Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Service Solution Planning Considering Priori Knowledge and Fast Retrieval

RUILIN LIU[1], (STUDENT MEMBER, IEEE), XIAOFEI XU[1], (MEMBER, IEEE), ZHONGJIE WANG[1], (MEMBER, IEEE), AND QUAN Z. SHENG[2], (Member, IEEE)
[1]School of Computer Science and Technology, Harbin Institute of Technology, Harbin, 150001, China (e-mail: ruilin@hit.edu.cn)
[2]Department of Computing, Macquarie University, Sydney, NSW 2109, Australia (e-mail: michael.sheng@mq.edu.au)

Corresponding author: Xiaofei Xu (e-mail: xiaofei@hit.edu.cn).

"Research in this paper is partially supported by the National Key Research and Development Program of China (No 2017YFB1400604) and the National Science Foundation of China (61772155, 61832014, 61832004, 61472106)."

ABSTRACT Service composition is widely used to build complex value-added composite services to meet various coarse-grained requirements of customers. Discovering relevant services as the constituents of composite services is a crucial task which needs to be frequently performed during the composition process. Due to the fact that the amount of services available on the Internet is increasing drastically, the efficiency of both service discovery and composition becomes a big challenge. To solve this challenge, we propose a Priori Knowledge Based Service Composition approach (PKBSC) to reduce the searching space of relevant service discovery so as to improve the efficiency of service composition. PKBSC utilizes an interoperable approach including an ontology construction and merging method to solve the problem of the cross-domain and heterogeneous services from different repositories. And *service pattern* is adopted to describe priori knowledge from massive historical solutions, which is a recurrent valuable fragment composed of services frequently invoked together in service solutions. PKBSC also adopts the Formal Concept Analysis (FCA) to extract the implicit relationship between service requests and service patterns. Compared with the approach of composing multiple services from scratch, PKBSC exhibits better performance since the search space is greatly reduced by the adoption of service patterns. Experiments demonstrate that the proposed approach significantly improves the efficiency of service composition by 22.44%.

INDEX TERMS Formal concept analysis, frequent pattern mining, service composition, service pattern

I. INTRODUCTION

In Service Oriented Architecture (SOA), Web service as a method of communication between two applications over the Internet has grown up to be an important part of software development. Web services are software components designed to assist interoperable machine-to-machine interactions without considering the development platform or operating environment. Subsequently, semantic Web service approaches give us the ability to describe the capabilities of services in a formal and machine-processable manner and the semantic relationships among services, which are stimulating automatic service discovery and composition. Over the past decade, the success of Semantic Web turns out to depend on the use of ontology as a means of communication and information sharing. Ontology is a paramount technology of the Semantic Web, which provides a formal and explicit specification of knowledge representation, with the advantage that they are reusable and shareable. However, as Web services go through different stages of development, the existing Web services on the Internet utilize various description languages. Not all service providers adopt semantic technology to define Web services. In general, Web service description methods fall into three categories [1]: SOAP (WSDL [2]), REST (WADL [3]), and Semantic WS (OWL-S [4], WSDL-S [5], WSMO [6]). Meanwhile, with the development and popularization of intelligent hardware, the booming of Internet of Things (IoT) represents the next most exciting technological revolution [7], [8]. The IoT connects billions of things that include sensors, actuators, services, and other Internet-connected objects which sets up the environment where things can automatically communicate with computers [9]. The implementation of the IoT system will seamlessly integrate the cyber world with our physical world, and computers will be able to learn and gain information and knowledge to solve real-world problems [10]. Although IoT services are categorized differently, this paper focuses

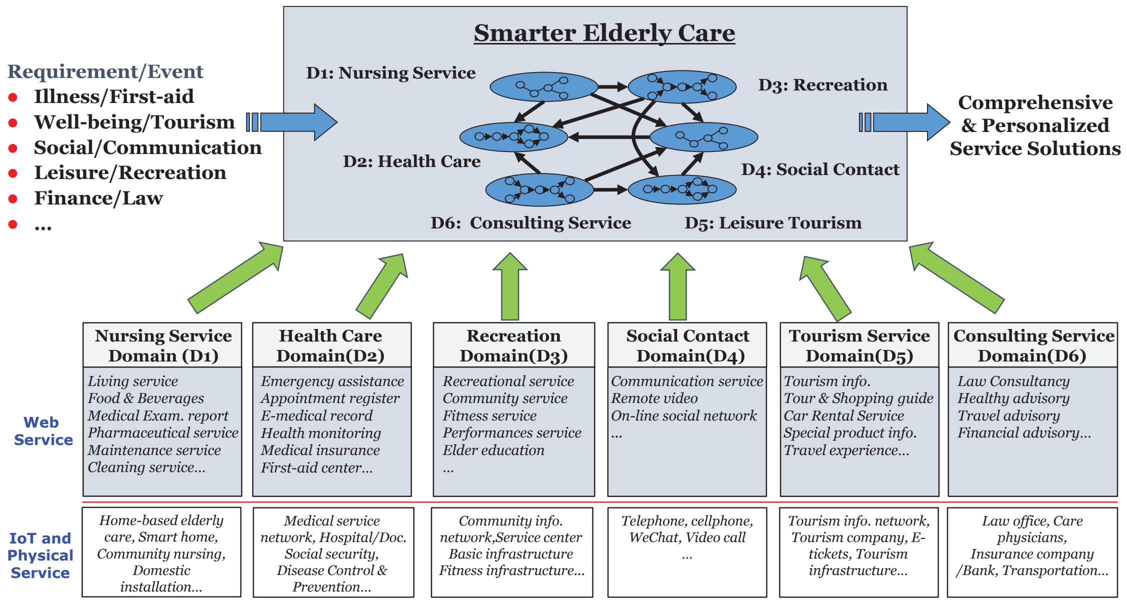


FIGURE 1: Smart Elderly Care [11].

on those IoT services that provide ambient data collection and analysis. This category of IoT services is growing rapidly especially in the eldercare domain where personalized living related data tracking and monitoring has become vital.

As a complex system that integrates a series of Web services and IoT services, Smart Elderly Care (SEC) focuses on providing comprehensive and personalized service solutions [12]. Massive services from multiple repositories converge on SEC (see Fig. 1), thus causing a series of related problems (RQ). **RQ1:** how to retrieval and consume these cross-domain and heterogeneous services is becoming an important problem.

Besides, SEC aims to construct composite services that could provide a one-stop shop for customers when no single service can fulfill customer's request on its own.

An analysis of the service composition literature highlights that the process of service discovery is a non-negligible task. It needs to be frequently carried out to discover relevant services during the generation of composition, no matter which composition approach is adopted, whether it is a fully automated approach based on Artificial Intelligence (AI) planning techniques [13], a semi-automatic approach relied on predefined workflow [14], or a graph-based approach which focuses on semantic Input/Output (I/O) parameter matching [15], [16]. In addition, due to the fact that the number of services is mushrooming in recent years and the connections among services become more and more complex, the search space of the service composition algorithm increases rapidly. Therefore, **RQ2:** how to improve the efficiency of service discovery is the key point of service composition, especially when a large number of potential services are involved.

Process reuse plays a vital role in service computing and business management domains, which encourages developers to fragment their applications into more services for reuse and benefit in scalability. Many studies pay attention to how to identify and describe valuable process fragments from a historical log. By analogy, we believe that reusable knowledge can be used in service composition. Dealing with service discovery for service composition by considering *priori knowledge* is a promising way, which contains the previous and implicit interactions among Web services. **RQ3:** how to manage and when to utilize priori knowledge is another issue to consider.

In order to tackle the previous problems, we propose a Priori Knowledge Based Service Composition (PKBSC) algorithm. For **RQ1**, we utilize an interoperable approach including an ontology construction method for services without semantic technology and a merging method for multiple ontologies. We also provide a method to calculate the similarity of semantic concepts in a merged ontology (see Section IV). For **RQ2**, we adopt *service pattern* to describe priori knowledge from massive historical solutions, which is a recurrent valuable fragment (or a coarse-grained component) composed of services frequently invoked together in service solutions. The priori knowledge is harvested from the historical log of existing solutions by service patterns mining algorithm (see Section V). For **RQ3**, we utilize Formal Concept Analysis (FCA) to extract the implicit relationship between service requests and service patterns (see Section VI). When a new service request arrives, the most relevant service patterns could be used empirically for providing coarse-grained components while constructing a service solution during the priori knowledge based forward search. After that, a heuristic backward search proceeds to identify the optimal service solution which is an executable solution without any redundancy (see Section VII). We conduct a series of experiments using both real and synthetic data to evaluate the efficiency and effectiveness of our approach. We improve the efficiency of A* based traditional graph searching service composition algorithm (A*-TSC) [17] by 22.44%. We analyze performance changing of PKBSC in terms of the time-aware frequency of service patterns, which both can affect the performance of the algorithm.

It is worth noting that Quality of Service (QoS) is also an indispensable measure of service computing. The work in this paper does not involve QoS at present, but readers can find QoS based services selection in the previous work [18]. The rest of this paper is organized as follows: In Section II, we discuss related works. In Section III, we introduce formal problem formulation and the whole process of approach. Section IV-VI elaborate the proposed method in detail and Section VII reports experimental results. Finally, Section VIII offers some concluding remarks.

II. RELATED WORKS

A. SEMANTIC WEB SERVICE

The last decade has witnessed the emergence of a number of approaches and tools leveraging Semantic Web technology that has been proven effective in service discovery, composition, and recommendation [19]–[21]. These approaches often adopt various markup languages to annotate service elements with the semantic concepts defined in ontologies, such as operations, inputs, and outputs. For example, Meteor-S [22] is a combined approach toward automatic semantic annotation which suggests concepts from domain ontologies to facilitate annotation task. The approach uses a simple aggregation function to combine string matcher, structural matcher, and synonym finder. Duo et al. [23] present a similar approach which also aggregates the results of several matchers from different perspectives. Salomie et al. [24] propose an approach named SAWS which enhances the Web Service Definition Language (WSDL) descriptions with semantic concepts provided by domain ontologies. Chabeb et al. [25] propose a method that executes semantic annotation on Web services and integrates the results into WSDL. Kino [26] automatically annotates Web services based on the similarity between service descriptions and vectors of available ontological concepts. A context-based semantic approach to the problem of matching and ranking Web services for possible service composition is suggested in [27].

B. FREQUENT PATTERN BASED COMPOSITION

Web service composition technique is widely used to make more complex and value-added applications to meet various requirements of Web customers, which is to compose Web services with different functionalities [28]. As the number of available Web services is rapidly increasing, the discovering process of relevant Web service from massive candidates demands a lot of efforts, and it is quite inefficient. Therefore, a large number of approaches and automatic composition techniques are proposed to enhance development efficiency, bringing a hot topic in the research community of Web service. Among these tools and techniques, frequent pattern based composition approaches receive a great deal of attention [28]–[32]. Although the number of created service compositions is growing rapidly, most of these compositions tend to follow some popular business models and usage patterns [32]. So, given the dataset of previously proposed service compositions, it can always find out some frequently occurred service composition patterns from history, and then based on these patterns make service composition in the future. A great number of service composition approaches based on this idea are proposed and proved to be very successful. These frequent patterns usually give rules that summarize the relationships of correlations, collaborations and complements between services in historical compositions.

C. FORMAL CONCEPT ANALYSIS

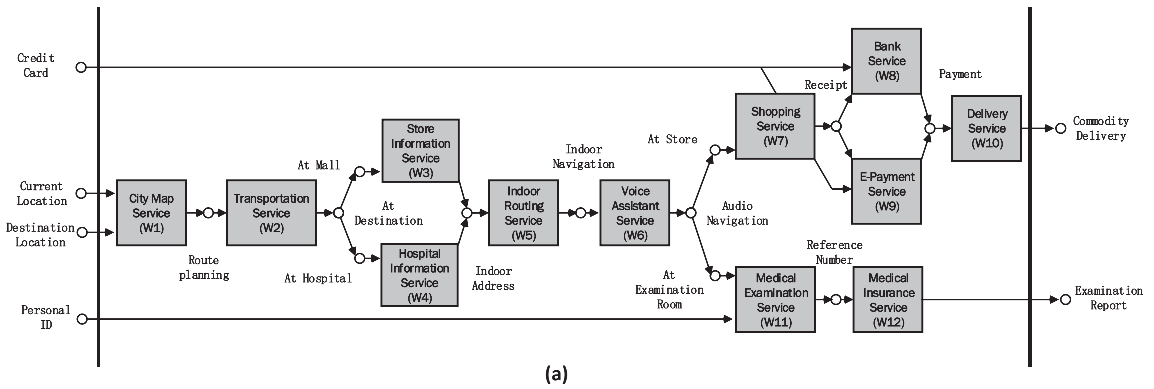
Formal concept analysis (FCA) is a mathematical model which offers conceptual knowledge representation in a hierarchical order. As a branch of lattice theory, FCA has been applied in various fields like knowledge representation [33], data analysis, information management and retrieval [34], [35], designing role based access control [36], and knowledge processing tasks [37]. [38] proposes an approach for Web service interface decomposition, which uses the FCA to identify the hidden relationships among service operations in order to improve the interface modularity and usability. [39] uses the FCA to generate service dependency network which is used to select the composition of discovered Web services set. [40] groups service descriptions into hierarchical clusters based on the topic correlation, and uses the FCA to organize the constructed hierarchical clusters into concept lattices according to their topics. [41] exploits the fuzzy extension of FCA theory to generate knowledge representations based on hierarchical structures for web resources retrieval. [42] shows how to combine rough set theory with fuzzy formal concept analysis to perform Semantic Web search and discovery of information.

III. SERVICE COMPOSITION PROBLEM

Service composition aims to construct composite services that could fulfill a request from the customer under the assumption that no single service can achieve such a request. In order to compose services together, we define a formal representation of services and the relationship between them.

***Definition** 3.1 (**Service**):* The functionality of a service (Web service and IoT service) s can be defined as a tuple $s =< In_s, Out_s >, s \in S, In, Out \subseteq P$ where In is a set of input parameters required to invoke s and Out is the set of output parameters returned by s after its execution, S is a set of all services in repository, and P is a set of parameters used to delineate the input/output of services in S.

***Definition** 3.2 (**Service Composition**):* Given a request $r =< In_r, Out_r >\in R, In_r, Out_r \neq \emptyset, In_r, Out_r \subseteq P$, and $In_r \cap Out_r = \emptyset$, where In_r is a set of available input parameters and Out_r a set of requested output parameters, we can define the problem of Web service composition as



(a)

Service Request	Service Solution
R1=<{CurrentLocation,DestinationLocation(Mall)},{AtStore}>	Sol1=W1-W2-W3-W5-W6
R2=<{IndoorAddress,CreditCard},{CommodityDelivery}>	Sol2=W5-W6-W7-W8-W10
R3=<{AtHospital,Personal ID},{ExaminationReport}>	Sol3=W4-W5-W6-W11-W12
R4=<{CurrentLocation,DestinationLocation(Hospital),Personal ID},{ExaminationReport}>	Sol4=W1-W2-W4-W5-W6-W11-W12

(b)

FIGURE 2: A toy example: (a) Construction process of a service solution, (b) Historical Record.

that of finding a service solution sol without any redundancy. A sol_i is a multilayer directed graph and is defined by a tuple: $sol_i =< S_{sol_i}, E(S_{sol_i}) >$. S_{sol_i} indicates the services contained by sol_i, $S_{sol_i} = \{s_1, s_2, \cdots, s_N\}, S_{sol_i} \subseteq S$. $E(S_{sol_i})$ describes the connection relationship between services. We say sol_i is a valid composition solution for request r if and only if the following expression is satisfied:

$$\begin{aligned}&(In_r \cap In_{s_1} = In_{s_1}) \wedge ((In_r \cup Out_{s_1}) \cap In_{s_2} = In_{s_2})\\ &\wedge \cdots \wedge ((In_r \cup Out_{s_1} \cup \cdots \cup Out_{s_N}) \cap Out_r = Out_r)\end{aligned} \quad (1)$$

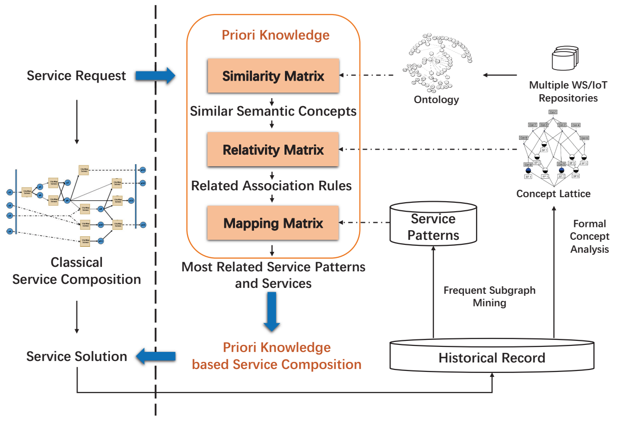


FIGURE 3: The whole process of approach.

Fig. 2 gives an example of service composition. When a service request arrives, we need to construct a feasible and optimized service solution with a series of services. This example describes how we help two groups of elders with different purposes at the same time. We could adopt the traditional graph based approach to construct the service solution. In graph based composition approach (left side in Fig. 3), a service network [16] is eventually generated based on the I/O matching information of the relevant services. This network contains all possible service compositions or solutions that fulfill the customer's request. Then, the service network is optimized by applying different techniques to reduce the number of services and redundancy. The optimal service network is a satisfied solution for the service request. As an eldercare service system, it assembles medical and health services, housekeeping services, life services, emergency relief services, cultural and entertainment services, and transportation services from different service repositories. Some services are Web service and some are IoT service, the biggest feature of IoT services is linking the cyber world and the physical world, such as the semantic concept "CurrentLocation" is automatically perceived by IoT service. How to implement interoperability among different service repositories is a problem that needs to be solved. In addition, we find it is a promising way using a priori knowledge of historical solutions to accelerate the process of service composition. We can extract the frequent fragment which is called a service pattern, such as "W1-W2", "W5-W6", "W10-W11". However, how to identify the most related service pattern according to a service request is problem to be solved. Fig. 3 (right side) provides a general idea of the framework, sketching the whole workflow through the main phases. First, we adopt ontology construction and merging method to solve multiple repositories interoperability and build a matrix (Similarity Matrix) to index the similarity between two different semantic concepts. Second, we adopt the apriori algorithm to mine the service patterns (frequent fragments) of historical solutions. Then, we adopt the formal concept analysis method to establish relations between the semantic concepts in service requests and service patterns. We use two matrices (Relativity Matrix and Mapping Matrix) to index the relations. Last, We use these three matrices together to complete priori knowledge based service composition.

IV. HETEROGENEOUS SERVICE INTEROPERABILITY

The goal of an ontology is to capture relevant domain knowledge and provide a common understanding to identify commonly recognized vocabulary in a domain, which gives a clear definition of the relationship between concepts from different levels of formalization. It also provides a viable solution for application interoperability, data access, and complex domain modeling by applying common standard languages such as OWL and RDF. In addition, the use of ontology to describe web resources promises validity, efficiency, and accuracy in information retrieval activities, especially since most knowledge is typically encapsulated in a collection of unstructured text documents. As an elder-care platform integrated various IoT services from different service repositories, we must propose an interoperable approach for handling different service description methods. For services used WSDL/WADL, we adopt an ontology bootstrapping approach [43] to construct a new ontology. Then a merging approach of heterogeneous domain ontologies [44], [45] is used to address the interoperability problem between the ontologies. At last, based on the merged ontology, Similarity Matrix is constructed which accelerates the service compatibility detection and enhances understanding in the intention of a service request.

A. ONTOLOGY CONSTRUCTION AND MERGING

As mentioned before, ontologies have become the de-facto modeling tool of choice, employed in many applications and prominently in the semantic web. Most automatic ontology learning methods still need of a significant manual effort in the completion, consolidation, and validation of the automatically generated ontology. Moreover, the issues of duplicate information across documents and redundant annotations are major challenges to automatic ontology creation as the automatically populating ontology from diverse and distributed web resources poses significant challenges. For services with an XML-based description, we adopt an ontology bootstrapping approach proposed in [43]. Ontological bootstrapping is a promising ontology construction technique with less manual effort, which aims at automatically generating concepts and their relations in a given domain. Bootstrapping an ontology based on a set of predefined textual sources address the problem of multiple, largely unrelated concepts. This approach exploits the advantage that Web services usually consist of both WSDL/WADL and free text descriptors. It proposed two methods to exact concepts in the WSDL/WADL descriptor, namely Term Frequency/Inverse Document Frequency (TF/IDF) analysis and web context generation. Ontology bootstrapping approach integrates the results of above methods and applies a method to validate the concepts using the service free text descriptor, thereby offering a more accurate definition of ontologies.

For services already used semantic technology, due to unavailability of any standard for ontology construction, ontologies from various repositories are different. It leads that the interoperability between multiple ontologies is very low. There is an increasing need to integrate or merge multiple ontologies, with the goal of creating a single ontology that provides a unified view while maintaining all the information from them. Several such multiple ontologies mapping approaches have already been proposed [46], [47], one increasingly adopted and promising idea is to decompose the complex integration problem into the match and merge subtasks. The match-merge base mappings support two very related operations, namely ontology alignment and merging. The ontology alignment process leverages the advances made for automatic ontology and schema matching. It takes two or more input ontologies of the same or similar domain and produces a set of relationships between semantically matching concepts. The ontology merging process can then utilize match relationships identifying corresponding concepts in the input ontologies that should be merged. It combines semantically matching concepts into a single concept and then generates a unique ontology from the source ontologies. Ontology mapping can solve multiple forms of mismatch, which can be caused by multiple ontology standards used by different knowledge engineers in different environments and at different time intervals.

B. SEMANTIC MATCHMAKING

A fundamental task for generating service compositions is the ability to analyze the compatibility between different available services. We extract comparable features available in each Web service description. Since services are formalized as Input-Output interfaces, only semantic features of input and output descriptions are taken into account. After ontology construction and merging, we assume that all I/O parameters are related to semantic concepts provided by a domain ontology $\mathcal{DO}$ through semantic annotations. Semantic matchmaking is in charge of assessing the level of semantic compatibility between semantic concepts, given an ontology.

In this context different ontological similarity measures have been proposed [48]. We first define the hierarchical relation as follows with respect to given two concepts in a domain ontology ($\mathcal{DO}$). Let a and b be two semantic concepts represented by the two nodes in a predefined ontology, i.e., $a, b \in \mathcal{DO}$.

***Definition** 4.1 (**Hierarchical relation**):*

- Equivalent ($\equiv$). If both concepts are equivalent, i.e., a and b represent the same same concept even though they could be expressed by means of equivalent synonyms, $a \equiv b$.
- Sub-concept ($\sqsubseteq$).If a is a sub-concept of b, i.e., a is a hierarchical specialization of b, $a \sqsubseteq b$.

$$Sim(a,b) = 1 - log_2(1 + \frac{\mid \Phi(a) \setminus \Phi(b) \mid + \mid \Phi(b) \setminus \Phi(a) \mid}{\mid \Phi(a) \setminus \Phi(b) \mid + \mid \Phi(b) \setminus \Phi(a) \mid + \mid \Phi(a) \cap \Phi(b) \mid}) \quad (3)$$

- Super-concept ($\sqsupseteq$). If a is a super-concept of b, i.e., a is a hierarchical generalization of b, $a \sqsupseteq b$.

To quantify semantic similarity, the measure proposed in [49] gives a more pertinent and practical result. The set of taxonomical features describing the concept a is defined in terms of the relation as follows:

$$\Phi(a) = \{c_i \mid c_i \sqsubseteq a \vee c_i \equiv a, c_i, a \in \mathcal{DO}\} \quad (2)$$

and the semantic similarity between two concepts is defined in Equ. (3). To accelerate the process of semantic concept similarity computation, we define a Similarity Matrix (*SimM* = *Semantic Concept* $\times$ *Semantic Concept*) to index the relationship between any two different semantic concepts.

$$SimM_{ij} = \begin{bmatrix} SimM_{11} & \cdots & SimM_{1|\mathcal{DO}|} \\ \vdots & \ddots & \vdots \\ SimM_{|\mathcal{DO}|1} & \cdots & SimM_{|\mathcal{DO}||\mathcal{DO}|} \end{bmatrix} \quad (4)$$

where $SimM_{ij} \in [0,1]$, $SimM_{ij} = SimM_{ji}$ and $SimM_{ii} = 1$.

The above defines how to calculate the similarity between individual semantic concepts, in order to measure the quality of compatibility, we also need a matchmaking mechanism that evaluates the semantic similarity of two sets of concepts. Two services are compatible if and only if the output parameters returned from the first service can satisfy the input parameters of the second service. That is also to say that each semantic concept annotating input parameters in the second service must has a same or similar concept offered by the first service. The similarity measure of two sets of concepts defines as follows:

***Definition** 4.2 (**Full Match**):* Given two sets of concepts $C_1, C_2 \subseteq \mathcal{DO}$, we define $C_1 \oplus C_2 = \{c_2 | Sim(c_1, c_2) \in [\theta, 1], c_1 \in C_1, c_2 \in C_2\}$, where θ is a predefined similarity threshold. Note that this operator $\oplus$ is incommutable. A full match between C_1 and C_2 exists if and only if $C_1 \oplus C_2 = C_2$.

We say that two services $s_1 = (In_{s_1}, Out_{s_1})$ and $s_2 = (In_{s_2}, Out_{s_2})$ are compatible if and only if $Out_{s_1} \oplus In_{s_2} = In_{s_2}$. And $sol_i =< S_{sol_i}, E(S_{sol_i}) >$ is a valid composition solution for request $r =< In_r, Out_r >$ if and only if the following expression is satisfied:

$$(In_r \oplus In_{s_1} = In_{s_1}) \wedge ((In_r \cup Out_{s_1}) \oplus In_{s_2} = In_{s_2}) \\ \wedge \cdots \wedge ((In_r \cup Out_{s_1} \cup \cdots \cup Out_{s_N}) \oplus Out_r = Out_r) \quad (5)$$

V. SERVICE SUBGRAPH MINING ALGORITHM

To promote the efficiency and accuracy of constructing a new service solution, considering the historical solutions could be a promising approach, which contains a set of already developed service solutions to satisfy customer's request. Often, there are lots of valuable subprocesses frequently appeared in service solutions. Such a large number of subprocesses could serve as a knowledge base for providing the guidance for solution construction effort. Therefore, the discovery and presentation of these valuable subprocesses (a.k.a. service patterns) have become of great importance. A service solution could be abstracted to a Directed Acyclic Graph (DAG), the valuable subprocesses mining can be considered as Frequent Subgraph Mining (FSM). Note that throughout this paper, the terms frequent subprocess, frequent subgraph and service pattern will be considered interchangeable. We define service pattern as follows:

***Definition** 5.1 (**Service Pattern**):* A service pattern $sp_i \in SP$ is a frequent subgraph mined from a set of solutions. sp_i is defined by a tuple: $sp_i =< In_{sp_i}, Out_{sp_i}, S_{sp_i}, E(S_{sp_i}) >$.

Refer to [50], a comprehensive survey about FSM algorithm, Apriori-based approach and pattern growth-based approach are two different kinds of widely adopted techniques to generate a set of frequent subgraphs. In most cases, pattern growth-based approach has better performance than Apriori-based approach due to the fact that it discovers all the frequent subgraphs without candidate generation and it combines the growing and checking of frequent subgraphs into one procedure. Additionally, it adopts two techniques, Depth-First Search (DFS) lexicographic order and minimum DFS code. However, according to our experimental studies, the average size of frequent subgraphs is small, that is to say that few number of nodes appear in a frequent subgraph. The computation complexity of Apriori-based approach is acceptable under the condition that sizes of frequent subgraphs are relatively smaller and services in a solution are rarely repetitive. Besides, it is easy to understand and implement.

Given a database, an Apriori-based approach adopts a Breadth First Search (BFS) strategy to explore the graph. The basic Apriori-based algorithm is presented in Alg. 1. This approach first considers all (k-1) subgraphs before considering k subgraphs. The Downward Closure Property (DCP) has been widely adopted with respect to candidate subgraph generation. If a graph is frequent, then all of its subgraphs will also be frequent. If any of the (k-1) candidate subgraphs are not frequent, then the DCP can be used to safely prune the candidates. Frequent subgraph mining approaches adopt an iterative pattern mining strategy, where each iteration can be divided into three phases:

- Candidate generation (line 6). The set of candidates is generated by a self-join of the candidates found in the last pass. In the k-th pass, a graph could be considered as a candidate only if each of its subgraphs is frequently found in the (k-1)-th pass.
- Counting candidates (lines 8-13). A new scan of the database calculates support for each candidate through

Algorithm 1: Apriori-based service pattern mining algorithm

```
Input: Historical service solutions $G$, minimum support $\varphi$
Output: $F_1, F_2, \cdots, F_k$ a set of frequent patterns of cardinality 1 to $k$
begin
    $F_1 \leftarrow$ Identify all frequent k=1 pattern in G
    $k \leftarrow 2$
    while $F_{k-1} \neq \emptyset$ do
        $F_k \leftarrow \emptyset$
        $C_k \leftarrow$candidate generation$(F_{k-1})$ /*Candidate generation by a self-join of $(F_{k-1})$*/
        foreach candidate $c \in C_k$ do
            $c.count \leftarrow 0$
            foreach $g \in G$ do
                if subgraph-ismorphism$(g, c)$ then
                    $c.count \leftarrow c.count + 1$ /*Counting candidates*/
                end
            end
            if $c.count \geq \varphi|G| \wedge c \notin F_k$ then
                $F_k \leftarrow F_k \cup c$ /*Pruning based on support*/
            end
        end
        $k \leftarrow k + 1$
    end
end
```

subgraph isomorphism detection.

- Pruning based on support (lines 14-16). Those candidates become the seed for the next pass only if their counts(or supports) are equal to or higher than the predefined threshold. The algorithm terminates when no frequent subgraph is found in a pass, i.e., when there is no candidate generated.

An example is shown in Fig. 4. Given a graph database which contains eight different graphs (solutions) in the left part and a minimum support $\varphi = 0.3$, the algorithm builds a subgraph lattice of this database in the right part after multiple iterative executions. This lattice consists of all frequent subgraphs generated in different iterations, and each level in Fig. 4 presents the times of iteration. Frequent nodes (only one node in each subgraph) are identified in level one, whose frequency of occurrence is more than three, e.g., $A, B, \cdots$. The permutation of these candidates with single nodes will be calculated in the next iteration, e.g., $C \rightarrow D, C \rightarrow F, \cdots$. Iterative process terminates until no candidate is generated. Note that, a service solution is abstracted to a DAG, so we adopt appropriate coding scheme to avoid unnecessary computation, such as the lexicographic order used in Fig. 4, $D \rightarrow C$ is a backward edge violating lexicographic order, which never appears in any solution. So, a candidate is only considered when all inside edges comply with the lexicographic order.

VI. FORMAL CONCEPT ANALYSIS

In the last section, we introduce how to mine service patterns in historical records. Service patterns can improve the retrieval efficiency of related services to a certain extent, however service patterns can appear anywhere in the service solution, some service patterns are directly semantic related to service requests, and some are indirectly statistical related. Therefore, how to establish the relationship between service requests and service patterns has become an urgent problem

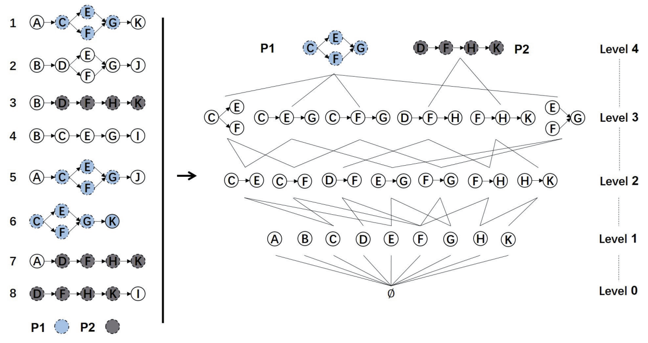


FIGURE 4: An example of Apriori-based pattern mining.

to be solved. Therefore, in this section, it details how to use the formal concept analysis (FCA) to retrieve relevant service patterns, including semantic correlation and statistical correlation, according to service requests.

FCA is a mathematical model which permits the identifications of groups of objects having common attributes and offers conceptual knowledge representation in a hierarchical order based on the applied lattice theory [51]. FCA starts the analysis from a given matrix called formal context, which comprises a set of objects, a set of attributes and a binary relationship between them. The basic outputs of FCA are formal concepts, formal concept lattice and attribute implications.

- A formal concept is a maximal pair of set of objects (extent) and its attributes (intent) closed with Galois connection, which is regarded as a basic unit of human thought, allowing meaningful comprehensible interpretation [52].
- The formal concept lattice provides a hierarchical order visualization between the discovered formal concepts.
- Attribute implications provide dependency between a given set of attributes and are applied in fields like healthcare [53], clustering [54], and decision formal context [55].

Since FCA allows the extraction of dependencies within the data, we exploit FCA to organize the historical solution into a formal concept lattice according to their semantic concepts. And it helps to discover the hidden dependency among semantic concepts. Then a set of association rules are exacted from this formal concept lattice. Finally, for making it easy to calculate, we build a related matrix to record the relationship between semantic concepts and valuable association rules. It is worth noting that the semantic concept and the formal concept are two easily confused but different terms. They are considered as similar terms because that they are both organized in a hierarchical order, but the semantic concept is from an ontology while the formal concept is from a formal concept lattice.

Definition 6.1 (Formal Context): A formal context is denoted by $K = (G, M, I)$ where G is a set of objects, M is a set of attributes, and I is a binary relation between G and $M (I \subseteq G \times M)$. The object $g \in G$ has the attribute $m \in M$

(or the attribute $m \in M$ applies to the object $g \in G$) if g and m are in relation I (denoted by gIm).
Given a context $K = (G, M, I)$, let $O \subseteq G$ and $A \subseteq M$ be two sets. We define the dual sets $O' = \{m \in M | \forall g \in O : gIm\}$ and $A' = \{g \in G | \forall m \in A : gIm\}$, where O' is the set of all attributes that are valid descriptions for all objects in O and A' is the set of those objects that have all attributes from A, respectively.
Definition 6.2 (Formal Concept): A formal concept is considered to be a unit of thought constituted of two parts: its extent and its intent. The extent consists of all objects belonging to the concept, while the intent comprises all attributes shared by those objects.
A formal concept of the context $K = (G, M, I)$ is a pair (O, A) such that $O \subseteq G$, $A \subseteq M$ and $O' = A$, $A' = O$. The set O and A represent the extensional and intensional components, which are referred to as the extent and the intent of a concept, respectively.
Definition 6.3 (Concept Lattice): A concept lattice defines a hierarchical representation of objects and attributes, in which a certain concept inherits all the extents (objects) of its descendants and all the intents (attributes) of its ascendants.
Definition 6.4 (Partial Order, $\leq$): Let $\mathfrak{c}_1 = (O_1, A_1)$ and $\mathfrak{c}_2 = (O_2, A_2)$ be two formal concepts of a formal context $K = (G, M, I)$, then $\mathfrak{c}_2$ is a subconcept of $\mathfrak{c}_1$ (equivalently, $\mathfrak{c}_1$ is a superconcept of $\mathfrak{c}_2$), if and only if $(O_1, A_1) \leq (O_2, A_2) \Leftrightarrow O_1 \subseteq O_2 (\Leftrightarrow A_1 \subseteq A_2)$.

Let M be a set of attributes of a formal context $K = (G, M, I)$. An association rule is a pair $X \Rightarrow Y$ with $X, Y \subseteq M$. The support is defined as:

$$Sup(X \Rightarrow Y) = \frac{|(X \cup Y)'|}{|G|} \tag{6}$$

and the confidence is computed as:

$$Conf(X \Rightarrow Y) = \frac{|(X \cup Y)'|}{|(X)'|} \tag{7}$$

where $|(X \cup Y)'|$ is the number of objects which have both X and Y attributes in their extents, $|G|$ is the number of objects in G, and $|(X)'|$ is the number of objects which have X attributes in their extents.

According to the support and confidence mentioned above, the association rules mining task based on formal concept lattice can be stated as follows.

- For any formal concept $\mathfrak{c}_i = (O_i, A_i)$ in the formal concept lattice, if $|O_i|$ (=$|(A_i)'|$) is greater than a threshold θ, then $\mathfrak{c}_i$ is called a frequent formal concept.
- If frequent formal concepts $\mathfrak{c}_1 = (O_1, A_1)$ and $\mathfrak{c}_2 = (O_2, A_2)$ satisfy $\mathfrak{c}_1 \leq \mathfrak{c}_2$, we can exact association rule $A_1 \Rightarrow A_2 - A_1$ which confidence is $|A1|/|A2|$, otherwise, $A_2 - A_1 \Rightarrow A_1$ which confidence is 100%.
- If frequent formal concepts $\mathfrak{c}_1 = (O_1, A_1)$ and $\mathfrak{c}_2 = (O_2, A_2)$ do not satisfy $\mathfrak{c}_1 \leq \mathfrak{c}_2$ or $\mathfrak{c}_2 \leq \mathfrak{c}_1$, and there exists nonempty common maximum-subconcept $\mathfrak{c} = (O, A)$, there are association rules between A_1 and A_2: $A_1 \Rightarrow A_2$, $A_2 \Rightarrow A_1$, their confidences are $|A|/|A1|$, $|A|/|A2|$ respectively.

We define a Relativity Matrix (*RelM = Semantic Concept × Association Rule*) to index the inclusion relationship between semantic concepts and association rules. $RelM_{ij} = 1$ indicates the semantic concept i appears in the antecedent of association rule j, otherwise $RelM_{ij} = 0$ indicates the semantic concept i is irrelevant to the association rule j. Given a set of semantic concepts, we say an association rule is related if and only if that the antecedent of an association rule is the subset of this semantic concept set. Therefore, given a service request, we can get a set of relative association rules easily. Note that an association rule is input-related when the antecedent of this association rule is the subset of input of given service request, otherwise, it is output-related when the antecedent of this association rule is the subset of the output of given service request.

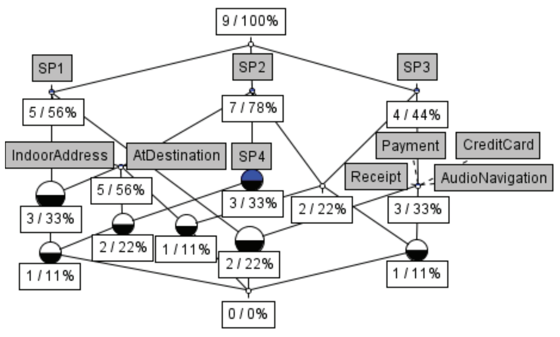
However, these attribute association rules are vulnerable to noise in the formal context. In addition, FCA's scalability and computability is another major focus of lattice-based applications. Given a formal context, the complexity of generating all formal concepts and their visualization is exponential [56]. Due to this fact, it is difficult to properly analyze the underlying knowledge using FCA when generating a large number of formal concepts [57]. Reducing the formal context to a lower dimension could help us in solving these problems. From one side, replacing the original solutions with the service patterns mined in the previous section can effectively reduce the number of attributes in the formal context. The minimal concept lattice can be found to avoid redundancy while maintaining structural consistency. By deriving the equivalence relationship between the concept lattice nodes, the number of nodes and edges of the concept lattice can be significantly reduced. From the other side, we can calculate the semantic similarity of service solutions and adopt clustering method to divide objects into different fragments, and generating concept lattice in each fragment respectively is another effective way to reduce the complexity.

We define a Mapping Matrix (*MapM = Association Rule × Semantic Concept*) to index the mapping relationship from the antecedent of association rule to the related semantic concepts in the consequent of the same association rule. As we discuss before, substituting semantic concepts by service pattern in each solution, we can identify the related service patterns statistically when they are semantical irrelevant to a given service request. Also, we can get a set of semantic concepts which are relevant, in the high probability. $MapM_{ij}$ indicates the consequent of association rule i contains the relative service pattern (or semantic concept) j, and records the support and confidence of association rule j. Continuing with the previous example, given a service request <{*Credit Card, Current Location, Destination Location,Personal ID*},{*Commodity Delivery, Examination Report*}>, we get nine most related service solutions by identifying the cluster which this request belongs in. And the first two columns of the table in Fig. 5(a) is the original solutions

	Service Solution		Semantic Concept / Service Pattern									
	Original Solution	Annotated with Service Pattern	SP1	At Destination	Indoor Address	SP2	Audio Navigation	Credit Card	Receipt	Payment	SP3	SP4
Sol 1	W1-W2-W3-W5-W6	**SP1**-W3-**SP2**	X	X	X	X						
Sol 2	W1-W2-W4-W5-W6	**SP1**-W4-**SP2**	X	X	X	X						
Sol 3	W1-W2-W7-W8-W10	**SP1**-W7-W8-**SP3**	X				X	X	X	X	X	
Sol 4	W1-W2-W7-W9-W10	**SP1**-W7-W9-**SP3**	X				X	X	X	X	X	
Sol 5	W1-W2-W4-W5-W6-W11-W12	**SP1**-W4-**SP2**-**SP4**	X	X	X	X						X
Sol 6	W5-W6-W7-W8-W10	**SP2**-W7-W8-**SP3**				X	X	X	X	X	X	
Sol 7	W3-W5-W6-W7-W9-W10	W3-**SP1**-W7-W9-**SP3**		X	X	X					X	
Sol 8	W5-W6-W11-W12	**SP2**-**SP4**				X						X
Sol 9	W4-W5-W6-W11-W12	W4- **SP2**- **SP4**		X	X	X						X

SP1: W1-W2 **SP2:** W5-W6 **SP3:** W10 **SP4:** W11-W12

(a)



(b)

1. { } => SP2;	[100%, 78%]
2. SP2 => AtDestination IndoorAddress;	[78%, 71%]
3. AtDestination => IndoorAddress SP2;	[56%, 100%]
4. IndoorAddress => AtDestination SP2;	[56%, 100%]
5. AtDestination IndoorAddress SP2 => SP1;	[56%, 60%]
6. SP1 => AtDestination IndoorAddress SP2;	[56%, 60%]
7. SP3 => AudioNavigation CreditCard Receipt Payment;	[44%, 75%]
8. SP1 SP2 => AtDestination IndoorAddress;	[33%, 100%]
9. AudioNavigation => CreditCard Receipt Payment SP3;	[33%, 100%]
10. CreditCard => AudioNavigation Receipt Payment SP3;	[33%, 100%]
11. Receipt => AudioNavigation CreditCard Payment SP3;	[33%, 100%]
12. Payment => AudioNavigation CreditCard Receipt SP3;	[33%, 100%]
13. SP4 => SP2;	[33%, 100%]

(c)

FIGURE 5: FCA example: (a) Annotated Formal Context with Service Patterns, (b) Concept Lattice, (c) Association Rules.

and the annotated solutions with service patterns respectively. According to these annotates solutions, a formal context is generated automatically, in which semantic concepts and service patterns constitute attributes. Using FCA method, we can get a formal concept lattice as shown in Fig. 5(b) and a set of association rules can be easily generated by given predefined minimal support and confidence. Fig. 5(c) shows association rules under the condition that minimal support (δ) and confidence (σ) is $[\delta, \sigma] = [30\%, 60\%]$. We can get $SP1, SP3, SP4$ by semantic correlation and get $SP2$ by statistical correlation respectively. Also, we can get some related semantic concepts, including "At Destination", "Indoor Address", "Indoor Navigation", "Credit Card", "Receipt and Payment". Using these semantic concepts could identify the related services, i.e. $W3, W4, W7, W8, W9$ in this example.

VII. PRIORI KNOWLEDGE BASED SERVICE COMPOSITION ALGORITHM

On the basis of the formal definition of the service composition problem, in this section we present a graph-based algorithm to generate service solution effectively and efficiently which fulfills customer requirement from available input concepts to desired output concepts.

A. PRIORI KNOWLEDGE FORWARD SEARCH FOR RELEVANT SERVICES

Service composition generates a composite solution, which consists of multiple services that can be executed in sequence or in a parallel process. Therefore how to identify the most relevant services is the key point of this section. Given a service request, a set of candidates (available services and service patterns) is dynamically generated layer by layer from inputs to outputs of this request (see Fig. 6). For each layer, the algorithm first traverses a priori search space which is a set of service patterns from historical solutions, then it searches available services from the repositories. That means each layer contains all services and service patterns that can be executed with a set of outputs provided by previous layers. The search process terminates until all the outputs of a request are obtained. This candidate set has the ability to construct all possible solutions that fulfill the request. A solution consists of a set of layered services (and service patterns) which could be connected in a specified executable path.

Priori knowledge based forward search algorithm is shown in Alg. 2. Initialization of the first layer is in lines 2-7, a dummy service S_{In} is created, which does not have any input concept and its output concepts are the same as input concepts of a request r. Variable i indicates current layer, L_i

is a set of candidates contained in layer i, and Out_i is a set of output concepts generated by these candidates. $\mathbb{C}$ indicates a set of semantic concepts achieved during the execution of the algorithm, or it can be considered as the intersection of all output concepts from previous layers. The algorithm first searches service patterns mined from three matrices, which presents the potential knowledge from historical service solutions (lines 11-17). After scanning each service pattern, the algorithm also considers available services from the priori knowledge when all input parameters can be matched with obtained semantic concepts (lines 18-24). Iterative search process terminates when the algorithm gets all output concepts of a request, or there is no new candidate generated which can be added into service solution. At last, if all desired semantic concepts are obtained, a dummy service S_{Out} is created, whose input concepts are the same as output concepts of a request r and it does not generate any output concepts (line 28), otherwise it continues with executing A*-TSC algorithm.

Algorithm 2: Priori knowledge based forward search algorithm

```
   Input: Service request $r = (In_r, Out_r)$, all services $S$, all service patterns
          $SP$, similarity matrix $SimM$, relativity matrix $RelM$, mapping matrix
          $MapM$
   Output: Layered candidates $LCS = \{L_1, L_2, \cdots\}$
 1 begin
 2     Calculate related services $S'$ and service patterns $SP'$ by
         $SimM$, $RelM$ and $MapM$
 3     $\mathbb{C} \leftarrow \emptyset, i \leftarrow 0, L_i \leftarrow \emptyset, Out_i \leftarrow \emptyset$ /*$i$ indicates current layer */
 4     Create a dummy service $S_{In} = (\emptyset, In_r)$
 5     $L_i \leftarrow L_i \cup S_{In}$ /*$L_i$ indicates candidate set in layer $i$*/
 6     $Out_i \leftarrow Out_i \cup In_r$ /*$Out_i$ indicates obtained semantic concepts by
         layer $i$*/
 7     $\mathbb{C} \leftarrow \mathbb{C} \cup Out_i$ /*$\mathbb{C}$ indicates obtained semantic concepts so far*/
 8     while $\mathbb{C} \oplus Out_r \neq Out_r \wedge L_i \neq \emptyset$ do
 9         $i \leftarrow i + 1$
10         $L_i \leftarrow \emptyset, Out_i \leftarrow \emptyset$
11         foreach $sp_j \in SP'$ /*Search for available service patterns*/ do
12             if $\mathbb{C} \oplus In_{sp_j} = In_{sp_j}$ then
13                 $L_i \leftarrow L_i \cup sp_j$
14                 $Out_i \leftarrow Out_i \cup Out_{sp_j}$
15                 $SP' \leftarrow SP' \setminus sp_j$
16             end
17         end
18         foreach $s_k \in S'$ /*Search for available services*/ do
19             if $\mathbb{C} \oplus In_{s_k} = In_{s_k}$ then
20                 $L_i \leftarrow L_i \cup s_k$
21                 $Out_i \leftarrow Out_i \cup Out_{s_k}$
22                 $S' \leftarrow S' \setminus s_k$
23             end
24         end
25         $\mathbb{C} \leftarrow \mathbb{C} \cup Out_i$
26     end
27     if $\mathbb{C} \oplus Out_r = Out_r$ then
28         Create a dummy service $S_{Out} = (Out_r, \emptyset)$
29         $i \leftarrow i + 1$
30         $L_i \leftarrow \emptyset$
31         $L_i \leftarrow L_i \cup S_{Out}$
32     else
33         Execute A*-TSC algorithm
34     end
35 end
```

B. HEURISTIC BACKWARD SEARCH FOR EXECUTABLE SOLUTION

Once the possible candidates including available services and service patterns are calculated after priori knowledge based forward search, a backward search needs to be performed to identify the optimal execution path. The backward search algorithm traverses the candidates backward, from the dummy service S_{In_r} to S_{Out_r} in Fig. 6, which is composed of two steps. The first step generates all possible service solutions by connecting these layered candidates. The second step finds out the optimal service solution with a heuristic approach.

1) Step 1:

For each layer, available candidates are put into a list, and these candidates can be executed in parallel. A set of candidates selected from each list constitutes a path, which represents the execution sequence. Firstly, the algorithm needs to identify the candidate that provides each input concepts of S_{Out_r} from the list of the last layer. If there is no candidate in the last layer for that input concept, a dummy service is created to keep the continuity of execution path, which indicates that this input concept has already been generated from a previous layer before the last layer. The input and output of this dummy service are the same as the missing concept. Secondly, the algorithm calculates combinations of these selected candidates. These combinations generate all possible neighbors from the current list. Step 1 ends after all possible paths have been connected.

For example, given the service S_{Out_r} in the layer L_5, with $In = \{X, Y\}$ and a set of candidates a, b, c, d, e in the layer L_4 where $Out_a = \{X\}$, $Out_b = \{Y\}$, $Out_c = \{X, Y\}$, $Out_d = \{X\}$, $Out_e = \{Y\}$, we construct a list of services for each input parameter of S_{Out_r}: Set(X) = {a,c,d} and Set(Y) = {b,c,e}. Then, we generate all combinations. Each combination will constitute a neighbor from S_{Out_r}. The possible combinations are: (a,b), (a,c), (a,e) (c,b), (c), (c,e), (d,e). All these combinations generate the required input parameters for S_{Out_r}.

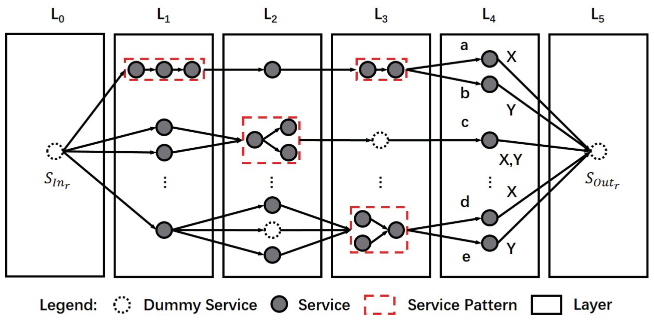


FIGURE 6: An example of heuristic backward search

2) Step 2:

Sol is a composite service obtained as a path over a set of candidates from different layers. The goal is to minimize the number of services in a composition, therefore, the function *cost* should calculate the length of a composition based on

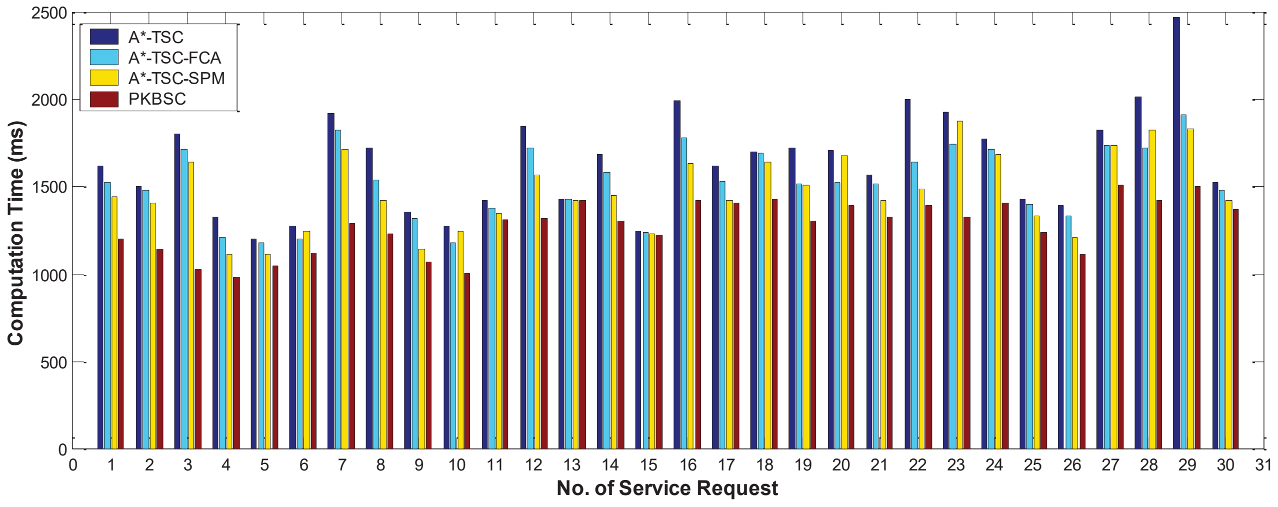


FIGURE 7: Performance with different approaches (color for better reading result).

the number of services. On this basis, we define a heuristic function $H(Sol)$ as follows.

$$H(Sol) = Minimize \sum_{i=1}^{N} cost(L_i) \tag{8}$$

where L_1 is the first layer of the current composition service, L_N is the last layer and $cost$ is a function that retrieves the number of services from each layer. The dummy services in a candidate list will not be computed.

VIII. EXPERIMENTAL ANALYSIS

In this section, we conduct two experiments to verify the effectiveness of our proposed approach. First, we compare the performance of Priori Knowledge Based Service Composition algorithm (PKBSC) with the A* based traditional graph searching service composition algorithm (A*-TSC) [17]. Then we aim to discover the impact factors of PKBSC in terms of search space variation, which could be affected by the time-aware frequency of service patterns.

A. DATASET

The dataset used in our experiments collects 2,037 real-world Web services from public datasets including QWS [58], WS-DREAM [59], OWLS [60] and Titan [61]. Exact I/O parameters of these services are extracted from their WSDL (Web Services Description Language) files. Since there are no standard datasets that aggregate customers' requests on real-world Web services, we generate a synthetic service request set based on Toronto 311 service dataset[1]. This dataset records the requests for a public service, which are submitted by citizens to report relevant problems. Apparent temporal (i.e., when the request is submitted) and geographical (i.e., where the request is from) distribution tendency can be identified, which have a direct influence on the construction of service solution. We transform each public request into a requirement of composite service by allocating semantic concepts in a domain ontology. The transformation follows the temporal/geographical distributions of the original requests so that the underlying variation tendency of requests are kept without loss. Due to limited space, we do not introduce details of this transformation. The dataset has 466,090 records in total from Jan. 2012 to Jun. 2013. We randomly divide the entire dataset into 80% training dataset and 20% test dataset. The large training dataset is used to generate historical solutions. For each request, a service solution is generated by the A*-TSC algorithm. The test dataset is used to verify the performance of PKBSC.

B. RESULT FOR PERFORMANCE COMPARISON

In the first experiment, we compare the performance of PKBSC with A*-TSC to show how the priori knowledge improves the effectiveness of graph searching in service composition. In addition, two variants of A*-TSC are also

[1]available for downloading from https://www.toronto.ca/311

TABLE 1: Results of performance

	Avg. computation time (ms)	Avg. number of layer	Avg. size of solution	Improvement
A*-TSC	1644.2	8.04	**8.96**	-
A*-TSC-FCA	1526.167	8.03	**8.96**	7.18%
A*-TSC-SPM	1475.367	**4.57**	**8.96**	10.27%
PKBSC	**1194.667**	**4.57**	**8.96**	**22.44%**

considered as comparative objects, which are A*-TSC with service pattern mining (A*-TSC-SPM) and A*-TSC with formal concept analysis (A*-TSC-FCA). This is to analyze which is more important in the components of PKBSC. In order to ensure the fairness of experiments, all methods utilize the same similar matrix which solves the problem of heterogeneity caused by service in different repositories. The total number of candidate services is 2,000, the number of historical solutions is 10,000. The minimal support for service pattern mining is $\varphi = 0.03$, and the minimal support and confidence for formal concept analysis is $[\delta, \sigma] = [0.03, 0.1]$. We randomly choose 30 requests from the test dataset to demonstrate the computation time of different approaches. The results are shown in Fig. 7 and the average experiment results are listed in Tab. 1.

The computation time is dynamically changed according to the search space which is affected jointly by the number of layers in a service solution and the candidates' amount in each layer. PKBSC has the best performance for each service request, compared with A*-TSC, efficiency increased by an average of 22.4%. The main reason is that SPM provides priori knowledge to reduce the number of layers for PKBSC, while FCA can effectively retrieve priori knowledge related to service requests, which reduce the number of candidate service patterns. A*-TSC-FCA only can retrieve related services as the priori knowledge, the granularity of related services is smaller than service patterns in most cases. It cannot reduce the number of layers, therefore it performs worst among the three improved approaches. Different from PKBSC, in A*-TSC-SPM all service patterns are considered as the priori knowledge, which are mined from the whole training dataset. However, prior knowledge contains a serious of irrelevant service patterns, the lack of effective retrieval strategy makes A*-TSC-SPM cannot get the desired results.

C. IMPACT ANALYSIS WITH DIFFERENT SELECTION STRATEGIES

We try to find out whether the frequency of service pattern is changing over time. We calculate the count of occurrence for each service pattern per month, which can exhibit a monthly variation tendency of frequency. In Fig. 8, we use a heatmap to show frequent variation tendency of 100 service patterns. The variance of service pattern is divided into three different types. The first type is stable, there are 34 service patterns belong to this type, that means these service patterns change little over time. The second type is volatile, there are 14 service patterns belong to this type. The counts of these service patterns are significantly high during some months, and it is arbitrary and irregular. The last one is gentle undulation, which is a large majority of service patterns, i.e, 52 service patterns. The conclusion from these experiments is that the frequency of service patterns indeed changes over time. So, we compare the performance of PKBSC under different selection strategies of service patterns. In each strategy, we select the frequent service patterns from different periods as reusable knowledge or a priori search space. Strategy 1 mines service patterns from historical solutions of last year; Strategy 2 considers service patterns from the corresponding month of last year; service patterns of Strategy 3 are from corresponding period of last year, which includes corresponding month and two months before it; Strategy 4 is also three month, i.e., from one month before corresponding month to next month after it.

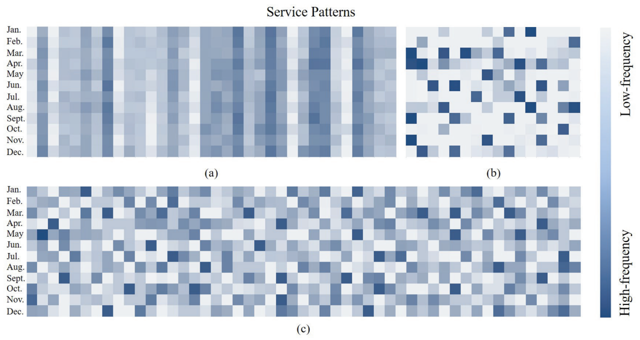


FIGURE 8: Time-aware frequency of service patterns: (a) Stable, (b) Volatile, (c) Gentle undulation (color for better reading result).

The performance with different strategies is shown in Fig. 9. We can see that the computation time of Strategy 2 is fluctuating drastically, that means if service patterns from corresponding month are used to construct solution the result is really well, otherwise, it takes more time to find relevant services. The average improvement is 5.18% comparing with Strategy 1. The performance of Strategy 3 and Strategy 4 has the same changing trend. They are both better than Strategy 1 in most cases. The average improvements are 9.15% and 9.83% respectively.

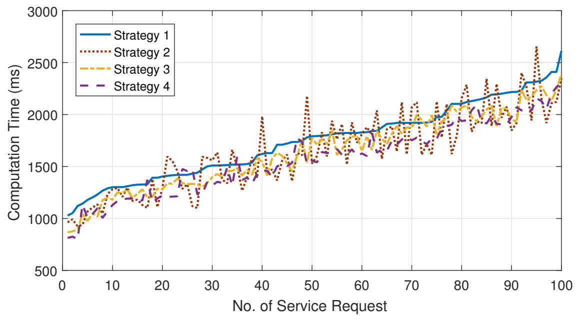


FIGURE 9: Performance with different strategies (color for better reading result).

IX. CONCLUSION AND FUTURE WORK

In this paper, we present priori knowledge based service composition (PKBSC) that assists the discovering and composition of services which best match user needs. We solve the heterogeneous service interoperability caused by different service repositories. We obtain prior knowledge by mining service patterns from historical records and quickly retrieve prior knowledge by establishing the relationship between requests and service patterns. We conduct a series of experiments using both real and synthetic data to evaluate the efficiency and effectiveness of our approach. We improve the efficiency of A* based traditional service composition algorithm (A*-TSC) by 22.44%. We conduct a series of experiments analyzing system performance in terms of the time-aware frequency of service patterns.

Our future work will focus on how to integrate QoS into service patterns and build service solutions through functional and non-functional requests.

REFERENCES

[1] A. L. Lemos, F. Daniel, and B. Benatallah, "Web service composition: a survey of techniques and tools," ACM Computing Surveys (CSUR), vol. 48, no. 3, p. 33, 2016.

[2] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana et al., "Web services description language (WSDL) 1.1," 2001. [Online]. Available: https://www.w3.org/TR/wsdl/

[3] M. J. Hadley, "Web application description language (WADL)," 2006.

[4] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne et al., "OWL-S: Semantic markup for web services," W3C member submission, vol. 22, no. 4, 2004.

[5] R. Akkiraju, J. Farrell, J. A. Miller, M. Nagarajan, A. P. Sheth, and K. Verma, "Web service semantics-WSDL-S," 2005. [Online]. Available: https://corescholar.libraries.wright.edu/knoesis/69

[6] D. Roman, U. Keller, H. Lausen, J. De Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel, "Web service modeling ontology," Applied ontology, vol. 1, no. 1, pp. 77–106, 2005.

[7] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," Computer networks, vol. 54, no. 15, pp. 2787–2805, 2010.

[8] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2347–2376, 2015.

[9] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, "IoT middleware: A survey on issues and enabling technologies," IEEE Internet of Things Journal, vol. 4, no. 1, pp. 1–20, 2017.

[10] Y. Qin, Q. Z. Sheng, N. J. Falkner, S. Dustdar, H. Wang, and A. V. Vasilakos, "When things matter: A survey on data-centric internet of things," Journal of Network and Computer Applications, vol. 64, pp. 137–153, 2016.

[11] X. Xu, Q. Z. Sheng, L.-J. Zhang, Y. Fan, and S. Dustdar, "From big data to big service," Computer, vol. 48, no. 7, pp. 80–83, 2015.

[12] X. Xu, R. Liu, Z. Wang, Z. Tu, and H. Xu, "RE2SEP: A two-phases pattern-based paradigm for software engineering," in Services (SERVICES), 2017 IEEE World Congress on. IEEE, 2017, pp. 67–70.

[13] E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. Nau, "HTN planning for web service composition using SHOP2," Web Semantics: Science, Services and Agents on the World Wide Web, vol. 1, no. 4, pp. 377–396, 2004.

[14] E. Sirin, B. Parsia, and J. Hendler, "Template-based composition of semantic web services," in Proceedings of the 1st Int. AAAI Fall Symposium on Agents and the Semantic Web, vol. 5. AAAI, 2005, p. 01.

[15] P. Hennig and W.-T. Balke, "Highly scalable web service composition using binary tree-based parallelization," in Proceedings of IEEE International Conference on Web Services (ICWS). IEEE, 2010, pp. 123–130.

[16] S.-C. Oh, D. Lee, and S. R. Kumara, "Web service planner (WSPR): An effective and scalable web service composition algorithm," International Journal of Web Services Research (IJWSR), vol. 4, no. 1, pp. 1–22, 2007.

[17] P. Rodriguez-Mier, M. Mucientes, and M. Lama, "Automatic web service composition with a heuristic-based search algorithm," in Proceedings of IEEE International Conference on Web Services (ICWS). IEEE, 2011, pp. 81–88.

[18] R. Liu, X. Xu, Z. Wang, Q. Z. Sheng, and H. Xu, "Probability matrix of request-solution mapping for efficient service selection," in Web Services (ICWS), 2017 IEEE International Conference on. IEEE, 2017, pp. 444–451.

[19] K. P. Joshi, Y. Yesha, and T. Finin, "Automating cloud services life cycle through semantic technologies," IEEE Transactions on Services Computing, vol. 7, no. 1, pp. 109–122, 2014.

[20] A. V. Paliwal, B. Shafiq, J. Vaidya, H. Xiong, and N. Adam, "Semantics-based automated service discovery," IEEE Transactions on Services Computing, vol. 5, no. 2, pp. 260–275, 2012.

[21] J. Zhang, J. Wang, P. Hung, Z. Li, N. Zhang, and K. He, "Leveraging incrementally enriched domain knowledge to enhance service categorization," International Journal of Web Services Research (IJWSR), vol. 9, no. 3, pp. 43–66, 2012.

[22] A. A. Patil, S. A. Oundhakar, A. P. Sheth, and K. Verma, "s," in Proceedings of the 13th International World Wide Web Conference(WWW). ACM, 2004, pp. 553–562.

[23] Z. Duo, L. Juan-Zi, and X. Bin, "Web service annotation using ontology mapping," in Proceedings of IEEE International Workshop on Service-Oriented System Engineering (SOSE). IEEE, 2005, pp. 235–242.

[24] I. Salomie, V. R. Chifu, I. Giurgiu, and M. Cuibus, "SAWS: A tool for semantic annotation of web services," in Proceedings of IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR). IEEE, 2008, pp. 387–392.

[25] Y. Chabeb, S. Tata, and D. Belaïd, "Toward an integrated ontology for web services," in Proceedings of the Fourth International Conference on Internet and Web Applications and Services (ICIW). IEEE, 2009, pp. 462–467.

[26] A. Ranabahu, P. Parikh, M. Panahiazar, A. Sheth, and F. Logan-Klumpler, "Kino: a generic document management system for biologists using SA-REST and faceted search," in Proceedings of the Fifth IEEE International Conference on Semantic Computing (ICSC). IEEE, 2011, pp. 205–208.

[27] A. Segev and E. Toch, "Context-based matching and ranking of web services for composition," IEEE Transactions on Services Computing, vol. 2, no. 3, pp. 210–222, 2009.

[28] Y. Han, S. Chen, and Z. Feng, "Mining integration patterns of programmable ecosystem with social tags," Journal of grid computing, vol. 12, no. 2, pp. 265–283, 2014.

[29] H. Khanh Dam, "Predicting change impact in Web service ecosystems," International Journal of Web Information Systems, vol. 10, no. 3, pp. 275–290, 2014.

[30] C.-H. Lee, S.-Y. Hwang, and I.-L. Yen, "A service pattern model for flexible service composition," in Proceedings of IEEE 19th International Conference on Web Services (ICWS). IEEE, 2012, pp. 626–627.

[31] O. Shafiq, R. Alhajj, and J. G. Rokne, "Reducing search space for web service ranking using semantic logs and semantic FP-tree based association rule mining," in Proceedings of IEEE International Conference on Semantic Computing (ICSC). IEEE, 2015, pp. 1–8.

[32] B. Vollino and K. Becker, "Usage profiles: A process for discovering usage patterns over web services and its application to service evolution," International Journal of Web Services Research (IJWSR), vol. 10, no. 1, pp. 1–28, 2013.

[33] C. Kumar, "Knowledge discovery in data using formal concept analysis and random projections," International Journal of Applied Mathematics and Computer Science, vol. 21, no. 4, pp. 745–756, 2011.

[34] U. Priss, "Formal concept analysis in information science," Annual review of information science and technology, vol. 40, no. 1, pp. 521–543, 2006.

[35] C. A. Kumar, M. Radvansky, and J. Annapurna, "Analysis of a vector space model, latent semantic indexing and formal concept analysis for information retrieval," Cybernetics and Information Technologies, vol. 12, no. 1, pp. 34–48, 2012.

[36] C. A. Kumar, "Designing role-based access control using formal concept analysis," Security and communication networks, vol. 6, no. 3, pp. 373–383, 2013.

[37] J. Poelmans, D. I. Ignatov, S. O. Kuznetsov, and G. Dedene, "Formal concept analysis in knowledge processing: A survey on applications," Expert systems with applications, vol. 40, no. 16, pp. 6538–6560, 2013.

[38] M. Daagi, A. Ouniy, M. Kessentini, M. M. Gammoudi, and S. Bouktif, "Web service interface decomposition using formal concept analysis," in Web Services (ICWS), 2017 IEEE International Conference on. IEEE, 2017, pp. 172–179.

[39] H. Naim, M. Aznag, M. Quafafou, and N. Durand, "Probabilistic approach for diversifying web services discovery and composition," in Web Services (ICWS), 2016 IEEE International Conference on. IEEE, 2016, pp. 73–80.

[40] M. Aznag, M. Quafafou, and Z. Jarir, "Leveraging formal concept analysis with topic correlation for service clustering and discovery," in Web Services (ICWS), 2014 IEEE International Conference on. IEEE, 2014, pp. 153–160.

[41] C. De Maio, G. Fenza, V. Loia, and S. Senatore, "Hierarchical web resources retrieval by exploiting fuzzy formal concept analysis," Information Processing & Management, vol. 48, no. 3, pp. 399–418, 2012.

[42] A. Formica, "Semantic web search based on rough sets and fuzzy formal concept analysis," Knowledge-Based Systems, vol. 26, pp. 40–47, 2012.
[43] A. Segev and Q. Z. Sheng, "Bootstrapping ontologies for web services," IEEE Transactions on Services Computing, vol. 5, no. 1, pp. 33–44, 2012.
[44] M. Nagy and M. Vargas-Vera, "Multiagent ontology mapping framework for the semantic web," IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, vol. 41, no. 4, pp. 693–704, 2011.
[45] S. Raunich and E. Rahm, "ATOM: Automatic target-driven ontology merging," in Data Engineering (ICDE), 2011 IEEE 27th International Conference on. IEEE, 2011, pp. 1276–1279.
[46] S. Amrouch and S. Mostefai, "Survey on the literature of ontology mapping, alignment and merging," in Information Technology and e-Services (ICITeS), 2012 International Conference on. IEEE, 2012, pp. 1–5.
[47] N. F. Noy, "Semantic integration: a survey of ontology-based approaches," ACM Sigmod Record, vol. 33, no. 4, pp. 65–70, 2004.
[48] L. Meng, R. Huang, and J. Gu, "A review of semantic similarity measures in wordnet," International Journal of Hybrid Information Technology, vol. 6, no. 1, pp. 1–12, 2013.
[49] D. Sánchez, M. Batet, D. Isern, and A. Valls, "Ontology-based semantic similarity: A new feature-based approach," Expert systems with applications, vol. 39, no. 9, pp. 7718–7728, 2012.
[50] C. Jiang, F. Coenen, and M. Zito, "A survey of frequent subgraph mining algorithms," The Knowledge Engineering Review, vol. 28, no. 01, pp. 75–105, 2013.
[51] R. Wille, "Restructuring lattice theory: An approach based on hierarchies of concepts," in International Conference on Formal Concept Analysis. Springer, 2009, pp. 314–339.
[52] B. Ganter, G. Stumme, and R. Wille, Formal concept analysis: foundations and applications. springer, 2005, vol. 3626.
[53] C. Aswani Kumar and S. Srinivas, "Mining associations in health care data using formal concept analysis and singular value decomposition," Journal of biological systems, vol. 18, no. 04, pp. 787–807, 2010.
[54] C. A. Kumar, "Fuzzy clustering-based formal concept analysis for association rules mining," Applied artificial intelligence, vol. 26, no. 3, pp. 274–301, 2012.
[55] J. Li, C. Mei, C. A. Kumar, and X. Zhang, "On rule acquisition in decision formal contexts," International journal of machine learning and cybernetics, vol. 4, no. 6, pp. 721–731, 2013.
[56] N. Moha, F. Palma, M. Nayrolles, B. J. Conseil, Y.-G. Guéhéneuc, B. Baudry, and J.-M. Jézéquel, "Specification and detection of soa antipatterns," in International Conference on Service-Oriented Computing. Springer, 2012, pp. 1–16.
[57] H. Wang, M. Kessentini, and A. Ouni, "Bi-level identification of web service defects," in International Conference on Service-Oriented Computing. Springer, 2016, pp. 352–368.
[58] E. Al-Masri and Q. H. Mahmoud, "QoS-based discovery and ranking of web services," in Proceedings of 16th International Conference on Computer Communications and Networks, 2007, pp. 529–534.
[59] Z. Zheng, Y. Zhang, and M. R. Lyu, "Distributed QoS evaluation for real-world web services," in Proceedings of IEEE International Conference on Web Services (ICWS), 2010, pp. 83–90.
[60] G. Meditskos and N. Bassiliades, "Structural and role-oriented web service discovery with taxonomies in OWL-S," IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 2, pp. 278–290, 2010.
[61] J. Wu, L. Chen, Y. Xie, and Z. Zheng, "Titan: a system for effective web service discovery," in Proceedings of the 21st International World Wide Web Conference (WWW), 2012, pp. 441–444.

RUILIN LIU is currently pursuing toward the PhD degree at the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China. His current research interests include service computing and service composition optimization.

XIAOFEI XU is a professor at School of Computer Science and Technology, Harbin Institute of Technology (HIT). He received the PhD degree in computer science from HIT in 1988. His research interests include enterprise computing, services computing, Internet of services, and data mining. He is the associate chair of IFIP TC5 WG5.8, chair of INTEROP-VLab China Pole, fellow of China Computer Federation (CCF), and vice director of the technical committee of service computing of CCF. He is the author of over 300 publications. He is a member of the IEEE and ACM.

ZHONGJIE WANG is a professor at School of Computer Science and Technology, Harbin Institute of Technology (HIT). He received the PhD degree in computer science from Harbin Institute of Technology in 2006. His research interests include services computing, mobile and social networking services, software architecture, social software engineering, and software repositories mining. He is a member of the IEEE.

QUAN Z. SHENG is a full Professor and Head of Department of Computing at Macquarie University, Australia. Professor Sheng holds a PhD degree in computer science from the University of New South Wales (UNSW), Sydney, Australia. His research interests include Internet of Things, Web of Things, big data analytics, service computing, distributed computing, and Internet technologies. He is the recipient of ARC Future Fellowship in 2014, Chris Wallace Award for Outstanding Research Contribution in 2012, and Microsoft Research Fellowship in 2003. He is the author of more than 330 publications. He is a member of the ACM and the IEEE.