Journal of Web Engineering, Vol. 16, No.3&4 (2017) 252-276

AN APPROACH OF WEB SERVICE ORGANIZATION USING BAYESIAN NETWORK LEARNING

JIANXIAO LIU*

College of Informatics, Huazhong Agricultural University, Wuhan
School of Computer & Software, Nanjing University of Information Science & Technology, Nanjing
**Corresponding author liujianxiao321@163.com*

ZHIHUA XIA

Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science & Technology, Nanjing
School of Computer & Software, Nanjing University of Information Science & Technology, Nanjing
xia_zhihua@163.com

Received August 7, 2015
Revised December 15, 2016
\

How to organize and manage Web services, and help users to select the atomic and a set of services with correlations to meet their functional and non-functional requirements quickly is a key problem to be solved in the era of services computing. Firstly, it uses the three-stage dependency Bayesian network structure learning method to organize service clusters which realize different functions. Then it uses the maximum likelihood estimation and Bayesian estimation methods to do the parameter learning, and the conditional probability table (CPT) of all the nodes can be got. This method can help users select a set of services with better function in the organized services quickly and accurately. Finally, the effectiveness of the proposed method is validated through experiments and case study.

Key words: Web Service, Bayesian Network Structure Learning, Bayesian Estimation, Service Organization
Communicated by: D. Schwabe & E-P Lim

1 Introduction

In the era of service-oriented computing, users not only have the functional requests for Web services, but also have the non-functional QoS requirements. In addition, users not only need the atomic service, but also a set of services that are related to a specific topic, such as the services which are related to gene information inquiry. How to organize and manage Web services effectively, and facilitate users to find services with the proper functional and non-functional values in the organized services is an important problem to be solved in the service-oriented software engineering [1].

Service organization refers to organize all kinds of Web services in the service registry using certain mechanism. Most of the existing research work use the clustering and classification approaches

to organize services [2]. Services with the same or similar function are clustered into different service clusters, and services in the same cluster often have different QoS values [3]. However, there are all kinds of services which realize different functions on the internet, and users often need a set of services with different functions. In addition, users usually need a set of services which are related to a specific topic or a particular domain problem [4]. Therefore, it needs to organize services with different functions, such as the services of inquiry gene information, searching the gene regulation relationships, searching the relationship between gene and protein. At present, there are some research work about service organization, such as the business service workflow-based method [5], community-based method [6], VINCA [7], logical Petri nets [8], etc. These approaches mainly consider the service execution process and behaviour, and use the semantic matching method to organize Web services. But some approaches are lacking of the consideration of the services with the same or similar function. This leads to the services which have different functions can't be found, and the QoS of the services can't meet users' request well. This is not compatible with users' personal requirements. Bayesian network combines the acyclic graphs and probability theory, and it has solid theory foundation of probability. It has the advantages of constructing causal relationship, doing reasoning, mining the implicit knowledge and so on. Aiming at solving the problems of existing service organization approaches, the main work of the paper is given as follows.

(1) It proposes a Web service organization method based on Bayesian network structure learning. The method makes full use of the service invocation history records, and it uses the three-stage dependency learning method to organize service clusters, and thus to get the service organization network structure.

(2) The Bayesian network parameter learning method (maximum likelihood estimation) is used to learn and get the conditional probability table of all the nodes in the service organization network graph. It can help to realize Web service recommendation using Bayesian network reasoning method.

(3) Experiments and case study are conducted to validate the proposed methods.

The rest of the paper is organized as follows: the basic definition is given in section 2. The algorithms of realizing Web service organization are elaborated in section 3. In section 4, the case study is used to explain the proposed algorithms. The related work is described in section 5. In section 6, we conduct experiments to verify the proposed approaches. The conclusion and future work are given finally.

2 Basic Definition

2.1 Bayesian theory

Definition 1. Conditional probability. Given A and B are two events, and $p(\mathrm{B})>0$, the occurrence probability of event A in the condition of event B is shown in Eq.(1).

$$p(\mathrm{A} \mid \mathrm{B}) = \frac{p(\mathrm{AB})}{p(\mathrm{B})} \tag{1}$$

Definition 2. Total probability formula. Supposing the events of B_1, B_2,…$B_n \in \Re$, and they constitute a complete events set. $\bigcup_{i=1}^{n} B_i = \Omega$, Ω is the sample set. For each event A, the occurrence probability of A is shown in Eq.(2).

$$p(\mathrm{A}) = \sum_{i=1}^{n} p(\mathrm{B}_i) p(\mathrm{A} \mid \mathrm{B}_i) \tag{2}$$

Definition 3. Bayesian formula. Supposing events of A_1, A_2,…$A_n \in \Re$, and each of them is not compatible with the others, $p(A_i)>0$, $i=1,2,\ldots,n$. For each event $B(p(B)>0)$, we can get $p(A_m \mid B)$, as shown in Eq.(3)

$$p(\mathrm{A}_m \mid \mathrm{B}) = \frac{p(B \mid \mathrm{A}_m) p(\mathrm{A}_m)}{\sum_{i=1}^{n} p(B \mid \mathrm{A}_i) p(\mathrm{A}_i)} \tag{3}$$

2.2 Bayesian network

Bayesian probability expresses the confidence of the occurrence of an event. Bayesian probability is the foundation of Bayesian network, which is called as belief network. A complete Bayesian network includes three parts: nodes in Bayesian network, edges between nodes and the conditional probability of all the nodes.

Definition 4. Bayesian Network. It is defined as a tuple of $B=\{S, P\}$.

- $S=\{X, E\}$, it expresses Bayesian network structure, and:
  $X=\{x_i, 0 \le i \le num\}$, it is denoted as the nodes in Bayesian network.
  $E=\{x_i \rightarrow x_j, 0 \le i \le num, 0 \le j \le num\}$, it is denoted as the edges between nodes.
- $P=\{p(x_i \mid pa(x_i)), x_i \in X\}$, it is denoted as the conditional probability table(CPT) of all the nodes in B, and:
  $pa(x_i)=\{x_p, x_p \rightarrow x_i \wedge (x_p \rightarrow x_i) \in E \wedge x_p \in X \wedge x_i \in X\}$, it is denoted as the parent node of x_i.

In the above definition, $p(x_i \mid pa(x_i))$ expresses the conditional probability of the $pa(x_i)$.

The conditional probability of all the nodes in B constitutes the conditional probability table of Bayesian network [9]. The joint probability distribution is equal to the product of conditional probability, as shown in Eq.(4). And $pa(x_i)$ is the parent node set of x_i.

$$p(X) = \prod_{\mathrm{x}_i \in X} p(\mathrm{x}_i \mid pa(\mathrm{x}_i)) \tag{4}$$

Bayesian network structure learning refers to find a network with the best fit for the given data set. It includes the following three steps to construct a Bayesian network.

(1) Determine the variables and domain of the variables. It will determine the variables of all the nodes in X of Bayesian network $S=\{X, E\}$.

(2) Structure learning. It will determine the dependency relationships between variables, and the directed acyclic graph is used to express the network structure.

(3) Parameter learning. It will learn the distribution between variables, and get the conditional probability table(*CPT*) of all the variables.

2.3 Web service

We use the status transition to express the capability of Web services. We define Web services as *ws*={*WSName*, *Interface*, *Capability*, *QoS*}, and:

WSName represents the name of *ws*.

Interface={*Input*,*Output*}, it denotes the input and output set of *ws*.

Capability={*Precondition*, *Effect*}, it indicates the prerequisite for service execution and the effect resulting from the execution of *ws*.

QoS={{$QosName_q$,$Value_q$}, $QosName_q$ can be time, cost, reliability and availability of *ws*. $Value_q$ represents the specific value of *QoS*.

The concrete definition of *ws* can be seen in [10].

Definition 5. Service cluster(*Cluster*). It is defined as *Cluster*={$clusws_c$, $1 \le c \le cnum$}, $clusws_c$={ws_{cw}, $1 \le w \le c_c$}.

Cluster expresses different service clusters, and the specific service cluster $clusws_c$ includes different Web services ws_{cw}. The services in the same cluster realize similar function, but have different QoS values.

3 Web Service Organization

3.1 Bayesian network structure learning

There are two kinds of Bayesian network structure learning methods: search score method and dependency analysis method [11]. This search score method uses the local or random search strategy. It is a combinatorial explosion problem as the number of nodes increases, and it leads to the efficiency of this method is too low. The efficiency of the dependency analysis method is relatively high, and it also can get the global optimal solution. Therefore, we mainly use the dependency analysis method to construct the Bayesian network. The three-phase dependency analysis algorithm (*TPDA*) is a commonly used dependency analysis method, and this algorithm determines the conditional independence between nodes through calculating the mutual information [12]. The execution correlation degree between Web services can be expressed by the mutual information between them. We construct the directed graph model to represent the inter-organization relationships between services. In addition, the number of conditional independency testing in sparse graph is relatively small, so the conditional independence test method is more suitable for sparse graphs. In the view of the relationship between services is sparse, we use the three-stage dependency learning algorithm to construct the service organization network graph.

The process of realizing *TPDA* method mainly includes three steps: *Drafting*, *Thickening* and *Thinning*. The first stage is *Drafting.* The correlation degree between any two nodes is measured through calculating the mutual information between them. When the mutual information is greater than the threshold, it means there exists an edge between the corresponding nodes. The initial network will be constructed using the above method. The second stage is conditional mutual information judgement (*Thickening*). It firstly finds the cut set *C* between two nodes when there is an open path between them

in the network. Then the conditional mutual information about the two nodes and C will be calculated, and we will judge whether it is conditionally independent. If it is not independent, the corresponding edge will be added into the graph, and then the network of *I-MAP* can be got. The third stage is *Thinning*. For each edge e in the graph, it will be removed temporarily. Then we will find the minimum cut set C_{min} between the nodes of e, and judge whether they are conditional independent or not. If they are conditional independent, e will be deleted. Otherwise, e will be added into the network again, and finally get *P-MAP*.

The first learning stage: *Drafting*

This stage mainly uses the history log information of Web service invocation. We map the history log information between Web services in different service clusters into the invocation information. The mutual information between service clusters will be calculated and the initial service organization network can be constructed.

Algorithm 1. The first stage learning algorithm (*Drafting*)

Input: $Cluster=\{clusws_c,\ 1\le c\le cnum\}$, $clusws_c=\{ws_{cw},\ 1\le w\le c_c\}$, $Rel_{ws}=\{rel_r:\ ws_{ij}\rightarrow ws_{mn}, 1\le r\le rnum, 0\le i,m\le cnum, 0\le j\le c_i, 0\le n\le c_m\}$

Output: *graph*, *R*

1: c_1, c_2←0, *S*←∅, v_I←0, *R*←∅
2: *Node*[] *nodes*←new *Node* [*cnum*]
3: *graph*←new *Graph*(*nodes*,*cnum*)
4: **for** *c*=1 **to** *cnum* **do**
5 $graph.nodes[c]\leftarrow Cluster.clusws_c$
6: **end for**
7: **for** c_1=1 **to** *Cluster.cnum* **do**
8: **for** c_2=1 **to** *Cluster.cnum* **do**
9: $v_I\leftarrow Imutual(clusws_{c1}, clusws_{c2}, Rel_{ws})$
10: **if**(v_I >ε) **then**
11: $S\leftarrow S\cup <clusws_{c1}, clusws_{c2,}\ v_I>$
12: **end if**
13:**end for**
14:*S*←*Sort*(*S*)//Sort *S* according to $Imutual(clusws_{c1}, clusws_{c2}, Rel_{ws})$
15:**for all** $<clusws_{c1}, clusws_{c2,}\ Imutual(clusws_{c1}, clusws_{c2}, Rel_{ws})$ in *S* **do**
16: **if**($ExistsPath(node_{c1}, node_{c2})$) **then** //exists the open path
17: $R\leftarrow R\cup <clusws_{c1}, clusws_{c2}>$
18: **else** *graph*.insert(new $Edge(clusws_{c1}, clusws_{c2})$)
19:**end for**
20:return *graph*, *R*

In Algorithm 1, the Rel_{ws} stores the service invocation information, including the services whose *Output* and *Input* are matched, the services whose *Effect* and *Precondition* are matched. It also stores the services that can be composited to realize specific function, and these services are often invocated together by users. In the algorithm, the service clusters are seen as nodes in graph. The initial network graph will be constructed firstly using step 2-6. The mutual information calculation formula $I(clusws_i, clusws_m, Rel_{ws})$ is used to calculate the mutual information between two service cluster nodes. The

edges whose nodes' mutual information is more than the threshold(ε) will be added into *S*. Then it will sort the node pair in *S* according to the value of mutual information, as seen in step 7-14. The node pair in *S* are judged in turn to see if there exists an open path between them. If there exists an open path, the node pair will be added into *R*. Otherwise, the edge of the node pair will be inserted into *graph*. Then the initial network diagraph will be constructed. The realization of $I(clusws_i, clusws_m, Rel_{ws})$ in step 9 is given in Algorithm 2.

Algorithm 2. Mutual information calculation algorithm (*Imutual*)

Input: $clusws_i$, $clusws_m$, Rel_{ws}

Output: val_i

1: val_i, $wsnum_1$, $wsnum_2 \leftarrow 0$, $total_{ws} \leftarrow 2*Rel_{ws}.rnum$, $sumclu[] \leftarrow \varnothing$, $sumws[] \leftarrow \varnothing$

2: **for all** rel_r: $ws_{ij} \rightarrow ws_{mn}$ in Rel_{ws} **do**

3: **if**($rel_r.ws_{ij} \in clusws_i$) **then** $sumclu[i]$++

4: **if**($rel_r.ws_{mn} \in clusws_m$) **then** $sumclu[m]$++

5: $wsnum_1 \leftarrow Num(ws_{ij})$ $sumws[wsnum_1]$++

6: $wsnum_2 \leftarrow Num(ws_{mn})$ $sumws[wsnum_2]$++

7: **end for**

8: $val_i = I(cluster_i, cluster_m)$

9: return val_i

On the basis of the service history invocation record Rel_{ws}, we use Algorithm 2 to calculate the mutual information between nodes. It gets the total number of services ($total_{ws}$), the service number in different service clusters ($sumclu[]$), number of services ($sumws[wsnum]$). The $Num(ws_{ij})$ is used to get the serial number in service lists for service ws_{ij}. $I(clusws_i, clusws_m)$ in step 8 is used to calculate the mutual information between two service cluster nodes, and it can be calculated through Eq.(5).

$$I(\text{cluster}_i, \text{cluster}_m) = \sum_{j=1}^{sumclu[i]} \sum_{n=1}^{sumclu[m]} p(\text{ws}_{ij}, \text{ws}_{mn}) \log \frac{p(\text{ws}_{ij}, \text{ws}_{mn})}{p(\text{ws}_{ij}) p(\text{ws}_{mn})} \quad (5)$$

The priori probability $p(ws_{ij})$ of service ws_{ij} in Eq.(5) can be get using $p(ws_{ij})=sumws[i]/sumclu[i]$. It means the number of ws_{ij} divided by the service total number. Similarly, $p(ws_{mn})$ can be got. $p(ws_{ij}, ws_{mn})=p(ws_{ij})*p(ws_{mn} \mid ws_{ij})$. And $p(ws_{mn} \mid ws_{ij})$ means the occurrence of ws_{mn} in service cluster *m* in the case of ws_{ij} occurring in service cluster *i*.

The second learning stage: *Thickening*

On the basis of constructing the initial network structure through *Drafting*, this stage will improve the graph in further. The conditional cut set *Cutset* which can *D*-separate the node pair in *R* will be calculated using Algorithm 5(*FindCutSet*). Then it will judge whether the two nodes are conditional independent or not. If the condition is not independent, the two nodes will be connected by the edge. Otherwise, it will judge the other two nodes in *R* in turn.

D-separate is an effective way of finding the conditional independence between nodes in Bayesian network, as seen in Definition **6**.

Definition 6. *D*-separate. For the directed acyclic graph $S=\{X, E\}$, *X* expresses the node set in the graph and *E* expresses the edge set between nodes. $A, B \in X$, $A \neq B$, $F \in X-\{A, B\}$. When the path

between any node in A and B is blocked by F, we call F D-separate A and B. It is denoted as $<A \mid F \mid B>_D$.

Whether the path is blocked or not is judged from the following three aspects: non-transfer connection, serial (transfer) connection and convergence connection. As shown in Figure 1, f in F D-separate the nodes in A and B.

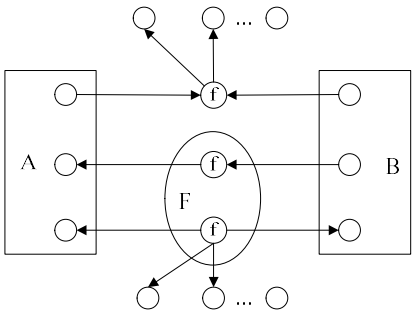

Figure 1. Cases of node blocked

Algorithm 3 gives the process of how to find the D-separate node set between two particular nodes.

Algorithm 3. Finding the D-cut set between two nodes (*FindCutSet*)

Input: *graph,* $node_i$, $node_m$

Output: *Cutset*

1: $Cutset \leftarrow \varnothing$, $InNode_i$, $OutNode_i$, $InNode_m$, $OutNode_m \leftarrow \varnothing$

2: **for all** $Edge(node_a, node_b)$ in *graph* **do**

3: **if**($a == i$) **then**

4: $OutNode_i \leftarrow OutNode_i \cup node_b$

5: **end if**

6: Similar to step 3~5, to get $InNode_i$, $InNode_m$, $OutNode_m$

7: **end for**

8: $Cutset = Cutset \cup (OutNode_i \wedge ... \wedge InNode_m)$ //$Node_i \rightarrow Cutset \rightarrow Node_m$

9: $Cutset = Cutset \cup (OutNode_m \wedge ... \wedge InNode_i)$ // $Node_m \rightarrow Cutset \rightarrow Node_i$

10: $Cutset = Cutset \cup (InNode_i \wedge ... \wedge InNode_m)$ // $Node_i \leftarrow Cutset \rightarrow Node_m$

11: **if**(($OutNode_i \wedge OutNode_m$)!= =*null* && $Sub(OutNode_i \wedge OutNode_m) ! \in F$) **then**

12: $Cutset = Cutset \cup (OutNode_i \wedge OutNode_m) \cup (Sub(OutNode_i \wedge OutNode_m))$

13: **end if**

14: Deal with the situation of existing more than 1 node

15: return *Cutset*

In the step 2-7 of Algorithm 3, the edges which have links with $node_i$ and $node_m$ are found firstly. Then the related nodes are added into the D-separate set(*Cutset*) from the aspects of non-transfer connection, serial connection and convergence connection. Finally, return *Cutset*.

The implementation process of the second stage learning algorithm (*Thickening*) in *TPDA* is given in Algorithm 4.

Algorithm 4. The second stage learning algorithm (*Thickening*)

Input: *graph*, *R*, *Cluster*, Rel_{ws}

Output: *graph*

1: i, m←0, *Cutset*←∅, val_c←0
2: **for all** <$clusws_i$, $clusws_m$> in *R* **do**
3: *Cutset*←*FindCutSet*(*graph*, $node_i$, $node_m$) //find the *D*-separate cut set
4: val_c=*Icondition*($clusws_i$, $clusws_m$ | *Cutset*)
5: **if**(val_c>ε) **then**
6: *graph.insert*(new *Edge*($node_i$, $node_m$))
7: **end if**
8: *Cutset*←∅
9: **end for**
10:return *graph*

The *FindCutSet* in step 3 of Algorithm 3 is used to find the *D*-separate set between two nodes. *Icondition*($cluster_i$, $cluster_m$ | *Cutset*) in step 4 is used to calculate the conditional mutual information between service cluster nodes of $clusws_i$ and $clusws_m$ in the case of *Cutset*, as seen in Eq.(6). If the conditional mutual information is more than the threshold (ε), the corresponding edge will be added into the graph, as seen in step 6-8.

$$Icondition(\mathrm{cluster}_i, \mathrm{cluster}_m \mid \mathrm{Cutset}) = \sum_{j=1}^{sumclu[i]} \sum_{n=1}^{sumclu[m]} \sum_{c=1}^{cnum} p(\mathrm{ws}_{ij}, \mathrm{ws}_{mn}, \mathrm{c}) \log \frac{p(\mathrm{ws}_{ij}, \mathrm{ws}_{mn} \mid \mathrm{c})}{p(\mathrm{ws}_{ij} \mid \mathrm{c}) p(\mathrm{ws}_{mn} \mid \mathrm{c})} \quad (6)$$

The c in Eq.(6) expresses each element in *Cutset*. The $p(ws_{ij}, ws_{mn} \mid c)$ means the probability of ws_{ij} and ws_{mn} in the occurence of c. And $p(ws_{ij}, ws_{mn} \mid c)=p(ws_{ij}, ws_{mn}, c)/p(c)$. The $p(ws_{ij}, ws_{mn}, c)$ denotes the joint probability of ws_{ij}, ws_{mn} and c in the cut set. It will be calculated according to the topology of c, ws_{ij} and ws_{mn}.

The third stage: *Thinning*

Based on the constructed network directed graph, this stage will find the minimum cut set $Cutset_{min}$ which can *D-cut* two service cluster nodes. Then it uses the conditional independence to judge whether the two nodes are conditional independent or not. The concrete process is given in Algorithm 5.

Algorithm 5. The third stage learning algorithm (*Thinning*)

Input: *graph*, *Cluster*, Rel_{ws}

Output: *graph*

1: $Cutset_{min}$←∅, val_c←0
2: **for all** *Edge*($node_i$, $node_m$) in *graph* **do**
3: if(*ExistPath*($node_i$, $node_m$)) **then**
4: delete *Edge*($node_i$, $node_m$)
5: $Cutset_{min}$←*FindMinCutSet*(*graph*, $node_i$, $node_m$)
6: val_c=*Icondition*($cluster_i$, $cluster_m$ | $Cutset_{min}$)

7: **if**(val_c>ε) **then**
8: *graph*.insert(new *Edge*($node_i$, $node_m$))
9: **end if**
10: $Cutset_{min}$←∅
11: **end if**
12:**end for**
13:return *graph*

The *ExistPath* in step 3 is used to judge whether there exists an open path between $node_i$ and $node_m$ or not except for the edge of *Edge*($node_i$, $node_m$). *FindMinCutSet* in step 5 is used to find the minimum cut set between two nodes. The step 6 is used to calculate the conditional mutual information, and *graph* will be updated in further.

3.2 Bayesian network parameter learning

Bayesian network parameter learning refers to learn the conditional probability distribution of each node in the network. Based on organizing service cluster nodes using Bayesian network structure learning method, this section mainly introduces how to learn the conditional probability of each node in the network. And the conditional probability table (*CPT*) can be got. Algorithm 6 gives the process of generating the conditional probability table of different nodes.

Algorithm 6. Bayesian network parameter learning algorithm (*BNPL*)
Input: *graph*, $total_{ws}$, *sumclu*[], *Cluster*, Rel_{ws}
Output: *CPT*
1: *CPT*←∅, *PreSet*←∅, CP_c←∅
2: **for** *c*=1 **to** *Cluster.cnum* **do**
3: **if**(*graph.nodes*[*c*].*indegree*= =0) **then**
4: get priori probability $p(w_{ij})$ of $clusws_i$, and CP_c
5: $CPT \leftarrow CPT \cup CP_c$
6: **end if**
7: **end for**
8: **for** *c*=1 **to** *Cluster.cnum* **do**
9: **for all** *Edge*($node_i$, $node_m$) in *graph* **do**
10: **if**(*m*= =*c*) **then**
11: $PreSet \leftarrow PreSet \cup clusws_i$
12: **end if**
13: **end for**
14: $CP_c=p(ws_{cj} \mid ws_{PreSet})$ //Maximum likelihood estimation method
15: $CPT \leftarrow CPT \cup CP_c$
16: *PreSet*←∅
17:**end for**
18:return *CPT*

In Algorithm 6, *sumclu*[*i*] refers to the service total number in the service cluster of $clusws_i$, $total_{ws}$ refers to the service total number, as seen in Algorithm 2. Algorithm 6 calculates the prior probability for the node whose indegree is equal to 0 using Eq.(12), as seen in step 2-6. Then all the

nodes ($clusws_c$) are judged in turn using the following approach: finding the predecessor node set *PreSet* of $clusws_c$, and calculating the conditional probability of this node, as seen in step 7-16. Finally, return *CPT*.

The step 14 in Algorithm 6 is used to calculate the conditional probability of nodes using the history invocation record Rel_{ws} between services, and thus to realize parameter learning. Parameter learning is also known as parameter estimation. Parameter learning mainly includes the maximum likelihood estimation (*MLE*) and Bayesian estimation (*BE*) in the case of complete data set. Bayesian estimation method needs to use the prior distribution information, but this information is not known when to do parameter learning on the initial network. Therefore, we use the *MLE* method to calculate the conditional probability firstly.

There are n variable nodes $X=\{X_1, X_2,\ldots, X_n\}$ in the Bayesian network N, and node X_i has r_i values: 1, 2, …, r_i. Its parent node $\pi(X_i)$ has q_i combined values, and it is denoted as $\{1, 2, \ldots, q_i\}$. If X_i has no parent node, we can get $q_i=1$. Then we can get the parameter of node i and parent node j, as shown in Eq.(7).

$$\theta_{ijk} = p(\mathrm{X}_i = \mathrm{k} \mid \pi(\mathrm{X}_i) = \mathrm{j}) \qquad (7)$$

In Eq.(7), the range of i is 1~n. For certain i, the range of j is 1~q_i and the range of k is 1~ r_i. Supposing the data sample set is D, the maximum likelihood estimation method is used to calculate the maximum likelihood estimation value θ^*_{ijk}. And θ^*_{ijk} can be got using the sample number of $\pi(X_i)=j$ divided by the sample number of $X_i=k$ and $\pi(X_i)=j$ in D.

After using the maximum likelihood estimation method to get *CPT*, the information of *CPT* needs to be updated as the sample data increases. The Bayesian estimation method could make full use of the priori and posteriori information, avoid the subjective bias of using priori information only, avoid doing the search and calculation blindly due to lacking sample information, and avoid the noise impact of using sample information only. We consider the initial *CPT* as priori information, and use the *BE* method to calculate the nodes' conditional probability, and thus to update *CPT*.

Supposing the Bayesian network structure is N, the sample data set is $D=(D_1, D_2,\ldots, D_m)$, the priori probability distribution of θ is $p(\theta)$. We can get $p(\theta \mid D)$ using Eq.(8).

$$p(\theta \mid D) = p(\theta)\prod_{i=1}^{n}\prod_{j=1}^{q_i}\prod_{k=1}^{r_i}\theta_{ijk}^{m_{ijk}} \qquad (8)$$

Supposing $p(\theta_{ij.})$ is *Dirichlet* distribution of $D[a_{ij1}, a_{ij2},\ldots, a_{ijri},]$, we can get $p(\theta)$ using Eq.(9) in the case of local and global independence.

$$p(\theta) = \prod_{i=1}^{n} p(\theta_{i..}) = \prod_{i=1}^{n}\prod_{j=1}^{q_i} p(\theta_{ij.}) \approx \prod_{i=1}^{n}\prod_{j=1}^{q_i}\prod_{k=1}^{r_i}\theta_{ijk}^{a_{ijk}-1} \qquad (9)$$

Using Eq.(8) and Eq.(9), we can get posterior distribution $p(\theta \mid D)$ using Eq.(10). $p(\theta \mid D)$ is also subjected to *Dirichlet* distribution, and it is local and global independent. $p(\theta_{ij.} \mid D)$ is subjected to *Dirichlet* distribution of $D[m_{ij1}+a_{ij1}, m_{ij2}+a_{ij2},\ldots, m_{ijri}+a_{ijri}]$.

$$p(\theta \mid D) \approx \prod_{i=1}^{n}\prod_{j=1}^{q_i}\prod_{k=1}^{r_i}\theta_{ijk}^{m_{ijk}+a_{ijk}-1} \qquad (10)$$

Using above Bayesian estimation method, we can calculate the conditional probability of different nodes, and update *CPT*. It can lay the foundation of Web service recommendation using Bayesian network reasoning method.

4 Case Study

Example 1. *Cluster*={$clusws_c$, $1 \le c \le 7$}. We use *A*~*G* to express the service clusters, and they are denoted as $clusws_A$~$clusws_G$. The number of services in $clusws_A$~$clusws_G$ is {5,3,6,7,7,3,5} respectively. We can see $clusws_A$ contains 5 services, and it denoted as $clusws_A$={ws_{Aw}, $1 \le w \le 5$}. Rel_{ws}={rel_r: $ws_{ij} \rightarrow ws_{mn}$, $1 \le r \le 51$, $0 \le i,m \le 7$, $0 \le j \le c_i$, $0 \le n \le c_m$}. The relationship between services in Rel_{ws} is shown in Table 1.

Table 1. The relationship between services in Rel_{ws}

Service cluster	Rel_{ws}
$clusws_A$	$ws_{Aj} \rightarrow ws_{Bn}(clusws_B)$: <$A_0$,$B_0$> <$A_0$,$B_1$> <$A_0$,$B_2$> <$A_1$,$B_0$> <$A_1$,$B_1$> <$A_1$,$B_2$>
	$ws_{Aj} \rightarrow ws_{Cn}(clusws_C)$: <$A_0$,$C_3$> <$A_1$,$C_4$> <$A_1$,$C_5$> <$A_2$,$C_4$> <$A_2$,$C_5$>
	$ws_{Aj} \rightarrow ws_{En}(clusws_E)$: <$A_0$,$E_0$> <$A_1$,$E_1$> <$A_2$,$E_2$> <$A_3$,$E_3$> <$A_4$,$E_1$>
$clusws_B$	$ws_{Bj} \rightarrow ws_{Cn}(clusws_C)$: <$B_0$,$C_0$> <$B_1$,$C_1$> <$B_2$,$C_2$> <$B_1$,$C_3$> <$B_0$,$C_4$>
$clusws_C$	$ws_{Cj} \rightarrow ws_{Dn}(clusws_D)$: <$C_1$,$D_4$> <$C_3$,$D_6$>
	$ws_{Cj} \rightarrow ws_{En}(clusws_E)$: <$C_1$,$E_1$> <$C_5$,$E_5$>
	$ws_{Cj} \rightarrow ws_{Fn}(clusws_F)$: <$C_0$,$F_0$> <$C_1$,$F_1$> <$C_2$,$F_2$> <$C_3$,$F_2$> <$C_4$,$F_1$> <$C_5$,$F_0$> <$C_4$,$F_2$> <$C_1$,$F_0$>
$clusws_D$	$ws_{Dj} \rightarrow ws_{En}(clusws_E)$: <$D_0$,$E_0$> <$D_1$,$E_1$> <$D_2$,$E_2$> <$D_3$,$E_3$> <$D_4$,$E_4$> <$D_5$,$E_5$> <$D_6$,$E_6$> <$D_4$,$E_4$> <$D_2$,$E_1$>
$clusws_E$	$ws_{Ej} \rightarrow ws_{Gn}(clusws_G)$: <$E_0$,$G_0$> <$E_1$,$G_1$> <$E_2$,$G_2$> <$E_3$,$G_3$> <$E_2$,$G_4$> <$E_1$,$G_3$> <$E_0$,$G_2$>
$clusws_F$	-
$clusws_G$	$ws_{Gj} \rightarrow ws_{Fn}(clusws_F)$: <$G_4$,$F_1$> <$G_4$,$F_2$>

Note: <A_0,B_0> in Table 1 expresses $ws_{A0} \rightarrow ws_{B0}$, it means there exists invocation record between service ws_{A0} in $clusws_A$ and ws_{B0} in $clusws_B$.

(1) Bayesian network structure learning

The first stage: *Drafting*

a) The mutual information of each two services in $clusws_A$~$clusws_G$ is calculated firstly. For example, when to calculate *Imutual*($clusws_A$, $clusws_B$, Rel_{ws}) using Algorithm 2, we can get the service total number $total_{ws}$($total_{ws}$=36) and the service number(*sumclu*[]) in different service clusters. For example, *sumclu*[1]=16 of $clusws_A$ and *sumclu*[2]=11 of $clusws_B$. The total number of different services is denoted as *sumws*[*wsnum*], such as *sumws*[1]=5 of ws_{A0}, *sumws*[6]=4 of ws_{B0}. We can calculate $I(clusws_A, clusws_B)$ using Eq.(11).

$$I(\text{cluster}_A, \text{cluster}_B) = \sum_{j=0}^{16} \sum_{n=0}^{11} p(\text{ws}_{Aj}, \text{ws}_{Bn}) \log \frac{p(\text{ws}_{Aj}, \text{ws}_{Bn})}{p(\text{ws}_{Aj}) p(\text{ws}_{Bn})} \qquad (11)$$

b) The mutual information between nodes can be calculated using step 2-8 of Algorithm 2. The result is shown in Table 2, and we use A~G to express $clusws_A$~$clusws_G$.

Table 2. Mutual information between nodes

$node_i$	$node_j$	v_I
A	B	0.0026521534
A	C	0.45123205
A	E	0.6611474
B	C	0.62302357
C	D	0.2628549
C	E	0.31027
C	F	0.33777514
D	E	0.89103544
E	G	0.37155902
G	F	0.05312646

From Table 1, we can get $I(clusws_A, clusws_B)$=0.0026521534, $I(clusws_G, clusws_F)$=0.05312646, and so on. Supposing ε=0.15, we can get $I(clusws_A, clusws_B)$<ε and $I(clusws_G, clusws_F)$<ε. The rest edges will be inserted into S using step 11 of Algorithm 1, such as $S \leftarrow S \cup <clusws_A, clusws_C, 0.45123205>$. The initial graph can be got, and it is shown in Figure 2(1).

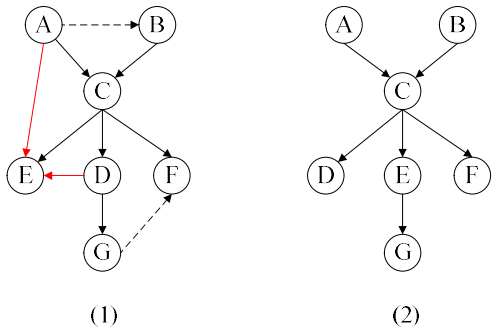

Figure 2. The structure graph of service nodes

c) The *Sort*(S) is used to sort the service cluster pair according to the mutual information between them. Through step 15-19 in Algorithm 1, we can see there exists open path between the node pair of <A, E> and <D, E> in Figure 2(1). Then we can get R={<A, E>, <D, E>}.

The second stage: *Thickening*

a) The *FindCutSet* in step 3 of Algorithm 4 is used to find the D-cut set *Cutset* of each node pair in R={<A, E>, <D, E>}. In Figure 2(2), the D-cut set between <A, E> is denoted as $Cutset_{AE}$={C}, and the D-cut set between <D, E> is denoted as $Cutset_{DE}$={C}.

b) Then we use *Icondition* in step 4 of Algorithm 4 to calculate the conditional mutual information between node pair in *R*. It mainly uses Eq.(12) to calculate *Icondition*($clusws_A$, $clusws_E$ | *Cutset*). We can get *sumclu*[1]=16 of $clusws_A$ and *sumclu*[4]=23 of $clusws_E$.

$$Icondition(\text{cluster}_A, \text{cluster}_E \mid \text{Cutset}) = \sum_{j=1}^{16}\sum_{n=1}^{23}\sum_{c=1}^{1} p(\text{ws}_{Aj}, \text{ws}_{En}, \text{c}) \log \frac{p(\text{ws}_{Aj}, \text{ws}_{En} \mid \text{c})}{p(\text{ws}_{Aj} \mid \text{c}) p(\text{ws}_{En} \mid \text{c})} \quad (12)$$

c) We can get *Icondition*($clusws_A$, $clusws_E$ | $clusws_C$)=0.0 and *Icondition*($clusws_D$, $clusws_E$ | $clusws_C$)=0.0. They are all less than the threshold, and the edges in *graph* are not changed, as shown in Figure 2(2).

The third stage: *Thinning*

a) Each edge in *graph* will be judged whether there exists an open path between the corresponding nodes except for the direct path using Algorithm 5. If there exists an open path, the edge will be removed temporarily. The *FindMinCutSet* will be used to get the minimum cut set $Cutset_{min}$ between the corresponding nodes. Then the *Icondition* is used to calculate whether it is conditional independent or not, and thus to decide whether to insert the edge into *graph* again or not.

b) All the edges in *graph* will be operated using the above method, we can get the edges are not changed. The final network structure is shown in Figure 2(2).

(2) Bayesian network parameter learning

On the basis of network structure, we use the maximum likelihood estimation parameter learning method to calculate the conditional probability of all the nodes(*A*~*G*). And thus the *CPT* can be got.

a) The *indegree* of node *A* and *B* is equal to 0. Using step 2-7 in Algorithm 6, we can get $p(A_0)$=0.3125, $p(A_1)$=0.375, $p(B_0)$=0.3636367, $p(B_1)$=0. 3636367, as shown in Table 3 and Table 4.

Table 3. The *CPT* information of node *A*

A_i	A_0	A_1	A_2	A_3	A_4
$p(A_i)$	0.3125	0.375	0.1875	0.0625	0.0625

Table 4. The *CPT* information of node *B*

B_i	B_0	B_1	B_2
$p(B_i)$	0.3636367	0.363637	0.272728

Table 5. The *CPT* information of node *F*

F / *C*	F_0	F_1	F_2
C_0	0.333	-	-
C_1	0.333	0.5	-
C_2	-	-	0.333
C_3	0.0	-	0.333
C_4	0.0	0.5	0.333
C_5	0.333	-	-

Note: '-' expresses the conditional probability is 0.0. The probability is accurate to 3 decimal places.

b) The conditional probability of the nodes whose *indegree* are not equal to 0 will be calculated using step 7-17 in Algorithm 7. When the indegree is 1, we can get $\pi(F)=\{C\}$ for node F. The $p(F=F_i \mid C=C_j)$ can be calculated through Eq.(8), and the result is shown in Table 5.

When the indegree is larger than 1, we can get $\pi(C)=\{A, B\}$ for node C. And $p(C=C_j \mid A=A_j, B=B_k)$ can be calculated, the result is shown in Table 6.

Table 6. The *CPT* information of node *C*

A_iB_j / C_x	A_0B_0	A_0B_1	A_0B_2	A_1B_0	A_1B_1	A_1B_2	A_2B_0	A_2B_1	A_2B_2	A_3B_0	A_3B_1	A_3B_2	A_4B_0	A_4B_1	A_4B_2
C_0	0.333	-	-	0.25	-	-	0.25	-	-	0.5	-	-	0.5	-	-
C_1	-	0.333	-	-	0.25	-	-	0.25	-	-	0.5	-	-	0.5	-
C_2	-	-	0.5	-	-	0.333	-	-	0.333	-	-	1.0	-	-	1.0
C_3	0.333	0.667	0.5	-	0.25	-	-	0.25	-	-	0.5	-	-	0.5	-
C_4	0.333	-	-	0.5	0.25	0.333	0.5	0.25	0.333	0.5	-	-	0.5	-	-
C_5	-	-	-	0.25	0.25	0.333	0.25	0.25	0.333	-	-	-	-	-	-

Note: '-' expresses the conditional probability is 0.0. The probability is accurate to 3 decimal places.

5 Related work

There exist two kinds of research work about service organization. The first one refers to organize services in real-time according to users' individual requirements in the service registry. The second one refers to use the relationship between services and organize them based on users' common requests. Services are organized in the view of specific topic using this method. Then services can be found and selected directly based on the relationship between them according to users' individual requirements. In this paper, we mainly consider the second research method.

There exists the following research work about the first aspect. Liu et al. have proposed a kind of service aggregation approach using consumer-driven technology [13, 14]. This method aggregates and organizes services mainly according to users' personal requirements and only the atomic services can be discovered. A user-centric service composition method synthesizing multiple views is proposed in [15]. The requirements are transferred into operations on multiple views. This approach starts from users' needs and realizes service organization in the exploratory manner. Han et al. have used the business user programming method to organize the virtual service resources according to users' requests [7]. In [16], a personalized requirement oriented virtual service resource aggregation method has been proposed. This method takes into account the characteristics of users' requests and resources. The virtual resource with large-granularity is dynamically integrated to satisfy the personal requests. Ye et al. have proposed a new concept of Autonomous Web Service (AWS) to search users' requirements autonomously [17]. An intention-behaviour-achievement mechanism based on environment ontology is proposed to specify the service request and the capability of AWS. This method models services to be autonomic entities which have certain intention and behaviours. Then services can aggregate autonomously to meet users' requests. Wen et al have proposed an on-demand service aggregation method in the orientation of requirements semantics [18]. This method concentrates on service organization from the aspect of interoperability.

There exists the following research work about the second aspect. Hu et al. have proposed a user-oriented service workflow construction method [5]. Services are clustered and the spanning tree approach is used to represent the services in the same cluster firstly. Then service clusters are organized through the workflow business logic method. This method organizes services in term of service execution process, and it mainly uses the hybrid particle swarm optimization algorithm to select services with the best QoS. Liu et al. have organized Web services using service group and service node [3]. Service group is similar to the service clusters that are formed through clustering, and service node is similar to the abstract service of specific service cluster. The services are organized through building service organization model, which uses business logic integration of service nodes. This method focuses on service aggregation process of multi-objective optimization of dynamic selection of services. The above two approaches cluster services with similar function firstly, and then organize service clusters in further. This is same to our proposed service organization method. But they mainly use the modelling method to get the service execution relationship in the view of business logic and workflow. It often needs the experts to participate in the process. Zhou et al. have concentrated on the research of data providing services discovery [19]. On the basis of clustering data providing services, they have mentioned organizing different service clusters into cluster network. But they have not elaborated the detail process of how to organize service clusters. Sellami et al. have used community to organize and manage Web services [6, 20]. The fuzzy clustering algorithm is used to cluster services to form service community [21], and the service communities are organized from the point of functionality. In addition, the related management operations (create, delete, merge, etc.) are used for the evolution of the organized services. And it can help to adapt to the dynamic changing environment. This method concentrates on the organization of service register, while our method mainly concentrates on the organization of different service clusters. Wu et al. have used a logical petri net-based approach to compose service clusters in a virtual layer [22]. The basic composition models of service cluster nets (SCNs) are presented. Aznag et al. in [23] have used the Formal Concept Analysis (FCA) formalism to organize the constructed hierarchical clusters into concept lattices according to their topics.

On the basis of Web service clustering, we have organized the service clusters from aspects of semantic interoperability and users' requirement features (role, goal, process) [24-26]. Based on clustering services with same or similar functions, the approach in this paper makes full use of the Web service history invocation records, uses Bayesian network structure and parameter learning method to organize different service clusters. Wu et al. in [27, 28] have proposed a composite service recommendation method using Bayesian theory. They mainly analyse the service execution log, including service function, QoS record, etc. Based on the used service execution process that is generated manually or automatically, this approach calculates the service correlation probability using Bayesian theory, and recommends the optimal service sequence for users. The Bayesian theory is also used in our method. The difference is our method mainly uses the Bayesian structure learning theory to organize service clusters. Then users can firstly select the services which they are interested in, and then they can select the services with proper QoS values in different service clusters in further.

6 Experiment

The experiment is carried out on the computer with the configuration of dual Intel (R) Core (TM)2 i5 CPU 760@ 2.80GHz, and 4G memory. BN Toolkit(BNT) [29] is a software development kit about

Bayesian network learning using Matlab by Kevin P.Murphy. This package provides a number of underlying libraries about Bayesian network learning, and supports structure learning, parameter learning, reasoning and some other functions. In addition, this package is completely free, and its code is entirely open and with good scalable. This package does not support the three-stage dependency analysis algorithm, we realize this algorithm and thus to realize Web service organization in further.

There are two kinds of Bayesian network structure learning method: search score method and dependency analysis method. The K2 algorithm, climbing algorithm(HC), greedy search algorithm(GS) and Markov chain Monte Carlo algorithm(MCMC) are four classical search score Bayesian network structure learning methods. We use the above four methods to compare with our TPDA method.

K2WS: using K2 algorithm to realize Web service organization;

HCWS: using HC algorithm to realize Web service organization;

GSWS: using GS algorithm to realize Web service organization;

MCMC: using MCMC algorithm to realize Web service organization;

TPDA: using the proposed three-stage dependency analysis learning method to realize Web service organization;

6.1 Experiment of Bayesian network structure learning

The experiment data is generated randomly, *cnum* refers to the number of different service clusters, *snum* refers to the service number in different service clusters, *rnum* refers to the number of service execution history records, as shown in Table 7.

Table 7. Experiment data

Type	Data									
cnum	5	10	15	20	25	30	35	40	45	50
snum	23	40	71	117	124	145	198	239	244	263
rnum	66	104	111	172	251	258	329	340	419	484

Experiment 1. Comparison of service organization efficiency of different methods

In the case of generating different numbers of service clusters (*cnum*), we use *K2WS*, *HCWS*, *GSWS*, *MCMC* and *TPDA* to realize service organization. We compare the service organization efficiency of the above five methods, the result is shown in Table 8.

Table 8. Comparison of service organization efficiency of different methods

cnum	5	10	15	20	25	30	35	40	45	50
K2WS	0.326	0.746	1.745	2.18	4.581	6.497	9.041	16.449	21.172	28.433
HCWS	1.257	18.862	62.432	91.728	1348.46	1980.991	7019.808	25571.2	41974.1	74409.02
GSWS	1.558	73.877	84.425	154.471	1590.54	2390.133	12984.34	24898.39	28057.04	42382.4
MCMC	3.217	6.442	10.985	24.922	76.998	94.373	157.585	247.224	485.322	743.677
TPDA	0.009	0.013	0.019	0.037	0.066	0.075	0.156	0.214	0.286	0.367

From Table 8 we can see the service organization efficiency is different for all the methods in the case of specific number of service clusters. For the specific number of service clusters, the service

organization time of TPDA method is much less than other four search score methods. Its efficiency is the highest, and this method can realize service organization quickly.

Experiment 2. Comparison of service organization accuracy of different methods

In the case of generating different numbers of service clusters, we use *K2WS*, *HCWS*, *GSWS*, *MCMC* and *TPDA* to realize service organization. We compare the service organization accuracy of the above five approaches. Table 9 elaborates the total edge number of the network which is generated by different service organization methods.

Table 9. Comparison of total edge number of different service organization methods

cnum	5	10	15	20	25	30	35	40	45	50
K2WS	4	5	4	6	19	18	19	42	45	52
HCWS	4	5	6	6	21	15	21	54	53	63
GSWS	4	5	5	6	21	15	21	51	57	64
MCMC	7	7	5	7	20	17	19	35	37	50
TPDA	3	8	9	8	17	14	23	34	36	40

In addition, we compare the common edge number, extra edge rate and loss edge rate of the standard network and the network using different methods, as shown in Figure 3-5.

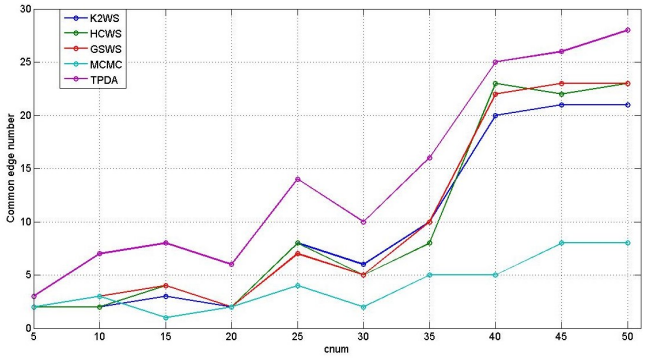

Figure 3. Comparison of common edge number of different methods

The threshold in *TPDA* is set to 0.15. We do not consider the direction of the edges in the service organization network graph when to statistic the edge numbers. From Figure 3~5, we can see the common edge number of *MCMC* method is the least of all, and its extra edge rate and loss edge rate is the largest. The learning effect of this approach is the worst. The extra edge rate and loss edge rate of our *TPDA* method is the least, this method can learn the network with the better structure. The learning effect of *K2WS*, *HCWS* and *GSWS* is about same. Their corresponding learning effect is better than *MCMC*, but it is less than *TPDA* method.

For example, we use *K2WS*, *HCWS*, *GSWS*, *MCMC* and *TPDA* to organize services in the case of *cnum*=30. The result of the total edge number using different service organization methods is shown in Table 10.

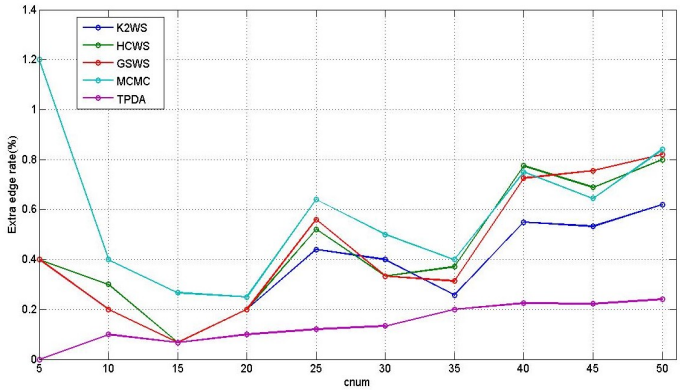

Figure 4. Comparison of extra edge rate of different methods

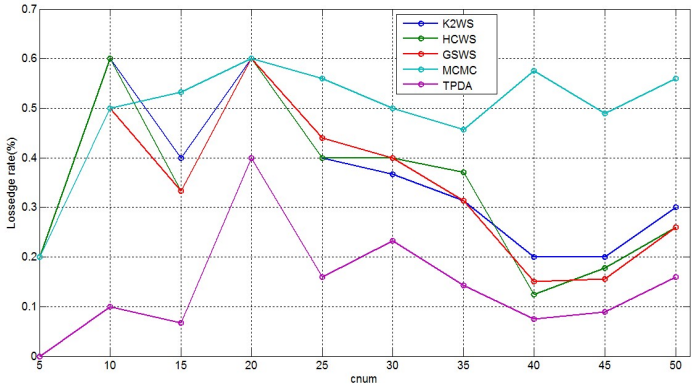

Figure 5. Comparison of loss edge rate of different methods

Table 10. The result of total edge number using different service organization methods (*cnum*=30)

Different methods	Result of total edge
K2WS	<8,19>, <9,7>, <10,26>, <12,3>, <12,14>, <16,25>, <28,27>, <29,1>, <29,22>, <1,22>, <1,23>, <25,20>, <25,4>, <26,30>, <20,4>, <20,30>, <23,3>, <23,17>
HCWS	<3,12>, <8,19>, <10,26>, <27,28>, <29,22>, <12,14>, <22,1>, <26,30>, <1,23>, <30,9>, <30,20>, <9,7>, <20,25>, <25,4>, <25,16>
GSWS	<8,19>, <10,26>, <12,3>, <12,14>, <27,28>, <29,22>, <22,1>, <26,30>, <1,23>, <30,9>, <30,20>, <9,7>, <20,25>, <25,4>, <25,16>
MCMC	<8,19>, <10,26>, <13,24>, <14,12>, <18,1>, <27,9>, <1,22>, <1,29>, <9,4>, <26,19>, <29,22>, <22,15>, <15,7>, <7,6>, <6,3>, <6,30>, <30,20>
TPDA	<1,3>, <1,23>, <4,21>, <9,7>, <12,3>, <14,6>, <14,18>, <17,11>, <18,7>, <21,17>, <25,20>, <29,1>, <29,22>, <30,3>

The result of the common edge number of standard network and the network using different methods is shown in Table 11.

Table 11. The result of common edge (*cnum*=30)

Different methods	Result of common edge
K2WS	<1,23>, <9,7>, <12,3>, <25,20>, <29,1>, <29,22>
HCWS	<1,23>, <9,7>, <12,3>, <25,20>, <29,22>
GSWS	<1,23>, <9,7>, <12,3>, <25,20>, <29,22>
MCMC	<29,1>, <29,22>
TPDA	<1,23>, <4,21>, <9,7>, <12,3>, <14,18>, <17,11>, <25,20>, <29,1>, <29,22>, <30,3>

For the specific service clusters to be organized, we can see the edge number in service organization network graph is about same for all the different methods. But the concrete edges of all the methods are largely different. The learning effect of *MCMC* is the worst and the *TPDA* method is the best of all. The learning effect of *K2WS*, *HCWS* and *GSWS* is in the middle.

Experiment 3. Comparison of learning time of three stages in TPDA method

In the case of different number of service clusters, we set the threshold to 0.15, and compare the learning time of the three stages in *TPDA*. The experiment result is shown in Table 12.

Table 12. Comparison of learning time of three stages in *TPDA*

cnum	5	10	15	20	25	30	35	40	45	50
Drafting	0.006	0.01	0.015	0.033	0.061	0.069	0.147	0.208	0.277	0.358
Thickening	0.002	0.002	0.002	0.001	0.002	0.002	0.004	0.002	0.003	0.004
Thinning	0.001	0.001	0.002	0.003	0.003	0.004	0.005	0.004	0.006	0.005

From Table 12, we can see the time is becoming more as the number of service clusters increases in the TPDA learning process. For the specific number of service cluster, the learning time of the three-stage is largely different. But the time used is relatively small, it is within 0.1*s*. The time of *Drafting* is the most of all. The time of *Thickening* and *Thinning* are more or less, and it is in the scope of 0.001~0.005*s*. Therefore, the total time of TPDA method is relatively small, and this method can organize services efficiently.

Experiment 4. Comparison of edge number of different thresholds

The different value of threshold in the three-stages of *TPDA* will have a greater impact on the structure of service organization network. In the case of different number of service clusters (*cnum*) and different threshold values, we compare the edge number of service organization graph. The result is shown in Table 13. From Table 13, we can see the number of edges in service organization network graph is becoming less as the threshold increases in the stage of *Drafting*. When the threshold is set to 0.5, the number of services that can be organized are becoming less. In addition, when the threshold is set to a specific value, the number of services that can be organized is becoming more as *cnum* increases.

For example, when set *cnum*=30, we compare the edge number in service organization graph when to set different thresholds in *Drafting* stage. The result is shown in Table 14.

Table 13. Comparison of service organization edge number of different thresholds

cnum thresholds	5	10	15	20	25	30	35	40	45	50
0.05	4	7	11	16	17	19	26	34	34	37
0.10	3	7	11	16	17	19	25	32	34	34
0.15	3	7	10	16	16	17	23	31	33	31
0.20	2	6	10	13	15	15	21	31	29	27
0.25	2	4	10	13	15	9	20	29	24	24
0.30	2	3	8	11	12	8	20	25	20	21
0.35	2	2	7	9	10	6	20	24	19	18
0.40	2	1	7	6	8	6	16	21	13	15
0.45	1	0	5	3	6	3	13	16	10	12
0.50	1	0	2	1	4	2	11	13	8	9

Table 14. Comparison of edge number using different thresholds in *Drafting*(*cnum*=30)

Threshold	Edges
0.05	<1,3>, <1,23>, <2,18>, <6,2>, <9,7>, <11,14>, <12,3>, <14,18>, <16,4>, <16,25>, <17,4>, <17,11>, <18,7>, <21,17>, <25,20>, <28,27>, <29,1>, <29,22>, <30,3>
0.10	<1,3>, <1,23>, <2,18>, <6,2>, <9,7>, <11,14>, <12,3>, <14,18>, <16,4>, <16,25>, <17,4>, <17,11>, <18,7>, <21,17>, <25,20>, <28,27>, <29,1>, <29,22>, <30,3>
0.15	<1,3>, <1,23>, <4,21>,<6,2>, <9,7>, <11,14>, <12,3>, <14,6>, <14,18>, <16,4>, <17,11>, <18,7>, <21,17>, <25,20>, <29,1>, <29,22>, <30,3>
0.20	<1,3>, <1,23>, <4,21>, <6,2>, <11,14>, <12,3>, <14,6>, <14,18>, <16,4>, <17,11>, <18,7>, <25,20>, <29,1>, <29,22>, <30,3>
0.25	<1,23>, <2,17>, <11,14>, <12,3>, <14,6>, <14,18>, <17,11>, <25,20>, <29,22>
0.30	<1,23>, <2,17>, <11,14>, <12,3>, <14,6>, <17,11>, <25,20>, <29,22>
0.35	<2,17>, <11,14>, <12,3>, <14,6>, <17,11>, <25,20>
0.40	<2,17>, <11,14>, <12,3>, <14,6>, <17,11>, <25,20>
0.45	<2,17>, <12,3>, <25,20>
0.50	<12,3>, <25,20>

We can see the number of edges in the service organization network graph is becoming less as the threshold increases. When the threshold is set to 0.25, the number of edges is reduced largely. Few edges that can be learned when the threshold is set to 0.50. This is because the mutual information that is greater than the threshold becomes less and less as the threshold increases.

6.2 Experiment of Bayesian network parameter learning

Experiment 5. Comparison of parameter learning time of different methods

On the basis of organizing services through *K2WS*, *HCWS*, *GSWS*, *MCMC* and *TPDA*, this experiment compares the parameter learning time of maximum likelihood estimation (*MLE*) and Bayesian estimation (*BE*) methods. The result is shown in Figure 6 and Figure 7.

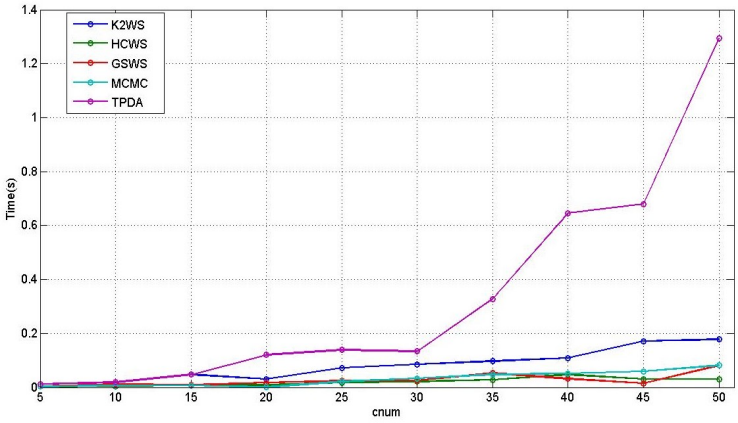

Figure 6. Comparison of parameter learning time (MLE)

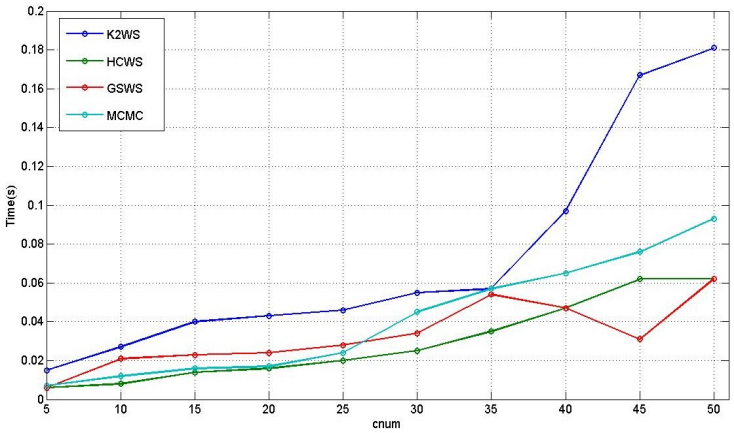

Figure 7. Comparison of parameter learning time (BE)

The Bayesian estimation method needs to use the priori distribution information in *TPDA* method. But this data is unknown when to realize parameter learning on the initial network. So this experiment only uses the maximum likelihood estimation (*MLE*) method to do parameter learning, as shown in Figure 6. And we have not compared the *TPDA* method in the parameter learning of Bayesian estimation method, as shown in Figure 7. From Figure 6 and Figure 7, we can see the parameter learning time of *MLE* and *BE* method of *K2WS*, *HCWS*, *GSWS* and *MCMC* are almost same. The time of *K2WS* method is the most of all, and the time of *HCWS* method is the least of all. At the same time, we can see the time of *MLE* in *TPDA* method is the most in Figure 6. But it does not use much time overall. When *cnum* is less than 50, the using time is less than 1*s*. In addition, the learning time of *MLE* method is slightly more than *BE* method when to use the approaches of *K2WS*, *HCWS*, *GSWS* and *MCMC*.

When *cnum*=30, the network structure that is learned through *TPDA* method is shown in Figure 8.

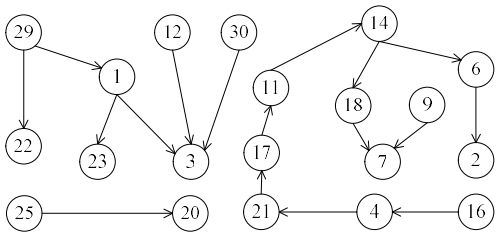

Figure 8. The network structure graph of TPDA method(cnum=30)

From Figure 8, we can see the indegree of node 7 is 2, its parent node is 9 and 18. Node 7 has 5 kinds of values, Node 9 has 4 kinds of values, Node 18 has 2 kinds of values. Through Bayesian estimation (*BE*) parameter learning method, we can get the conditional probability table of node 7, as shown in Table 15.

Table 15. The CPT information of node 7

9_i18_j / 7_x	9_018_0	9_018_1	9_118_0	9_118_1	9_218_0	9_218_1	9_318_0	9_318_1
7_0	0.2	0.0	0.2	0.0	0.333	0.111	0.2	0.077
7_1	0.0	0.25	0.2	0.375	0.167	0.333	0.1	0.231
7_2	0.4	0.25	0.2	0.125	0.167	0.111	0.3	0.231
7_3	0.2	0.25	0.2	0.25	0.167	0.222	0.2	0.231
7_4	0.2	0.25	0.2	0.25	0.167	0.222	0.2	0.231

When *cnum*=30, the network structure that is learned through *K2WS* method is shown in Figure 9.

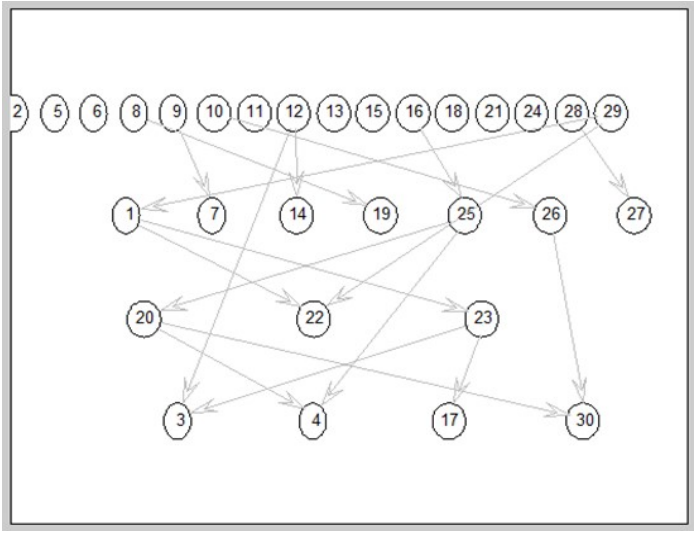

Figure 9. The network structure graph of K2WS method (cnum=30)

From Figure 9, we can see the indegree of node 3 is 2, its parent node is 12 and 23. The conditional probability table of node 3 using maximum likelihood estimation (*MLE*) and Bayesian estimation (*BE*) methods is shown in Table 16 and Table 17 respectively.

Table 16. The CPT information of node 3(*MLE*)

Cases	4=yes	4=no
12=yes,23=yes	0.2273	0.7727
12=yes, 23=no	0.1	0.9
12=no, 23=yes	0.1361	0.8639
12=no, 23=no	0	1.0

Table 17. The CPT information of node 3(*BE*)

Cases	4=yes	4=no
12=yes,23=yes	0.2276	0.7724
12=yes, 23=no	0.1098	0.8902
12=no, 23=yes	0.1362	0.8638
12=no, 23=no	0.0044	0.9956

We can see the conditional probability of *MLE* and *BE* method is slightly different, but the result is about same. The learning efficiency of *BE* method is higher than *MLE* method.

7 Conclusions and Future Work

In the era of service computing, how to realize service organization and management, and help user to find the atomic and a set of services with correlations to meet their functional requirements quickly is a key problem to be solved. On the basis of users' history service invocation record and Web service clustering, this paper uses the Bayesian network structure and parameter learning method to organize service clusters. It can lay the foundation of personal service selection. Finally, the experiments and case study are used to do the validation and explanation.

The next step research work mainly includes the following aspects: use Bayesian network reasoning method to realize Web service recommendation and selection; organize Web services from the semantic level to improve the accuracy; optimize the Bayesian network structure learning algorithm and improve the efficiency of service organization.

Acknowledgements

The authors would like to thank anonymous reviewers for their valuable comments. This research is supported by the National Natural Science Foundation of China under Grant No.31601078, the Natural Science Foundation of Hubei Province under Grant No.2016CFB231, the PAPD fund and Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology(CICAEET) fund.

References

1. Papazoglou, M. P., Traverso, P., Dustdar, S. and Leymann, F., Service-Oriented Computing: A Research Roadmap. International Journal of Cooperative Information Systems, 17(2), 223-255, 2008.
2. Zheng, Y. H., Jeon, B., Xu, D. H., Wu, Q. M. J. and Zhang, H., Image segmentation by generalized hierarchical fuzzy C-means algorithm, Journal of Intelligent and Fuzzy Systems, 28(2), 961-973, 2015.
3. Liu, S. L., Liu, Y. X., Zhang, F., Tang, G. F. and Jing, N., A Dynamic Web Services Selection Algorithm with QoS Global Optimal in Web Services Composition. Chinese Journal of Software, 18(3), 646-656, 2007.
4. Xia, Z. H., Wang, X. H., Song, X. M. and Wang, Q., A Secure and Dynamic Multi-keyword Ranked Search Scheme over Encrypted Cloud Data. IEEE Transactions on Parallel and Distributed Systems, 27(2), 340-352, 2015.
5. Hu, C. H., Wu, M. and Liu, G. P., An Approach to Constructing Web Service Workflow Based on Business Spanning Graph. Chinese Journal of Software, 18(8), 1870-1882, 2007.
6. Sellami, M., Bouchaala, O., Gaaloul, W. and Tata, S., Communities of Web Service Registries: Construction and Management. Journal of Systems and Software, 86(3), 835-853, 2013.
7. Zhao, Z. F., Han, Y. B., Yu, J. and Wang, J. W., A Service Virtualization Mechanism for Business User Programming. Chinese Journal of Computer Research and Development, 41(12), 2224-2230, 2004.
8. Wu, H. Y. and Du, Y. Y., A Logical Petri Net-Based Approach for Web Service Cluster Composition. Chinese Journal of Computer, 38(1), 204-218, 2015.
9. Gu, B., Sun X. M. and Sheng, V. S., Structural Minimax Probability Machine. IEEE Transactions on Neural Networks and Learning Systems, 1-11, 2016.
10. Liu, J. X., Liu, F., Li, X. X., He, K. Q., Ma Y. T. and Wang J., Web Service Clustering Using Relational Database Approach. International Journal of Software Engineering and Knowledge Engineering, 25(8), 1365-1393, 2015.
11. Zhang, Y. P. and Zhang, L., Machine Learning Theory and Algorithm. Science Press, 246-269, 2012.
12. Cheng, J., Grainer, G., Kelly, J., Bell, D. and Liu, W. R., Learning Bayesian Networks From Data: An Information-Theory Based Approach, Artificial Intelligence, 137, 43-90, 2002.
13. Liu, X. Z., Huang, G. and Mei, H., Consumer-Centric Service Aggregation: Method and Its Supporting Framework. Chinese Journal of Software, 18(8), 1883-1895, 2007.
14. Fu, Z. J., Sun, X. M., Liu, Q., Zhou, L. and Shu, J. G., Achieving Efficient Cloud Search Services: Multi-keyword Ranked Search over Encrypted Cloud Data Supporting Parallel Computing. IEICE Transactions on Communications, E98-B(1), 190-200, 2015.
15. Ding, W. L., Wang, J. and Zhao, S., A User-Centric Service Composition Method Synthesizing Multiple Views. Chinese Journal of Computers, 34(1), 131-142, 2011.
16. Chu, D. H., Wang, X. Z., Wang, Z. J. and Xu, X. F., Personalized requirement oriented virtual service resource aggregation method. Chinese Journal of Computers, 34(12), 2370-2380, 2011.
17. Ye, R. H., Jin, Z., Wang, P. W., Zhen, L. W. and Yang, X. F., Approach for Autonomous Web Service Aggregation Driven by Requirement. Chinese Journal of Software, 21(6), 1181-1195, 2010.
18. Wen, B., He, K. Q. and Wang, J., Requirements Semantics-Driven Aggregated Production for On-demand Service. Chinese Journal of Computers, 33(11), 2163-2176, 2010.
19. Zhou, Z. B., Sellami, M., Gaaloul, W., Barhamgi, M. and Defude, B., Data Providing Services Clustering and Management for Facilitating Service Discovery and Replacement. IEEE Transactions on Automation Science and Engineering, 10(4), 1131-1146, 2013.

20. Sellami, M., Gaaloul, W. and Tata, S., An Implicit Approach for Building Communities of Web Service Registries. The 13th International Conference on Information Integration and Web-Based Applications and Services, Ho Chi Minh City, Vietnam, December 5-7, 2011.
21. Wen, X. Z., Shao, L., Xue, Y. and Fang, W., A Rapid Learning Algorithm for Vehicle Classification. Information Sciences, 295(1), 395-406, 2015.
22. Wu, H. Y. and Du, Y. Y., A Logical Petri Net-Based Approach for Web Service Cluster Composition. Chinese Journal of Computer, 38(1), 204-218, 2015.
23. Aznag, M., Quafafou, M., Jarir, Z., Leveraging Formal Concept Analysis with Topic Correlation for Service Clustering and Discovery. IEEE International Conference on Web Services, 153-160, 2014.
24. Liu, J. X., He, K. Q., Wang, J., Feng, Z. W. and Ning, D., A Semantic Interoperability Oriented Method of Service Aggregation. Chinese Journal of Software, 22(2), 27-40, 2011.
25. Liu, J. X., He, K. Q., Wang, J., Yu, D. H., Feng, Z. W., D. Ning and Zhang, X. W., An Approach of RGPS-Guided On-demand Service Organization and Recommendation. Chinese Journal of Computers, 36(2), 238-251, 2013.
26. Liu, J. X., Wang, J., He, K. Q., Liu, F. and Li, X. X., Service organization and recommendation using multi-granularity approach. Knowledge-Based Systems, 73, 181-198, 2015.
27. Wu, J., Liang, Q. H. and Jian, H. Y., Bayesian Network Based Services Recommendation. IEEE Asia-Pacific Services Computing Conference, 313-318, 2009.
28. Wu, J., Chen, L., Jian, H. Y. and Wu, Z. H., Composite Service Recommendation Based on Bayes Theorem. International Journal of Web Services Research, 9(2), 69-93, 2012.
29. Murphy, K. P., The Bayes Net Toolbox for Matlab. 2001.