# A Generic Approach for Process Performance Analysis using Bipartite Graph Matching

Chiao-Yun Li[1], Sebastiaan J. van Zelst[1,2], and Wil M. P. van der Aalst[1,2]

[1] Fraunhofer Institute for Applied Information Technology (FIT), Germany
{chiao-yun.li,sebastiaan.van.zelst,wil.van.der.aalst}@fit.fraunhofer.de
[2] Chair of Process and Data Science, RWTH Aachen University, Germany
{s.j.v.zelst,wvdaalst}@pads.rwth-aachen.de

**Abstract.** The field of process mining focuses on the analysis of event data, generated and captured during the execution of processes within companies. The majority of existing process mining techniques focuses on process discovery, i.e., automated (data-driven) discovery of a descriptive process model of the process, and conformance and/or compliance checking. However, to effectively improve processes, a detailed understanding in differences of the actual performance of a process, as well as the underlying causing factors, is needed. Surprisingly, few research focuses on generic techniques for process-aware data-driven performance measurement, analysis and prediction. Therefore, in this paper, we present a generic approach, which allows us to compute the average performance between arbitrary groups of activities active in a process. In particular, the technique requires no a priori knowledge of the process, and thus does not suffer from representational bias induced by any underlying process representation. Our experiments show that our approach is scalable to large cases and especially robust to recurrent activities in a case.

**Keywords:** Process Mining · Process Performance Analysis · Bipartite Graph Matching · Integer Linear Programming

## 1 Introduction

The field of *process mining* has gained its significance as a technology to objectively obtain insights into business processes by exploiting *event logs*, i.e., records of events executed in the context of a business process. To further understand the execution of business processes, *process performance analysis* aims to measure and analyze the performance of business processes, e.g., throughput time, by extracting information from event logs [19].

The techniques for process performance analysis [2, 5, 12] can be classified into two approaches: model-based and log-based approaches [19]. The first one typically projects the event log on a predefined or discovered process model [2, 12, 13]. The advantage is that the performance analysis results can be interpreted in the context of a process model. However, this assumes that there exists a suitable model with high conformance.

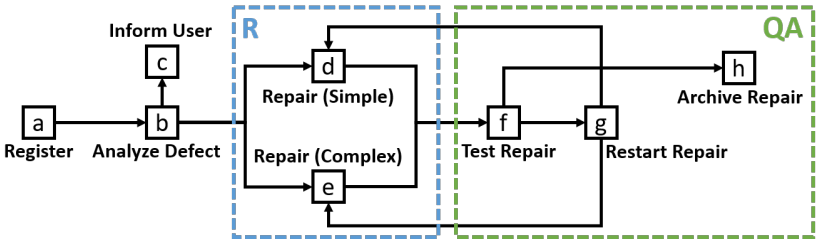



Fig. 1: Process model of repairing telephones in a company [15]. A telephone can be fixed by either the team for simple defects or the other team for complex defects whereas some defects can be handled by either team.

The other approach is purely based on an event log [5, 11, 19]. The techniques based on such approach are not limited to the constraints of the model-based techniques and more flexible to users' need. However, current work fails to provide robust performance metrics in the presence of repeated activities in a *case*, i.e., a run of a business process.

In this paper, we present a novel approach for business process performance analysis. Figure 1 serves as a motivating example of a process of repairing telephones in a company [15]. The process starts with a registration of a telephone sent by a customer (a). The telephone is then analyzed (b) and the problem is reported to the customer (c). Afterwards, one of the two teams in the Repair (R) department repairs the defect; one for simple defect (d) and the other for complex defect (e) while some defects can be handled by either team. Then, the Quality Assurance (QA) department tests if the defect is repaired (f) and decides whether to restart another repair (g) or to archive the case (h).

Suppose we are interested in the performance of R and QA department, i.e., the average duration between `d` or `e` to `g` or `h`. Given a case in which the activities are executed in the order of $\langle a, b, c, d, f, g, e, f, g, e, f, h\rangle$, Figure 2 shows the events on a timeline for an overview of the duration of interest.

Intuitively, the average duration would be $\frac{1+60+60\times 2}{3} = 60.33$ minutes. We propose a novel approach which supports the intuition and the flexibility of analysis while being robust to recurrent activities, e.g., all `d`, `e`, `g` and `h` are included for analysis. Our approach applies bipartite graph matching to pair the events for computation. Moreover, by allowing multiple selection of activities, our approach can be applied to analysis at a higher abstraction level, e.g., performance of department `R` and `QA`, without building another model in advance.

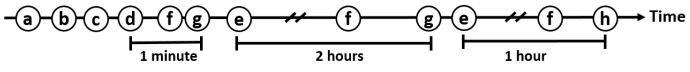



Fig. 2: A case of the process in Figure 1. The events are plotted on a timeline with the duration of interest shown, i.e., duration between `d` or `e` to `g` or `h`.

It is common that real processes do not conform with a pre-defined structured process model. Interchangeability and concurrency among activities impose challenges on the existing methods for process performance analysis. These methods either need to exclude the non-conforming cases [2], or explicitly specify the relationships between events [5]. For instance, considering processes in a hospital, suppose we are interested in the duration between an injection and a medical examination for the reaction after the injection. It is impossible to pre-define what and how many injections that a patient would need in advance. By specifying two sets of activities, our approach allows for a certain degree of uncertainty among the activities in the event log, and further provides the flexibility when measuring the performance of complex processes.

To assess our approach, we conduct a quantitative evaluation of its scalability and a qualitative evaluation by means of comparing our approach with analysis obtained from commercial tools. The first experiment shows that our approach is scalable to large cases while the comparative evaluation shows its robustness against recurrent activities in a case.

The remainder of this paper is organized as follows. Section 2 compares our approach with existing work in academia and with the typical functionalities provided by commercial tools. Section 3 introduces the definitions and notations used in the paper. We present our approach in Section 4, followed by the evaluation and discussion in Section 5. Finally, Section 6 concludes the paper.

## 2 Related Work

Existing work on process performance analysis provides metrics at different levels. Most of them focus on the level of cases, e.g., throughput time, and/or the level of individual activities, e.g., waiting time. Despite various metrics, most techniques can be classified into model-based and log-based approaches [19].

Model-based performance analysis accounts for most techniques proposed [2, 3, 13, 14, 16, 17]. These techniques attempt to map an event log on a given process model. Regardless of the amount of work [4, 6, 20], relatively few methods provide solutions on performance analysis for cases deviating from the model. Hornix provides the option to include the deviated cases when computing performance using token-based replay technique [12]. However, the results can be misleading when there are activities with the same label or artificial activities in a model. Adriansyah measures the performance of activities from events with transaction type attributes, e.g., start, complete, and suspend [2]. Nevertheless, there is no generic rule to determine the timestamps of the missing events in the work. Generally speaking, model-based approach confronts the challenges of (1) the reliability of the prescribed model, (2) the need for recalculation if the model changes, (3) complex and flexible business processes which result in low conformance of the model, and (4) the flexibility of the analysis, e.g, the metrics provided are dependent on the model.

Alternatively, one can analyze process performance based on an event log only [5, 11, 19]. This approach provides the flexibility and static results due to

Table 1: Comparison of works on process performance analysis.

| Algorithm/Method | Model Independent | Two Arbitrary Activities | Two Sets of Activities |
|---|---|---|---|
| [2] Alignments | − | − | − |
| [3] Robust Performance Analysis | − | − | − |
| [5] First/Last to First/Last events | + | + | − |
| [9] Analysis on Segmented Journey Model | + | + | − |
| [10] Disco | − | − | − |
| [11] Context-aware KPI | + | − | − |
| [12] Log reply on Petri Nets | − | − | − |
| [13] Hierarchical Performance Analysis | − | − | − |
| [14] Analysis with Advanced Constructs | − | − | − |
| [16] Alignments with Stochastic Petri Nets | − | − | − |
| [17] Log Replay on Petri Nets | − | + | − |
| [18] Queue Enabling CSPNs (QCSPN) | + | − | − |
| [19] Dotted Chart | + | − | − |
| **This Paper: Bipartite Graph Matching** | + | + | + |

the independence of a model. In this paper, we proposed a novel performance measurement technique which is scalable to large event logs. We apply bipartite graph matching for pairing as many events for measuring as possible. By the design of the algorithm, users can analyze the performance between two arbitrary groups of activities. Compared to the existing log-based techniques, our approach is more flexible to the need of analysis and, supported by the evaluation, more robust to recurrent activities. Table 1 lists some representative work on process performance analysis. We compare if a method is dependent to a model, i.e., suffers from the reliability and limitations, and flexible to the analysis, i.e., the selection of activities.

## 3 Preliminaries

In this section, we introduce the related concepts and notations used in this paper, including process mining and bipartite graph matching.

As a preliminary, we first define the required mathematical notations for a *sequence*. We introduce a function: $\sigma : \{1, 2, ...n\} \to X$ for a finite *sequence* of length $n$ over an arbitrary set $X$. The function assigns every element $x \in X$ to an index $i \in \{1, 2, ..., n\}$. We write $\sigma = \langle x_1, x_2, , ..., x_n \rangle \in X^*$, where $x_1 = \sigma(1)$, $x_2 = \sigma(2)$, ..., $x_n = \sigma(n)$ and $X^*$ denotes the set of all possible sequences over $X$ of arbitrary length. Given a sequence $\sigma \in X^*$ and an element $x \in X$, let $|\sigma|$ denotes the length of the sequence, we overload the notation and write $x \in \sigma$ if and only if $\exists 1 \leq i \leq |\sigma| \big(\sigma(i) = x\big)$.

### 3.1 Event Logs

An *event log* is a collection of sequences of *events*, i.e., *traces*. Each event describes the execution of an *activity*, i.e., a well-defined step in a process [1]. During the execution of processes, i.e., *cases*, additional information, i.e., *attributes*, are associated to the events and/or cases. In the following part of this section, we explain the relationships among *event*, *activity*, *traces*, and *event log*. For simplicity, in the remainder of the paper, $\mathscr{E}$ denotes the universe of events and $\mathscr{C}$ denotes the universe of cases.

**Definition 1. (Event, Event Attributes)** *Let $\mathscr{A}$ be the universe of activities and $\mathscr{T}$ be the universe of timestamps. We define the projection functions:*

- $\pi_{act} : \mathscr{E} \to \mathscr{A}$*, which* $\pi_{act}(e)$ *represents the activity of the event* $e \in \mathscr{E}$*,*
- $\pi_{ts} : \mathscr{E} \to \mathscr{T}$*, which* $\pi_{ts}(e)$ *represents the occurring time of the event* $e \in \mathscr{E}$*.*

**Definition 2. (Trace)** *We define a projection function $\pi_{tra} : \mathscr{C} \to \mathscr{E}^*$ for a trace which describes a finite sequence of events in a case such that*

- $\forall 1 \leq i < j \leq |\sigma| \big(\pi_{ts}(\sigma(i)) \leq \pi_{ts}(\sigma(j))\big)$
- $\forall 1 \leq i < j \leq |\sigma| \big(\sigma(i) \neq \sigma(j)\big)$

*We define an event index as the order of the event in a trace.*

**Definition 3. (Event Log)** *Given a collection of cases $\mathscr{L} \subseteq \mathscr{C}$, an event log $L$ is a collection of traces, i.e., $L = \{\sigma \in \mathscr{E}^* | \exists c \in \mathscr{L} \big(\sigma = \pi_{tra}(c)\big)\} \subseteq \mathscr{E}^*$ s.t. $\forall \sigma, \sigma' \in L \big(\exists e \in \mathscr{E} (e \in \sigma \wedge e \in \sigma' \Rightarrow \sigma = \sigma')\big)$.*

For the sake of performance analysis, we additionally define the notation of measurements between events in the context of a trace.

**Definition 4. (Measurement)** *Let $M$ denote all the measurable entities, e.g., duration, cost. Given a measurable entity $m \in M$ and a trace $\sigma$, we define the function: $\phi_m : \mathscr{E}^* \times \mathscr{E} \times \mathscr{E} \to \mathbb{R}$ for which $\forall e, e' \in \sigma$.*

In this paper, we assume that any measurable entities for performance analysis are always available given an event log.

### 3.2 Bipartite Graph Matching

A matching in a bipartite graph is to select the edges, i.e., pairs of nodes, such that no edges share the same nodes. This section formally defines a (weighted) bipartite graph and the corresponding matching problem as follows.

**Definition 5. ((Weighted) Bipartite Graph)** *Let $G = (V, E, w)$ be a weighted graph where $V$ is a set of nodes, $E$ is a set of edges and $w$ is a weight function $w : E \to \mathbb{R}$. Given a weighted graph $G$, it is bipartite if and only if $\exists V_1, V_2 \subseteq V \big(V_1 \cup V_2 = V \wedge V_1 \cap V_2 = \emptyset\big)$ s.t. $\nexists (v, v') \in E \big(v, v' \in V_1 \vee v, v' \in V_2\big)$. We denote a weighted bipartite graph as $G = (V_1 \cup V_2, E, w)$.*

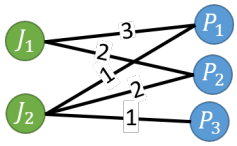



Fig. 3: An example of maximum weighted bipartite matching. The edges $(P_1, J1)$ and $(P_2, J_2)$ are selected due to maximum weights.

**Definition 6. (Maximum Weighted Bipartite Matching)** *Given a weighted bipartite graph $G = (V_1 \cup V_2, E, w)$, a bipartite graph matching is the selection of $E' \subseteq E$ s.t. $\forall(v, v') \in E'\big(\exists(v, v'') \in E' \Rightarrow v' = v'' \wedge \exists(v'', v') \in E' \Rightarrow v'' = v\big)$.*

*Let $E_M$ denotes all possible matching. A maximum weighted bipartite matching is a matching $E' \in E_M$ such that $\nexists E'' \in E_M\big(\sum_{e \in E''} w(e) > \sum_{e \in E'} w(e)\big)$.*

An example is shown in Figure 3 as a bipartite graph which $J_i$ is a set of jobs and $P_j$ is the sets of applicants. An edge indicates that an applicant applies for the job with the qualification score implied as its weight. A maximum weighted bipartite matching is the optimal assignments of the applicants to the jobs such that the total qualification score is maximum, i.e., $(P_1, J_1)$ and $(P_2, J_2)$.

In this paper, Integer Linear Programming (ILP) should be applied to find the maximum weighted bipartite matching. ILP is a mathematical optimization method that, given a set of variables, assigns an integral value to every variable to achieve the best result, e.g., minimum cost, maximum benefits, with the *objective* and the *constraints* formulated in linear relationships [7].

## 4 Generic Process Performance Analysis using Bipartite Graph Matching

Our approach aims for measuring the performance between two arbitrary groups of activities. By projecting the events of interests into a bipartite graph for each trace, we find the maximum weighted bipartite matching indicating the pairs of events from which the performance metric of a case is derived. Figure 4 depicts a global overview of our approach and we illustrate each step in more detail in the sections specified on the arcs.

### 4.1 Bipartite Graphs Generation

Given an event log, a user specifies two groups of activities of interest for analyzing the performance from one to the other. We construct a weighted bipartite graph for each case by taking the events of interests as nodes and connecting them according to the direction specified. The weights of the edges are computed using a monotonic function of the event index of the nodes. Figure 5 shows an example of a bipartite graph for the trace which we only show the events of interest. The projection of a trace to a bipartite graph is formalize as follows.

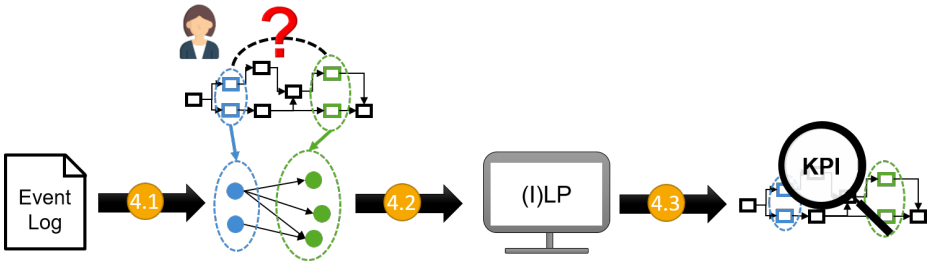



Fig. 4: An overview of the proposed approach. We analyze the performance of a case by finding a maximum weighted bipartite graph matching using ILP.

**Definition 7. (Trace Projection)** *Let $\mathscr{A}$ be the universe of activities. Given two sets of activities $A_S, A_T \subseteq \mathscr{A}$ and an event log $L \subseteq \mathscr{E}^*$, we define a trace projection function $\rho(\sigma)$ which generates a weighted bipartite graph $G = (S \cup T, E, w)$ from $\sigma \in L$ such that*

- $S = \{e \in \sigma \mid \pi_{act}(e) \in A_S\}$,
- $T = \{e \in \sigma \mid \pi_{act}(e) \in A_T\}$,
- $E = \{(s,t) \in S \times T | \exists 1 \leq i < j \leq |\sigma| \big(\sigma(i) = s \wedge \sigma(j) = t\big)\}$, *and*
- *given $(s,t) \in E$ and let $1 \leq i < j \leq |\sigma|$ s.t. $\sigma(i) = s$ and $\sigma(j) = t$, $w(s,t) = \frac{1}{j-i}$.*

*Given a graph $\rho(\sigma)$, $\rho(\sigma)_E$ denotes the edges of the graph.*

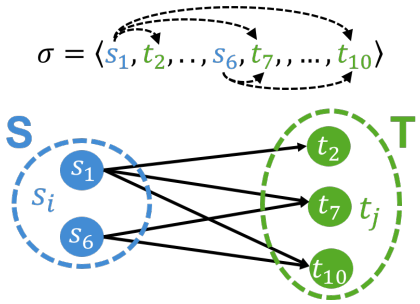



Fig. 5: An example of the bipartite graph given a trace

### 4.2 Maximum Weighted Bipartite Graph Matching Formulation

Given a bipartite graph built as described in Definition 7, we select pairs of events for measurements by finding a maximum weighted bipartite matching using ILP. The ILP model is formulated as below.

**Definition 8. (ILP Formulation)** *Given a weighted bipartite graph $G = (S \cup T, E, w)$, we assign a variable $x_{(s,t)}$ to every edge $(s,t) \in E$ and limit each*

*variable to* 0 *or* 1*, indicating whether the edge is selected provided the integral solutions. We formulate the constraints and the objective as follows to find the maximum weighted bipartite matching (Definition 6).*

$$\begin{aligned} &\textit{Maximize} \quad Z = \sum_{(s,t)\in E} w_{(s,t)} x_{(s,t)} \\ &\textit{Subject to} \quad \sum_{s\in S} x_{(s,t)} \leq 1, && \forall t \in T \\ &\qquad\qquad \sum_{t\in T} x_{(s,t)} \leq 1, && \forall s \in S \\ &\qquad\qquad x_{(s,t)} \in \{0,1\} \end{aligned}$$

*We denote the value for every variable* $x_{(s,t)}$ *as* $x^*_{(s,t)}$ *such that* $Z$ *is maximum.*

In principle, the problem above should be solved by ILP. However, we alternatively relax the integral constraints by applying Linear Programming (LP), i.e., $x_{(s,t)} \in [0,1]$, and compare their performance in Section 5.1. We found that LP always selects as much fraction of an edge as possible, i.e., 1, due to the constraints on the nodes. Thus, given the integral solutions provided by LP, we use LP in our implementation for better performance.

### 4.3 Aggregation

For each case, we average the desired measurements, e.g., duration between two events, given the pairs of events selected as described in Section 4.2. The performance metrics in terms of case and event log is computed as follows.

**Definition 9. (Case Performance)** *Let* $M$ *denote all the measurable entities of a trace. Given a trace* $\sigma = \pi_{tra}(c)$ *for which* $c \in \mathscr{C}$*, refer to Definition 8, we compute the case performance in terms of* $m \in M$ *as:*

$$\delta_m(\sigma) = \frac{\sum_{(s,t)\in\rho(\sigma)_E} \phi_m(\sigma,s,t) * x^*_{(s,t)}}{\sum_{(s,t)\in\rho(\sigma)_E} x^*_{(s,t)}}.$$

Suppose all the cases in an event log are independent to each other, then we compute the log performance as defined below.

**Definition 10. (Log Performance)** *Let* $M$ *denote all the measurable entities of a trace. Given an event log* $L$*, for every trace* $\sigma \in L$*,* $\delta_m(\sigma)$ *denotes the case performance in terms of* $m \in M$*. We select traces* $T = \{\sigma \in L || \rho(\sigma)_E| \geq 1|$ *and compute the log performance regarding* $m \in M$ *as:*

$$\Delta_m(L) = \frac{\sum_{\sigma\in T} \delta_m(\sigma)}{|T|}$$

By transforming a trace into a bipartite graph, our approach is capable of measuring the performance between two arbitrary groups of activities. The grouping can also be applied to analyze the performance at a higher level of abstraction without constructing another model at the desired granularity.

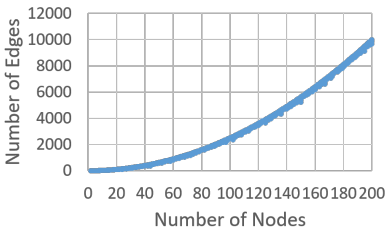



Fig. 6: Number of nodes and edges of the synthetic dataset.

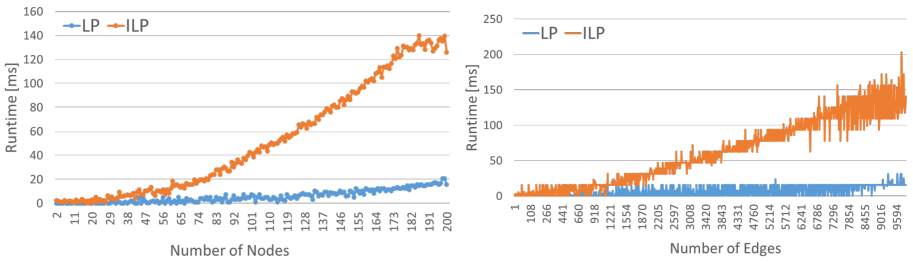
## 5 Evaluation

In this section, we evaluate the scalability using both ILP and LP in Section 5.1, followed by a comparative evaluation in Section 5.2.

### 5.1 Scalability with LP and ILP

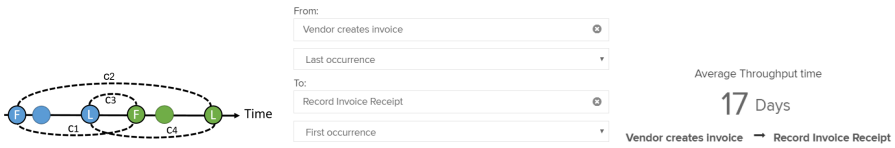
We evaluate the performance and scalability of our approach by measuring the runtime of solving LP and ILP. Since the complexity of the problem is reflected by the number of variables (edges) and the number of constraints (nodes), we generate a synthetic event log with different number of nodes and edges. The event log simulates the results after filtering out events not of interest. Each event is randomly assigned as either an event in the start or target activity group and an event index between 1 and 1000, indicating the maximum length of a trace before filtering. The length of the trace is limited to 2 to 200 representing the log after the filtering. For each length, 15 traces are generated. Figure 6 shows the number of nodes and the corresponding number of edges.

Figure 7 shows the runtime using LP and ILP in terms of the number of edges and nodes. As expected, the runtime increases with complexity of the graph, i.e., the number of variables (edges) as shown in Figure 7b and constraints (nodes) as shown in Figure 7a. However, the edges selected by using LP and ILP are different. Nevertheless, compared to using ILP, the optional solutions from LP provide the same number of edges given integral solutions. Given the better performance and scalability of LP, we apply LP instead of ILP for our approach.




(a) Runtime[ms] versus Number of Nodes (b) Runtime[ms] versus Number of Edges

Fig. 7: Performance of the approach using LP and ILP. The performance using LP is better than ILP and highly scalable.




(a) Four configurations. (b) Configuration selected. (c) Analysis result.

Fig. 8: Performance analysis using Celonis [5]

### 5.2 Comparative Evaluation

We compare our approach with other techniques with the dataset from the BPI Challenge 2019 [8]. The methods from Celonis [5] and Disco [10] are used for comparison as the representatives of log-based and model-based techniques, respectively. In the experiment, we evaluate the performance between *Vendor creates invoice* and *Record Invoice Receipt*. The two activities are chosen due to the limitation of analyzing with Disco [10], i.e., only selection of two adjacent activities in the model is allowed. To emphasize the difference of analysis result in the presence of recurrent activities, we further select the cases in which each activity is executed at least twice. After the filtering, we obtain 4915 cases with 14498 *Vendor creates invoice* and 17210 *Record Invoice Receipt* in total.

Table 2 shows the performance statistics in terms of the frequency and duration from different approaches. The case frequency shows that the number of cases considered while the absolute frequency shows the number of measurements used for calculation. These two frequencies are the same using Celonis due to only one measurement out of four possible configurations is selected. Figure 8a depicts the four possible configurations: first to first ($c1$), first to last ($c2$), last to first ($c3$), last to last ($c4$). Each refers to the duration between the first/last event of one activity to the first/last event of the other activity in a trace. For each trace, only one configuration is chosen as the performance metrics. Thus, it produces skewed results depending on the configuration, e.g., the measurement between $c2$ and $c3$ can differ much. Figure 8c shows the analysis from one of the configuration as in Figure 8b. From the frequency information in Table 2, it shows that some cases are ignored if the events in the cases do not have the configured order, e.g., the last execution of *Vendor creates invoice* is after the first execution of *Record Invoice Receipt*.

Figure 9 shows the mean duration and absolute frequency between activities from Disco [10]. The cases that do not comply to the model are ignored, i.e., only 2684 out of 4915 cases are considered. In addition, the resulting metrics tends to be smaller since it only considers the measurement if *Record Invoice Receipt* directly follows *Vendor creates invoice* without other activities in between.

The results show that our approach is robust for that it covers as many measurements as possible given two activities. Moreover, Table 2 shows that even a simple question as the time between two activities is not as simple as it seems. Besides, our approach allows for measuring the performance between two groups of activities.

Table 2: Comparative evaluation results. Our approach computes the performance based on the most measurements, i.e., absolute frequency.

| | | Celonis | | | | Disco | Our Approach |
|---|---|---|---|---|---|---|---|
| | | $F \to F$ | $F \to L$ | $L \to F$ | $L \to L$ | | |
| Case Freq. | | 4881 | 4915 | 2613 | 4889 | 2684 | 4915 |
| Absolute Freq. | | 4881 | 4915 | 2613 | 4889 | 3797 | 14441 |
| $\Delta_{dur}(L)$ | Min. | 5h | 19h | 5h | 84m | 84m | 8h |
| | Max. | 70y | 70y | 159d | 447d | 15.7w | 35y |
| | Median. | 14d | 41d | 12d | 21d | 3.7d | 18d |
| | Mean. | 61d | 105d | 17d | 33d | 10.2d | 43.7d |

F: Timestamp of the earliest occurrence of *Vendor creates invoice* in a case
L: Timestamp of the latest occurrence of *Record Invoice Receipt* in a case
$\Delta_{dur}(L)$: Duration between two activities of the event log $L$

## 6 Conclusion

Process performance can be analyzed based on a model or an event log only. Most techniques are model-based and are, thus, limited to the representational bias of the underlying models and not flexible to the need of analysis, e.g., measuring the performance between two milestones. Existing techniques that are based on an event log face the challenge of providing a robust result in the presence of recurrent activities. This paper introduces a novel and generic process-aware log-based technique which is robust to recurrent activities and can be further extended to analysis of two groups of activities. By applying bipartite graph matching, we pair as many events as possible and measure them accordingly. Our experiments show that the proposed approach is scalable to huge event logs and outperforms the existing methods by providing a robust metrics given recurrent activities.

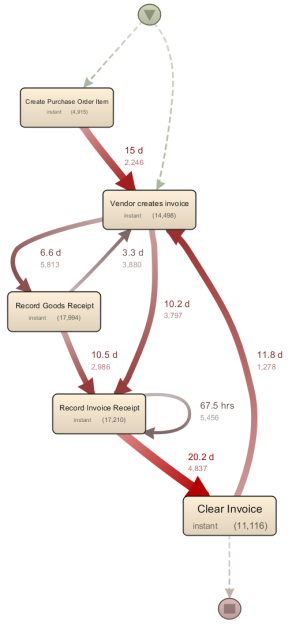



Fig. 9: Analysis using Disco [10]

To better locate and diagnose the performance of a process, we aim to extend our approach by incorporating the control-flow constructs into the bipartite graph and clustering the cases based on the measured performance. Furthermore, more investigation on different business context or rules can be applied to determine complementary performance indicators and the weights of the bipartite graph. Meanwhile, we also aim to automate the selection of the activities to facilitate the application of our approach.

## References


1. van der Aalst, W.M.P.: Data science in action. In: Process Mining, pp. 3–23. Springer (2016)
2. Adriansyah, A.: Aligning observed and modeled behavior (2014)
3. Adriansyah, A., van Dongen, B., Piessens, D., Wynn, M., Adams, M.: Robust performance analysis on yawl process models with advanced constructs. Journal of Information Technology Theory and Application (JITTA) **12**(3), 5–26 (2012)
4. Castellanos, M., Casati, F., Shan, M.C., Dayal, U.: iBOM: A platform for intelligent business operation management. In: 21st International Conference on Data Engineering (ICDE'05). pp. 1084–1095. IEEE (2005)
5. Celonis SE: Academic cloud. https://academiccloud.celonis.com (2019), [Online; accessed 26-April-2019]
6. Costello, C., Molloy, O.: Building a process performance model for business activity monitoring. In: Information Systems Development, pp. 237–248. Springer (2009)
7. Dantzig, G.: Linear programming and extensions. Princeton university press (2016)
8. van Dongen, B.F.: Dataset BPI Challenge 2019. 4TU.Centre for Research Data. https://doi.org/10.4121/uuid:d06aff4b-79f0-45e6-8ec8-e19730c248f1 (2019), [Online; accessed 10-April-2019]
9. Gal, A., Mandelbaum, A., Schnitzler, F., Senderovich, A., Weidlich, M.: Traveling time prediction in scheduled transportation with journey segments. Information Systems **64**, 266–280 (2017)
10. Günther, C.W., Rozinat, A.: Disco: Discover your processes. In: BPM (2012)
11. Hompes, B.F., Buijs, J.C., van der Aalst, W.M.P.: A generic framework for context-aware process performance analysis. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". pp. 300–317. Springer (2016)
12. Hornix, P.T.: Performance analysis of business processes through process mining. Master's Thesis, Eindhoven University of Technology (2007)
13. Leemans, M., van der Aalst, W.M.P., Van Den Brand, M.G.: Hierarchical performance analysis for process mining. In: Proceedings of the 2018 International Conference on Software and System Process. pp. 96–105. ACM (2018)
14. Piessens, D., Wynn, M.T., Adams, M.J., van Dongen, B.: Performance analysis of business process models with advanced constructs (2010)
15. ProM: Running example - process to repair telephones in a company. http://www.promtools.org/prom6/downloads/example-logs.zip (2019), [Online; accessed 16-May-2019]
16. Rogge-Solti, A., van der Aalst, W.M.P., Weske, M.: Discovering stochastic petri nets with arbitrary delay distributions from event logs. In: International Conference on Business Process Management. pp. 15–27. Springer (2013)
17. Rozinat, A.: Process mining: conformance and extension (2010)
18. Senderovich, A., Rogge-Solti, A., Gal, A., Mendling, J., Mandelbaum, A., Kadish, S., Bunnell, C.A.: Data-driven performance analysis of scheduled processes. In: International Conference on Business Process Management. pp. 35–52. Springer (2016)
19. Song, M., van der Aalst, W.M.P.: Supporting process mining by showing events at a glance. In: Proceedings of the 17th Annual Workshop on Information Technologies and Systems (WITS). pp. 139–145 (2007)
20. Wetzstein, B., Leitner, P., Rosenberg, F., Brandic, I., Dustdar, S., Leymann, F.: Monitoring and analyzing influential factors of business process performance. In: 2009 IEEE International Enterprise Distributed Object Computing Conference. pp. 141–150. IEEE (2009)