Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/080649

Breadth First Search Sequence based Method for Efficient Process Retrieval

Ye Yanming[1,2,*], *Yin Yuyu*[3], *Xu Yueshen*[1], *Cao Bin*[1] *and Yin Jianwei*[1]

[1] College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China
[2] College of Information Engineering, Hangzhou Dianzi University, Hangzhou 310018, China
[3] College of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China

Received: 19 Nov. 2013, Revised: 17 Feb. 2014, Accepted: 18 Feb. 2014
Published online: 1 Nov. 2014

Abstract: With increasing improvement of the business process management (BPM) technology, large-scale business process repositories have been adapted widely. However, due to the explosion of the number of business processes, a large part of enterprises are confronting with the challenge on effective management of those massive processes. Usually, each business process is modeled as a process graph, and therefore most existing approaches are based on graph mining algorithms. This paper puts forward a new method, which first utilizes the breadth first search (BFS) algorithm to label the process model, and then calculates the similarity based on the matching distance. The experimental results show that our method is efficient enough for practical use, especially suitable for fuzzy retrieval.

Keywords: process retrieval, breadth first search, BFS code, process matching

1 Introduction

As a widely employed approach in enterprises to regulate business logic and handle business processes, workflow techniques gain increasing development with continuous enterprises information construction. There are various business processes in a company, which cooperate with each other to accelerate the work efficiency in an organization. In the meantime, most companies have to modify some processes to keep pace with the frequent requirement changes. Therefore, a company usually has a large number of candidate business processes. In order to manage these business processes efficiently, many enterprises have built their own business process model repositories, and regard them as important knowledge bases in business process management within organizations. There are often thousands of process models in a single repository [1,2], and enterprises tend to accumulate large numbers of process models along with time. For example, Suncorp, one of the largest Australian insurers, maintains a repository containing more than 6000 process models [3], and Chinese Railway Corporation holds more than 200000 models [4].

It is an intractable challenge to manage such enormous business process model repositories. In practice, it is a common way to retrieve existing process models to reuse the whole process or process fragments, when a new process model is created. Therefore, the ability to retrieve efficiently from a business process model repository is critical for enterprises. There is another technology called process recommendation that uses similar basic definitions. Generally, the process retrieval may be the preceding step when process recommendation is conducted, but there is no inclusion relation between the two techniques. The main differences of the two technologies are explicated as follows.

1. The granularity of conduct is different. Process recommendation requires precise retrieval, and a certain process along with its process fragments may be the intermediate result (in most proposed methods, the problem is to find all the subgraphs of the corresponding process graph). While the common process retrieval only needs to return the whole process or some certain process fragments.
2. Process retrieval is the basic technique in a process repository. It may be useless when a process repository contains numerous process models, so that we cannot find the certain process effectively that

* Corresponding author e-mail: yeyanming@126.com

matches the search condition. However, process recommendation is not the requisite technology for a process repository except for intelligent modeling purpose.

3. In process recommendation and process retrieval techniques, their core process sets are different. Process recommendation technologies use candidate process sets that are a subset of the process repository and all frequent subgraphs of the candidate processes as the basic analysis data. While process retrieval technologies generally use all processes in a repository as the compared data.
4. The usage intentions are different. Process recommendation is mainly used in intelligent modeling, while process retrieval is mainly used in maintenance of the process repository.

Next, the research background will be introduced for deeper comprehension of the paper.

A process model can be depicted with a graphical model, which describes the way that a certain process is composed. Usually, a process consists of different tasks, in which resources are involved in carrying out these tasks, and objects are being manipulated [5]. Therefore, the core task of most of the existing process retrieval methods is to find out all process models in a repository that contains the given process fragment as a subgraph. For example, a versatile graph matching algorithm based on fix-point computation was proposed in [6], which was announced to be usable across various scenarios. Paper [7] focused on the application of graph matching algorithms to the similarity search problem and studied four kinds of graph matching algorithms. Paper [8] proposed a DFS Code-SED method for process retrieval that adopted depth first search (DFS) code to label the process model and calculated DFS codes based on Levenshtein distance to get the similarity values, which improved the computation efficiency of measuring the similarity between two graphs. Nevertheless, to find all subgraph isomorphisms is a NP-complete problem [9]. Different approaches have been proposed to solve this problem, for instance, paper [10] proposed a two-stage approach that reduced the number of models that needed to be checked for subgraph isomorphism.

However, all the methods are either complex or inefficient, and most of them are not suitable for process retrieval with loop structures. This paper presents a BFS sequence based process retrieval method. Firstly, we transforms each process in the process repository into the corresponding BFS sequences, based on the matching distance between the targeted process and reference process in the repository. Then, we calculate the largest process matching distance as the matching degree. Finally, this approach will output the retrieval results that reach or exceed the predefined matching degree threshold. Our approach does not need to handle subgraph isomorphism problem, which greatly reduces the complexity. Though the process isomorphism is still needed to be determined for filtering the process repository, it is not a NP-complete problem. Especially, our approach is much appropriate for fuzzy retrieval, in all cases that the processes contain loop structures or not, the problem of which has not been wholly solved in other methods.

The presented paper is organized as follows. After the above introduction, Section 2 gives some definitions with related basic instructions, and at the end of this section, highlights how to calculate process matching degree. In Section 3, we discuss the implementation of BFS based process retrieval algorithm. Furthermore, in Section 4, the method is evaluated through sufficient experiments. Finally, Section 5 discusses the research perspective.

2 Preliminary

Business process is a series of activities that are performed by different performers to specific target respectively, and the order of activities represents collaboration of these performers. In workflow systems, the process reflects the actual business process, and the activity node represents the business operation in enterprises. Information or operations will flow or conduct in turn according to the nodes sequence. Therefore, the business process model can be abstracted as a directed graph, in which the nodes represent activities, with a corresponding label (implicating activity type, content, and the serial number etc.) and the arcs indicate nodes orders.

2.1 Business Process Graph

There are many different formalized definitions for business process, which are used in different occasions to describe the different features of the real business process. In these definitions, activities and arcs are widely used as standard terms. For simplicity, this paper gives the definition as follows.

Definition 1(Business Process Graph): Let T be the activity type, and a business process is a sextet $P = (A, R, f, s, e)$ where:

1. A is the finite set of activities, which is represented by nodes in a process graph.
2. $R \in A \times A$ is the relationship between activities, which is represented by arcs or edges in a process graph.
3. $f : A \to T$ is the activity type function and $T = \{and\,join - and\,split, and\,join - or\,split, or\,join - and\,split, or\,join - or\,split\}$.
4. $s \in A$ is the start activity.
5. $e \in A$ is the end activity.

Activity $x \in A$ is the input of activity $y \in A$ if and only if there exists a directed arc connecting x with y (that is, $(x, y) \in R$). Node $x \in A$ is the output of node $y \in A$ if and only if there exists a directed arc connecting y with x (that is, $(y,x) \in R$). The activity that has no input node is called *start* activity and the activity that has no output activity is called *end* activity. In a workflow repository, any business process can be modeled as a process graph containing only one *start* activity s and one *end* activity e. So the business process in Definition 1 can be simplified as $P = (A,R,f)$.

Definition 2 (Business Process Isomorphism): Let $P = (A,R,f)$ and $P' = (A',R',f')$ be two business processes. The business isomorphism between P and P' (denoted as $P \cong P'$) is a mapping $g : A \to A'$ such that:

1. $\forall u \in A, (f(u) = f'(g(u)))$
2. $\forall u,v \in A, ((u,v) \in R \Rightarrow (g(u),g(v)) \in R')$

The presented method in this paper only adopts the entire business process as the process pattern, and does not need to handle subgraph isomorphism. However, for efficiency, before a new process is created or added into the process repository, the method still needs to determine whether the process is isomorphism with one of the processes in the repository. Canonical label is widely used to solve the graph isomorphism problem. The canonical label for a graph (denoted as cl(G)) is a unique code which is a sequence of bytes, characters or numbers. It is irrelative with the order of nodes and arcs of the graph and fully depends on the graph topology. If the canonical labels of two graphs are the same , then these graphs are isomorphic to each other. This paper adopts BFS (Breadth first search) standard sequence to construct canonical labels. Related definitions of BFS standard sequence will be discussed in the following.

2.2 Process Canonical Label

DFS (Depth First Search) and BFS (Breadth First Search) are widely used in graph mining. Compared with BFS, DFS consumes less memory, but runs slower due to the stack utilization. On the other hand, BFS consumes more memory, but runs faster than DFS. The fact in construc -tion practice of process resource library and process patterns shows that the vast majority of business processes are of simple structure and less nodes. In the selected process library used in this paper, there are 86% processes including less than 20 nodes, and nearly 71% processes have less than 10 nodes. Therefore, the problem of memory usage is not a matter. On the contrary, because the number of processes may be huge, the time performance is more important. Therefore, this paper uses BFS to realize the canonical label of the process graph.

Breadth first search by different nodes orders of the same hierarchy may lead to different BFS sequences. Therefore, the paper presents standard BFS sequence to ensure that the BFS sequence of the same process is unique. And we can reconstruct unique process with the standard BFS sequence. Next, we give some definitions on the BFS sequence.

In Definition 1, the activity and its type are defined separately and the type is the output of function f with certain activity input. For simplicity, we can define an activity label mapping function $L : (A, f(A)) \to N$ that combines the activity A and its type $f(A)$ into a unique activity label N. Then the business process in Definition 1 can be simplified as $P = (N,R)$. Next, we give some definitions on BFS sequence.

Definition 3 (BFS Sequence) : The breadth first traversal order on a process P is a linear order. As Definition 1 shows, the directed arc of P can be labeled by ordered activities. Therefore, the BFS sequence of P can be represented as follows:

$$\{s\},\{sn_i : sn_i \in R\},\{n_in_j : n_in_j \in R\},\ldots,\{n_te : n_te \in R\},\{e\}$$

Then the BFS sequence of P is:

$$BFSsequence(P) = s\sharp\{sn_i\}\sharp\{n_in_j\}\sharp\ldots\{n_te\}\sharp\{e\}$$

Where, the symbol $\sharp$ is the separator that divides the different traversal hierarchies. For example, as shown in

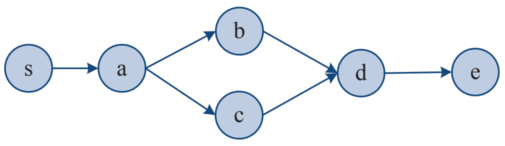


Fig. 1: Process Sample P

Fig. 1, the BFS sequence of the process sample P is

$$BFSsequence(P) = s\sharp sa\sharp ab,\ ac\sharp bd,\ cd\sharp de\sharp e \qquad (1)$$

Definition 4 (Standard BFS Sequence): A BFS sequence of process P is called standard BFS sequence if the labels of the same hierarchies are by lexicographic order, which is denoted as $BFSsequence(P)$.

Except for Eq. (1), the BFS sequence of process shown in Fig. 1 can also be $s\sharp sa\sharp ac,\ ab\sharp cd,\ bd\sharp de\sharp e$ or others. But only Eq. (1) follows the Definition 4, so only Eq. (1) can be called standard BFS sequence of the process.

Generally, if there are continuous repeated fragments in BFS sequence of a process, the process may contain loop structures. However, as shown in Fig. 2(a)(b), the fragments ab, bc, ca in the given processes are continuously repeated in both BFS sequences and the Fig. 2(b) apparently does not have loop structures. Therefore, we can not determine whether a process has loop structures or not only based on the existence of continuous repeated fragments of its BFS sequence. It is easy to see that for a process with N activities and no loops, its biggest BFS sequence hierarchy is not more than $N+1$. Therefore, if the number of BFS sequence hierarchies is more than the number of nodes in a process, the process will certainly contain loop structures.

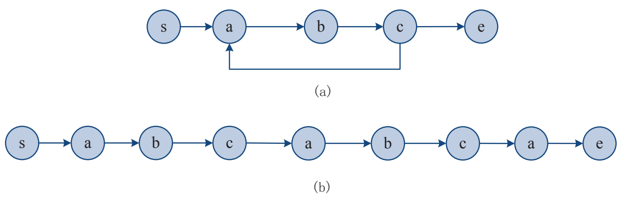


Fig. 2: Process samples with similar BFS sequence

Definition 5 (Extensional Standard BFS Sequence): The extensional standard BFS sequence of the process that contains N nodes has at most $N+2$ hierarchies, and the first $N+1$ hierarchies are the same as standard BFS sequence and the $(N+2)^{th}$ hierarchy consists of all nodes hierarchies numbers that the arcs in $(N+1)^{th}$ hierarchy may be connected to. The extensional standard BFS sequence is denoted as $EBFSsequence(P)$.

For example, the extensional standard BFS sequence of process that is shown in Fig. 2(a) is:

$$s\sharp sa\sharp ab\sharp bc\sharp ca,\ ce\sharp ab, e\sharp 4$$

The set expression of the extensional BFS standard sequence is $EBFSSsequence(P) = p_1, p_2, \ldots p_m$, where p_i is the arcs set of the i_{th} hierarchy and can be denoted as $\{p_{i1}, p_{i2}\ldots\}$. For example, the set expression of the extensional BFS standard sequence of process that is shown in Fig.2(a) is:

$$\begin{aligned} S\_EBFSSsequence(P) =& \{p_1, p_2, p_3, p_4, p_5, p_6, p_7\} \\ =& \{s, \{sa\}, \{ab\}, \{bc\}, \{ca, ce\}, \\ & \{ab, e\}, \{4\}\} \end{aligned}$$

Definition 6 (BFS Determination of Process Isomorphism): Given process P and process P', if there is isomorphism between P and P', their extensional BFS standard sequences of P and P' must be the same. That is,

$$\begin{aligned} P \cong P' &\Leftrightarrow EBFSSsequence(P) = EBFSSsequence(P') \\ &\Leftrightarrow S\_EBFSSequence(P) = S\_EBFSSsequence(P') \end{aligned}$$

Using extensional standard BFS sequence to build process repository every process can be uniquely identified and whether a new process can be added into repository depends on whether there is not isomorphism between it and any other processes of the repository.

2.3 Process Matching Degree

As mentioned above, extensional standard BFS sequence can help us build process repository. This part mainly discusses how to calculate the process matching degree, which is the basis of process retrieval.

Definition 7 (Process Matching Matrix) : Let $P = (N,R)$ and $Q = (N',R')$ be two process graphs. $S\_EBFSSsequence(P) = \{p_1, p_2, \ldots, p_M\}$ and $S\_EBFSSsequence(Q) = \{q_1, q_2, \ldots, q_M\}$ are their extensional standard BFS sequences respectively. The process matching matrix can be expressed as:

$$Mat(P,Q) = \begin{pmatrix} \psi(p_M, q_1) & \ldots & \psi(p_M, q_N) \\ \ldots & \ldots & \ldots \\ \psi(p_1, q_1) & \ldots & \psi(p_1, q_N) \end{pmatrix} \quad (2)$$

where $\psi(p_i, q_j)$ is the comparison function which is defined as:

$$\psi(p_i, q_j) = \frac{SED(p_i, q_j) - |length(p_i) - length(q_j)|}{min(length(p_i), length(q_j))} \quad (3)$$

where, $SED(x,y)$ is the string edit distance of x and y , which is the minimum number of insertions, deletions and substitutions to transform x into y. $Length(x)$ is the number of characters that string x contains.

Note that it has been mentioned in Sect. 2.1 that any business process can be modeled as a process graph containing only one *start* node s and only one *end* node *e*. The process matching path can be defined as follows.

Definition 8 (Process Matching Path): For process P and Q in Definition 7, let $Mat(P,Q)$ be the process matching matrix. A process matching path W, is a contiguous set of matrix elements that define a mapping between P and Q. The v^{th} element of W is defined as $w_v = (i,j)_v$, so we get:

$$W = w_1, w_2, , w_V, where\ max(M,N) \le V < M+N-1 \quad (4)$$

The process matching path is typically subject to several constraints.

Boundary conditions : $w_1 = (1,1)$ and $w_V = (M,N)$, this requires the path to start and end in diagonally opposite corner cells of the matrix.

Continuity: Given $w_v = (a,b)$ then $w_{v-1} = (a',b')$ where $a - a' \le 1$ and $b - b' \le 1$. This restricts the allowable steps in the path to adjacent cells (including diagonally adjacent cells).

Monotonicity : Given $w_v = (a,b)$ then $w_{v-1} = (a',b')$ where $a - a' \ge 0$ and $b - b' \ge 0$. This forces the points in W to be monotonically spaced in turn.

There are various matching paths that satisfy the above conditions, but we are only interested in the path that minimizes the distance, and the minimized distance is also called process matching distance. The path distance can be gained using dynamic programming to evaluate the following recurrence which defines the cumulative distance $d(i,j)$ as the value of current cell $(\psi(i,j))$ and the minimum of the cumulative distances of the adjacent elements:

$$d(i,j) = \psi(i,j) + min\{d(i-1,j-1), d(i-1,j), d(i,j-1)\} \quad (5)$$

For simplicity, the process matching path is referred to the path with minimized distance in the latter.

Definition 9 (Process Matching Degree) : For process P and Q in Definition 7 , let $Mat(P,Q)$ be process matching matrix and $W = w_1, w_2, , w_V$ is the matching path, then the process matching degree can be calculated by the Eq. (4):

$$MatchDegree(P,Q) = 1 - \frac{d(M,N)}{max(M,N)} \tag{6}$$

where M and N are the BFS sequence hierarchy of process P and Q, and $d(M,N)$ is the cumulative distance of cell $Mat[M,N]$ in the top right corner of the process matrix.

When a querying process or process fragment is retrieved, the matching degree between the querying one and each one in the repository will be calculated and all the processes that make the matching degree larger than the given threshold will be returned as results. Next, the implementation of the method will be discussed in detail.

3 Implementation

Let θ be the threshold of process matching, the BFS sequence based process retrieval method can be divided into four modules as shown in Fig. 3.

Step 1. This step transforms the process into BFS sequence form and abandons the isomorphism process. When a new process is being added into the standard process repository, it must be transformed into BFS sequence form if it is not in this form and then process isomorphism will be determined between the new process and each one in repository. Only if there is no any isomorphism, the new process can be added into the repository.

Step 2. In this step, a user can use a graphical interface to input the parameter θ that stands for the precision demand that the user expects and the use the transforming tool to transform the query process into BFS sequence.

Step 3. In this step, the standard process repository will be reduced to form the process candidate set depending whether the process contains the nodes that the input process does.

Step 4. This step is the core part that calculates matching degree between querying process and each one in the repository and output those that meet the user's precision demand.

3.1 Build the Process Candidate Set

Building process candidate set is actually to cut the standard process repository. Obviously, a process that has no intersection or rarely intersects intersected with the querying process on the nodes can not match the querying process well. Therefore, we can just remove the processes whose nodes sets have less intersection with that of querying process. Next, we give the pseudo code (shown in Algorithm 1).

Algorithm 1 Pseudo code of building process candidate set

```
Input:
    Process: p, Standard process repository: p_rep, Parameter:$\theta$
Output:
    Candidate process set: CPS
 1: Initialize candidate process set: CPS
 2: $Hp \leftarrow$ get length of BFS sequence of $p$
 3: $Np \leftarrow$ get number of nodes in $p$
 4: NodeSetof P $\leftarrow$ get nodes set of $p$
 5: for each $q$ in $p\_rep$ do
 6:    Hq $\leftarrow$ get length of BFS sequence of $q$
 7:    if $Hq < Hp$ then
 8:       continue
 9:       $Nq \leftarrow$ get number of nodes in $q$
10:    else if $Nq < Np$ then
11:       continue
12:       NodeSetofQ $\leftarrow$ get nodes set of $q$
13:       diffSetofPQ $\leftarrow$ NodeSetofP $-$ NodeSetofQ
14:    else if diffSetofPQ.length $>$ Hp $\times(1-\theta)$ then
15:       continue
16:    else
17:       CPS.add(p)
18:    end if
19: end for
20: return CPS
```

3.2 Matching Degree Calculation

Process retrieval is to get the processes that satisfy the given process matching degree from the process candidate set. Next, we give the pseudo code to show how to calculate the matching degree (shown in Algorithm 2).

Algorithm 2 Pseudo code of building process candidate set

```
Input:
    Process: $p, q$
Output:
    Matching degree: $md$
 1: Hp $\leftarrow$ get length of BFS sequence of $p$
 2: Hq $\leftarrow$ get length of BFS sequence of $p$
 3: $d[\,] \leftarrow$ new $[Hp, Hq]$
 4: for $i = i$ to $Hp$ do
 5:    for $j = i$ to $Hq$ do
 6:       $d[i,j] = \Psi(i,j) + \min\{\Psi(i-1,j), \Psi(i,j-1), \Psi(i-1,j-1)\}$
 7:    end for
 8: end for
 9: $md = 1 - d[Hp,Hq]\,\max(Hp,Hq)$
10: return $md$
```

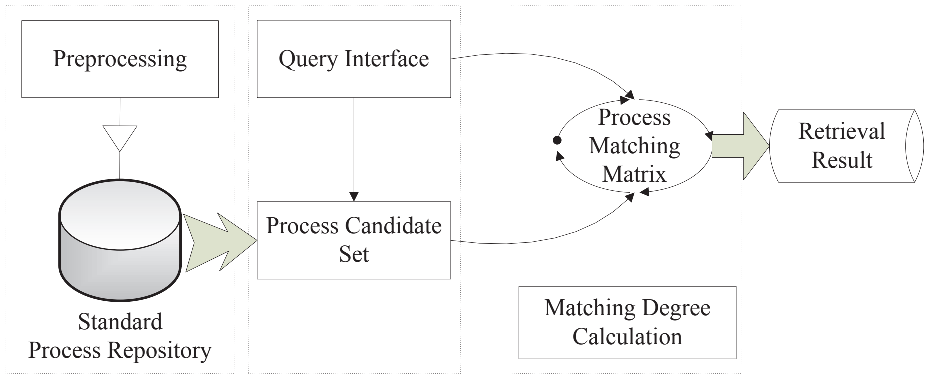


Fig. 3: The process of BFS sequence based process retrieval method

4 Experimental Evaluation

In this section, experiments on the retrieval accuracy and performance of the method and comparison between our BFS sequence based method and DFSCode-SED based method [8] (the latter method was claimed to be more efficient) are carried out on the dataset that is collected from administrative examination and approval processes from the administrative department of a local government (China), which contains 221 business processes involving in totally 52 activities. All experiments are done on a 2.4GHz Intel Core 2 Duo P8600 PC with 4GB main memory, running on Windows 7.

In order to facilitate the discussion, the experiments are grouped into full retrieval and fragment retrieval. The full retrieval is to judge whether the repository contains the querying process and output it if it does. The fragment retrieval is to find and output all processes that contain the querying process fragments to a certain degree. The fragment retrieval is also divided into exact retrieval and fuzzy retrieval for better analysis.

4.1 Full Retrieval Evaluation

In our BFS sequence based method, one to full retrieve only needs to set the parameter $\theta = 1$. We randomly select five processes with simple structure (no branch and no loop) and five processes with complex structure (4 contain branch structure and 1 contains loop structure) from the collected 221 processes as the testing querying processes. These selected processes are divided into two groups and labeled as 'S1, S2, S3, S4, S5' and 'C1, C2, C3, C4, C5' by their nodes number order in each group respectively. The test results are shown in Fig. 4.

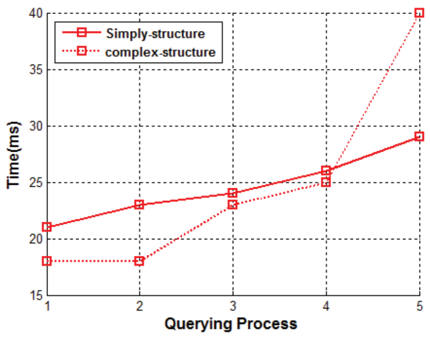
In Fig. 4(a), the retrieval time rises when the nodes number of the processes rises in turn although varied little. And the first four processes of simply structure consumes little more retrieval time than the first processes of complex structure respectively. It may be because the BFS sequence length of a process with simply structure is longer than that of complex structure if their nodes number is the same. Similarly, the retrieval time of process 'C5' suffers from a sudden increase and this may be because the BFS sequence length of process with loop is even longer than that of simply structure with the same nodes number. The comparison experiment between our BFS sequence method and DFSCode-SED is also carried out in this section and the result is shown in Fig. 4(b). As it can been seen in Fig. 4(b), the average retrieval time of the two methods is approximately the same for simple structure . For complex structure process retrieval, the BFS method seems need more time than DFSCode-SED from the figure alone. That is because the DFSCode-SED method can not retrieve the process with loop structure and the output average time only employs the values of the first four processes.

In conclusion, the retrieval time depends on both the nodes number and the structure of the querying process in our BFS sequence based method and the loop structure has more impact on the efficiency. Our method is similar in efficiency with the DFSCode-SED method, but our method supports the querying process containing loop structure, while the DFSCode-SED method can not.

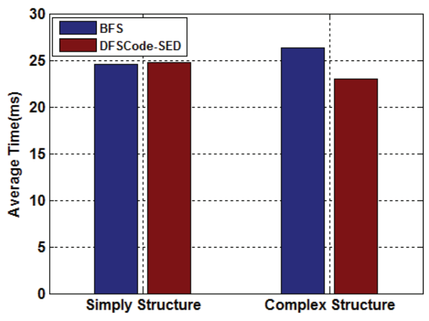
4.2 Fragment Retrieval Evaluation

In fragment retrieval experiment, we cut out 5 process fragments (labeled as F1, F2, F3, F4, F5) from the collected processes and build 5 process fragements (labeled as B1, B2, B3, B4, B5) based on the 52 activities randomly as the testing querying processes to evaluate the exact retrieval and fuzzy retrieval respectively. By setting the parameter =0.9, 0.75, 0.6 respectively, our BFS sequence based method is experimented in turn. Some results are shown in Table 1 and Fig. 5.

As we can see from Table 1 and Fig. 5, when the matching degree threshold θ decreases, the number of retrieved processes and the total retrieval time both rise. On the one hand, the matching degree threshold is used to control the return of retrieval results and is irrelevant to

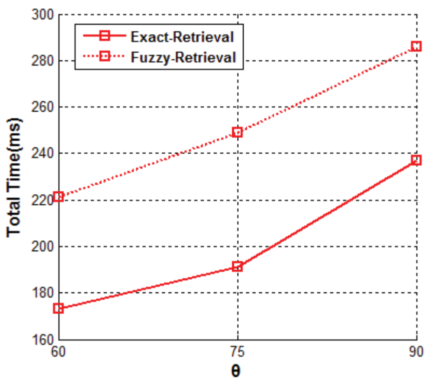


(a) Querying Process



(b) Simple Structure and Complex Structure

Fig. 4: Full Retrieval Evaluation



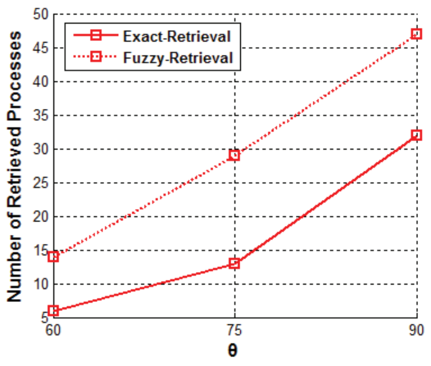


Fig. 5: Fuzzy Retrieval Evaluation

Table 1: Fuzzy Retrieval Result

Parameterθ	Total Time(ms)		The Number of Retrieved Processes	
	Exact Retrieval	Fuzzy Retrieval	Exact Retrieval	Fuzzy Retrieval
0.9	173	221	6	14
0.75	191	249	13	29
0.6	237	286	32	47

the executing time of matching degree calculation algorithm. On the other hand, it has been discussed in full retrieval that the executing time of matching degree calculation is only based on the the nodes number and process structure. However, in the experiment results, the total time changes so much for different threshold θ under the same querying process. In Sect. 3.1, building candidate process set must input the matching degree threshold θ, and obviously, the number of the processes that the candidate set contains will rise by the θ being larger. The size of candidate set increases along with the threshold θ decreasing and thus, although with the same querying process, the retrieval time will increase, as a result that the retrieval efficiency is reduced.

Both the total time and the number of retrieved processes in fuzzy retrieval are more than in exact retrieval and it may be also because the size of candidate process set in fuzzy retrieval is larger than that in exact retrieval.

It is important to note that the BFS sequence based method can not control the switch to exact retrieval or fuzzy retrieval, and in the experiment, it is determined by the querying process fragments.

5 Conclusion and Future Work

Process retrieval is an important technology in management of large business process model repositories and most of the existing process retrieval methods use graphical notations to find matching processes that contain given process fragments as subgraphs. However, the complexity of finding all subgraph isomorphisms are known to be NP-complete, so most methods adopt canonical labels to solve the problem. Unfortunately, these methods are inadequate or limited in some circumstances, such as the direct retrieval of process with branches or loops, fuzzy retrieval and so on. This paper proposes a BFS sequence based method that can solve these problems, especially support fuzzy retrieval and the retrieval of process with loop structure. The most contribution of this paper is that it is the first time to propose BFS sequence to label process, while other methods mainly adopt DFS code to represent process and can not solve the loop problem. Meanwhile, our method can avoid the subgraph isomorphism determination that is known to be NP-complete.

Still, much work has to be done in the future. The BFS sequence automatic construction is important to improve the performance of the method and the matching degree calculation algorithm can be improved to achieve better performance. Furthermore, the application of BFS sequence is another issue for future work.

Acknowledgement

This work is supported by the National Natural Science Foundation of China under Grant (No.61272129), National High-Tech Research Program of China (NO. 2013AA01A213), New-Century Excellent Talents Program by Ministry of Education of China (No. NCET-12-0491), Zhejiang Provincial Natural Science Foundation of China (LR13F020002).

References

[1] Fahland D, Favre C, Jobstmann B, et al. Instantaneous soundness checking of industrial business process models. Business Process Management. Springer Berlin Heidelberg, 278-293 (2009).

[2] Dijkman R M, La Rosa M, Reijers H A. Managing large collections of business process models-current techniques and challenges. Computers in Industry, **63**, 91-97 (2012).

[3] La Rosa M, Dumas M, Uba R, et al. Business process model merging: an approach to business process consolidation. ACM Transactions on Software Engineering and Methodology (TOSEM), **22**, 11 (2013)

[4] Ekanayake C C, La Rosa M, Ter Hofstede A H M, et al. Fragment-based version management for repositories of business process models. On the Move to Meaningful Internet Systems: OTM 2011. Springer Berlin Heidelberg, 20-37 (2011)

[5] Dijkman R, Dumas M, Garcła-Bauelos L. Graph matching algorithms for business process model similarity search. Business Process Management. Springer Berlin Heidelberg, 48-63 (2009).

[6] Melnik S, Garcia-Molina H, Rahm E. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. Proceedings of 18th IEEE International Conference on Data Engineering, 117-128 (2002)

[7] Dijkman R, Dumas M, Garcła-Bauelos L. Graph matching algorithms for business process model similarity search[M]//Business Process Management. Springer Berlin Heidelberg, 48-63 (2009).

[8] Cao B,Yin J W,Chen H X. A Levenshtein Distance Based Method for Process Retrieval. Computer Intergrated Manufacturing Systems, **18**, 1766-1773 (2012).

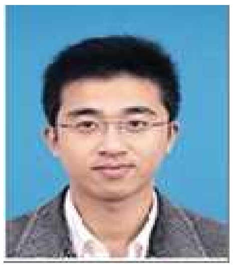
[9] Shasha D, Wang J T L, Giugno R. Algorithmics and applications of tree and graph searching. Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. ACM, 39-52 (2002).

[10] Jin T, Wang J, La Rosa M, et al. Efficient querying of large process model repositories. Computers in Industry, (2012).

Yanming Ye is a Ph.D candidate at the College of Computer Science and Technology, Zhejiang University, Hangzhou, China. His research interests include workflow & business process management, cloud computing and social computing.

Yuyu Yin received the Doctor?s degree in computer science from Zhejiang University, Hangzhou, China, in 2010. He is currently an assistant professor in Hangzhou Dianzi University. His research interests include service computing, cloud computing and middleware techniques.

Yueshen Xu is a Ph.D candidate at the College of Computer Science and Technology, Zhejiang University, Hangzhou, China. His research interests include service computing, recommendation system, business process management and applied machine learning.



Bin Cao received the Doctor?s degree in computer science from Zhejiang University, Hangzhou, China, in 2013. His research interests include workflow management, event processing and spatial database.

Jianwei Yin is currently a professor in the College of Computer Science, Zhejiang University (China). He received his Ph.D. in Computer Science from Zhejiang University in 2001. He is the visiting scholar of Georgia Institute of Technology, America in 2008. His research interests include distributed network middleware, software architecture and information integration.