Research of Web Service Recommendation Using Bayesian Network Reasoning

Jianxiao Liu, Zonglin Tian, Yifei Liu, and Liang Zhao(✉)

College of Informatics, Huazhong Agricultural University, Wuhan, China
zhaoliang323@mail.hzau.edu.cn

Abstract. How to recommend the atomic and a set of services with correlations to meet users' functional and non-functional requests is a key problem to be solved in the era of services computing. On the basis of organizing service clusters with different functions using the three-stage Bayesian network structure learning method. It uses the parameter learning method to obtain the conditional probability table (CPT) of all the nodes. The Bayesian network reasoning method (Gibbs Sampling) is used to recommend a set of service types that are interested to users. Finally, it selects a set of services in the specific service clusters to meet users' functional and QoS requirements. The case study and experiments are used to explain and validate the effectiveness of the proposed method.

Keywords: Bayesian network learning · Web service
Service recommendation · Gibbs sampling

1 Introduction

In the era of service-oriented computing, how to recommend the atomic service and a set of services with correlations to meet users' functional and non-functional *QoS* requirements is an important problem to be solved in the service-oriented software engineering [1].

There are a large number of Web services on the internet, and the most frequently used method is recommending services according to users' personal requirements. At present, there exists a lot of research work about service recommendation, such as collaborative filtering, using users' history usage information, *QoS*-aware method, latent semantic probabilistic model, Bayesian theory and some other approaches. The above research work mainly use certain approach to recommend services for users, and it mainly concentrates on the aspect of service function. However, there are a lot of services with similar function but have different *QoS* values on the internet. In addition, users usually need a set of services that can be composited to realize specific function. In order to solve the above problem, we can cluster services firstly, and then organize the service clusters to realize service organization network graph. Then we can recommend a set of service with correlations effectively and conveniently according to users' personal requirements. Bayesian network combines the acyclic graphs and probability theory, and it has solid foundation of probability theory. It has the advantages of constructing causal relationship, doing reasoning, mining the implicit

J. E. Ferreira et al. (Eds.): SCC 2018, LNCS 10969, pp. 19–35, 2018.
https://doi.org/10.1007/978-3-319-94376-3_2

knowledge, and so on. There are two kinds of Bayesian network structure learning methods: search score method and dependency analysis method [2]. This search score method uses the local or random search strategy. It is a combinatorial explosion problem as the number of nodes increases, and this 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. The three-phase dependency analysis algorithm (*TPDA*) [3] is a commonly used dependency analysis method. Therefore, we mainly use the three-phase dependency analysis Bayesian network structure learning method to organize services and thus to construct the service organization network in this work. The main work is given as follows.

(1) It uses the three-stage Bayesian network structure learning method to organize Web services on the basis of using the service invocation history records. The Bayesian network parameter learning method is used to learn the conditional probability of all the nodes in the service organization network graph.
(2) On the basis of realizing service organization, it proposes a Web service recommendation method based on Bayesian network reasoning. The Gibbs sampling approach is used to calculate the conditional probability between particular service nodes. This method can recommend and help users to select the atomic and a set of services with the proper function and *QoS*.
(3) Experiments are conducted to validate the proposed methods, and the case study is used to do the Explanation.

2 Web Service Organization and Recommendation

2.1 Web Service Organization

The process of realizing *TPDA* method mainly includes three steps: *Drafting*, *Thickening* and *Thinning*.

(1) The first learning stage: *Drafting*

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

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

Output: *graph*, *R*

1: $c_1, c_2\leftarrow 0, S\leftarrow\varnothing, v_I\leftarrow 0, R\leftarrow\varnothing$
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**($v_I > \varepsilon$) **then**
11:       $S\leftarrow S\cup\langle clusws_{c1}, clusws_{c2}, v_I\rangle$
12:     **end if**
13:**end for**
14:$S\leftarrow Sort(S)$//Sort S using $Imutual(clusws_{c1}, clusws_{c2}, Rel_{ws})$
15:**for all** $\langle clusws_{c1}, clusws_{c2}, Imutual(clusws_{c1}, clusws_{c2}, Rel_{ws})\rangle$ in S **do**
16:   **if**(*ExistsPath*($node_{c1}$, $node_{c2}$)) **then** //exists the open path
17:     $R\leftarrow R\cup\langle clusws_{c1}, clusws_{c2}\rangle$
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. The initial network graph will be constructed firstly using step 2–6. The $I(clusws_i, clusws_m, Rel_{ws})$ is used to calculate the mutual information between two service cluster nodes, and the concrete process can be seen in [5]. 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.

(2) The second learning stage: *Thickening*

The second stage firstly finds the cut set between two nodes when there is an open path between them in the network. Then the conditional mutual information about the two nodes and cut set 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.

(3) The third stage: *Thinning*

In the third stage, for each edge e in the graph, it will be removed temporarily. Then we will find the minimum cut set 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 the network.

2.2 Web Service Recommendation

According to users' personal requirements, we can recommend the atomic and a set of services with proper function and *QoS* in the organized services. This section uses the Bayesian network reasoning method to get the service cluster node that can meet users' functional requirements firstly. Then it selects a set of services with the proper *QoS* values in different service clusters for users in further.

Given specific network and evidence variable set, Bayesian network reasoning refers to calculate the posterior probability $P(X \mid E)$ of an event occurrence using the joint probability formula. It mainly includes two ways: causal reasoning and diagnostic reasoning. This section mainly introduces how to recommend a set of service types for users using Bayesian network causal reasoning method. These service types mean the service cluster nodes that are interested for users. Algorithm 2 gives the process of how to realize Web service recommendation based on Bayesian network reasoning method.

Algorithm 2. Web service recommendation based on Bayesian network Reasoning(*BRWSR*)
Input: *RE*, *graph*, *CPT*, *Cluster*, Rel_{ws}
Output: *wsnodes*

```
1: vr←0, wsnodes←∅, rnode←∅, pathnode←∅
2: fori=1 tograph.cnumdo
3:   if(matchreq(RE, cluswsi)> α ) then
4:     rnodes←rnodes ∪ <nodes[i], 1.0>
5:   end if
6: end for
7: forall rnodes[r] in rnodes do
8:   wsnodes←wsnodes ∪ <rnodes[r], 1.0>
9:   pathnode←getpathnode(rnodes[r])
10:   forall pathnode[p] in pathnode do
11:     vr←p(pathnode[p] | rnodes[r])
12:     if(vr> γ ) then
13:       wsnodes←wsnodes ∪ <pathnode[p], vr>
14:     end if
15:   endfor
16:   pathnode←∅
17:endfor
18:wsnodes←DelDuplicate(wsnodes)
19:wsnodes←Sort(wsnodes)
20:returnwsnodes
```

In Algorithm 2, all the service clusters are done matching calculation according to users' functional requirements *RE* firstly. The service nodes whose matching degree are larger than the threshold will be found and recommended for users, and thus to form service cluster node set *rnodes*, as seen in step 2–6. Then all the nodes *rnodes*[*r*] in *rnodes* are judged in turn. It will add *rnodes*[*r*] and its matching degree (1.0) into the result node set *wsnodes*. It will also find the execution path *pathnode* of *rnodes*[*r*] in *graph*. The causal reasoning method is used to calculate the conditional probability *p* (*pathnode*[*p*] | *rnodes*[*r*]) of the related nodes in *pathnode*. When the conditional probability is larger than the threshold, we will add *pathnode*[*p*] and the matching degree into result node set *wsnodes*. Finally, the duplicate node in *wsnodes* will be removed, and all the nodes will be sorted according to the matching degree. Finally, return *wsnodes*.

The step 3 in Algorithm 2 is used to calculate the matching degree between users' request and services in service clusters considering of service interface and execution capability. The step 11 in Algorithm 2 is used to calculate the conditional probability between nodes. However, the complexity of precise reasoning is relatively high and the efficiency is too low for the large-scale and multi-connectivity Bayesian network. This leads to the inoperability for the large-scale Bayesian network, and it is a NP Hard problem. Therefore, it needs to use the approximate reasoning method. Markov Chain Monte Carlo (*MCMC*) method is a commonly used approximate reasoning method, including Gibbs sampling algorithm (Gibbs Sampling) and hybrid MCMC algorithms (Hybrid Monte Carlo Sampling) etc. This kind of algorithm is very effective when there is no extreme probability in the network. There is no extreme probability distribution between Web service cluster and Gibbs sampling algorithm using Markov chain theory (Markov coverage), it can ensure the results of the algorithm returns the convergence in real posterior probability. Therefore, we mainly use the approximate reasoning algorithm. Algorithm 3 is used to calculate the conditional probability $P(ws_{pi} \mid ws_{ej})$ between different services (like ws_{pi} and ws_{ej}) in specific service cluster node (like $node_p$ and $node_e$). Then we can calculate the conditional probability $p(node_p \mid node_e)$ of the corresponding service cluster node in further.

Algorithm 3. Service conditional probability calculation using Gibbs sampling (*GSWCP*)
Input: $graph$, $Cluster$, Rel_{ws},$node_p$, ws_{pi}, $node_e$, ws_{ej}
Output: $p(node_p{=}ws_{pi} \mid node_e{=}ws_{ej})$
$m_q \leftarrow 0$, $Set_{sample} \leftarrow \varnothing$, $m \leftarrow 0$, $D[] \leftarrow \varnothing$, $nodes_{mb} \leftarrow \varnothing$, $val_{mb} \leftarrow \varnothing$, $nodes_{ne} \leftarrow graph.nodes\text{-}node_e$
$Set_{sample} \leftarrow Getsampleset(Cluster, Rel_{ws})$
$m \leftarrow Set_{sample}.length$
generate $D[1]$ with $node_e{=}ws_{ej}$ from Set_{sample} randomly
if $(D[1].node_p{=}{=}ws_{pi})$ **then**
    $m_q \leftarrow m_q+1$
end if
for$i$=2 **to**m**do**
    $D[i] \leftarrow D[i\text{-}1]$
    forall $nodes_{ne}[j]$ in $nodes_{ne}$ **do**
        $nodes_{mb} \leftarrow MB(nodes_{ne}[j])$ //getting Markov coverage nodes of $nodes_{ne}[j]$
        $val_{mb} \leftarrow D[i].nodes_{mb}$ // getting the value of Markov coverage nodes in $D[i]$
        $D[i].nodes_{ne}[j] \leftarrow Sampleing(p(nodes_{ne}[j].val \mid val_{mb}))$
    endfor
    if $(D[\mathrm{i}].node_p{=}{=}ws_{pi})$ **then**
        $m_q \leftarrow m_q+1$
    end if
endfor
return m_q/m

In Algorithm 3, m in *Input* represents the sample size, $node_p$ and ws_{pi} represent the query variable node and the corresponding value. The $node_e$ and ws_{ej} represent the evidence variable node and the corresponding value. The $nodes_{ne}$ represents non-evidence variable node set, and $nodes_{mb}$ represents the Markov coverage nodes of a node. A node's Markov coverage nodes include the parent node, child node and other parent nodes of its child node. *MB*() in step 11 is used to get the Markov coverage nodes of a particular node. Set_{sample} in step 2 represents sample set. Each sample can be got through constructing the exact match path between services in Rel_{ws}.

The step 4–19 in Algorithm 3 gives the process of how to calculate the conditional probability between nodes. It generates the sample $D[1]$ which is consistence with the evidence variable node ($node_e = ws_{ej}$) firstly. If $D[1]$ meets $node_p = ws_{pi}$, then m_q plus 1, as seen in step 4–7. Step 10–14 is used to operate on all the non-evidence variable node $nodes_{ne}[j]$ in turn according to the topological order. The Markov coverage nodes $nodes_{mb}$ of $nodes_{ne}[j]$ will be got firstly, then get val_{mb} of $nodes_{mb}$ in $D[i]$. Then it will calculate $p(nodes_{ne}[j] \mid val_{mb})$, sample and update the $nodes_{ne}[j]$ in $D[i]$ using the sample result. Step 15–17 is used to judge $D[i]$ whether it meets $node_p = ws_{pi}$ or not according to the sample result. If it meets the condition, $m_q$will be added 1. It will operate m times in turn using the above methods. Finally, it calculates m_q/m and return.

2.3 Web Service Selection of QoS

The service node set *wsnodes* that can meet users' specific requirements can be got using Web service recommendation method. Each node can correspond to specific service cluster. Then it will select a set of services with better *QoS* values in different service clusters according to *RE*. We mainly use the following two approaches.

(1) On the basis of selecting services with proper function, the services with better *QoS* values will be selected. It mainly uses the following steps.
   (a) After calculating the conditional probability $p(node_m = ws_{mn} \mid node_i = ws_{ij})$ between specific service nodes using Gibbs sampling algorithm (Algorithm 3), we store the probability of recommending ws_{pi} in the condition of service ws_{ej}.
   (b) For specific service $node_e = ws_{ej}$, it sorts all the services in $node_p$ according to $p(node_p = ws_{pi} \mid node_e = ws_{ej})$.
   (c) The users' request *RE* and services will be done matching calculation from the functional level, service ws_{ij} in specific service cluster $clusws_i$ which can meet users' functional requirements can be got.
   (d) The service cluster $clusws_m$ of all the nodes in *wsnodes* are operated in turn to get the probability $p(ws_{mn} \mid ws_{ij})$ of each service ws_{mn} in $clusws_m$. It sorts service ws_{mn} in the descending order, and calculates the matching value between *RE.ReQoS* and $ws_{mn}.QoS$. When the matching value is larger than the threshold, it will recommend service ws_{mn} for users.
   (e) According to above methods, the services in service cluster $clusws_m$ of all the nodes in *wsnodes* are judged in turn. Then the service set related to ws_{ij} can be got, thus it can recommend a set of services with better function and *QoS* values for users.
(2) Select a set of services with better *QoS* values from different service clusters directly

On the basis of getting service execution path node set *wsnodes*, this method will judge each service ws_{ij} in $clusws_i$ of $node_i$ in *wsnodes*. The matching value between *RE*.

$ReQoS$ and $ws_{mn}.QoS$ will be calculated, and it will select the services with the largest QoS matching value.

3 Case Study

Example 1. $Cluster = \{clusws_c,\ 1 \le c \le 7\}$. We use $A \sim G$ to express the service clusters, and it is denoted as $clusws_A \sim clusws_G$. The service number in $clusws_A \sim clusws_G$ is {5, 3, 6, 7, 7, 3, 5} respectively. We can see $clusws_A$ contains 5 services, $clusws_A = \{ws_{Aw},\ 1 \le w \le 5\}$. $Rel_{ws} = \{rel_r\!: ws_{ij} \rightarrow ws_{mn},\ 1 \le r \le 51,\ 0 \le i, m \le 7,\ 0 \le j \le c_i,\ 0 \le n \le c_m\}$. 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}
$clusws_A$	$ws_{Aj} \rightarrow ws_{Bn}(clusws_B)$: $<A_0, B_0> <A_0, B_1> <A_0,B_2> <A_1, B_0> <A_1, B_1> <A_1, B_2>$ $ws_{Aj} \rightarrow ws_{Cn}(clusws_C)$: $<A_0, C_3> <A_1, C_4> <A_1, C_5> <A_2, C_4> <A_2, C_5>$ $ws_{Aj} \rightarrow ws_{En}(clusws_E)$: $<A_0, E_0> <A_1, E_1> <A_2, E_2> <A_3, E_3> <A_4, E_1>$
$clusws_B$	$ws_{Bj} \rightarrow ws_{Cn}(clusws_C)$: $<B_0, C_0> <B_1, C_1> <B_2, C_2> <B_1, C_3> <B_0, C_4>$ $ws_{Cj} \rightarrow ws_{Dn}(clusws_D)$: $<C_1, D_4> <C_3, D_6>$
$clusws_C$	$ws_{Cj} \rightarrow ws_{En}(clusws_E)$: $<C_1, E_1> <C_5, E_5>$ $ws_{Cj} \rightarrow ws_{Fn}(clusws_F)$: $<C_0, F_0> <C_1, F_1> <C_2, F_2> <C_3, F_2> <C_4, F_1> <C_5, F_0> <C_4, F_2> <C_1,F_0>$
$clusws_D$	$ws_{Dj} \rightarrow ws_{En}(clusws_E)$: $<D_0, E_0> <D_1, E_1> <D_2, E_2> <D_3, E_3> <D_4, E_4> <D_5, E_5> <D_6, E_6> <D_4, E_4> <D_2, E_1>$
$clusws_E$	$ws_{Ej} \rightarrow ws_{Gn}(clusws_G)$: $<E_0, G_0> <E_1, G_1> <E_2, G_2> <E_3, G_3> <E_2, G_4> <E_1, G_3> <E_0, G_2>$
$clusws_F$	–
$clusws_G$	$ws_{Gj} \rightarrow ws_{Fn}(clusws_F)$: $<G_4, F_1> <G_4, F_2>$

(1) **Web service organization using TPDA**

The concrete process of service organization can be seen in [5], and the initial graph will be got, as shown in Fig. 1(1). Using the third stage of *Thinning*, we can get the edges are not changed. The final network structure is shown in Fig. 1(2).

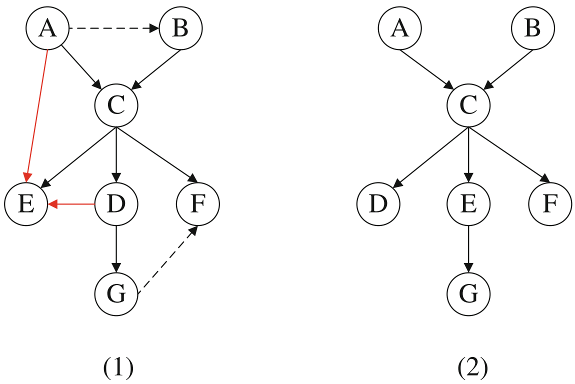

Fig. 1. The structure graph of service nodes

(2) **Web service recommendation based on Bayesian network reasoning**

(a) Supposing $rnodes = \{A\}$, $wsnodes \leftarrow wsnodes \cup \{<A, 1.0>\} \Rightarrow wsnodes = \{<A, 1.0>\}$ through step 4 in Algorithm 2, the path is denoted by $getpathnode(rnodes[r]) = \{\{A, C, D\}, \{A, C, E, G\}, \{A, C, F\}\} \Rightarrow pathnode = \{A, C, D, E, F, G\}$.

(b) All the nodes in *pathnode* are done the calculation of $p(pathnode[p] \mid rnodes[r])$ using step 10–15 in Algorithm 2. For example, when to calculate $p(E \mid A) = 0.14283879$, supposing $\gamma = 0.1$, we can get $p(E \mid A) > \gamma$, and $wsnodes \leftarrow wsnodes \cup <E, 0.14283879>$. Then we can get $wsnodes = \{<A, 1.0>, <C, 0.13331947>, <D, 0.1428377>, <E, 0.14283879> <F, 0.3999311>, <G, 0.19997229>\}$.

(c) All the nodes in *wsnodes* will be sorted using step 19 in Algorithm 2, and we can get $wsnodes = \{A, F, G, E, D, C\}$.

When to calculate $p(pathnode[p] \mid rnodes[r])$ in above step (b), such as calculating $p(E \mid A)$, it can be got using Eq. (1). There are 5 services in the $clusws_E$ of node E, and 7 services in the $clusws_A$ of node A.

$$p(E \mid A) = \sum_{i=1}^{5} \sum_{j=1}^{7} p(ws_{Ei} | ws_{Aj}) \tag{1}$$

The $p(ws_{Ei} \mid ws_{Aj})$ in Eq. (1) can be calculated through Algorithm 3. For example, we use the following steps to calculate $p(ws_{E2} \mid ws_{A1})$.

(a) The service ws_{A1} in node A is evidence variable, and service ws_{E2} in node E is query variable. In Fig. 1(2), we can get non-evidence node $nodes_{ne} = \{B, C, D, E, F, G\}$.

(b) Using the given *Cluster* and Rel_{ws}, we can get Set_{sample}, and then $m = 345$.

(c) Generate $D[1]$ whose evidence variable $node_e$ is ws_{A1} in Set_{sample}, and $D[1] = \{ws_{A1}, ws_{B0}, ws_{C2}, ws_{D6}, ws_{E2}, ws_{F2}, ws_{G0}\}$. Then we can get $D[1].node_p = ws_{E2} \Rightarrow m_q = m_q + 1 \Rightarrow m_q = 1$.

(d) $D[2] = D[1] \Rightarrow D[2] = \{ws_{A1}, ws_{B0}, ws_{C2}, ws_{D6}, ws_{E2}, ws_{F2}, ws_{G0}\}$. The nodes in $nodes_{ne}$ are operated using the following steps.

When $j = 0$, $nodes_{ne}[0] = B$, $nodes_{mb} \leftarrow MB(B) \Rightarrow nodes_{mb} = \{A, C\} \Rightarrow val_{mb} = \{ws_{A1}, ws_{C2}\}$. Using step 13 in Algorithm 3, we can get $p(nodes_{ne}[0] \mid val_{mb}) = p(ws_{B0} \mid ws_{A1}, ws_{C2}) = 0.5$. Through the sampling calculation, the $D[2].B$ of node B in $D[2]$ will be updated to ws_{B1}. Then $D[2] = \{ws_{A1}, ws_{B1}, ws_{C2}, ws_{D2}, ws_{E2}, ws_{F2}, ws_{G0}\}$.

The calculation approach of $p(ws_{B1} \mid ws_{A1}, ws_{C2})$ is shown in Eq. (2), and $p(ws_{C2} \mid ws_{A1}, ws_{B1})$ can be got from CPT of node C.

$$p(ws_{B1}|ws_{A1}, ws_{C2}) = \frac{p(ws_{A1}, ws_{C2}, ws_{B1})}{p(ws_{A1}, ws_{C2})} = \frac{p(ws_{A1}) * p(ws_{B1}) * p(ws_{C2}|ws_{A1}, ws_{B1})}{p(ws_{C2}|ws_{A1}) * p(ws_{A1})} \tag{2}$$

When $j = 1$, then $nodes_{ne}[1] = C$, $nodes_{mb} \leftarrow MB(C) \Rightarrow nodes_{mb} = \{A, B, D, E, F\} \Rightarrow val_{mb} = \{ws_{A1}, ws_{B1}, ws_{D2}, ws_{E2}, ws_{F2}\}$. Using step 13 in Algorithm 3, we get $p(nodes_{ne}[1] \mid val_{mb}) = p(ws_{C2} \mid ws_{A2}, ws_{B1}, ws_{D2}, ws_{E2}, ws_{F2}) = 0.333$. Through the sampling calculation, the $D[2]$. C of node C in $D[2]$ is ws_{C2}. And its value is not changed. Then $D[2] = \{ws_{A1}, ws_{B1}, ws_{C2}, ws_{D2}, ws_{E2}, ws_{F2}, ws_{G0}\}$.

All the nodes in $nodes_{ne}$ are operated using above steps, we can get all the value of $D[i].D[2] = \{ws_{A1}, ws_{B1}, ws_{C2}, ws_{D2}, ws_{E2}, ws_{F2}, ws_{G0}\}$, and $D[2].node_p = ws_{E2} \Rightarrow m_q = m_q + 1 \Rightarrow m_q = 2$.

(e) Then $D[3] = D[2]$. It will operate $345(m)$ times. And we can get $m_q = 87$, then $m_q/m = 0.2522$.

4 Related Work

At present, the research work about Web service recommendation includes the following approaches: collaborative filtering, using users' history usage information, *QoS*-aware, latent semantic probabilistic model, Bayesian theory and some other approaches. Most research work main uses the collaborative filtering method. Zheng et al. have proposed a *QoS*-aware Web service recommendation method by collaborative filtering [6]. The collaborative filtering method is used to predict the *QoS* values of Web services, and it mainly takes advantages of the past usage experiences of service users. In [7], a novel collaborative filtering algorithm is designed for large scale Web service recommendation. It mainly employs the characteristic of *QoS* and achieves considerable improvement on the commendation accuracy, and the recommendation visualization technique is also used as the auxiliary method. Nguyen *et al.* in [8] have proposed a collaborative filtering technique for Web service recommendation method based on user-operation combination. This method makes full use of the history usage records between users and operation, and it can recommend the services for users with the most similar service user preferences. Jiang *et al.* have proposed an effective Web service recommendation method based on personalized collaborative filtering [9]. It takes into account the personalized influence of services when computing similarity measurement between users and personalized influence of services. The personalized hybrid collaborative filtering (*PHCF*) technique by integrating personalized user-based algorithm and personalized item-based algorithm is proposed. Chen *et al*. have proposed a scalable hybrid collaborative filtering algorithm for personalized Web service recommendation [10], and their method can promote the personal Web service discovery. Kuang et al. have proposed a personalized services recommendation method based on context-aware *QoS* prediction [11]. This method refers the previous service invocation experiences under similar context with the current consumer. It clusters the service invocation records according to the similarity on context properties and selects the cluster that is most similar to the context of current consumer. And it predicts the *QoS* of an unused service for current consumer based on the filtered recommendation records by Bayesian inference. The above several mentioned methods mainly use the collaborative filtering method to recommend the proper services in the view of different aspects (such as *QoS*, users' operation, context, etc.). The services with different functions are organized in advance in our method, and the services are then recommended based on users' request information.

In addition, Kang *et al*. have proposed an active Web service recommendation (AWSR) [12] method based on usage history. It extracts users' functional interests and *QoS* preferences from his/her usage history. This method firstly calculates the similarity between users' functional interests and a candidate Web service. The hybrid new metric of similarity is used to combine functional similarity measurement and nonfunctional similarity measurement based on comprehensive *QoS* of Web services. The Top-K Web service list is recommended for users. This method can recommend the proper atomic service. However, our approach concentrates on recommending a set of services with correlations based on service organization in the view of function and *QoS*. In [13], a personalized Web service recommendation method based on latent semantic probabilistic model is proposed. It establishes the latent semantic relations among users, users' preferences and service situations. Then it uses the trained model to predict users' criteria preferences. Pan *et al*. have proposed a service classification and recommendation method based on software network [14]. The software network is used to describe the compositional strength between services, and the corresponding service algorithms have been proposed. Lee et al. have used the approach of member organization-based group similarity measures to realize service recommendation in Internet of Things environments [15]. Yu have proposed a framework named CloudRec to realize personalized service Recommendation in the Cloud. It exploits a user-centric strategy to achieve personalized QoS assessment of cloud services [16]. Kumara *et al*. have proposed a cluster-based Web service recommendation method [17]. It considers semantic similarity between services in the clustering process and the association between services. Cao *et al*. have proposed a mashup service recommendation method based on usage history and service network [18]. This approach firstly extracts users' interests from their Mashup service usage history and builds a service network based on social relationships information among Mashup services, APIs and their tags. Meng et al. mainly concentrate on the service recommendation for big data application [19]. This method aims at presenting a personalized service recommendation list and recommending the most appropriate services to users.

On the basis of organizing service from aspects of users' role, request goal and execution process, Liu *et al.* have proposed several service recommendation algorithms using users' different request information in [20, 21]. Wu *et al.* in [22, 23] have proposed a composite service recommendation method using Bayesian theory. They mainly analyze 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 recommend the optimal service sequence for users. The Bayesian theory is also used in our method. The difference is our method mainly concentrates on using the Bayesian structure learning theory to organize service clusters. Then users can firstly select the services that they are interested in, and thus use Bayesian network reasoning method to recommend services with correlations for users. This is different with the above method of recommending services based on the used service sequence. In addition, users will select the services with proper *QoS* values in different service clusters in further based on recommending different service types in our method.

5 Experiment

BN Toolkit(BNT) is a software development kit about Bayesian network learning using Matlab by Murphy [24]. This package does not support the algorithm of three-stage dependency analysis, and we implement this algorithm in this work. The experiment mainly compares our method with the algorithms of *K2*, hill-climbing (*HC*), greedy search (*GS*) and Markov chain Monte Carlo (*MCMC*) of realizing service organization. We denote these algorithms as *K2WS*, *HCWS*, *GSWS*, *MCMC* and *TPDA*. The experiment is carried out on the computer with the configuration of dual Intel (R) Core (TM)2 i5 CPU 760@ 2.80 GHz, and 4 G memory.

5.1 Web Service Organization Experiment

The experiment data is generated randomly. The *cnum* refers to the number of different service clusters, *snum* refers to the service numbers in different service clusters, *rnum* refers to number of service execution history records, as shown in Table 2.

Table 2. 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 accuracy.

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 Figs. 2, 3 and 4.

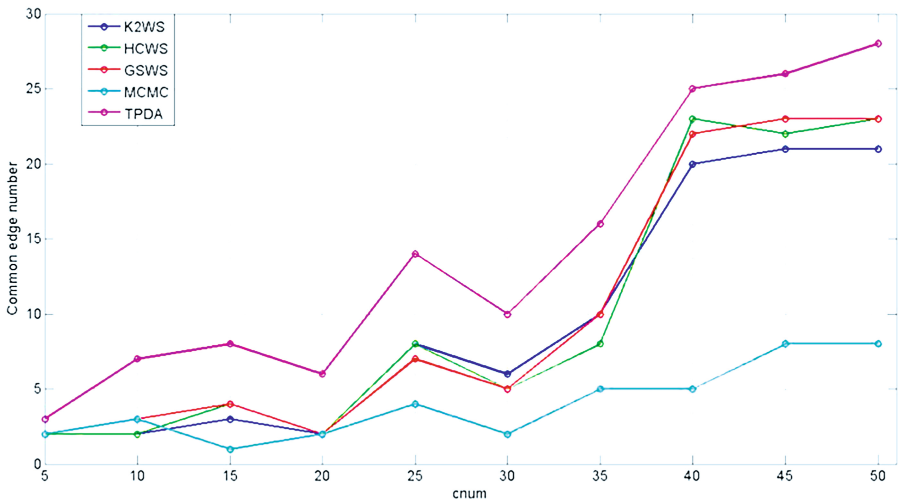

Fig. 2. Comparison of common edge number of different methods

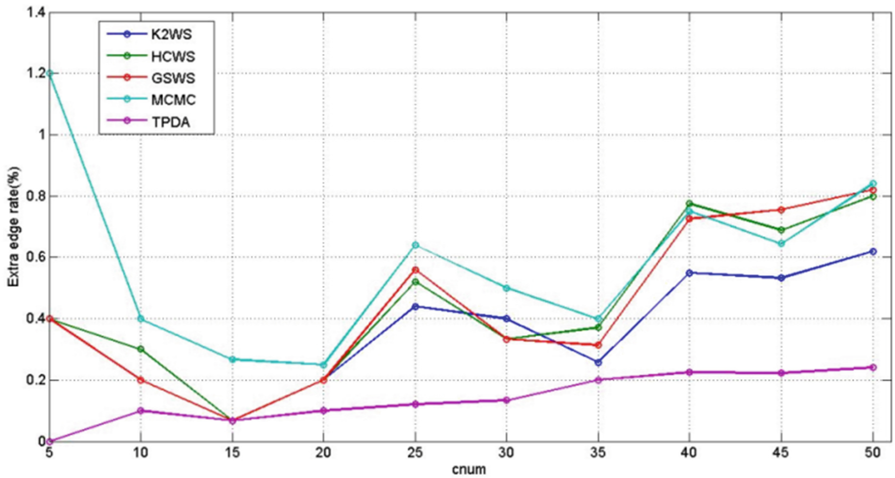

Fig. 3. Comparison of extra edge rate of different methods

The threshold in *TPDA* is set to 0.15. 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, it can learn the network with the better structure. The learning effect of *K2WS*, *HCWS* and *GSWS* is about same. The corresponding learning effect is better than *MCMC*, but it is less than *TPDA* method.

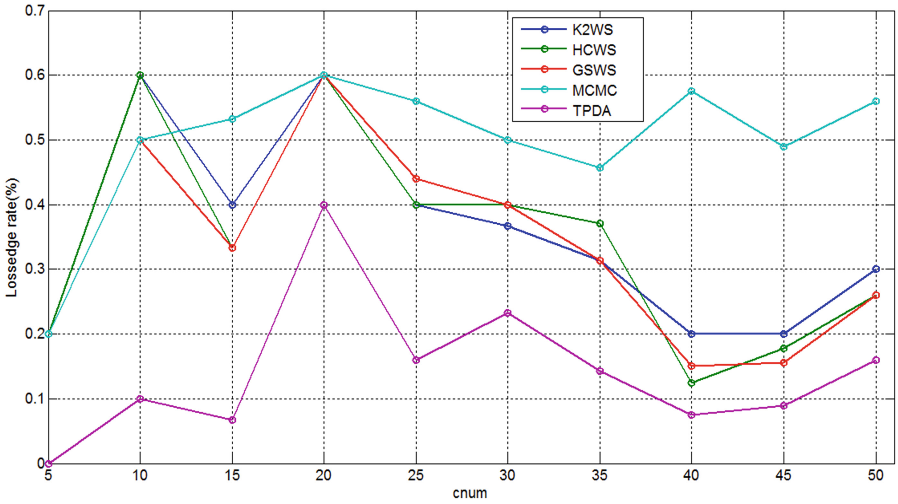

Fig. 4. Comparison of loss edge rate of different methods

5.2 Web Service Recommendation Experiment

Experiment 2. Comparison of service recommendation efficiency.

On the basis of realizing service organization using the methods of *K2WS*, *HCWS*, *GSWS*, *MCMC* and *TPDA*, we use three Bayesian network reasoning methods (*Variable Elimination*, *Join Tree*, *Gibbs Sampling*) to realize Web service recommendation respectively when to use two Bayesian network parameter learning approaches (*MLE* and *BE*). In the case of setting *cnum* to 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 and 14, this experiment compares the service recommendation time of the above mentioned methods. The result is shown in Table 3.

Note: the time in Table 3 refers to the using time of recommending related nodes for all the nodes when we look the service cluster node as evidence node. And it is measured in seconds.

In Table 3, the service recommending time of all the approaches are becoming more as the service cluster number *cnum* increases. For the specific Bayesian network reasoning method, the recommending time of maximum likelihood estimation (*MLE*) parameter method is more than the Bayesian estimation method. For the specific Bayesian network structure learning method, the time of the three Bayesian network reasoning approaches is all about same. The time of Join Tree recommendation method is the least of all, and the time of Gibbs sampling method is the most. The time of variable elimination method is in the middle. In addition, the service recommendation time of *K2WS*, *HCWS*, *GSWS* and *MCMC* is about same, but it is less than the *TPDA* method.

Table 3. Comparison of service recommendation efficiency

cnum Methods			4	5	6	7	8	9	10	11	12	13	14
K2WS	Variable elimination	MLE	0.49	0.71	0.8	0.90	1.67	2.3	2.83	3.56	4.74	5.70	6.67
		BE	0.31	0.56	0.67	0.70	1.6	2.19	2.62	2.98	3.98	5.57	6.44
	Join tree	MLE	0.38	0.25	0.25	0.30	0.78	0.93	1.4	1.46	1.63	1.84	1.91
		BE	0.08	0.19	0.20	0.30	0.46	0.69	1.22	1.39	1.48	1.59	1.66
	Gibbs sampling	MLE	0.68	0.82	0.86	0.91	1.76	2.26	2.71	3.11	3.66	3.99	4.52
		BE	0.48	0.79	0.79	0.82	1.7	2.23	2.51	2.73	2.99	3.3	4.34
HCWS	Variable elimination	MLE	0.37	0.50	0.71	0.80	1.59	2.13	2.84	4.02	5.72	5.75	6.25
		BE	0.25	0.45	0.7	0.70	1.36	1.82	2.63	3.88	5.14	5.51	5.66
	Join tree	MLE	0.16	0.21	0.34	0.40	0.64	0.92	1.36	1.75	1.92	2.32	3.65
		BE	0.07	0.13	0.24	0.30	0.45	0.78	1.16	1.66	1.79	2.22	2.98
	Gibbs sampling	MLE	0.38	0.53	0.75	0.81	1.80	2.28	2.88	3.34	4.36	5.65	6.26
		BE	0.29	0.45	0.53	0.61	1.62	2.28	2.77	3.11	3.59	5.53	6.234
GSWS	Variable elimination	MLE	0.38	0.52	0.70	0.91	1.77	2.46	2.8	4.10	5.59	5.69	6.86
		BE	0.27	0.44	0.52	0.70	1.72	2.30	2.61	3.77	5.33	5.65	6.83
	Join tree	MLE	0.16	0.20	0.37	0.51	0.74	0.99	1.35	1.72	2.01	3.33	4.26
		BE	0.07	0.13	0.24	0.41	0.55	0.88	1.24	1.63	1.86	2.21	4.08
	Gibbs sampling	MLE	0.42	0.53	0.76	0.81	1.82	2.26	2.77	2.89	3.04	4.61	5.54
		BE	0.35	0.45	0.64	0.8	1.75	2.22	2.72	2.8	2.77	4.35	5.38
MCMC	Variable elimination	MLE	0.42	0.85	0.85	1.0	1.6	2.33	2.56	3.18	5.04	5.57	6.39
		BE	0.25	0.68	0.78	0.96	1.49	2.17	2.48	2.04	5.0	5.42	6.1
	Join tree	MLE	0.2	0.28	0.38	0.61	0.83	1.2	1.44	1.69	1.79	2.29	2.66
		BE	0.08	0.14	0.25	0.49	0.64	1.11	1.31	1.49	1.54	2.15	2.33
	Gibbs sampling	MLE	0.36	1.05	1.22	1.31	1.72	2.3	2.6	3.25	5.43	5.93	7.11
		BE	0.32	0.91	1.16	1.22	1.66	2.22	2.35	2.53	4.93	5.0	7.0
TPDA	Gibbs sampling	BE	2.01	2.16	5.2	7.55	11.1	13.9	15.9	21	24.5	28.1	31.3

Experiment 3. Comparison of service recommendation number of different methods.

On the basis of organizing services using *K2WS*, *HCWS*, *GSWS*, *MCMC* and *TPDA*, we use Gibbs sampling method to realize service recommendation. In the case of setting *cnum* to 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 and 14, the number of recommending services using different methods is shown in Table 4.

In Table 4, for the several methods, the number of recommending service is different when setting *cnum* to different values. For specific *cnum*, the recommending service number of *MCMC* is the least of all, and *TPDA* method is the most. The methods of *K2WS*, *HCWS* and *GSWS* are in the middle.

Through Experiment 2 and 3, we can see the service recommendation time of *TPDA* is slightly larger than other methods. But this method can recommend the most number of services.

Table 4. Comparison of service recommendation number of different methods

cnum / *Methods*	4	5	6	7	8	9	10	11	12	13	14
K2WS	10	7	8	15	10	15	16	19	25	28	27
HCWS	9	7	6	14	10	13	16	17	20	26	27
GSWS	9	6	6	10	8	13	12	14	17	20	18
MCMC	5	3	5	7	4	8	7	10	9	14	9
TPDA	10	9	11	19	12	20	18	22	32	33	34

Experiment 4. Comparison of service recommendation number of different thresholds.

In the case of setting different value of threshold γ in Algorithm 4, this experiment compares the service recommendation number using Gibbs sampling method in *TPDA*. It also analyzes the impact of threshold to service recommendation number. The result is shown in Table 5.

Table 5. Comparison of service recommendation number of different thresholds

cnum / Thresholds	4	5	6	7	8	9	10	11	12	13	14
0.05	12	13	17	21	23	23	26	32	36	37	41
0.10	12	12	17	21	22	21	25	31	36	33	36
0.15	10	9	11	19	12	20	18	22	32	33	34
0.20	5	3	8	12	9	13	16	16	25	28	34
0.25	4	3	5	11	6	6	16	10	21	22	28
0.30	4	3	3	6	5	6	16	10	21	16	20
0.40	0	3	0	1	3	0	12	8	13	10	14
0.5	0	0	0	0	2	0	11	5	7	8	10

For the specific number of service cluster in Table 5, the service recommendation number is becoming less as the thresholds increases. The number of recommending service is becoming seldom when the threshold is set to 0.4. In addition, the number of recommending service shows the growing trend as the number of service clusters increases. The results are in good agreement with the experiment data.

6 Conclusion

In the era of service-oriented software engineering, how to effectively organize services and further to recommend a set of services for users is an urgent problem to be solved. In this work, we see different service clusters as nodes, and see the execution relationships between services as the edges between nodes in graph. We use the three-stage Bayesian network structure learning method to organize service clusters, and thus to form service cluster organization network graph. Two Bayesian network parameter learning methods (*MLE* and *BE*) are used to calculate the conditional probability of all the nodes, and thus to get the conditional probability table (CPT). The Bayesian network reasoning method (Gibbs sampling) is used to calculate the conditional probability and thus to realize Web service recommendation. A set of service type with correlations which can meet users' functional requirements will be recommended for users. On the basis of users' different QoS requirements, it will select services in further in different service clusters. Finally, the experiments and case study are used to do the validation. The next step research work mainly includes the following aspects: organizing Web services from the semantic level to improve the accuracy; optimizing the Bayesian network structure learning algorithm and improving the efficiency of service organization.

Acknowledgment. The authors are grateful to the valuable comments from the anonymous reviewers. 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, 2016CKB705. The Fundamental Research Funds for the Central Universities under grant No. 2662018JC030.

References

1. Papazoglou, P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: a research roadmap. Int. J. Cooper. Inf. Syst. **17**(2), 223–255 (2008)
2. Liu, F.: Research on Bayesian Network Learning Algorithm. Doctoral Dissertation of Beijing University of Posts and Telecommunications, pp. 1–13 (2007)
3. Cheng, J., Grainer, G., Kelly, J., Bell, D., Liu, W.R.: Learning bayesian networks from data: an information-theory based approach. Artif. Intell. **137**, 43–90 (2002)
4. Zhang, Y.P., Zhang, L.: Machine Learning Theory and Algorithm. Science Press, pp. 246–269 (2012)
5. Liu, J.X., Xia, Z.H.: An approach of web service organization using bayesian network learning. J. Web Eng. **16**(3&4), 252–276 (2017)
6. Zheng, Z.B., Ma, H., Michael, R.L., King, I.: QoS-aware web service recommendation by collaborative filtering. IEEE Trans. Serv. Comput. **4**(2), 140–152 (2011)
7. Chen, X., Zheng, Z.B., Liu, X.D., Huang, Z.C., Sun, H.L.: Personalized QoS-aware web service recommendation and visualization. IEEE Trans. Serv. Comput. **6**(1), 35–47 (2013)
8. Ngoc Chan, N., Gaaloul, W., Tata, S.: Collaborative filtering technique for web service recommendation based on user-operation combination. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 222–239. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16934-2_17
9. Jiang, Y.C., Liu, J.X., Tang, M.D., Liu, X.Q.: An effective web service recommendation method based on personalized collaborative filtering. In: IEEE International Conference on Web Services, pp. 211–218 (2011)
10. Chen, X., Liu, X.D., Huang, Z.C., Sun, H.L.: RegionKNN: A scalable hybrid collaborative filtering algorithm for personalized web service recommendation. In: IEEE International Conference on Web Services, pp. 9–16 (2010)
11. Kuang, L., Xia, Y.J., Mao, Y.X.: Personalized services recommendation based on context-aware QoS prediction. In: IEEE 19th International Conference on Web Services, pp. 400–406 (2012)

12. Kang, G.S., Liu, J.X., Tang, M.D., Liu, X.Q., Cao, B.Q., Xu, Y.: AWSR: active web service recommendation based on usage history. In: IEEE 19th International Conference on Web Services, pp. 186–193 (2012)
13. Hu, Y., Peng, Q.M., Hu, X.H.: A personalized web service recommendation method based on latent semantic probabilistic model. J. Comput. Res. Develop. **51**(8), 1781–1793 (2014)
14. Pan, W.F., Li, B., Shao, B., He, P.: Service classification and recommendation based on software networks. Chin. J. Comput. **34**(12), 2355–2369 (2011)
15. Lee, J.S., Ko, I.Y.: Service recommendation for user groups in internet of things environments using member organization-based group similarity measures. In: IEEE International Conference on Web Services (ICWS), pp. 276–283 (2016)
16. Yu, Q.: CloudRec: a framework for personalized service recommendation in the cloud. Knowl. Inf. Syst. **43**(2), 417–443 (2015)
17. Kumara, B.T.G.S., Paik, I., Siriweera, T.H.A.S., Koswatte, K.R.C.: Cluster-based web service recommendation. In: IEEE International Conference on Services Computing (SCC), pp. 348–355 (2016)
18. Cao, B.Q., Liu, J.X., Tang, M.D., Zheng, Z.B., Wang, G.R.: Mashup service recommendation based on usage history and service network. Int. J. Web Serv. Res. **10**(4), 82–101 (2013)
19. Meng, S.M., Dou, W.C., Zhang, X.Y., Chen, J.J.: KASR: a keyword-aware service recommendation method on mapreduce for big data applications. IEEE Trans. Parallel Distrib. Syst. **25**(12), 3221–3231 (2014)
20. Liu, J.X., He, K.Q., Wang, J., Yu, D.H., Feng, Z.W., Ning, D.: An approach of RGPS-guided on-demand service organization and recommendation. Chin. J. Comput. **36**(2), 238–251 (2013)
21. Liu, J.X., Wang, J., He, K.Q., Liu, F., Li, X.X.: Service organization and recommendation using multi-granularity approach. Knowl. Based Syst. **73**, 181–198 (2015)
22. Wu, J., Liang, Q.H., Jian, H.Y.: Bayesian network based services recommendation. In: IEEE Asia-Pacific Services Computing Conference, pp. 313–318 (2009)
23. Wu, J., Chen, L., Jian, H.Y., Wu, Z.H.: Composite service recommendation based on bayes theorem. Int. J. Web Serv. Res. **9**(2), 69–93 (2012)
24. Murphy, K.P.: The bayes net toolbox for matlab (2001)