Efficiently Querying Business Process Models with BeehiveZ

Tao Jin[*#+‡], Jianmin Wang[#+‡], and Lijie Wen[#+‡]

[*]Department of Computer Science and Technology, Tsinghua University, China
[#]School of Software, Tsinghua University, China
[+]Key Laboratory for Information System Security, Ministry of Education, China
[‡]Tsinghua National Laboratory for Information Science and Technology, China
{jint05,wenlj00}@mails.thu.edu.cn,jimwang@tsinghua.edu.cn

Abstract. The technology of business process management is being more widely used, and there are more and more business process models. In this demonstration, we show how to query a large number of models efficiently with BeehiveZ. Four types of queries are all supported in BeehiveZ, i.e. exact query based on structure, similarity query based on structure, exact query based on behavior, similarity query based on behavior. BeehiveZ provides different indexes to facilitate the efficient query processing of different types of queries. To make BeehiveZ more applicable, label similarity is considered.

1 Introduction

With the technology of business process management being more widely used, there are more and more business process models. For example, there are more than 6,000 models in Suncorp, an Australian bank and insurance company, and there are more than 200,000 models in China CNR Corporation Limited. How to query such a large number of models efficiently is challenging. For example, before a designer creates a new process model, if the designer can retrieve the related models and work on them instead of starting from scratch, it would save a lot of time and is less error-prone. Another example is that, before China CNR came into being there were more than 20 subsidiary companies, when these small companies were merged into a bigger one, the similar business processes needed to be integrated, so similarity query is needed.

Since a business process can be represented as topologically different graphs, behavior is their essential characteristic. Therefore, besides the query based on structure the query based on behavior is also needed. In some cases, we cannot find the exact matches according to the user's requirement, but we can find some sufficiently similar ones. Therefore, apart from the exact query the similarity query is also needed. Based on these two orthogonal dimensions (i.e. (structure, behavior) and (exact, similar)), we can classify the queries into four types: (1) exact query based on structure, (2) similarity query based on structure, (3) exact query based on behavior, (4) similarity query based on behavior.

There have already been several research prototypes on business process model query. BPMN-Q[1] is a graph-based query language, which extends the notations of BPMN with some new elements (e.g. variable node). WISE[2] is a workflow information search engine, where workflow models are represented hierarchical. It takes keywords from users as input and returns synthesized workflow hierarchies containing the matched keywords. The results contain only minimal views of relevant workflow hierarchies. The distinguishing feature of VisTrails[3] is its capability to leverage provenance information of workflows. It allows users to query workflows by example and to refine workflows by analogies. BP-QL[4] is based on an abstraction of the BPEL standard for distributed environment and supports query by example. In [5], feature-based similarity estimation is used to improve the efficiency of similarity search. However, all the above work only deals with the queries based on structure, while all the four types of queries are supported in BeehiveZ. Moreover, since the number of models is large, query efficiency is very important, BeehiveZ provides different indexes to speed up the query processing. All these indexes are inverted indexes, which set up the mapping between the indexed items and the models. The indexed items are different according to different types of queries. These indexes can be used as filters to discard many models that are impossible to be resulting models, so that the query efficiency can be improved. Our contributions can be summarized as follows.

- All four types of process model queries are supported in BeehiveZ, and indexes are used to improve query efficiency, as presented in Section 3.
- Since different business analysts would use different labels to describe identical business activities, label similarity is considered in BeehiveZ so that two tasks having the label similarity not less than a specified threshold are regarded as identical, as presented in Section 2.

2 Implementation of BeehiveZ

The BeehiveZ system is implemented in Java, and it can be found at `http://code.google.com/p/beehivez/`. There are many functions in BeehiveZ system. Here we only focus on the query function of BeehiveZ. All the models are stored as `Text` in RDBMS(e.g. MySQL). There are many different notations to capture business processes, to deal with the models with different formats in a uniform way, we assume that all the models in the repository are represented as or transformed to Petri nets. All the indexes used in this paper are managed by Lucene[1] or stored in B+ tree (only the path indexes are stored in B+ trees). We use the source code from ProM[6] to represent and display Petri nets.

Since different analysts would use different labels for identical business activities, label similarity is considered. Two tasks having the label similarity not less than a specific threshold are regarded as identical. Whether to enable label similarity and the label similarity threshold both can be configured by users

[1] http://lucene.apache.org/

during query time. To process queries efficiently when label similarity is considered, the retrieval of similar labels should be efficient, so a label index is used in BeehiveZ. The label index sets up the mapping between words and labels where the investigated word appears.

Let $W(l)$ be the number of words can be extracted from the label string l. Let $SCW(l_1, l_2)$ be the number of words in the label l_1 whose synonymous can be found in the words of the label l_2. In BeehiveZ, the similarity between two labels is calculated using Equation 1, which is similar to Dices Coefficient[7] except that the synonymous are considered now.

$$labelSim(l_1, l_2) = \frac{2 \times SCW(l_1, l_2)}{W(l_1) + W(l_2)}. \quad (1)$$

Note that before this equation is used, all the words must be extracted from the labels, and converted into lower case. Stop words (e.g. a, the) must be removed and the remaining words are replaced with their stem. Of course, we can replace the Equation 1 with other term based similarity measures. To retrieve synonyms quickly, we mapped WordNet[2] synonyms into memory, which consumed approximately 10MB. During the query processing, BeehiveZ will expand queries with the synonymous labels existing in the repository with the help of the label index. More details of how label index is constructed and used can be found in [8].

3 Querying with BeehiveZ

The four types of business process model queries are all supported in BeehiveZ. We will discuss them seperately. Fig. 1 shows the screenshot of querying business process models with BeehiveZ. A model example is input. The user can choose different indexes and configure model similarity threshold for similarity queries. The results are shown in a list and they can be viewed in a viewer.

Exact query based on structure Before a designer creates a new business process model, s/he can draw a small part and then use it as a query to retrieve the models containing the given query condition model as a subgraph. Since subgraph isomorphism algorithm is NP-complete, we cannot scan all the models in the repository sequentially. Path indexes (e.g. *LengthOnePathIndex*) can be used in BeehiveZ, and they can be used as a filter to discard many models that are impossible to be the resulting models and obtain a set of candidate models whose size is always much smaller than the size of the repository. More details of our path indexes can be found in [9]. Since we only need to use subgraph isomorphism algorithm to check whether the candidate models contain the given query condition model as a subgraph, the query efficiency can be improved greatly. For example, if we use the model in Fig. 3(a) as a query on a repository containing the models in Fig. 2, we will get *pn*3 in Fig. 2 as the resulting model.

[2] http://wordnet.princeton.edu/

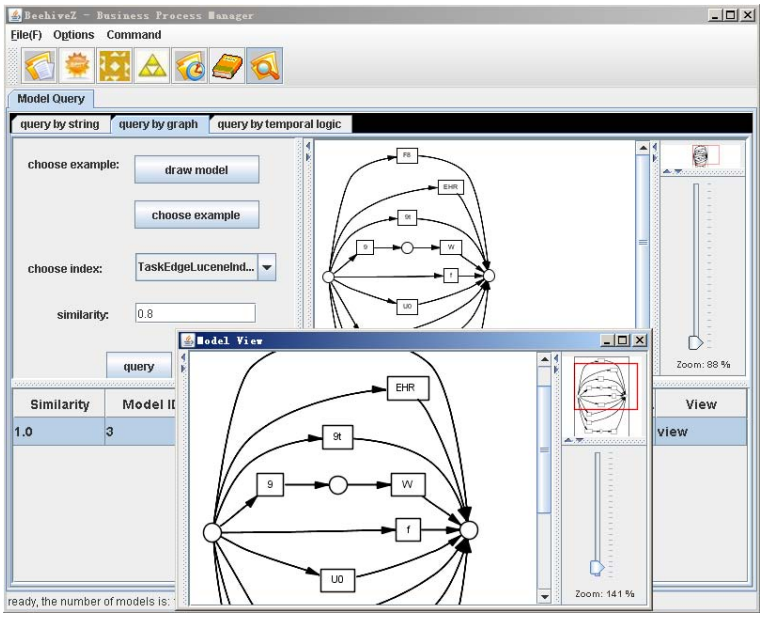

Fig. 1. The screenshot of querying with BeehiveZ

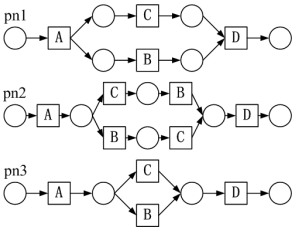

Fig. 2. Business process models represented as Petri nets

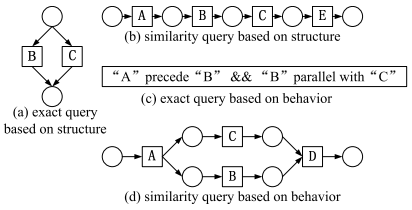

Fig. 3. Query examples

Similarity query based on structure In some cases, we cannot find a model through the exact query based on structure, but if we relax the query we can find sufficiently similar ones. In other words, we can get some models containing a subgraph sufficiently similar to the given query condition model. There are many methods to compute the similarity between graphs based on their structure such as the method based on graph edit distance, maximum common subgraph. In BeehiveZ, we use the maximum common edge subgraph (MCES) based similarity, which is widely used. MCES-based similarity is superior to graph edit based similarity in that no particular edit operations together with their costs need to be defined. Since the computation of MCES-based similarity is NP-hard, we use a filtering-verification framework to reduce the number of times of MCES-based similarity computation. In the filtering stage, an index based on task edges (i.e.

TaskEdgeLuceneIndex) can be used. For example, if we use the model in Fig. 3(b) as a query on the repository containing the models in Fig. 2, we cannot find a model containing this query as a subgraph, but with the MCES-based similarity threshold configured as 0.6, we can get $pn2$ as the resulting model.

Exact query based on behavior The behavior characteristic is the essential characteristic of business process models. When we query the repository, we often want to obtain some models satisfying the given behavior requirements. For example, find the models in which task "A" can be executed parallelly with "B". To process this type of queries, BeehiveZ provide a language for users to describe the requirements. To compute the ordering relations between tasks efficiently, we use the unfolding technology[10] instead of the methods based on reachability graph so that the problem of state-explosion can be solved. To improve the efficiency of query processing, an index based on the ordering relations between tasks (i.e. *TaskRelationIndex*) is used. More details of our language and task relation index can be found in [8]. For example, if we use the string in Fig. 3(c) as a query on the repository containing the models in Fig. 2, we can get $pn1$ as the resulting model.

Similarity query based on behavior When some small companies are merged into a bigger one, the similar business processes should be integrated, so the function of similarity query based on behavior is needed. We measure the similarity between business process models based on their task adjacency relations[11]. The similarity measure based on task adjacency relations has been proved to be a metric. To compute the task adjacency relations efficiently, we use the unfolding technology in BeehiveZ. An index based on task adjacency relations (i.e. *TARLuceneIndex*) is used to facilitate the efficient query processing. For example, if we use the model in Fig. 3(d) as a query with the model similarity threshold configured as 1.0 on the repository containing the models in Fig. 2, we can get $pn1$ and $pn2$ as the resulting models.

4 Demonstration

What will be shown in the demo? We will demonstrate the query processing of four types of queries, i.e. exact query based on structure, similarity query based on structure, exact query based on behavior, similarity query based on behavior. Videos for the demonstration are available at `http://code.google.com/p/beehivez/downloads/list`. Two collections of models are used. One consists of hundreds of thousands of synthesized models which are generated by our generator. The other consists of 591 SAP Reference models. Through the demonstration on the first collection of models, we can see the query performance is good with use of indexes. Through the demonstration on the second collection of models, we can see the query performance is still good with the label similarity considered. Besides the query demonstration, we will also show

the generators used to generate a large number of models for experiments and discuss about the indicators to evaluate different indexes.
Significance in BPM area. This paper shows that four types of business process model queries can be processed by BeehiveZ efficiently. It is helpful for efficient model design, business process model integration and so on.

Acknowledgments. We would like to thank Arthur H.M. ter Hofstede for his suggestion to improve the writing of the first half of this paper. The work is supported by a HGJ project of China (No. 2010ZX01042-002-002-01), the National Basic Research Program (973 Plan) of China (No. 2009CB320700), the National High-Tech Development Program (863 Plan) of China (No. 2008AA042301) and an NSF Project of China (No. 61003099).

References

1. Sherif Sakr and Ahmed Awad. A Framework for Querying Graph-based Business Process Models. In *WWW*, pages 1297–1300, 2010.
2. Qihong Shao, Peng Sun, and Yi Chen. WISE: A Workflow Information Search Engine. In *ICDE*, pages 1491–1494, 2009.
3. Carlos Eduardo Scheidegger, Huy T. Vo, David Koop, Juliana Freire, and Cláudio T. Silva. Querying and Re-Using Workflows with VisTrails. In *SIGMOD Conference*, pages 1251–1254, 2008.
4. Catriel Beeri, Anat Eyal, Simon Kamenkovich, and Tova Milo. Querying Business Processes. In *VLDB*, pages 343–354, 2006.
5. Zhiqiang Yan, Remco M. Dijkman, and Paul Grefen. Fast Business Process Similarity Search with Feature-Based Similarity Estimation. In *OTM Conferences (1)*, pages 60–77, 2010.
6. Boudewijn F. van Dongen, Ana Karla A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and Wil M. P. van der Aalst. The ProM Framework: A New Era in Process Mining Tool Support. In *ICATPN*, pages 444–454, 2005.
7. L.R. Dice. Measures of the Amount of Ecologic Association between Species. *Ecology*, 26(3):297–302, 1945.
8. Tao Jin, Jianmin Wang, and Lijie Wen. Querying Business Process Models Based on Semantics. In *DASFAA*, pages 402–409, 2011.
9. Tao Jin, Jianmin Wang, Nianhua Wu, Marcello La Rosa, and Arthur H. M. ter Hofstede. Efficient and Accurate Retrieval of Business Process Models through Indexing - (Short Paper). In *OTM Conferences (1)*, pages 402–409, 2010.
10. Javier Esparza, Stefan Römer, and Walter Vogler. An Improvement of McMillan's Unfolding Algorithm. *Formal Methods in System Design*, 20(3):285–310, 2002.
11. Haiping Zha, Jianmin Wang, Lijie Wen, Chaokun Wang, and Jiaguang Sun. A Workflow Net Similarity Measure Based on Transition Adjacency Relations. *Computers in Industry*, 61(5):463–471, 2010.