ESANN 2010 proceedings, European Symposium on Artificial Neural Networks - Computational Intelligence
and Machine Learning. Bruges (Belgium), 28-30 April 2010, d-side publi., ISBN 2-930307-10-2.

Heuristics Miner for Time Intervals

Andrea Burattin and Alessandro Sperduti

Department of Pure and Applied Mathematics
University of Padua, Italy

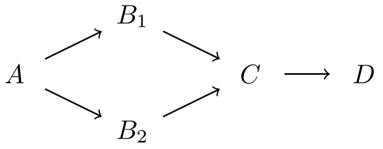
Abstract. Process Mining attempts to reconstruct the workflow of a business process from logs of activities. This task is quite important in business scenarios where there is not a well understood and structured definition of the business process performed by workers. Activities logs are thus mined in the attempt to reconstruct the actual business process. In this paper, we propose the generalization of a popular process mining algorithm, named Heuristics Miner, to time intervals. We show that the possibility to use, when available, time interval information for the performed activities allows the algorithm to produce better workflow models.

1 Introduction

In the IEEE Glossary [1], one of the definitions for "process" is: "*a sequence of steps performed for a given purpose; for example, the software development process.*" In this paper, we will use the term to refer to the so called "business processes". Recently, the number of systems for business processes management is increased incredibly, thanks to the availability of improved tools and systems for their support. These systems, however, often are used to support the activities performed by single workers, with no explicit blueprint for the whole workflow. In fact, the actual workflow emerges by the combination of activities performed by the workers, according to business rules that are owned by the workers. In these cases, it has been recognized that, in order to optimize the business process, it is of vital importance to try to identify the actual workflow. A number of data mining algorithms have been developed, over the years, that attempt to reconstruct the business process, or related information, from logs of performed activities. These algorithms constitute a branch of Data Mining named Process Mining. In this paper, we refer to algorithms for control-flow discovery, i.e. algorithms that focus on constructing the process model that underlies the observations, expressed with some formalization, such as Petri Net, Heuristics Net, etc. For example, the Heuristics Miner algorithm [2, 3] produces a Heuristics Net (a network that can be easily converted into a Petri Net), while Cook et al. approaches, [4, 5, 6], which actually constitute a collection of three techniques (one purely statistical, based on recurrent neural networks; one purely algorithmic and another that is a mix of statistical and algorithmic approach) produce Finite State Machines. One feature of all process mining algorithms proposed up to now, is the assumption that each activity is considered instantaneous. This is due to the fact that usually a single log for each preformed activity is recorded, regardless of the duration of the activity. In many practical cases, however, activities involve a span of time, so they can be described by time intervals (couples of time points). Of course, not recording the duration

$$W = \left\{(A, B_1, B_2, C, D)^5 \ ; \ (A, B_2, B_1, C, D)^5\right\}$$

$$X \Rightarrow_W Y = \frac{|X >_W Y| - |Y >_W X|}{|X >_W Y| + |Y >_W X| + 1} \qquad X \Rightarrow_W (Y \wedge Z) = \frac{|Y >_W Z| + |Z >_W Y|}{|X >_W Y| + |X >_W Z| + 1}$$



$$C \Rightarrow_W D = \frac{10 - 0}{10 + 0 + 1} = 0.9\overline{09}$$

$$A \Rightarrow_W (B_1 \wedge B_2) = \frac{5 + 5}{5 + 5 + 1} = 0.9\overline{09}$$

Fig. 1: Top: example of process log W with 10 process instances (^{n} indicates n repetitions of the same sequence). Middle: two main Heuristics Miner formulas. Bottom left: the process sample that generated the log W. Bottom right: values returned by the shown formulas for some activities.

of activities makes mining quite hard. In some cases, information about duration of some activities is available, and it is wise to use this information. For this reason, in this paper we propose to generalize Heuristics Miner, which is a well known and robust process mining algorithm, to allow the treatment of time intervals. This way a better (more precise and similar to the real one) process model can be mined, without modifying the overall complexity of the original algorithm, and in addition preserving backward compatibility.

2 Heuristics Miner

Heuristics Miner is a process mining algorithm that uses a statistical approach to mine the dependency relations among activities represented by logs. In Fig. 1, we have tried to summarize the main concepts underpinning the mining process. Given the set of activities $\{A, B_1, B_2, C, D\}$, the first row shows an example of log W with 10 process instances (^{n} indicates n repetitions of the same sequence), where a process instance is a sequence of performed activities. The second row shows two of the main formulas used by Heuristics Miner to determine the presence of dependency between activity X and Y in the log W (left side), and the presence of an AND-split (right side), i.e. the parallel activities Y and Z jointly depend on the completion of the X activity.

In these formulas, the notation $|X >_W Y|$ refers to the number of times that, in W, activity Y directly follows activity X. If the $\Rightarrow_W$ is greater than a *dependency threshold*, a parameter of the algorithm, then the algorithm assumes the presence of the dependency. After that, in order to mine the relationship type between activities (if in AND or XOR split) the second formula is used: if $X \Rightarrow_W (Y \wedge Z)$ is greater than an *AND threshold* then Y and Z are considered as

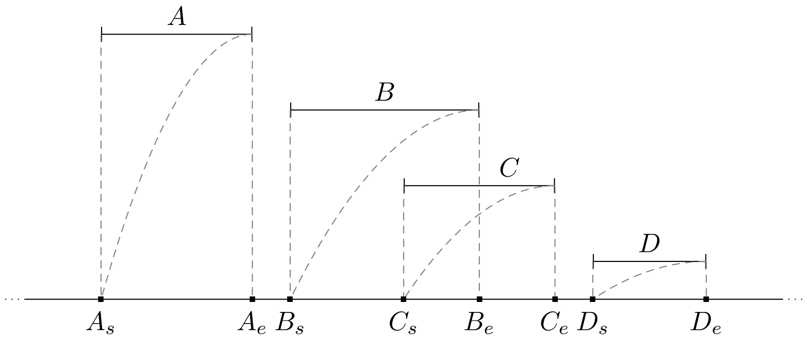


Fig. 2: Example of a process model composed by activities with completely different behaviours if mined as time intervals or instantaneous events.

in AND relation, otherwise they are considered as activities in mutual exclusion (XOR relation).

3 Activities as time intervals

Heuristics Miner considers each activity as an instantaneous event, either considering the start or the finish time as log of the activity. In the example shown in Fig. 2, regardless the starting or the finishing time is used, the algorithm will mine always a "linear dependency" between all activities (so D depends on C, C on B and B on A). However, as we can see, there is actually no causal dependency between activities B and C, since they are partially overlapped in time.

In order to extend the algorithm to be able to cope with time intervals, it is necessary to provide a new definition for the direct succession relation in the time intervals context. With an activity represented as a single event, we have that $X >_W Y$ iff $\exists\ \sigma = t_1 \ldots, t_n$ and $i \in \{1, \ldots, n-1\}$ such that $\sigma \in W$, $t_i = X$ and $t_{i+1} = Y$. This definition has to be modified so to cope with activities represented by time intervals. First of all, given an event e let define with $activityName[e]$ the activity name the event belongs to, and with $typeOf[e]$ the type of the event (either *start* or *end*). The new succession relationship $X \gtrdot_W Y$ between two activities is defined as follow:

Definition 1 (Direct succession relation, $\gtrdot$)**.** Let X and Y be two interval activities (not instantaneous) in a log W, then

$$X \gtrdot_W Y \text{ iff } \exists\ \sigma = t_1, \ldots, t_n \text{ and } i \in \{2, \ldots, n-2\},\ j \in \{3, \ldots, n-1\}$$
$$\text{such that } \sigma \in W,\ t_i = X_{end} \text{ and } t_j = Y_{start} \text{ and}$$
$$\forall k \text{ such that } i < k < j \text{ we have that } typeOf[t_k] \neq start$$

Less formally, we can say that two activities, to be in a direct succession relation, must meet the condition for which the termination of the first occurs before the start of the second and, between the two, no other activity should start.

There is also a new concept to be introduced: the parallelism between two activities. With the instantaneous activities we have X and Y considered as parallel when they are observed in no specific order (sometimes X before Y and other times Y before X), so $(X >_W Y) \wedge (Y >_W X)$. Actually, this definition may seem odd, but without the notion of duration, it's complex to express parallelism. In the new context, this definition is easier and more intuitive:

Definition 2 (Parallelism relation, $\|$)**.** Let X and Y be two interval activities (not instantaneous) in a log W, then

$$X \|_W Y \text{ iff } \exists\ \sigma = t_1, \dots, t_n \text{ and } i, j, u, v \in \{1, \dots, n\}$$
$$\text{with } t_i = X_{start},\ t_j = X_{end} \text{ and } t_u = Y_{start},\ t_v = Y_{and}$$
$$\text{such that} \quad u < i < v \quad \vee \quad i < u < j$$

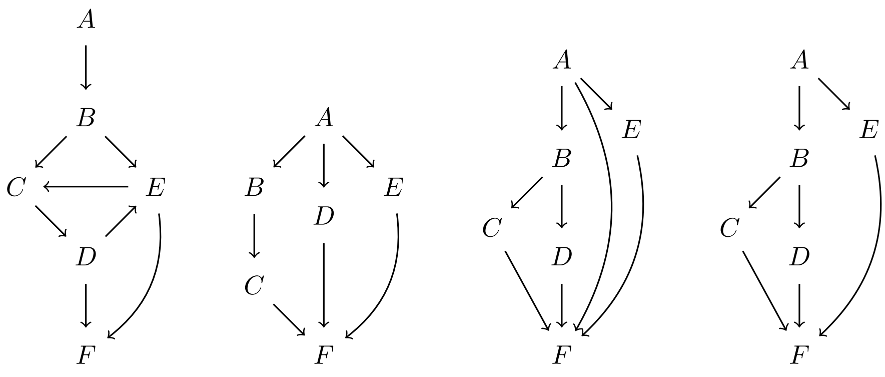
More intuitively, this definition indicates two activities as parallel if they are overlapped or if one contains the other. Referring to the notion of "intervals algebra" as introduced by Allen [7] and the macro-algebra $A_3 = \{\prec, \cap, \succ\}$ as Golumbic and Shamir described in [8], we can think the direct succession relation as the "preceedings" $(X \prec Y)$ one and the parallelism relation as the "intersection" $(X \cap Y)$ one.

We not only modified the notions of relations between two activities, we improved the algorithm performance modifying the formulas for the statistical dependency and to determine the relation type (AND or XOR). The new formulas are:

$$X \Rightarrow_W Y = \frac{|X \overline{>}_W Y| - |Y \overline{>}_W X|}{|X \overline{>}_W Y| + |Y \overline{>}_W X| + 2|X \|_W Y| + 1}$$

$$X \Rightarrow_W (Y \wedge Z) = \frac{|Y \overline{>}_W Z| + |Z \overline{>}_W Y| + 2|X \|_W Y|}{|X \overline{>}_W Y| + |X \overline{>}_W Z| + 1}$$

where the notation $|X \|_W Y|$ refers to the number of times that, in W, activity X and Y are in parallel relation. In the first formula, in addition to the usage of the new direct succession relation, we introduced the parallel relation in order to reduce the likelihood to see, in the mined model, the activities in succession relation if in the log they were overlapped. In the second formula, we inserted the parallelism counter in order to prefer the selection of an AND relation if the two activities were overlapped in the log. In both cases, because of the symmetry of the $\|$ relation, a factor 2 was introduced for parallel relations. With the new formulas, we obtain "backward compatibility" with the original Heuristics Miner algorithm: if we use special intervals, where for each activity its start and finish points coincide, we'll obtain the same behaviour. This happens because any two activities X and Y will never be in parallel relation, i.e. $|X \|_W Y| = 0$. We can use this feature to tackle logs with a mixture of activities expressed as time intervals and instantaneous, improving the performances without losing the Heuristics Miner benefits, as suggested in Fig. 2.



(a) 0% as int. (b) 16% as int. (c) 33% as int. (d) 50% as int.

Fig. 3: In this figure the first 4 networks mined from the algorithm are shown: from left to right we have the network mined with (a) 0, (b) 1, (c) 2 and (d) 3 activities as time intervals. The last network presented here is already the correct one, so adding more interval does not improve the algorithm performance.

4 Experimental Results

We implemented the new algorithm as a plugin for the ProM framework [9] and we built a process log generator for testing purposes. Actually, we produced 100 processes and, for each, 10 logs (with 500 instances each): in the first one, no activity is expressed as time interval, in the second one only one activity (randomly different) is expressed as time interval, in the third two of those are intervals and so on, until the last where all activities are expressed as intervals.

We executed our algorithm in the log observing an improvement of the output network proportional to the number of activities as time intervals. In Fig. 3 we present one particular process mined with different logs (with increasing number of activities expressed as intervals); and, in Fig. 4, we present results of the mining expressed as the average of the F1 measure; the minimum and maximum average values of F1 are reported as well. It can be observed that already with a small percentage of activities expressed as intervals, the performances are improved.

5 Conclusion and future works

In this paper we presented a generalization of the Heuristics Miner algorithm that uses the activity expressed as time intervals instead of single events. We introduced this notion into the previous algorithm paying attention to the backward compatibility. Future works may involve the autonomous identification of the best algorithm parameters with, for example, a Minimum Description Length approach that minimize the cost of the network definition and the number of log

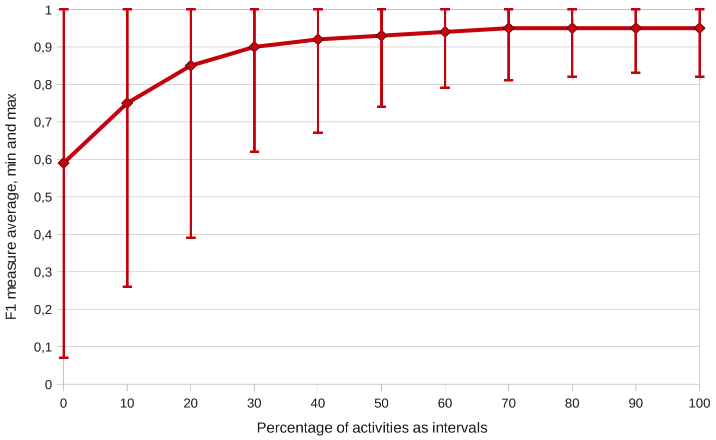


Fig. 4: Plot of the F1 measure averaged over 100 processes logs (true positives are the correctly mined dependences; false positives are dependences present in the original model but not in the mined one; and false negatives are dependences present in mined model but not in the original one). Minimum and maximum average values are reported as well. Already with a small number of activities expressed as intervals the performances are improved.

observations that can't be explained by the model.

References

[1] *IEEE standard computer dictionary : a compilation of IEEE standard computer glossaries*. IEEE Computer Society Press, New York, NY, USA, January 1991.

[2] A. 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.

[3] W. M. P. van der Aalst and A. J. M. M. Weijters. *Process-Aware Information Systems: Bridging People and Software through Process Technology*, chapter 10: Process Mining, pages 235–255. John Wiley & Sons Inc, 2004.

[4] Jonathan E. Cook, Zhidian Du, Chongbing Liu, and Alexander L. Wolf. Discovering models of behavior for concurrent workflows. *Computers in Industry*, 53(3):297 – 319, 2004.

[5] Jonathan E. Cook, Alexander L. Wolf, and Er L. Wolf. Automating process discovery through event-data analysis. In *In Proceedings of the 17th International Conference on Software Engineering*, pages 73–82. ACM Press, 1995.

[6] Jonathan E. Cook and Alexander L. Wolf. Discovering models of software processes from event-based data. Technical Report 3, University of Colorado, November 1996.

[7] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[8] Martin Charles Golumbic and Ron Shamir. Complexity and algorithms for reasoning about time: a graph-theoretic approach. *J. ACM*, 40(5):1108–1133, 1993.

[9] B. F. van Dongen, A. K. A. 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. *Application and Theory of Petri Nets*, 3536:444–454, 2005.