Decomposed Process Mining with DivideAndConquer

H.M.W. Verbeek

Department of Mathematics and Computer Science,
Eindhoven University of Technology, Eindhoven, The Netherlands
h.m.w.verbeek@tue.nl

Abstract. Many known process mining techniques scale badly in the number of activities in an event log. Examples of such techniques include the ILP Miner and the standard replay, which also uses ILP techniques. To alleviate the problems these techniques face, we can decompose a large problem (with many activities) into a number of small problems (with few activities). Expectation is, that the run times of such a decomposed setting will be faster than the run time of the original setting. This paper presents the DivideAndConquer tool, which allows the user to decompose a large problem into small problems, to run the desired discovery or replay technique on each of these decomposed problems, and to merge the results into a single result, which can then be shown to the user.

1 Introduction

Although process mining has become a mature technique for analyzing all kinds of business processes described in an event log [1], some challenges still remain. One of these challenges is to improve the usability of known process mining techniques in the context of event logs that contain many activities. Many of these known techniques scale very badly, like exponential or worse, in the number of these activities [2]. As a result, as the event logs are growing, these techniques start to fail producing results.

To overcome this problem, [2] has proposed a solution that is based on decomposition. Instead of applying the technique on a single big event log, we decompose the event log into a collection of smaller event logs, apply the technique on each of these smaller event logs, and merge the results of these applications into a single result. Clearly, the technique might very well still work on the smaller event logs, while it might fail on the single big event log.

This decomposition approach works for both discovery and replay. For discovery, we decompose the big event log, into smaller event logs, discover a process model from each of these event logs, and merge the resulting process models into a single process model. For replay, we decompose both the event log and the process model into smaller event logs and smaller process models, replay every smaller event log on the corresponding smaller process model, and merge the resulting alignments into a single alignment.

The DivideAndConquer tool supports both decomposed discovery and decomposed replay. This tool has been built as the `DivideAndConquer` package for ProM 6[3]. As a result, many discovery and replay techniques that have been implemented in ProM 6 can be used by the tool. This paper discusses this tool in detail, using the ILP

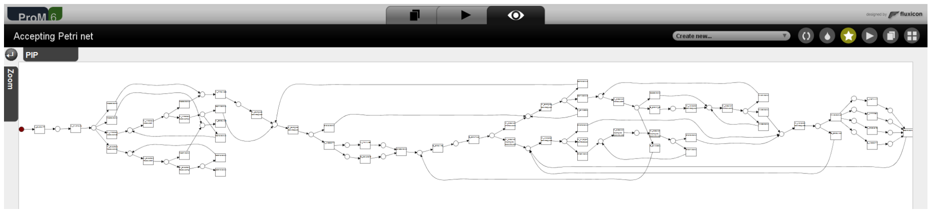


Fig. 1. The Petri net discovered from the noise-free event log.

Miner as discovery technique and the ILP-based replayer as replay technique. Both ILP-based techniques have been researched thoroughly but scale badly in the number of activities, which makes them ideal candidates for our tool.

2 Running Example

As a concrete running example, we take an event log that is based on the BPI Challenge of 2012 [4]. The advantages of this event log are that it is a real-life event log, but that it is still very structured, which suits our purposes fine. Nevertheless, the ILP Miner is very sensitive to noise in the event log, and while the event log is very structured, it still contains some noise. For this reason, as we would have done the same when using the ILP Miner without decomposition, we allow ourselves to first remove the noise from the event log when doing discovery.

For sake of completeness, we mention that we have removed this noise by *aligning* the original event log onto a known process model (see [5] for this process model). Every trace in the original event log is replaced in the resulting noise-free event log by its aligned trace, that is, the activity sequence that corresponds to the best fitting execution sequence of the process model. This guarantees that every trace in the resulting event log can be successfully replayed by the process model used.

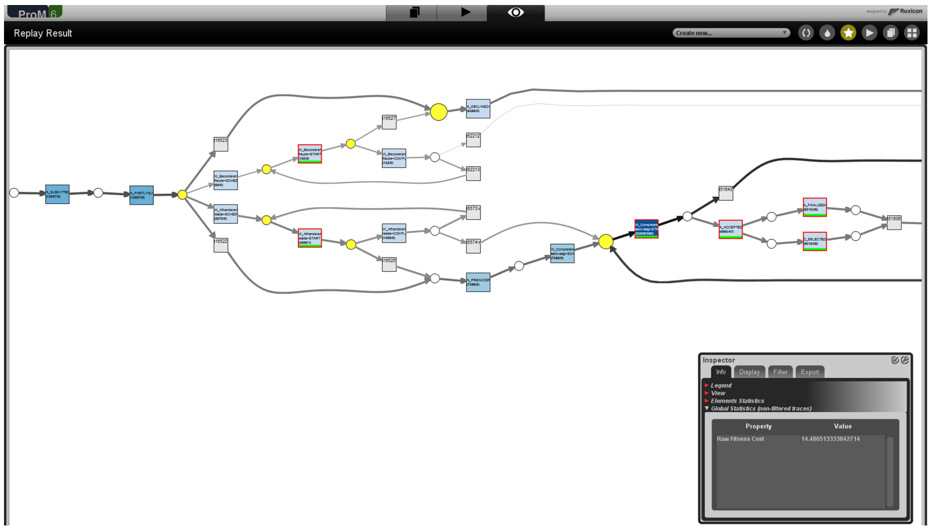
The resulting noise-free event log contains a single process, $13,087$ instances (Cases) of this process, and $383,836$ events for these instances. Furthermore, the event log contains 58 different activities (Event classes).

3 Discovery

We can discover a process model (a Petri net) from this event log using the standard ILP Miner as implemented in ProM 6. On this event log, this takes approx. 37.5 minutes[1], and results in the Petri net as shown by Fig. 1.

Instead of using the ILP Miner as-is, we can also use it in a decomposed setting using the "Divide-and-Conquer" tool. First, the tool discovers a number of activity clusters, that is, sets of related activities, from the event log. For this discovery, the tool

[1] All running times mentioned in this paper were obtained on a Dell Optiplex 9020 desktop computer with Inter(R) Core(TM) i7-4770 CPU @ 3.40 GHz processor, 16 GB of RAM, running Windows 7 Enterprise 64-bit, Service Pack 1, and using revision 13924 of the `DivideAndConquer` package.



Property	Value
Raw Fitness Cost	14.486513333842714

Fig. 2. The result of replaying the original event log on the discovered net.

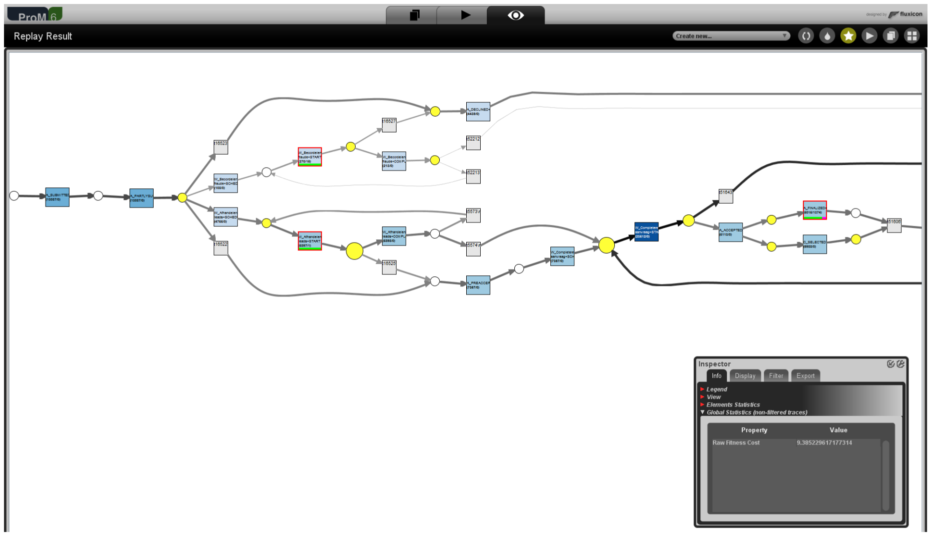
uses a combination of available heuristics. As a result, 34 activity clusters are discovered from the noise-free event log. Second, the tool decomposes the event log into as many event logs as there are activity clusters, where the i-th event log contains only the activities contained in the i-th cluster. As a result, the event log is decomposed into 34 event logs. Third, the tool discovers a Petri net from every of these event logs using the ILP Miner. As a result, we now have a collection of 34 Petri nets. Fourth and last, the tool merges (by fusing transitions that refer to the same activity) all resulting Petri nets into a single Petri net. As a result, a single Petri net is 'discovered' by the tool. This decomposed application of the ILP Miner takes approx. 1.5 minutes and results in the same Petri net (see Fig. 1).

4 Replay

We can replay the original, noisy, event log on the discovered model using the standard replayer as implemented in ProM 6, which uses ILP-based techniques to ensure optimality of the resulting alignment. On this event log and this net, this takes about 90 seconds, and results in a replay costs 14.49 (see [2] for details on replay costs). Fig. 2 shows the results of the replay projected onto the Petri net.

Like with discovery, we can apply decomposition on the standard replayer using the "Divide-and-Conquer" tool[2]. First, as we now have a Petri net to start with, the tool uses this net to find the clusters, see [2] for details. As a result, 11 activity clusters are found. Second, the tool decomposes the event log into as many event logs as there are

[2] The description of the decomposed replay has been simplified in this section to keep it readable. See [6] for details on the full decomposed replay.



Property	Value
Raw Fitness Cost	9.385229617177314

Fig. 3. The result of replaying the original event log on the discovered net using decomposition.

clusters, and the Petri net into as many Petri nets as there are clusters. As before, the i-th event log contains only the activities contained in the i-th cluster. In a similar way, the i-th Petri net contains only visible transitions that are related to this cluster, that is, all transitions not related to this cluster have been made invisible. As a result, the event log is decomposed into 11 event logs, and the Petri net is decomposed into 11 Petri nets. Third, the tool runs the standard ILP-based replayer on every of these 11 combinations of a filtered event log and a filtered Petri net. As a result, we now have 11 replays, each with its own costs and its own sets of alignments (which matches a trace in the log with an execution sequence of the net). Fourth, the tool merges these replays (including costs and alignments) into a single replay (with single costs and single alignment). This results in a replay with costs 9.39 in less than 70 seconds. Note that the run times are indeed faster, but that the quality of the result is not perfect (9.39, where 14.49 is perfect). This is due to the fact that the decomposed replay does not take the replay of the other problems into account. As a result, the merged alignment may not always match, as every subreplay could have been too optimistic.

5 Implementation

The DivideAndConquer tool has been implemented as the `DivideAndConquer` package in ProM 6 [3]. This package contains a coherent collection of so-called *provided objects* and a collection of so-called *base plug-ins*. The decomposed discovery using the ILP Miner and the decomposed replay using the standard replayer have also been implemented as plug-ins: The "Discover with ILP using Decomposition" plug-in and the "Replay with ILP using Decomposition" plug-in. Both plug-ins are basically *macro* plug-ins that only call (6 and 10) base plug-ins in the correct order with the correct parameters to get the job done.

Instead of running these *macro* plug-ins, it is also possible for the user to run the base plug-ins one by one in the correct order. As a result, the user has much more control over what these plug-ins are doing, as many parameters can then be set which are hidden from the user by the *macro* plug-ins. As a result, the user can quite often achieve better results as he has better control over these parameter values. In this way, we cater for both the novice user (*macro* plug-ins) and the expert user (*base* plug-ins).

6 Links

The DivideAndConquer tool can be installed by installing the `DivideAndConquer` package in ProM 6. At the moment of writing, this is only possible in the so-called Nightly Build version of ProM 6, which can be downloaded from the ProM 6 Nightly Build folder[1]. Nevertheless, we plan for this package to be part of the next release of ProM 6, ProM 6.4, which is scheduled around September 2014. Once released, it can be downloaded from its release page[2]. Documentation of the tool can be found at the "DivideAndConquer" folder[3] in the ProM 6 documentation, which includes in-depth documentation on the package, screencasts, and inputs like sample event logs and sample Petri nets. The original event log as used in this paper is available as "Aalst13.xes.gz", the noise-free event as "Aalst13Aligned.mxml.gz", and the Petri nets that can be used for replaying both logs as "Aalst13.pnml" and "Aalst13Aligned.pnml". Basically, these two nets are similar except for the use of capitals in the label names. Without the proper capitalization, the *macro* plug-ins will fail to relate transitions and activities in an automated way.

References

1. Aalst, W.M.P.v.d.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. 1st edn. Springer Publishing Company, Incorporated (2011)
2. Aalst, W.M.P.v.d.: Decomposing Petri nets for process mining: A generic approach. Distributed and Parallel Databases **31**(4) (2013) 471–507
3. Verbeek, H.M.W., Buijs, J.C.A.M., Dongen, B.F.v., Aalst, W.M.P.v.d.: ProM 6: The process mining toolkit. In: Proc. of BPM Demonstration Track 2010. Volume 615., CEUR-WS.org (2010) 34–39
4. Dongen, B.F.v.: BPI Challenge 2012 (2012) http://dx.doi.org/10.4121/uuid: 3926db30-f712-4394-aebc-75976070e91f.
5. Aalst, W.M.P.v.d., Verbeek, H.M.W.: Process discovery and conformance checking using passages. Fundamenta Informaticae **131**(1) (2014) 103–138
6. Verbeek, H.M.W., Aalst, W.M.P.v.d.: Decomposed process mining: The ILP case. In: BPI 2014 Workshop, Haifa, Israel (September 2014) Accepted for publication.

[1] http://www.promtools.org/prom6/nightly

[2] http://www.promtools.org/prom6/prom64.html

[3] https://svn.win.tue.nl/trac/prom/browser/Documentation/DivideAndConquer