LocalProcessModelDiscovery: Bringing Petri Nets to the Pattern Mining World

Niek Tax[1,2(✉)], Natalia Sidorova[1], Wil M.P. van der Aalst[3], and Reinder Haakma[2]

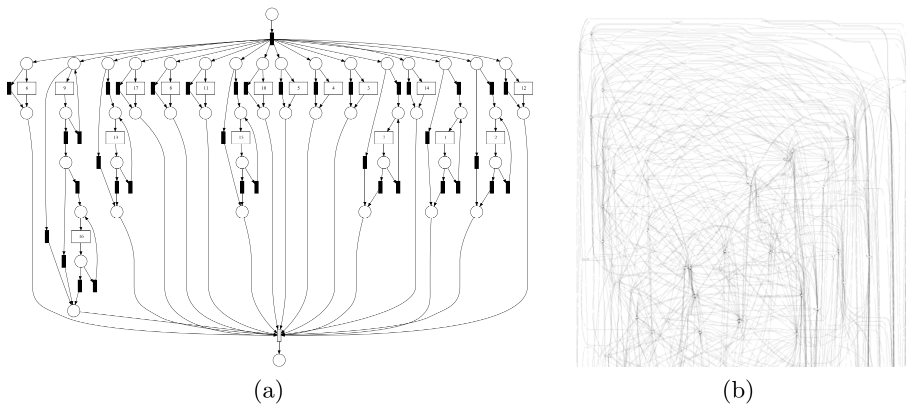
[1] Eindhoven University of Technology, Department of Mathematics and Computer Science, P.O. Box 513, 5600MB Eindhoven, The Netherlands
{n.tax,n.sidorova}@tue.nl
[2] Philips Research, Prof. Holstlaan 4, 5665 AA Eindhoven, The Netherlands
{niek.tax,reinder.haakma}@philips.com
[3] RWTH Aachen, Germany
wvdaalst@pads.rwth-aachen.de

Abstract. This paper introduces the tool *LocalProcessModelDiscovery*, which is available as a package in the process mining toolkit ProM. *LocalProcessModelDiscovery* aims to discover *local process models*, i.e., frequent patterns extracted from event logs, where each frequent pattern is expressed in the form of a Petri net. Local process models can be positioned in-between process discovery and Petri net synthesis on the one hand, and sequential pattern mining on the other hand. Like pattern mining techniques, the *LocalProcessModelDiscovery* tool focuses on the extraction of a set of frequent patterns, in contrast to Petri net synthesis and process discovery techniques that aim to describe all behavior seen in an event log in the form of *a single model*. Like Petri net synthesis and process discovery techniques, the models discovered with *LocalProcessModelDiscovery* can express a diverse set of behavioral constructs. This contrasts sequential pattern mining techniques, which are limited to patterns that describe sequential orderings in the data and are unable to express loops, choices, and concurrency.

Keywords: Petri nets, Frequent Pattern Mining, Process Discovery

1 Introduction

LocalProcessModelDiscovery is a novel tool for the discovery of frequent patterns in the form of Petri nets from event logs. This paper aims to present this Petri-net-based tool and provide some insights into the discovery techniques used. The tool is implemented as a package in the Java-based *process mining* framework ProM [12] and is publicly available at `https://svn.win.tue.nl/repos/prom/Packages/LocalProcessModelDiscovery/` and in the ProM package manager. After installing the *LocalProcessModelDiscovery* package to ProM, the tool can be started by importing an event log in XES [26] format into ProM and then running the ProM plugin *Search for Local Process Models* using this event log as input. The algorithms that we developed for the mining of frequent Petri net patterns from event logs [10, 22–24] form the core of the *LocalProcessModelDiscovery* tool.



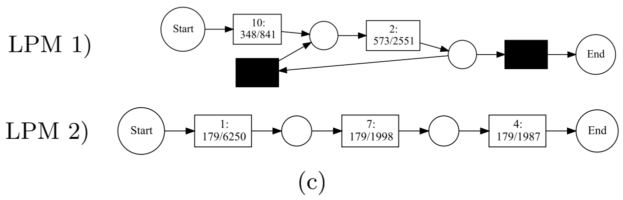


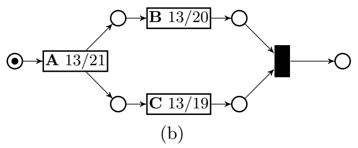
Fig. 1. The Petri net model mined from the MSNBC dataset with *(a)* the Inductive Miner [21], *(b)* the ILP Miner [25], and *(c)* two LPMs mined from the MSNBC dataset. Black transitions correspond to silent transitions.

Mining of Local Process Models (LPMs) can be positioned in-between the research areas of *Petri net synthesis* and *process discovery* on the one hand and *frequent pattern mining* on the other hand. *Frequent pattern mining* [17] techniques focus on extracting local patterns from data. *Sequential pattern mining* [14] techniques are a type of frequent pattern mining that focuses on the extraction of frequent patterns from sequence data. While process discovery [1] and Petri net synthesis techniques aim to discover an *end-to-end process model*, sequential pattern mining techniques aim to extract a *set of patterns* where each pattern describes a subsequence that frequently occurs in the event log. Sequential pattern mining techniques can be used to generate insights from event data that only contain weak relations between the activities, i.e., that have a relatively high degree of randomness. From such event logs, process discovery and Petri net synthesis techniques generate either overgeneralizing models that allow for too much behavior, or generate a 'spaghetti'-model that is accurate in the allowed behavior but is not understandable and often overfitting. Figure 1a gives an example of an overgeneralizing process model, showing the process model discovered with the Inductive Miner [21] from web click data from the MSNBC.com news website[1] (note that most activities can be skipped and/or repeated allowing for any behavior). Figure 1b gives an example of a non-interpretable 'spaghetti'-like process model, discovered from the same dataset with the ILP Miner [25]. The first LPM of Figure 1c, however, shows that activity 10 is generally followed by multiple instances of activity 2.

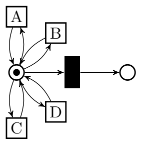
[1] http://kdd.ics.uci.edu/databases/msnbc/msnbc.data.html

Event sequences
⟨A,**A,C,B**,A,**A,C,B**,B,C⟩
⟨C,**A,C,B**,A,A,**A,B,C**,B⟩
⟨A,**A,B**,D,**C**,D,**A,B,C**,B⟩
⟨C,**A,C,B**,B,B,**A**,D,**B,C**⟩
⟨B,**A,B,C**,C⟩
⟨D,**A,C,B**,C,A,**A,C**,A,**B**⟩
⟨D,**A,B,C**,D,C,A,C,**A,B,C**⟩

(a)



(b)



(c)

Sequential patterns	Support
$\langle A, B, C\rangle$	7
$\langle B, A, B\rangle$	7
$\langle A, B, A\rangle$	6
$\langle A, C, A\rangle$	6
$\langle A, C, B\rangle$	6
$\langle B, A, C\rangle$	6
$\langle C, A, C\rangle$	6

(d)

Fig. 2. *(a)* A log L with highlighted instances of the frequent pattern. *(b)* An example local process model that shows some frequent behavior in L. *(c)* The Petri net discovered from L with the Inductive Miner tool [21]. *(d)* The sequential patterns discovered from L with the PrefixSpan algorithm [18] (with minimum support=6).

2 What is Local Process Model Mining?

The process models that can be discovered with process discovery and Petri net synthesis techniques can describe a rich set of process constructs, such as concurrency, inclusive and exclusive choices, loops, and sequential execution. In contrast, the patterns that are discovered with sequential pattern mining techniques are limited to sequential orderings. The mining of *Local Process Models* (LPMs) [24] extends sequential pattern mining techniques to Petri nets, *allowing for the discovery of local patterns of non-sequential behavior, including choices, concurrency, and loops.* Figure 2 illustrates an example local process model on an example event log, and highlights the instances of the LPM in the event log in **blue**. Note that instances of an LPM in the event log do not have to consist of consecutive events, i.e., there can be *gaps* within a pattern instance. The LPM instances that contain gaps are indicated in the event log with underline.

The *LocalProcessModelDiscovery* tool provides an implementation of the local process model mining algorithm presented in [24]. Intuitively, the local process model mining algorithm works by iteratively expanding patterns into larger candidate patterns, for which it calculates the support by calculating an alignment [2] between the pattern and the event log. Patterns that satisfy a minimum *support threshold* that is provided by the user are then expanded further in the next expansion iteration of the algorithm. The local process model mining algorithm returns a ranked list of patterns, where the patterns are ranked according to a weighted average over the following quality criteria:

Support Relates to the number of times that the behavior that is described by the Petri net pattern is found in the event log.

Confidence A pattern has high confidence when a high ratio of the events in the event log of the activities that are described in the pattern belong to instances of the pattern.

Language fit Relates to how much behavior that is allowed by the Petri net pattern is actually observed in the event log at least once. A Petri net that allows for many behavior that was never observed has low language fit.

Determinism Relates to the average number of enabled transitions during replay of the pattern instances on the pattern. A purely sequential model has optimal determinism, while a model that allows for more behavior might have higher support but will have lower determinism.

Coverage Relates to how many events in the event log are described by the pattern.

Related Tools

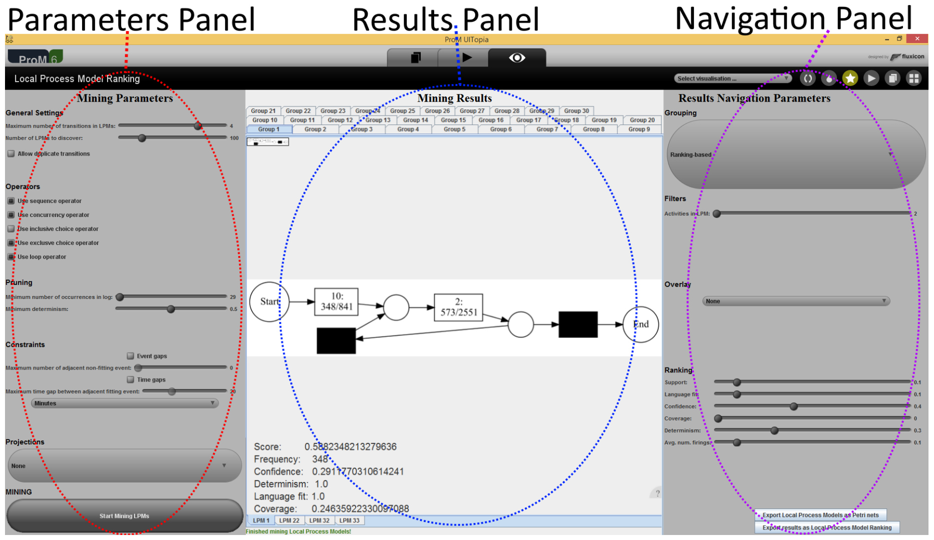
Several tools for Petri net synthesis (see [3] for an overview of synthesis techniques) have been implemented throughout the years, including APT [5], GENET [6], and Petrify [9]. APT [5] allows for the synthesis and analysis of Petri nets and transition systems. GENET [6] allows for the synthesis of a Petri net from automata. Petrify [9] allows for the synthesis of Petri nets and asynchronous circuits. VipTool [4] allows for the synthesis of a Petri net from a finite partial language. Furthermore, ProM [12], APROMORE [20], PMLAB [7], and bupaR [19] are tools that provide implementations of process discovery algorithms. SPMF [13] is a pattern mining tool that provides implementations of a wide variety of frequent pattern mining algorithms, including sequential pattern mining algorithms. Additionally, several frequent pattern mining algorithms have been implemented in the data mining toolkit WEKA [16]. Another related tool is WoMine [8] which provides an implementation of the *w-find* algorithm [15] for mining frequent behavioral patterns from a process model, allowing a user to find frequent subprocesses in a process model. The Episode Miner [21] provides functionality to mine frequent partial orders from an event log. Diamantini et al. [11] provide a related tool to mine frequent behavioral patterns from event logs. Unlike *LocalProcessModelDiscovery*, the behavioral patterns mined with this tool and with the Episode Miner are not able to discover choice relations and loops.

3 The LocalProcessModelDiscovery tool

Figure 3 shows the main screen of the *LocalProcessModelDiscovery* tool, consisting of three panels: on the left side a panel to configure the mining parameters, in the middle a panel that presents the mining results when mining has completed, and on the right side a panel to interact with and navigate through the mined local process models.

3.1 Configuring the Local Process Model Miner

Figure 4 shows the mining parameters panel. Located at the top of the panel is a **maximum number of transitions in the LPMs** slider, which allows the user to set the maximum number of non-silent transitions for the local process models. The maximum number of non-silent transitions puts an upper bound on the number of expansion iterations in the local process model mining algorithm, therefore, setting this slider to higher values enables the tool to discover larger patterns at the cost of higher computation time.



Mining Parameters

General Settings
Maximum number of transitions in LPMs: 4
Number of LPMs to discover: 100
Allow duplicate transitions

Operators
Use sequence operator
Use concurrency operator
Use inclusive choice operator
Use exclusve choice operator
Use loop operator

Pruning
Minimum number of occurrences in log: 29
Minimum determinism: 0.5

Constraints
Event gaps
Maximum number of adjacent non-fitting event: 0
Time gaps
Maximum time gap between adjacent fitting event: 20
Minutes

Projections
None

MINING

Mining Results

Group 21 Group 22 Group 23 Group 24 Group 25 Group 26 Group 27 Group 28 Group 29 Group 30
Group 10 Group 11 Group 12 Group 13 Group 14 Group 15 Group 16 Group 17 Group 18 Group 19 Group 20
Group 1 Group 2 Group 3 Group 4 Group 5 Group 6 Group 7 Group 8 Group 9

Score: 0.5882348213279636
Frequency: 348
Confidence: 0.2911770310614241
Determinism: 1.0
Language fit: 1.0
Coverage: 0.24635922330097088

LPM 1 LPM 22 LPM 32 LPM 33

Finished mining Local Process Models!

Results Navigation Parameters

Grouping
Ranking-based

Filters
Activities in LPM: 2

Overlay
None

Ranking
Support: 0.1
Language fit: 0.1
Confidence: 0.4
Coverage: 0
Determinism: 0.3
Avg. num. firings: 0.1

Fig. 3. The home screen of the *LocalProcessModelDiscovery* tool.

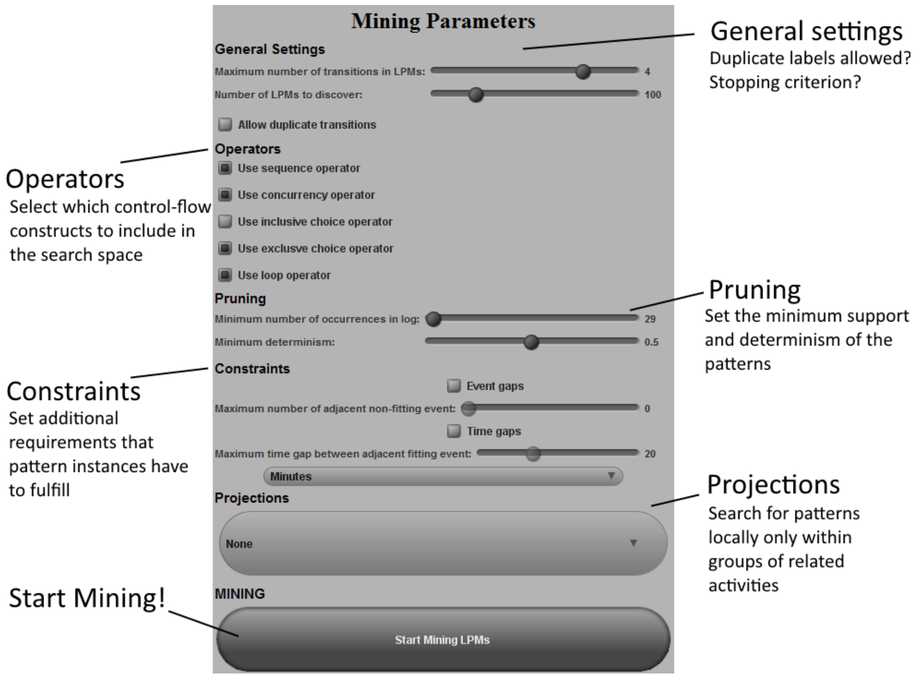
The **number of LPMs to discover** slider lets the user specify a maximum number of patterns that he or she wants to obtain, allowing him or her to prevent mining an overload of patterns. Note that the local process model discovery algorithm returns a ranked list of patterns, therefore, the algorithm returns the patterns with the highest weighted score to the user.

The **allow duplicate transitions** toggle allows the user to specify whether or not he or she wants to mine patterns that contain multiple transitions with the same label. Enabling this option comes at the price of higher computation cost.

The **operator** section of the parameter panel allows the user to include or exclude patterns with certain control-flow constructs in the search space of the mining procedure.

In the **pruning** section of the parameter panel the user can specify a **minimum number of occurrences** of the pattern in the log (i.e., minimum support), and a minimum **determinism** value for the pattern. Patterns that do not meet these thresholds set by the user are not presented in the results and are not expanded into larger patterns, thereby pruning the search space of the local process model mining algorithm.

Instances of a local process model in the event log do not have to be consecutive, i.e., there can be other events that do not fit the behavior specified by the LPM in-between. When this is not desired, constraint-based mining of LPMs can be used to put restrictions on the instances of LPMs, thereby not including instances in the event log that do not comply with there constraints, even when it fits the behavior specified by the LPM. The parameters section allows the user to specify **event gap constraints** and **time gap constraints**. An event gap constraint puts a maximum on the number of non-fitting events in-between two fitting events of an instance of an LPM in the log. For example, a sequence $\langle a, b, x, x, c \rangle$



Mining Parameters

General Settings

Maximum number of transitions in LPMs: 4

Number of LPMs to discover: 100

Allow duplicate transitions

Operators

Use sequence operator

Use concurrency operator

Use inclusive choice operator

Use exclusve choice operator

Use loop operator

Pruning

Minimum number of occurrences in log: 29

Minimum determinism: 0.5

Constraints

Event gaps

Maximum number of adjacent non-fitting event: 0

Time gaps

Maximum time gap between adjacent fitting event: 20

Minutes

Projections

None

MINING

Start Mining LPMs

General settings
Duplicate labels allowed? Stopping criterion?

Operators
Select which control-flow constructs to include in the search space

Pruning
Set the minimum support and determinism of the patterns

Constraints
Set additional requirements that pattern instances have to fulfill

Projections
Search for patterns locally only within groups of related activities

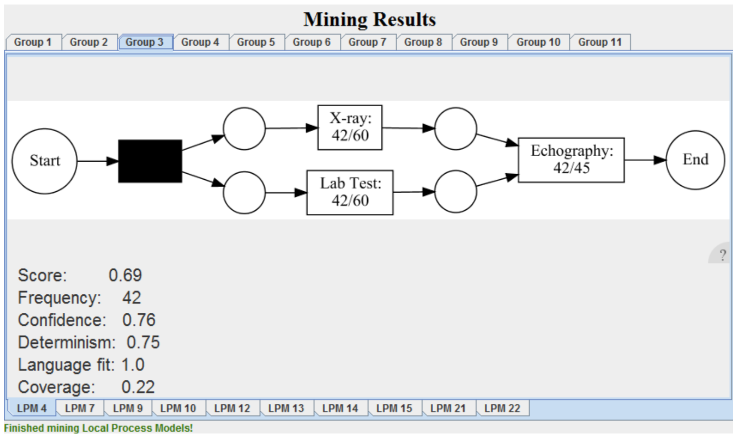
Start Mining!

Fig. 4. The parameters panel of the *LocalProcessModelDiscovery* tool.

is considered to be an instance of a Petri net N with language $\mathfrak{L}(N) = \{\langle a, b, c\rangle\}$ when the event gap constraint is set to 2, but is not considered to be an instance when the event gap constraint is set to 1. For event logs containing timed events, time gap constraints can be used to specify an upper bound on the time difference between two consecutive events that fit the behavior of an LPM.

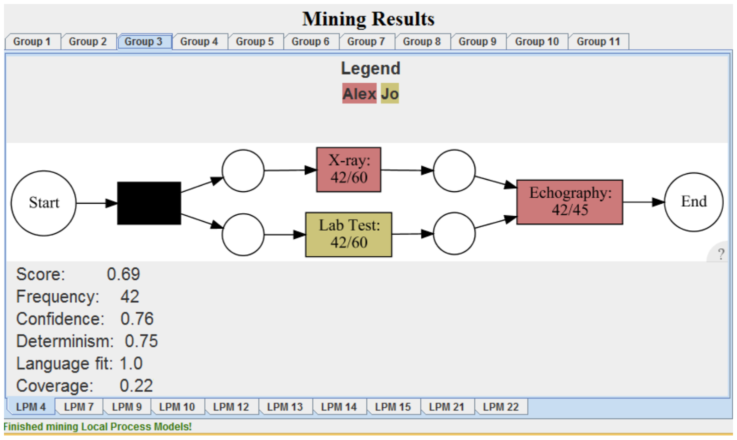
The computational complexity of mining LPMs is exponential in the number of activities in the event log [24]. Several heuristic mining techniques have been proposed in [22] to make the mining of LPMs on event logs with large numbers of events feasible. These heuristic techniques work by detecting clusters of activities that frequently occur close to each other in the event log, and then mining the LPMs for each cluster individually, restricting the expansion steps of the LPM mining to activities that are part of the same cluster of activities. The **projections** section of the mining parameters panel allows the user to configure the tool to use these approaches.

When starting the *LocalProcessModelDiscovery* tool, it automatically sets the minimum support parameter and the projections configuration to a default value that is dependent on the event log, using information such as the number of events and the number of activities in the event log.



```
Score:        0.69
Frequency:    42
Confidence:   0.76
Determinism:  0.75
Language fit: 1.0
Coverage:     0.22
```

(a)



```
Score:        0.69
Frequency:    42
Confidence:   0.76
Determinism:  0.75
Language fit: 1.0
Coverage:     0.22
```

(b)

Fig. 5. *(a)* An example LPM in the results panel of the *LocalProcessModelDiscovery* tool, and *(b)* an example of the *overlay* feature in the *navigation* panel, projecting the *resource* information on top of the pattern.

3.2 Interpreting Local Process Model Results

Figure 5a shows grouping of local process models in the mining results panel. The figure shows a single local process model mined from a hospital event log that describes the behavior that a *lab test* and an *X-ray* are performed in arbitrary order, finally followed by an *echography*. Printed below the Petri net are the scores of the pattern in terms of the local process model quality criteria. The local process models are ranked by the aggregated score of the LPM (shown as **score**), and the tabs in the bottom of the panel can be used to navigate through the LPMs in the ranking. Typically, the mining procedure results in multiple local process models that specify behavior over the same alphabet of activities. By default, the resulting local process models are grouped by the alphabet of

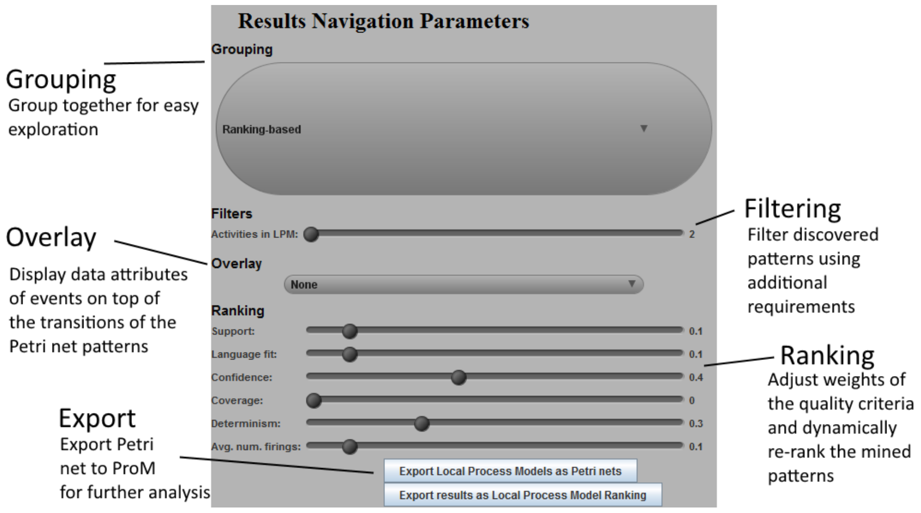

Results Navigation Parameters
Grouping
Ranking-based
Filters
Activities in LPM: 2
Overlay
None
Ranking
Support: 0.1
Language fit: 0.1
Confidence: 0.4
Coverage: 0
Determinism: 0.3
Avg. num. firings: 0.1
Export Local Process Models as Petri nets
Export results as Local Process Model Ranking
Grouping
Group together for easy exploration
Overlay
Display data attributes of events on top of the transitions of the Petri net patterns
Export
Export Petri net to ProM for further analysis
Filtering
Filter discovered patterns using additional requirements
Ranking
Adjust weights of the quality criteria and dynamically re-rank the mined patterns

Fig. 6. The navigation panel of the *LocalProcessModelDiscovery* tool, which allows the user to interact with the mining results and perform more in-depth follow-up analysis.

activities that they describe. The **group** tabs above the Petri net can be used to explore the LPMs that describe different alphabets of activities.

3.3 Navigating Local Process Model Results

The navigation panel provides several functionalities to interactively navigate through the resulting local process models obtained through mining. A weighted average over the quality criteria of local process models is used to rank the resulting local process models, and the results are presented in the order of the ranking. The user can reconfigure the weights assigned to each of the quality criteria in the **ranking** section of the navigation panel, resulting in an updated ranking of local process models in the **results panel**.

The **overlay** functionality in the navigation panel allows the user to project data attributes of the events in the log onto the local process models. The overlay functionality consists of a drop-down selector where the user can select one of the global event attributes (i.e., an event attribute that is set for every event in the log). Figure 5b illustrates the overlay functionality and shows the mining results panel when selecting the *org:resource* event attribute for the local process model of Figure 5. The results show that the *X-ray* and *Echography* events that fit the behavior of this local process model are most frequently performed by employee *Alex*, while the *Lab Test* events that fit the behavior of this pattern are most frequently performed by employee *Jo*. Note that this does not say anything about the *X-ray* events that do not fit the behavior of this pattern, i.e., the X-ray events that are not performed concurrently to the *Lab Test* and before *Echography*.

The **filters** section of the results panel allows the user to filter out local process models from the results that do not comply with certain specifications that are provided by the user, such as a minimum number of **activities in the log**. In the **grouping** section, the user can select a strategy for grouping mined local process models into groups of local process models for the visualization in the results panel. By default, the **ranking-based** grouping strategy is used, which adds one local process model A to the same group as another local process model B if 1) the set of activities of A is a subset of the set of or equal to the activities of B and 2) A has a lower aggregated score than B.

4 Conclusion

This paper presents the tool *LocalProcessModelDiscovery*, which allows for the mining of frequent patterns (called local process models) from event logs that are expressed as Petri nets. Local process models are positioned in-between process discovery and Petri net synthesis on the one hand, and frequent pattern mining on the other hand. The local process models that can be mined with this tool extend existing sequential pattern mining approaches: while sequential patterns are restricted to mining frequent sequential behavior, local process models allow the frequent patterns to describe a more general language over the activities by expressing the patterns as Petri nets. *LocalProcessModelDiscovery* supports the mining of local process models as well as functionality to navigate through the mining results and to relate discovered local process models back to the event log for a more in-depth analysis.

References

1. van der Aalst, W.M.P.: Process mining: data science in action. Springer (2016)
2. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.F.: Replaying history on process models for conformance checking and performance analysis. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 2(2), 182–192 (2012)
3. Badouel, E., Bernardinello, L., Darondeau, P.: Petri net synthesis. Springer (2015)
4. Bergenthum, R., Desel, J., Lorenz, R., Mauser, S.: Synthesis of Petri nets from finite partial languages. Fundamenta Informaticae 88(4), 437–468 (2008)
5. Best, E., Schlachter, U.: Analysis of Petri nets and transition systems. In: Proceedings of the 8th Interaction and Concurrency Experience (2015)
6. Carmona, J., Cortadella, J., Kishinevsky, M.: GENET: A tool for the synthesis and mining of Petri nets. In: Application of Concurrency to System Design. pp. 181–185. IEEE (2009)
7. Carmona, J., Solé, M.: PMLAB: An scripting environment for process mining. In: Proceedings of the BPM Demo Sessions. pp. 16–20. CEUR-ws.org (2014)
8. Chapela-Campa, D., Mucientes, M., Lama, M.: Towards the extraction of frequent patterns in complex process models. Jornadas de Ciencia e Ingeniería de Servicios pp. 215–224 (2017)
9. Cortadella, J., Kishinevsky, M., Kondratyev, A., Lavagno, L., Yakovlev, A.: Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. IEICE Transactions on Information and Systems 80(3), 315–325 (1997)

10. Dalmas, B., Tax, N., Norre, S.: Heuristics for high-utility local process model mining. In: Proceedings of the International Workshop on Algorithms & Theories for the Analysis of Event Data. pp. 106–121. CEUR-ws.org (2017)
11. Diamantini, C., Genga, L., Potena, D., Storti, E.: Discovering behavioural patterns in knowledge-intensive collaborative processes. In: International Workshop on New Frontiers in Mining Complex Patterns. pp. 149–163. Springer (2014)
12. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The ProM framework: A new era in process mining tool support. In: International Conference on Application and Theory of Petri nets. pp. 444–454. Springer (2005)
13. Fournier-Viger, P., Gomariz, A., Gueniche, T., Soltani, A., Wu, C.W., Tseng, V.S.: SPMF: a Java open-source pattern mining library. The Journal of Machine Learning Research 15(1), 3389–3393 (2014)
14. Fournier-Viger, P., Lin, J.C.W., Kiran, R.U., Koh, Y.S., Thomas, R.: A survey of sequential pattern mining. Data Science and Pattern Recognition 1(1), 54–77 (2017)
15. Greco, G., Guzzo, A., Manco, G., Pontieri, L., Saccà, D.: Mining constrained graphs: The case of workflow systems. In: Constraint-Based Mining and Inductive Databases, pp. 155–171. Springer (2006)
16. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. ACM SIGKDD explorations newsletter 11(1), 10–18 (2009)
17. Han, J., Cheng, H., Xin, D., Yan, X.: Frequent pattern mining: current status and future directions. Data Mining and Knowledge Discovery 15(1), 55–86 (2007)
18. Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.C.: PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In: Proceedings of the International Conference on Data Engineering. pp. 215–224. IEEE (2001)
19. Janssenswillen, G., Depaire, B.: BupaR: Business process analysis in R. In: Proceedings of the BPM Demo Sessions. pp. 160–164. CEUR-ws.org (2017)
20. La Rosa, M., Reijers, H.A., Van Der Aalst, W.M.P., Dijkman, R.M., Mendling, J., Dumas, M., García-Bañuelos, L.: APROMORE: An advanced process model repository. Expert Systems with Applications 38(6), 7029–7040 (2011)
21. Leemans, M., van der Aalst, W.M.P.: Discovery of frequent episodes in event logs. In: International Symposium on Data-Driven Process Discovery and Analysis. pp. 1–31. Springer (2014)
22. Tax, N., Sidorova, N., van der Aalst, W.M.P., Haakma, R.: Heuristic approaches for generating local process models through log projections. In: Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining. pp. 1–8. IEEE (2016)
23. Tax, N., Genga, L., Zannone, N.: On the use of hierarchical subtrace mining for efficient local process model mining. In: International Symposium on Data-driven Process Discovery and Analysis. CEUR-ws.org (2017)
24. Tax, N., Sidorova, N., Haakma, R., van der Aalst, W.M.P.: Mining local process models. Journal of Innovation in Digital Ecosystems 3(2), 183–196 (2016)
25. van der Werf, J.M.E.M., van Dongen, B.F., Hurkens, C.A.J., Serebrenik, A.: Process discovery using integer linear programming. In: International Conference on Applications and Theory of Petri nets. pp. 368–387
26. XES Working Group: IEEE standard for eXtensible Event Stream (XES) for achieving interoperability in event logs and event streams. IEEE Std 1849-2016 pp. 1–50 (Nov 2016)