A benchmark framework for evaluating process mining algorithms based on process similarity

Jianmin Wang[1], Haiping Zha[1,2], Nianhua Wu[1], Tengfei He[1], and Lijie Wen[1]

[1] School of Software, Tsinghua University, Beijing 100084, China
jimwang@tsinghua.edu.cn,{chp04,wnh08,htf08,wenlj00}@mails.tsinghua.edu.cn
[2] Institute of Specifications and Standards, Shanghai 200235, China

Abstract. In recent years, many process mining algorithms have been proposed. In spite of the efforts of researchers, a standard benchmark to evaluate and compare various process mining algorithms is still lacking. In this paper, we propose a benchmark framework for evaluating process mining algorithms by measuring the similarity between the original process and the mined model. We use the original model as the prior knowledge and use the average of such similarities as the single indicator of the discovery capability of the process mining algorithm under test. The testing procedure in our benchmark framework is fully automatic, and the test result is simple. Moreover, the original models can be selected according to the evaluation purpose. We have conducted intensive experiments on evaluating the typical process mining algorithms using our benchmark framework, which shows that our benchmark framework is valid and useful.

Key words: benchmark framework, process mining, process similarity, event log

1 Introduction

Process mining aims to discover process models from event logs [25]. Up to now, a lot of process mining algorithms have been proposed [18, 24, 17, 7, 19]. At the same time, researchers have been aware of the importance of the evaluation of process mining algorithms. They proposed their own criterions for evaluating their process mining algorithms, such as *completeness*, *irredundancy*, and *minimality* [21], or *fitness*, *structural appropriateness*, and *behavioral appropriateness* [2, 3]. However, a common benchmark to evaluate and compare different algorithms is still lacking.

The pioneering efforts towards a common evaluation framework for process mining algorithms were done by Rozinat et al. recently [1, 4]. The proposal focuses on the compliance between the mined process model and the event logs. The evaluation dimensions include *fitness*, *precision*, *generalization*, and *structure*. Users have to make a tradeoff among the multiple dimensions to get an overall evaluation of the process mining algorithm.

In this paper, we propose a new benchmark framework for evaluating process mining algorithms by measuring the average similarity between the original processes and the mined models. The test dataset of our benchmark framework is a set of process models. Then, it generates artificial event logs for each process. After that, the framework invokes the process mining algorithm under test to rediscover the process model from the event logs. Finally, it calculates the similarity between the original process and the mined one. The procedure will repeat for each process in the test dataset. At the end of the test, *the average similarity* is calculated which serves as the single indicator of the discovery capability of the process mining algorithm. If the average similarity of one process mining algorithm is greater than the average similarity of the other, we can think that the discovery capability of the former is more powerful than the latter on average.

We believe our benchmark framework for evaluating process mining algorithms is useful for the following reasons.

- Simple and clear. It uses a single number (i.e., the average similarity) to indicate the discovery capability of the process mining algorithm. A larger value of the average similarity means more powerful discovery capability of the process mining algorithm, and vice versa.
- Automatic and objective. After given the test dataset (i.e., a set of process models), the benchmark test procedure will be automatically executed until the final result is produced. The test result is objective lies that: firstly, the benchmark test does not need manual intervention and does not depend on human judgments; secondly, the final result is an averaged number of the results on a large quantity dataset.
- Open and configurable. The framework is open and configurable. We define the interfaces for log generation algorithms, process mining algorithms, and process similarity measurement algorithms. Thus, users can substitute their preferred implementations of these interfaces instead of the defaults.

Our main contribution is proposing such a new benchmark framework for evaluating process mining algorithms by measuring process similarity. Using the benchmark framework, the discovery capability of a process mining algorithm is indicated by a single number for the first time. We have conducted intensive experiments to evaluate the typical process mining algorithms implemented as plugins in the ProM framework [10]. These experiments validate the usefulness of our benchmark framework.

The remainder of this paper is organized as follows. Section 2 presents the details of the benchmark framework for evaluating process mining algorithms. Section 3 shows the experiments to evaluate process mining algorithms using the benchmark framework. Section 4 reviews the related work. Section 5 concludes the paper and outlines future work.

2 The benchmark framework

In this section, we present the benchmark framework for evaluating process mining algorithms in details.

2.1 An overview

The central idea of the benchmark framework is using the average similarity between the original processes and the mined processes as the indicator of the discovery capability of the process mining algorithm under test. The mined processes are produced by the process mining algorithm being tested from the event logs, while the event logs are generated from the processes in the test dataset.

Usually, a process mining algorithm cannot guarantee to exactly rediscover all original processes from the event logs. Thus, it is not suitable to evaluate the discovery capability of a process mining algorithm based on whether it can or cannot rediscover a given process. However, the similarity between the original process and the mined process quantifies the degree of conformance between them. Therefore, the average similarity on a large quantity of typical processes reflects the discovery capability of a process mining algorithm. A more powerful process mining algorithm will lead to a larger value of the average similarity, and vice versa.

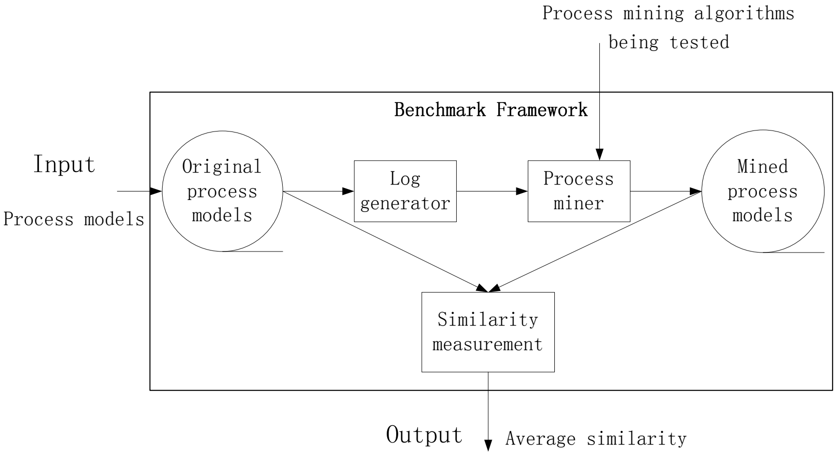

Fig. 1. The benchmark framework architecture.

The architecture of the benchmark framework is shown in Figure 1. *The input* of the benchmark framework is a set of process models which are provided by users. *The output* is the average similarity between the processes and the corresponding mined models. The framework includes three key functional modules, i.e., *the log generator*, *the process miner* which is a container of the process mining algorithms being tested, and *the similarity measurement*.

The function of the log generator is to automatically generate event logs of a process model. The event logs are the input of the process miner which invokes the process mining algorithm by a unified interface. Then, the process mining algorithm which has implemented the interface will rediscover the process model from the event logs. Finally, the similarity measurement module will invoke a similarity measuring algorithm to calculate the similarity between the original process model and the process produced by the process miner. The above procedure will be repeatedly executed for each process model in the test dataset. At the end, the average of these similarities is presented which is the indicator of the discovery capability of the process mining algorithm being tested. The details of each module will be discussed in the following subsections.

2.2 The test dataset

The test dataset of the benchmark framework is a set of process models. The process models can be either artificial models generated for the test purpose, or real-life processes collected from industrial applications. The process models should be typical so that they can represent the application requirements on the process mining algorithms. In order to get an objective evaluation of the tested process mining algorithms, a test dataset usually consists of a large quantity of process models. Although this paper focuses on the benchmark framework, a standard test dataset of process models is essential for a benchmark test.

A test dataset of real-life processes are valuable because they represent real requirements in organizations. In fact, any process mining algorithm does have its own strengths and its own weaknesses. In spite of that, a process mining algorithm is useful when it meets the requirements of the intended application. For example, the α algorithm proves to be able to discover all SWF-nets if complete event logs are presented [24]. Although the α algorithm has limitation on discovering processes beyond SWF-nets, it is still good enough for an organization whose process models are almost structural processes. Therefore, if an organization wants to evaluate which process mining algorithm is suitable to its requirements, the best test dataset must be their own real-life processes.

Sometimes, artificial processes with certain characteristics are necessary for the evaluation purpose, especially in the development phase of a process mining algorithm. Artificial processes have their advantages, such as availability and customization. As a facility of the benchmark framework, we provide a process generator in the package which can automatically generate customized WF-nets in large quantities. With the process generator, users can specify the quantity and the average size of the intended process models. Then, the customized process models will be available in seconds.

The processes used in the benchmark framework are default WF-nets. However, users can either extend the benchmark framework to directly support processes modeled in other languages (e.g., EPC , XPDL), or transform them into WF-nets. In the ProM framework[10], a process model in one modeling language can be transformed into its corresponding model in another modeling language.

2.3 Event logs

Event logs are important to the benchmark framework because they bridges the original processes and the mined models. Event logs are automatically generated by a configurable algorithm for each process in the test dataset. Then, the event logs serve as the input of the process miner. Usually, event logs in the benchmark framework are transparent to the end users.

We have implemented a log generator in the benchmark framework. It is intended to benchmark the discovery capability of the control flow of process mining algorithms. We adopt the meta model and the MXML format for the event logs [9]. The common MXML event logs support the process mining algorithms for different purposes, so that we leave the room for users to extend the benchmark framework to test process mining algorithms for various purposes other than control flow discovery. At the same time, we provide the flexibility for users to configure their own log generation policy. For example, to test the noise toleration of a process mining algorithm, we need another log generation policy in which noise is intentionally added into the event logs.

A challenging problem of artificial event logs is how to meet the requirement of *completeness* as to control flow discovery. Different process mining algorithms have different requirements on completeness of event logs. By definition, complete event logs of a process should record all observed behavior of a process. For example, if the set of firing sequences of a process is finite, the set of event logs is complete if it contains the full firing sequences of the process. However, many processes contain loop constructs, so that their full firing sequences are infinite. Therefore, the behavior of such a process model cannot be completely recorded in a finite set of event logs.

To cope with the above challenge, we take two measures in the implementation of our benchmark framework. Firstly, the event logs of a process is generated based on the fairness principle, i.e., each enabled transition has the same probability to be fired in each step. So that, each possible execution path of the process can be covered with equal probabilities. Secondly, depending on the size of the process model, we require that the quantity of traces in event logs should be large enough. The pseudocode of the default algorithm for event log generation is as follows.

//***Input*: An initially marked Petri net N.

//***Output*: A MXML file f for event logs.

Algorithm 1 *(Algorithm of event log generation)*

```
creatMXMLFile(f);
iCaseNumber=0;
WHILE (iCaseNumber < MAX_CASE_NUMBER)
{
    iCaseNumber++;
    beginCase(iCaseNumber,f);
    iRunStep=0;
    WHILE ((bEnabledTransition(N) == TRUE) AND (iRunStep < MAX_RUN_STEP))
    {
        iRunStep++;
        listEnabledTransitions=getAllEnabledTransitions(N);
//** each enabled transition has the same probability to be fired
```

```
        i = Random(listEnabledTransitions.size());
        fire(listEnabledTransitions.elementAt(i-1))
        writeEvent(iCaseNumber,f);
    }
    endCase(iCaseNumber,f);
}
```

2.4 The process miner

The function of process miner is to rediscover the process model from event logs using the process mining algorithm under test. The input of the process miner is the event logs and the output is the mined process. The key feature of the process miner is that it should be able to integrate different process mining algorithms.

From the perspective of control flow, the output result of a process mining algorithm is a process model. However, the process model may be presented in very different forms. For example in the ProM framework, the result of the α algorithm plugin is a *PetriNetResult*, while the result of the Genetic algorithm plugin is a *HeuristicsNetResult*. Therefore, we need to abstract a unified interface for different process mining algorithms. In the benchmark framework, the similarity measurement is based on Petri nets. So that, it requires the result of process miner be a Petri net. Therefore, we specify a standard interface for implementation of different process mining algorithms is as follows.

```
public interface CtrolflowMiner{
  public PetriNet Mine(LogReader);
}
```

2.5 The similarity measurement

Process similarity measurement plays a key role in the benchmark framework. In our benchmark framework, similarity measure approaches should be behavior oriented.

After the mined process model has been produced by the tested process mining algorithm, the similarity between the mined process and its corresponding original process will be calculated by a similarity measuring approach. The procedure will repeat for each process in the test dataset. Finally, the average of the similarity is presented which serves as the indicator of the discovery capability of the process mining algorithm under test.

We provide the option for users to use their preferred similarity measuring approach. We also define a standard interface for the implementation of the similarity measuring approaches. Similarity measuring approaches which have implemented the interface can be integrated into the benchmark framework. The definition of the interface for similarity measurement is as follows.

```
public interface ProcessSimilarityMeasure{
  public float similarityMeasure(PetriNet, PetriNet);
}
```

Note that, the interface is Petri nets oriented by default. Therefore, a model converter is needed here to transform the processes in other modeling languages to Petri nets.

2.6 Features and limitations

The outstanding feature of our benchmark framework is using the average similarity as the single indicator of the discovery capability of the process mining algorithm being tested. Although the result is straightforward and the procedure is fully automatic, the benchmark framework has its limitations which should be addressed in future work.

Firstly, considering the fact that process mining algorithm have different features, a single evaluation may be not sufficient. For example, let us compare the α algorithm and the heuristic algorithm. The experiments using the benchmark framework suggests that performance of the heuristic algorithm seems to be a bit weaker than the α algorithm (low similarity). However, one important feature of the heuristic algorithm is that it can cope with noise in the logs. Therefore, the benchmark framework should be extended to include some other performance related factors. Moreover, there are optimal parameter settings for some mining algorithms, e.g., the genetic algorithm. The default parameter settings for such algorithms may not be the proper ones.

Secondly, the framework relies on the presence of original process models. For the benchmark purpose, the assumption is reasonable which can greatly simplify the testing and comparison. However, the benchmark result may be not conducive to real environments where event (or transactional) logs often exist, but process models do not.

Thirdly, the challenging problem of the framework is how to meet the requirement of *completeness* of event logs. Currently, we only reach an empirical solution, i.e., n^2 traces for a process of size n.

The trace set volume of a process model is not just depended on the size of the process.For example, when all transitions in a process model are sequential, this model only produce one trace, when all transitions in a process model are parallel, the process model will generate 2^n potential traces, where n is the transition number of the process. Therefore, to judge the completeness of a trace set purely based on the size of a process model seems to be questionable. The number of traces producible by a model is not linear to the number of transitions in the process model.

3 Experiments

This section introduces the experimental evaluation of several typical process mining algorithms using our benchmark framework. The benchmark framework is implemented as a module in the BeehiveZ project [1]. Note that, the implementation of the benchmark framework is closely related to the ProM project [10]. We directly invoke the process mining algorithm plugins and use the models transformation approaches provided in the ProM.

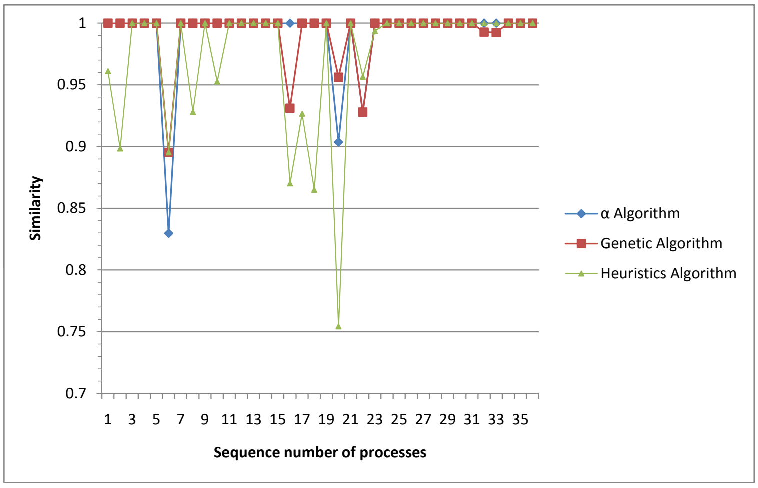

Fig. 2. The evaluation results of different process mining algorithms using real life structural processes.

We first use the benchmark framework to evaluate different process mining algorithms which are available in the ProM. We selected three typical process mining algorithms, i.e., the α algorithm [24], the genetic algorithm [7], and the heuristics algorithm [6]. The default similarity measuring approach is the CF similarity [11].

The evaluation result using 37 typical real-life process models is shown in Figure 2. The average size of these models is about 20, and most of them contain parallel or loop constructions. The average similarity of the α algorithm, the genetic algorithm, and the heuristics algorithm is 0.9905, 0.9915, and 0.9723 respectively. It shows that all process mining algorithms have acceptable capability on rediscovery the processes. However, the genetic algorithm is a little more powerful than the other two on average.

We conduct the above experiments again, but using eight artificial processes, to investigate effect on the evaluation of the quality of processes and event logs. The eight artificial processes include four structural processes, two processes

---

[1] http://sourceforge.net/projects/beehivez/

with short loops (length-one loops and length-two loops) [8], a process with a typical non-free-choice structure [19], and a process whose event logs contain noise. The evaluation results is shown in Figure 3. The average similarity of the α algorithm, the genetic algorithm, and the heuristics algorithm is 0.5637, 0.5748, and 0.5628 respectively. The result of each process mining algorithm is significant lower than the result on structural processes. The result shows the poor performance of these algorithms on discovery the unstructrual processes, and event logs containing noise.

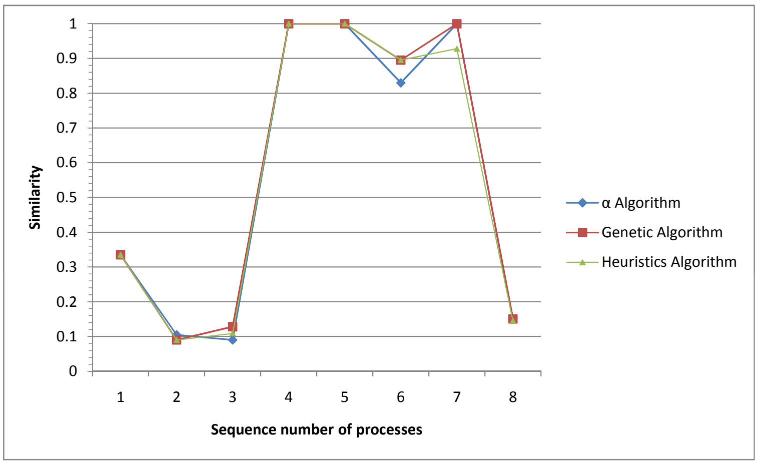

Fig. 3. The evaluation results of different process mining algorithms using artificial unstructural processes, and event logs containing noise.

However, the poor performance of the genetic algorithm is unexpected because it should be able to discover the unstructural processes, as well as to deal with the event logs containing noise. We did find that the genetic algorithm could produce a high fitness (greater than 0.9) during the experiments. The true reason of such a lower similarity is caused by the CF similarity. The genetic algorithm usually introduces a few routing tasks (or invisible tasks) to get a high fitness with the event logs if necessary. However, neither the definition of causal footprints nor the implementation of causal footprints in the ProM has been optimized for such routing tasks. So that, the CF similarity presented a low value between the original process and the mined process with additional routing tasks in spite of a high fitness between them.

4 Related work

Process mining aims to discover exact process models from event logs. In [24], an algorithm, i.e., the α algorithm, was proposed which theoretically constructed the final process model in WF-nets from event logs. The strengths and weaknesses of the α algorithm had been well studied. Therefore, [24] provides a good start point for further research. After that, a lot of different process mining approaches have been proposed [17, 15, 23, 13, 7, 19]. For more information on process mining you can refer to a survey paper [25].

At the same time, different tools for process mining is also presented, such as Little Thumb [5], EMiT [12], Process Miner [14], and InWoLvE [17]. These tools are intended for special process mining algorithms. The ProM framework [10] is a common framework to facilitate process mining implementation. Many process mining algorithms have been implemented as plugins in the ProM framework which are available for public. The implementation of our benchmark framework has taken full advantage of the facility of the ProM framework, such as log reading, process mining, and model transformation.

Most researchers have been aware of the importance of performance evaluation of the algorithms, they proposed the own criterion along with their process mining algorithms, such as completeness, irredundancy, and minimality [21], or fitness, structural appropriateness, and behavioral appropriateness [2, 3]. The pioneering efforts towards a common evaluation framework for process mining have been done [1, 4]. The proposal focuses on the test of compliance between the mined process model and the events logs from four dimensions, i.e., fitness, precision, generalization and structure.

Our work is aligned with the efforts toward a common benchmark for process mining algorithms. The benchmark is based on similarity measurement between process models. Approaches for process similarity measurement have been proposed recently [22, 20, 16] which aim to quantify the difference between business processes. In the benchmark framework, the average similarity between original process models and the mined process models serves as an indicator of the discovery capability of a process mining algorithm.

5 Conclusion

In this paper, we propose a novel benchmark framework for process mining algorithms by measuring the similarity between the original processes and the mined models. The average similarity serves as the single indicator of the discovery capability of the process mining algorithm under test. The advantages of our benchmark framework are as follows: the benchmark result is simple and clear , the benchmark procedure is fully automatic , the components in the framework are substituted easily, the test dataset can be selected as necessary.

We believe our benchmark framework can be used to evaluate process mining algorithms, which will benefit the inventions and the applications of the process mining algorithms. By experiments, we also learn that the test dataset (i.e., the processes) and the similarity measuring approaches used in the benchmark framework are relevant to the evaluation results of process mining algorithms. In the future, we will build a reference dataset of process models for the benchmark purpose and incorporate the performance metric of a process mining algorithm into our benchmark framework.

Acknowledgements

We would like to thank the contributors to the ProM process mining framework.

The work is supported by the National Basic Research Program of China (No. 2009CB320700), the National High-Tech Development Program of China (No. 2008AA042301 and No. 2007AA040607), and the Program for New Century Excellent Talents in University of China.

References

1. A. Rozinat, A.K.A. de Medeiros, A.J.M.M. Weijters, and W.M.P. van der Aalst. The need for a process mining evaluation framework in research and practice. In A. ter Hofstede, B. BBenatallah, and H.Y. Paik, editor, *BPM 2007 Workshops*, volume 4928 of *Lecture Notes in Computer Science*, pages 84–89. Springer-Verlag, Berlin, 2008.
2. A. Rozinat and W.M.P. van der Aalst. Conformance testing: measuring the fit and appropriateness of event logs and process models. In C. Bussler et al., editor, *BPM 2005 Workshops*, volume 3812 of *Lecture Notes in Computer Science*, pages 163–176. Springer-Verlag, Berlin, 2005.
3. A. Rozinat and W.M.P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64–95, 2008.
4. A. Rozinat, A.K. Alves de Medeiros, C.W. Günther, A.J.M.M. Weijters, W.M.P. van der Aalst. Towards an evaluation framework for process mining algorithms. In *BPM Center Report BPM-07-06, BPMcenter.org*, 2007.
5. A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering workflow models from event-based data using little Thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.
6. A.J.M.M. Weijters, W.M.P. van der Aalst, and A.K. Alves de Medeiros. Process mining with heuristicsminer algorithm. In *BETA Working Paper Series, WP 166, Eindhoven University of Technology, Eindhoven*, 2006.
7. A.K. Alves de Medeiros, A.J.M.M. Weijters, and W.M.P. van der Aalst. Genetic process mining: an experimental evaluation. *Data & Knowledge Engineering*, 14:245–304, 2007.
8. A.K. Alves de Medeiros, B.F. van Dongen, W.M.P. van der Aalst, and A.J.M.M. Weijters. Process mining: extending the α-algorithm to mine short loops. BETA Working Paper Series, WP 113, Eindhoven University of Technology, Eindhoven, 2004.
9. B.F. van Dongen and W.M.P. van der Aalst. A meta model for process mining data. In M. Missikoff and A. de Nicola, editors, *Proceedings of the CAiSE'05 Workshops*, volume 2, pages 309–320, 2005.
10. B.F. van Dongen, A.K. Alves de Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, and W.M.P. van der Aalst. The prom framework: a new era in process mining tool support. In G. Ciardo and P. Darondeau, editors, *International Conference on Application and Theory of Petri Nets 2005*, volume 3536 of *Lecture Notes in Computer Science*, pages 444–454. Springer-Verlag, Berlin, 2005.

11. B.F. van Dongen, R.M. Dijkman, and J. Mendling. Measuring similarity between business process models. In *CAiSE 2008*, pages 450–464, 2008.
12. B.F. van Dongen and W.M.P. van der Aalst. EMiT: a process mining tool. In J. Cortadella and W. Reisig, editors, *Inernational Conference on Application and Theory of Petri Nets*, volume 3099 of *Lecture Notes in Computer Science*, pages 454–463. Springer-Verlag, Berlin, 2004.
13. G. Greco, A. Guzzo, L. Pontieri, and D. Saccá. Discovering expressive process models by clustering log traces. *IEEE Transactions on Knowledge and Data Engineering*, 18(8):1010–1027, 2006.
14. G. Schimm. Process miner - a tool for mining process schemes from event-based data. In S. Flesca and G. Ianni, editors, *the 8th European Conference on Artificial Intelligence (JELIA)*, volume 2424 of *Lecture Notes in Artificial Intelligence*, pages 525–528. Springer-Verlag, Berlin, 2002.
15. G. Schimm. Mining exact models of concurrent workflows. *Computers in Industry*, 53(3):265–281, 2004.
16. H. Zha, J. Wang, L. Wen, C. Wang, and J. Sun. A workflow net similarity measure based on transition adjacency relations. *Computers in Industry*, page (to appear), 2010.
17. J. Herbst and D. Karagiannis. Workflow mining with InWoLvE. *Computers in Industry*, 53(3):245–264, 2004.
18. J.E. Cook and A.L. Wolf. Discovering models of software processes from event-based data. *ACM Transactions on Software Engineering and Methodology*, 7(3):215–249, 1998.
19. L. Wen, W.M.P. van der Aalst, J. Wang, and J. Sun. Mining process models with non-free-choice constructs. *Data Mining and Knowledge Discovery*, 15(2):145–180, 2007.
20. M. Dumas, L. García-Bañuelos, and R. Dijkman. Similarity search of business process models. *Bulletin of the IEEE Technical Committee on Data Engineering*, 32(3):25–30, 2009.
21. R. Agrawal, D. Gunopulos, and F. Leymann. Mining process models from workflow logs. In I. Ramos, G. Alonso, and H.J. Schek, editors, *Sixth International Conference on Extending Database Technology*, pages 469–483, 1998.
22. R. Dijkman, M. Dumas, and L. García-Bañuelos. Graph matching algorithms for business process model similarity search. In U. Dayal et al., editor, *BPM 2009*, volume 5701 of *Lecture Notes in Computer Science*, pages 48–63. Springer-Verlag, Berlin, 2009.
23. W. Gaaloul and C. Godart. Mining workflow recovery from event based logs. In W.M.P. van der Aalst et al., editor, *the Third International Conference on Business Process Management (BPM 2005)*, volume 3649 of *Lecture Notes in Computer Science*, pages 169–185. Springer-Verlag, Berlin, 2005.
24. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow mining: discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
25. W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow mining: a survey of issues and approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.