 as 80% of train data and 20% of test data. To avoid bias, we randomly shuffled the train data and test data every time before testing. The experiment was repeated for 1,000 times, and we computed the arithmetic mean of all resulting numbers. Fig. 4 shows when the service number changes, how the MAP and MAR will change.



(a) (b)

Fig. 4 MAP & MAR

Our method outperforms Lucene both in MAP and MAR, mainly because we consider not only service descriptions or tagging but also the structural meaning of OWL-S documents. For Lucene, the structure information cannot be released by field search. No filter to the result and too many segments will be created in indexing as well as searching.

We also tested the time complexity of our DADBF method comparing with the Lucene approach. The results are shown in Fig. 5, where x-axis represents the number of services and y-axis represents the time consumption in the unit second.

Once the service number increases, the time complexity of Lucene remains low. BF-only method increases exponentially. The main reason is that once the number of services increases, the Bloom Filter address will increase exponentially in order to keep low error rate. As shown in Fig. 6, our DADBF eliminates the time cost issue. As explained in the previous sections, our solutions are two-fold: do not apply BF to the leaf nodes and conduct dynamic network adjustment.

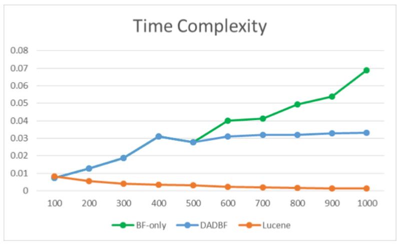


Fig. 5 Time complexity comparison.

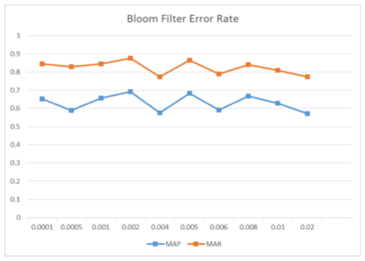
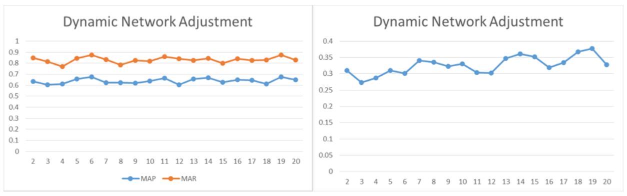


Fig. 6 False positive



(a) (b)

Fig. 7 Dynamic network adjustment.

C. False Positive Rate Study

The Bloom Filter method generally allows certain allowable errors, caused by False Positive. We tested the MAP and MAR for different error rates. Fig. 6 (x-axis means error rate) illustrates the impact of False Positive. It can be shown that even though downtrend with larger error rate is not remarkable, the larger the error rate the lower the MAP and MAR.

When a new service joins the network, if our predicted probability of connectivity reaches a threshold, the BF network will be re-computed to generate a new network. We studied the impact of threshold to MAP, MAR and Time Consumption. The results are shown in Fig. 7(a), where x-axis represents the threshold value and y-axis represents the MAP and MAR, respectively in two curves. In Fig. 7(b), where x-axis represents the threshold value and y-axis represents the time consumption in the unit of second.

Once the threshold value increases, the time complexity keeps increasing and the MAP and MAR remain stable. Lower threshold means less leaf nodes for search. Hence, as explained in the previous sections, conducting dynamic network adjustment could reduce the time complexity.

VI. Conclusions

In this paper, we have presented an approach of applying Bloom Filter, a popular network routing mechanism to service discovery in service network. Deterministic annealing is used to guide the service clustering and service network construction. With the rapid advancement of service oriented computing that leads to enormously increasing scale of service network, our approach addresses the scalability issue of existing semantic service discovery methods.

We plan to further our research in the following directions. First, we plan to explore to integrate supervised LDA and deterministic annealing to further enhance the effectiveness and efficiency of service clustering. Second, we will study encoding richer service information, including semantic data, QoS data, and past usage data. Third, we plan to construct an OWL-S annotated large-scale service network and further evaluate the scalability of our approach.

Acknowledgment

This work is partially supported by National Aeronautics and Space Administration, under grants NASA NNX16AB22G and NNX16AE15G; National Science Foundation, under grant NSF ACI-1443069; and Ericsson Research Gift.

References

1. I. Foster, "Service-Oriented Science," Science, vol. 308, col. 5723, pp. 814-817, 2005.
2. A.V. Paliwal, B. Shafiq, J. Vaidya, H. Xiong, and N. Adam, "Semantics-Based Automated Service Discovery," IEEE Transactions on Services Computing (TSC), vol. 5, col. 2, pp. 260-275, 2012.
3. M. Stollberg, M. Hepp, and J. Hoffmann, "A Caching Mechanism for Semantic Web Service Discovery," Lecture Notes in Computer Science, vol. 4825, col., pp. 480-493, 2007.
4. J. Zhang, W. Tan, J. Alexander, I. Foster, and R. Madduri, "Recommend-As-You-Go: A Novel Approach Supporting Services-Oriented Scientific Workflow Reuse," in Proceedings of *IEEE International Conference on Services Computing (SCC)*, Washington DC, USA, pp. 48-55, 2011.
5. B.H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," Communications of the ACM, vol. 13, col. 7, pp. 422-426, 1970.
6. A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A Survey," Internet Mathematics, col., pp. 636–646, 2002.
7. S. Chen, C.K. Chang, and L.-J. Zhang, "An Efficient Service Discovery Algorithm for Counting Bloom Filter-Based Service Registry," in Proceedings of *IEEE International Conference on Web Services (ICWS)*, Los Angels, CA, USA, pp. 157-164, 2009.
8. K. Rose, "Deterministic Annealing for Clustering, Compression, Classification, Regression, and Related Optimization Problems," Proceedings of the IEEE, vol. 86, col. 11, pp. 2210-2239, 1998.
9. M. Klusch, B. Fries, and K. Sycara, "Automated Semantic Web Service Discovery with OWLS-MX," in Proceedings of *fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, New York, NY, USA, pp. 915-922, 2006.
10. J. Zhang, W. Wang, X. Wei, C. Lee, S. Lee, L. Pan, and T.J. Lee, "Climate Analytics Workflow Recommendation as a Service - Provenance-Driven Automatic Workflow Mashup," in Proceedings of *The 22nd IEEE International Conference on Web Services (ICWS)*, New York, NY, USA, pp. 89-97, 2015.
11. P. Goering and G. Heijenk, "Service Discovery Using Bloom Filters," in Proceedings of *12th Annual Conference of the Advanced School for Computing and Imaging*, pp., 2006.
12. M. Zhou, S. Bao, X. Wu, and Y. Yu, "An Unsupervised Model for Exploring Hierarchical Semantics from Social Annotations," Lecture Notes in Computer Science, vol. 4825, col., pp. 680-693, 2007.
13. X. Liu, Y. Ma, G. Huang, J. Zhao, H. Mei, and Y. Liu, "Data-Driven Composition for Service-Oriented Situational Web Applications," IEEE Transactions of Services Computing (TSC), vol. 8, col. 1, pp. 2-16, 2015.
14. M. Klusch and F. Kaufer, "WSMO-MX: A Hybrid Semantic Web Service Matchmaker," Web Intelligence and Agent Systems, vol. 7, col. 1, pp. 23-42, 2009.
15. M.L. Sbodio, D. Martin, and C. Moulin, "Discovering Semantic Web Services using SPARQL and Intelligent Agents," Web Semantics: Science, Services and Agents on the World Wide Web, vol. 8, col. 4, pp. 310-328, 2010.
16. M. Junghans, S. Agarwal, and R. Studer, "Towards Practical Semantic Web Service Discovery," Lecture Notes in Computer Science (The Semantic Web: Research and Applications), vol. 6089/2010, col., pp. 15-29, 2010.
17. G. Cassar, P. Barnaghi, and K. Moessner, "Probabilistic Matchmaking Methods for Automated Service Discovery," IEEE Transactions of Services Computing (TSC), vol. 7, col. 4, pp. 654-666, 2014.
18. C. Li, R. Zhang, J. Huai, X. Guo, and H. Sun, "A Probabilistic Approach for Web Service Discover," in Proceedings of *IEEE 10th International Conference on Services Computing (SCC)*, Santa Clara, CA, USA, pp. 49-56, 2013.
19. L. Chen, Y. Wang, Q. Yu, Z. Zheng, and J. Wu, "WT-LDA: User Tagging Augmented LDA for Web Service Clustering," in *Service-Oriented Computing*, Eds.: Springer, 2013, pp. 162-176.
20. Y. Zhong, Y. Fan, K. Huang, W. Tan, and J. Zhang, "Time-Aware Service Recommendation for Mashup Creation," IEEE Transactions on Services Computing (TSC), vol. 8, col. 3, pp. 356-368.

21. M. Porter, "An Algorithm for Suffix Stripping Program*,"* Automated Library and Information Systems, vol. 14, col. 3, pp. 130-137, 1980.
22. K.S. Jones, "A Statistical Interpretation of Term Specificity and Its Application in Retrieval*,"* Journal of Documentation, vol. 28, col. 1, pp. 11-21, 1972.
23. J. Zhang, R. Madduri, W. Tan, K. Deichl, J. Alexander, and I. Foster, "Toward Semantics Empowered Biomedical Web Services," in Proceedings of *IEEE International Conference on Web Services (ICWS)*, Washington DC, USA, pp. 371-378, 2011.
24. C. Leacock and M. Chodorow, "Combining Local Context and WordNet Similarity for Word Sense Identification," in *WordNet: An Electronic Lexical Database*, C. Fellbaum, Editor, Eds.: MIT Press, 1998, pp. 265–283.
25. R. Jonker and T. Volgenant, "Improving the Hungarian assignment algorithm*,"* Operations Research Letters, vol. 5, col. 4, pp. 171-175, 1986.
26. T. Hofmann, "Probabilistic Latent Semantic Indexing," in Proceedings of *22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 50-57, 1999.
27. R. Albert and A.-L. Barabási, "Topology of Evolving Networks: Local Events and Universality*,"* Physical Review Letters, vol. 85, col. 24, pp. 5234-5237, 2000.