

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/360593566>

Data-Driven Analysis of Batch Processing Inefficiencies in Business Processes

Chapter in Lecture Notes in Business Information Processing · May 2022

DOI: 10.1007/978-3-031-05760-1_14

CITATIONS

8

READS

696

4 authors:



Katsiaryna Lashkevich
University of Tartu

6 PUBLICATIONS 28 CITATIONS

[SEE PROFILE](#)



Fredrik Milani
University of Tartu

72 PUBLICATIONS 932 CITATIONS

[SEE PROFILE](#)



David Chapela-Campa
University of Tartu

20 PUBLICATIONS 134 CITATIONS

[SEE PROFILE](#)



Marlon Dumas
University of Tartu

547 PUBLICATIONS 31,782 CITATIONS

[SEE PROFILE](#)



Data-Driven Analysis of Batch Processing Inefficiencies in Business Processes

Katsiaryna Lashkevich, Fredrik Milani^(✉), David Chapela-Campa,
and Marlon Dumas

University of Tartu, Narva mnt 18, 51009 Tartu, Estonia
{katsiaryna.lashkevich,fredrik.milani,david.chapela,marlon.dumas}@ut.ee

Abstract. Batch processing reduces processing time in a business process at the expense of increasing waiting time. If this trade-off between processing and waiting time is not analyzed, batch processing can, over time, evolve into a source of waste in a business process. Therefore, it is valuable to analyze batch processing activities to identify waiting time wastes. Identifying and analyzing such wastes present the analyst with improvement opportunities that, if addressed, can improve the cycle time efficiency (CTE) of a business process. In this paper, we propose an approach that, given a process execution event log, (1) identifies batch processing activities, (2) analyzes their inefficiencies caused by different types of waiting times to provide analysts with information on how to improve batch processing activities. More specifically, we conceptualize different waiting times caused by batch processing patterns and identify improvement opportunities based on the impact of each waiting time type on the CTE. Finally, we demonstrate the applicability of our approach to a real-life event log.

Keywords: Process mining · Batch processing · Cycle time efficiency

1 Introduction

Companies seek to continuously improve their business processes by incrementally addressing process weaknesses, such as process wastes [33]. Waiting time is a waste in business processes [13]. Waiting times are inevitable in real-life processes, but they can be reduced to improve process efficiency. For instance, waiting times occur with batch processing, i.e., accumulating cases to process them collectively as a group [31]. While batch processing reduces processing time and cost by exploiting economies of scale [32], it also introduces increased waiting times due to case accumulation [7, 16, 19, 32]. Therefore, an optimal batch processing strategy can efficiently balance the processing and waiting times associated with batching. In this regard, practitioners use Cycle Time Efficiency (CTE) to measure the ratio of processing time to the total cycle time [14]. CTE measures the value-adding part, i.e., the processing time compared to the waiting time [14]. As such, CTE-driven process improvements focus on increasing the value-adding part of the process by reducing the waiting time [14, 37].

Modern systems log execution data that can be used in process mining techniques. Such techniques use event logs to enable discovery, analysis, and identification of improvement opportunities in business processes [1]. For instance, process mining techniques enable the discovery of wastes, such as hand-offs between process resources [6], over-processing [38], and waiting times [1]. Process mining techniques have also been used to discover batch processing [19, 24, 25, 39, 41]. However, existing techniques focus on batch processing discovery. Therefore, there is a gap in using process mining for batch processing analysis, i.e., to provide analysts with information on how batch processing, for instance, impacts the CTE. More specifically, there is a gap in how much different types of waiting times associated with batch processing contribute to process inefficiencies. To address this gap, we ask (RQ1) *What types of waiting times are associated with batch processing?*, (RQ2) *How can waiting times caused by batch processing be identified from event logs?*, and (RQ3) *How can improvement opportunities, expressed as inefficiencies due to batch processing, be identified from event logs?*

We propose a process mining-based approach for the discovery and analysis of inefficiencies caused by batch processing. In our approach, we first identify batch processing from an event log. Then, we analyze the incurred waiting times per type. For this step, we first define the types of waiting times incurred by batch processing. Finally, we identify improvement opportunities by analyzing how each type of waiting time contributes to CTE inefficiencies. As such, the contribution of this paper is an approach for the discovery and analysis of inefficiencies due to batch processing. The approach enables process analysts to identify improvement opportunities in processes that have the batch processing. We apply our approach to a real-life case using an event log of a production process of a manufacturing company to identify batch processing inefficiencies.

The rest of the paper is structured as follows. Section 2 provides the background and related work. In Sect. 3, the approach is presented. Section 4 covers the evaluation of the approach, and Sect. 5 concludes the paper.

2 Background and Related Work

Batch processing is when a resource accumulates cases to process them together as a group [31]. Batch processing is often used to solve scheduling problems [34], reduce processing times and costs [32, 34] by executing multiple cases together. However, introducing batch processing requires accumulating cases, which increases waiting time [7, 16, 19, 32]. As such, batch processing implies a trade-off between processing and waiting time [19, 25]. As batch processing impacts processing and waiting times, we can use the CTE metric to measure batch processing efficiency. CTE calculates the ratio of the processing time relative to the cycle time of batch processing activities. CTE ratio close to 1 indicates that the process has comparatively low waiting to processing time and, thus, little room for improvement. However, low CTE indicates a comparatively high waiting time to processing time. Therefore, there is an improvement opportunity [14]. CTE is used to identify time-related process inefficiencies [4, 18, 20].

For instance, in [20], it has been applied for measuring time-related performance in factories to detect inefficiencies. Ignizio [18] reports that CTE is the most efficient metric in identifying workstation instability in production processes. Furthermore, CTE has been used to assess the efficacy of process redesigns [37]. We, therefore, use the CTE metric to analyze batch processing (in)efficiencies.

Process mining techniques enable the discovery of batch processing behavior from event logs. For instance, Nakatumba et al. [26] address the problem of accurately reproducing batch processing behavior in simulation models. Wen et al. [41] propose a process mining technique to discover batch processing from events logs. Pufahl et al. use event logs to enhance process models with batch processing [31]. Andrews et al. [2] propose an approach to identify and quantify shelf time, i.e., idle time that exceeds acceptable duration, in business processes. Pika et al. [27] propose an approach for discovering batch processing from event logs and discusses, in particular, how to identify batch processing from multiple perspectives of a business process such as activity, resource, and data perspectives. Martin et al. [24] focus on identifying rules that trigger a batch processing activity. Similarly, Martin et al. [25] use batch processing metrics, such as frequency of batch processing, batch size, duration of activity instances, and waiting time in a batch, to describe batching behavior. This paper focuses on discovering and analyzing the waiting times associated with batch processing. Thereby, we build on existing research to identify batch processing inefficiencies.

Process mining techniques have also been applied to assess batch processing performance and explore their impact on process performance. Thus, in addition to batch discovery, in [39] batch processing behavior is visualized and quantitatively analyzed to identify specific patterns such as detecting outliers. Similarly, Klijn [19] propose quantifying batch processing performance using measures such as intra-batch case inter-arrival time, case inter-arrival time, batch interval, batch size, and batch frequency. Pufal et al. [31] examine how overall process performance can be improved in terms of time and cost by simulating batch processing. While these studies use event logs to analyze various performance dimensions of batch processing, they do not consider the different types of waiting times associated with batch processing. Furthermore, their focal point is on batch-processing performance measures. We extend existing work by identifying the different types of waiting times related to batch processing and their impact on process performance to identify potential improvement opportunities.

3 Identification of Batch Processing Inefficiencies

In this section, we present the proposed approach for discovering batch processing inefficiencies. First, we present an overview of the approach and then provide a more detailed description of each step.

The approach for identifying batch processing inefficiencies consists of three steps (see Fig. 1). First, given an event log, the batch processing activities —i.e., the activities that are executed in batches— are discovered. The output of this stage is a report listing such activities, as well as the frequency of their execution being part of a batch. The second step is to analyze their batch processing

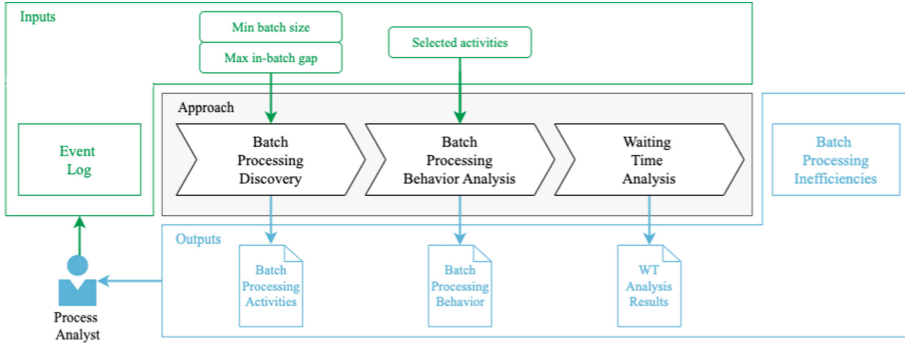


Fig. 1. Overview of the proposed approach

behavior, i.e., type of batch processing, batch size distribution, and batch activation rules. The result of the step is a report describing the batch processing behavior per activity. The final step is to analyze the time that the cases spend waiting before they are batch-processed. This includes an analysis of how batch processing and their associated waiting times impact the CTE. The final output is a report presenting results on batch-processing activities that, based on CTE, can be used to identify improvement opportunities.

3.1 Batch Processing Discovery

For the discovery of batch processing activities, we use the approach proposed by Martin et al. [23] that identifies different types of batch processing activities. They use the resource, start, and end time data to detect when an activity instance is being executed as part of a batch, i.e., by the same resource, and either in sequence, concurrently, or in parallel w.r.t. other activity instances in the batch. Thus, the necessary requirement for the event log is to have the data on the resources, start and end timestamps. In this paper, we focus on batches intentionally processed as such, i.e., the activities were not executed once they were enabled (available for processing) but accumulated and processed as a group. If a case becomes available for processing after the start of the batch, it is excluded as it was not accumulated for batch processing. Accordingly, we extend the definition of a batch with the constraint that all cases part of a batch must be enabled before the batch starts. Therefore, we filter the results obtained by Martin et al. [23] technique to remove such cases.

To perform such filtering, we need to calculate the enabled time of each activity instance. For cases where this information is not available in the event log, we set the enabled time as the end time of the previous activity if no concurrency relation exists. To obtain the concurrency relations between the activities, we use the concurrency heuristics from the Heuristics Miner [40], as their sensibility

to outliers provides more robust results than other naïve techniques.¹ As part of this filtering, we also allow the analyst to define a minimum batch size, i.e., the minimum number of cases that should be accumulated and processed as a group for it to be considered a batch (see Inputs in Fig. 1). The analyst can also define a maximum time gap between the end of processing of a particular case and the start of the next one within one batch (max in-batch gap). This input allows for the discovery of batch processing when interruptions occur. Such interruptions are common for processes executed by human resources [27], e.g., when a resource needs a break or a set-up for particular cases is required [23].

The output of this step is a report listing the activities where the batch processing occurs. For these activities, we identify their case frequency (i.e., the number of cases in which an activity is processed w.r.t. the total number of cases of the process) and batch processing frequency (i.e., the number of times an activity is processed as part of a batch w.r.t. its total number of executions). The list of batch processing activities is sorted by the batch processing frequency in descending order. An analyst can decide which batch processing activities to analyze further based on the frequencies. For instance, the list might have an activity with only 60% batch-processed cases, but the case frequency might be close to 100%. Therefore, the improvement of such batch processing would target 60% of all cases. On the other hand, the improvement of the activity with batch processing frequency close to 100% (all cases are batch-processed) and case frequency of 5% would affect a relatively small number of cases and thus, the overall impact would be negligible. Therefore, a combination of batch processing frequency and case frequency indicates the impact of the batch processing activity. Based on this information, the analyst can determine which batch processing activities to analyze further.

3.2 Batch Processing Behavior Analysis

For each identified batch processing activity, we discover its behavior, i.e., the characteristics of how the batch is processed. We describe batch processing behavior with batch processing frequency, type, size, and activation rule/-s. *Batch processing frequency* indicates the proportion of cases of an activity that are processed in a batch w.r.t. its total executions. *Batch size* illustrates the number of cases processed in a batch. The batch size can be constant (e.g., batch size = 5 cases in 100% of cases) or variable (e.g., batch size = 5 in 40% of cases, batch size = 6 in 60% of cases). *Batch processing type* describes how the cases are batch-processed. In this research, we discover five batch processing types defined by Martin et al. [23]. Parallel batch processing is when all cases are processed at the same time. In a sequential task-based batch processing, one activity is executed for all the cases in the batch, one after another. In sequential case-based batch processing, a set of activities are sequentially executed for each

¹ We note that other notions of concurrency have been proposed in the field of process mining. An in-depth treatment of concurrency notions in process mining is provided in Abel Armas-Cervantes et al. [3].

case in a batch. Concurrent task-based batch processing is when one activity is executed for all cases, but with a partial overlap in case processing. In concurrent case-based batch processing, multiple activities are performed for each case with partial processing time overlap [23].

We also identify *batch activation rule*, i.e. the condition(s) that trigger the batch processing [24]. In the batch activation rule discovery, we follow the approach and batch activation rule types proposed by Martin et al. [24]. The activation rules can be volume-based (batch processing is triggered when a certain volume, weight, or number of cases are accumulated [5, 11, 17, 22, 29, 30]), time-based (batch processing is time-triggered, e.g., by scheduled date and time [17, 36], when a case reaches a waiting time limit [17, 28]), resource-based (batch processing activation is dependent on the resource attributes, e.g., the batch is processed when the resource workload allows processing this amount of work [24]), case-based (batch processing is initiated based on the case-related attributes, e.g., arrival of the case of a particular type such as emergency or high-priority case [28]), or context-based (rules embracing other aspects out of the process scope such as meteorological conditions [24]).

Batch can be activated with a single rule, multiple rules, or on an ad-hoc basis. A single rule consists of one condition to be fulfilled to initiate batch processing, e.g., accumulated orders are shipped on scheduled date [17]. Multiple rules present a combination of different types of rules are used to activate a batch processing [28, 35], e.g., the combination of volume and time-based rules [9, 10, 17]: accumulated orders are shipped based either on the scheduled date or when certain weight, volume, or number of cases are collected [17]. However, when no specific pattern that confidently correlates with the start of batch processing can be identified (i.e., no activation rule is assigned [31]), it is assumed that the batch processing is determined by the resources, i.e., executed ad-hoc. In our approach, the batch activation rules are elicited using the RIPPER technique². The quality of the discovered batch activation rule is measured by support and confidence parameters. The output of this step is a report detailing the behavior of each selected batch processing activity. This information serves as an input for the next step and can be used later by the analyst to identify what parameters to alter to improve batch processing.

3.3 Waiting Time Analysis of Batch Processing Activities

In this section, we describe the characteristics of the five batch processing types considered by our approach, and the waiting times they might induce. We determine three types of waiting times associated with the batch processing – waiting time for batch accumulation, waiting time of a ready batch, and waiting time to process other cases of the batch.

Parallel Batch Processing. In parallel batch processing, a resource starts and completes processing all cases of a batch at the same time [23]. First cases

² <https://github.com/imoscovitz/wittgenstein.git>.

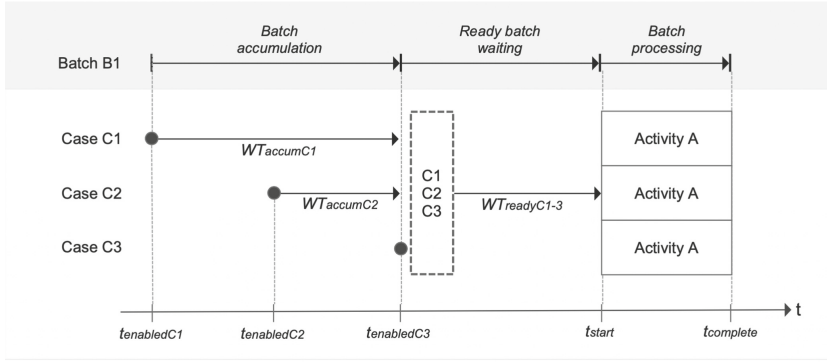


Fig. 2. Waiting times in parallel batch processing

are accumulated, and once accumulated, they can be processed. There are two waiting time types in parallel batch processing (Fig. 2). The first is *waiting time for batch accumulation*, i.e., the time it takes to build up a batch (WT_{accum} in Fig. 2). When all the cases are accumulated, a batch is ready to be processed. But a batch might not be processed as soon as it is ready. Thus, there is a second waiting time - *waiting time of a ready batch* (WT_{ready} in Fig. 2).

To identify these waiting times from the event log, we need enabled time (when a case becomes available for processing), start time (when a case starts being processed), and completion time (when a case processing is finished) [12]. The enabled time is often the same as the completion time of the preceding activity. However, they might differ where parallel activity execution occurs. Therefore, when there are two parallel activities, both of which must be completed before the processing can continue, enabled time is the completion time of the latter activity [8]. Thus, once enabled time is recorded, the case is ready to be processed ($t_{enabled}$ in Fig. 2).

Waiting time for batch accumulation is the time cases wait before a batch is accumulated and ready to be processed. A batch is accumulated when all the included cases are available for processing. Therefore, a batch is accumulated when the last-arriving case in the batch is enabled. Until the enablement of the last case, all earlier cases wait for the batch to be accumulated. Thereby, *waiting time for batch accumulation* ($WT_{accumC1}$, $WT_{accumC2}$ in Fig. 2) is calculated as the difference between the enabled time of the cases in a batch and the enabled time of the last-arrived case. In Fig. 2, a batch is activated when three cases have been accumulated. Cases (see Cases C1, C2) wait until the batch is ready to be processed (accumulated). Their waiting time is, therefore, from their enabled times (see $t_{enabledC1}$, $t_{enabledC2}$) until the enabled time of the last case included in the batch, i.e., the third case (see $t_{enabledC3}$). The earlier a case arrives, the longer it waits. The last-arriving case (see Case C3) has no waiting time for batch accumulation since its arrival marks the batch as ready to be processed.

Waiting time of a ready batch (see $WT_{readyC1-C3}$ in Fig. 2), on the other hand, is the time from when the cases are ready to be processed (batch accumulated)

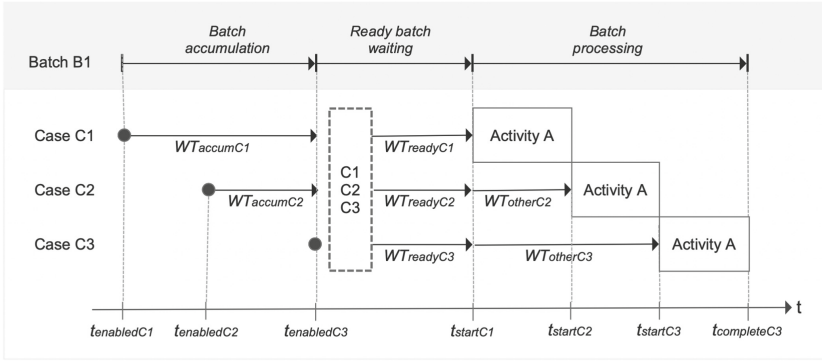


Fig. 3. Waiting times in sequential task-based batch processing

until they are processed (batch processing starts). Therefore, it is calculated as the difference between the enabled time of the last-arrived case ($t_{enabledC3}$) and the start time of the batch processing (t_{start}).

Sequential Batch Processing. In sequential batch processing, a resource accumulates a group of cases and processes them one after another. In sequential task-based batch processing (Fig. 3), the same activity is executed for all cases, one by one. However, in sequential case-based batch processing (Fig. 4), several activities are performed for each case in a batch [23].

Sequential task-based (Fig. 3) and case-based (Fig. 4) batch processing have, similar to parallel batch processing, *waiting time for batch accumulation* and *waiting time of a ready batch*. But sequential task-based and case-based batch processing also have *waiting time for other cases to be processed* within the batch processing, i.e., the time when a case waits while work is done on other cases in the same batch. This waiting time occurs when the resource is processing some case/-s in the batch (e.g., Case C1 in Fig. 3), other cases have to wait for their turn ($WT_{otherC3}$ for Case C2 and $WT_{otherC3}$ for Case C3 in Fig. 3).

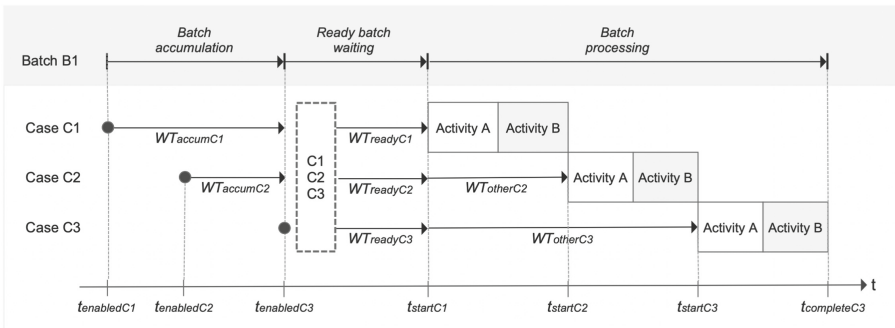


Fig. 4. Waiting times in sequential case-based batch processing

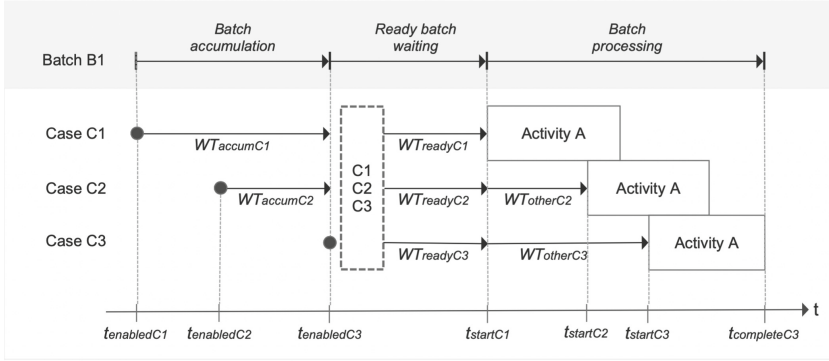


Fig. 5. Waiting times in concurrent task-based batch processing

Waiting time for other cases to be processed is calculated as the difference (interval) between the start time of the first case/s ($t_{startC1}$ in Fig. 3) in a batch and the start time of the other case/s ($t_{startC2}$ and $t_{startC3}$ in Fig. 3). In a sequential case-based batch processing (Fig. 4), several consecutive activities are performed for each case of a batch [23]. Waiting time types in sequential case-based batch processing correspond to those in sequential task-based batch processing (see Fig. 4).

Concurrent Batch Processing. Concurrent batch processing is when cases are processed sequentially but with partial time overlap. Concurrent task-based batch processing (Fig. 5) is to execute the same activity for all cases in the batch with an overlap in processing time. In contrast, for the case-based type (Fig. 6), multiple activities are performed for each case with an overlap in processing time [23]. In concurrent task-based (Fig. 5) and case-based (Fig. 6) batch processing, the waiting times are the same as sequential batch processing. Although there is an overlap in processing time, the cases still have to wait for their turn to be processed. Therefore, *waiting time for other cases to be processed* occurs (e.g., $WT_{otherC2}$, $WT_{otherC3}$ in Fig. 5). Given the similar activity processing times, the difference between these batch processing types lies in a shorter *waiting time for other cases to be processed* due to processing time overlaps.

Waiting Time Analysis. We analyze discovered waiting times to identify inefficiencies. We, first, determine how long cases wait on average before they are batch-processed. Then we examine the batch processing efficiency by calculating its CTE, i.e., the ratio of the processing time to the cycle time of the batch processing activity/-ies. Finally, we measure the impact of each waiting time type on batch processing efficiency. We express the impact as the potential CTE improvement, i.e., how much improvement of the CTE can be obtained if a particular waiting time type is eliminated.

The first step is to examine the average waiting time per type. We measure, for each waiting time type, the *average waiting time for batch accumulation*, *average waiting time of a ready batch*, *average waiting time for other cases to be*

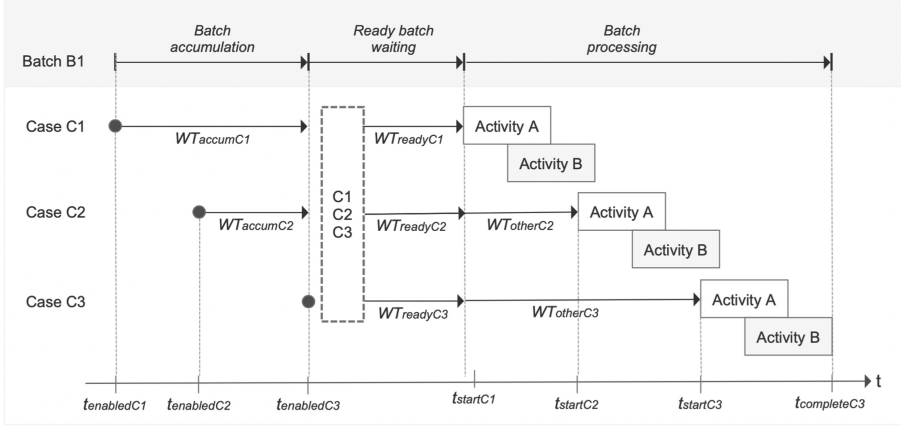


Fig. 6. Waiting times in concurrent case-based batch processing

processed, and average total waiting time. These metrics depict how long cases wait during batch accumulation and processing.

The analysis of waiting times provides the input required to assess the impact of discovered waiting time types on the batch processing CTE. It enables the discovery of batch processing inefficiencies from event logs. The impact of each waiting time (CTE_{im}) on the batch processing CTE is measured as the difference between the CTE of the batch processing activity/-ies and the CTE if that particular waiting time is eliminated. Thus, this metric illustrates the potential CTE improvement if that particular waiting time is eliminated. Thus, if the CTE of a batch processing activity is 50% and, for instance, after eliminating its *waiting time of a ready batch*, the CTE increases to 70%, the potential improvement opportunity of addressing this activity is $CTE_{im} = 40\%$.

The output of this step is a report that presents discovered waiting times per type and their impact on the batch processing activity CTE (CTE_{im}). Process analysts can use the obtained results to identify batch processing inefficiencies and from where (what particular waiting time type) the inefficiencies stem. Thus, the results can aid the analyst in identifying which batch processing activities to focus on in their improvement initiatives and how they can be improved.

4 Evaluation

In this section, we present the evaluation of our approach. First, we use a synthetic event log with artificially injected batches. With this experimentation, we validate the ability of our technique to discover batching behavior and analyze the different types of waiting times. Second, we apply our approach to a real-life event log to demonstrate its applicability in real-life scenarios. The implementation of the approach, as well as the event logs and results of the experiments, are available on GitHub³.

³ <https://github.com/AutomatedProcessImprovement/batch-processing-analysis>.

4.1 Experiments with Synthetic Data

For the synthetic evaluation, we artificially added batches to a simulated event log of a loan application process with the specific purpose of evaluating if our technique was able to detect them. To do this, we first grouped the activity instances of three activities of the process. Then, we assigned a new resource to each group to ensure that the activity instances of a batch did not share their resource with other activity instances. Finally, we delayed the start and end timestamps of the activity instances of each group (as well as the timestamps of succeeding activity instances) to force their execution as a batch.

To increase the variety in the synthetic evaluation, we added batches of different sizes (10, 12, and 14) and types (parallel, sequential, and concurrent) to three different activities. We left some activity instances unaltered, so not all their executions were processed in a batch. Furthermore, for some batches, we designed the batch activation to be performed based on temporal rules, e.g., at a specific time of a specific day of the week. Finally, the waiting times were also altered to create batches with different characteristics, e.g., with or without the waiting time of a ready batch.

We evaluated our approach with this event log, resulting in the discovery of all artificially added batches. Moreover, due to the displacement of the batch processing activities and their successors, batch processing was also detected in other activities. The discovered activation rules corresponded to the designed temporal constraints, indicating the day of the week and hour of activation. Finally, the performance analysis detected the correct waiting times and CTE, accurately reporting cases in which some waiting times were set to 0.

4.2 Experiments with a Real-Life Log

This section demonstrates how batch processing inefficiencies can be identified by applying our approach to a real-life event log of a manufacturing production process [21]. This event log has 225 traces, 26 activities, 4953 events, 48 resources and does not contain multitasking (i.e. the resources do not work in more than a task at the same time) [15].

Batch Processing Discovery. We applied our approach to a real-life event log. First, we discovered batch processing activities from the event log. We set the minimum batch size threshold to two cases with no in-batch time gaps (intervals between the case processing within a batch) allowed. The output of this step is a report showing each batch processing activity, their case frequency, and batch processing frequency (see Table 1). From the report, we see that, for this event log, 12 activities had batch processing. The activities are sorted in descending order based on batch processing frequency. For each activity, the report shows the case frequency and the batch processing frequency. From the report (Table 1), we note that activities “Lapping”, “Packing”, and “Turning Rework”, have high batch processing frequencies (91.07%, 83.75%, and 66.67% respectively). These cases are predominantly processed in batches and, therefore, relevant for batch efficiency analysis. However, the “Turning Rework” activity is executed for a

Table 1. Batch processing discovery results.

Activity	Case frequency	Batch processing frequency
Lapping	58.67%	91.07%
Packing	77.78%	83.75%
Turning rework	1.33%	66.67%
Turning & Milling	72.00%	45.78%
Final Inspection Q.C.	78.22%	35.64%
Turning & Milling Q.C.	75.11%	27.78%
Grinding rework	14.67%	23.71%
Laser marking	74.22%	18.25%
Round grinding	49.33%	14.08%
Turning Q.C.	12.00%	10.91%
Flat grinding	26.22%	7.02%
Turning	10.67%	2.35%

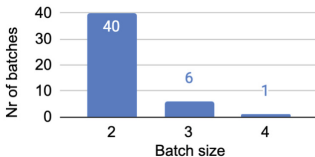
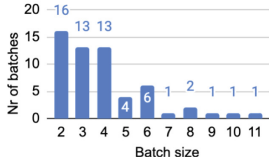
comparatively negligent amount of cases (for 1.33% of total cases) and, therefore, has little impact on the process efficiency and is not considered for further analysis. Thus, the report shows that “Lapping” and “Packing” have high batch processing and case frequencies, and therefore, an analyst might decide to select these as an input for the next step.

Batch Processing Analysis. The next step is to analyze the behavior of selected batch processing activities. The output of this step is a report that captures batch processing type, activation rules, and batch size distribution for each batch processing activity (see Table 2). As can be seen from Table 2, the batch processing activity named “Lapping” follows parallel, sequential task-based, and concurrent task-based batch processing types, i.e., in this activity cases can be processed at the same time, sequentially one after another, with or without an overlap in processing time. It indicates that there is no specific style of activity execution. The activity “Packing”, on the contrary, follows only a parallel batch processing type, i.e., cases in a batch are processed simultaneously.

Next, we discover the batch activation rules. For instance, for “Lapping”, batch processing occurs from 4 to 6 h. This rule has a confidence of 0.94 and a support of 0.12. Finally, the report (Table 2) also captures the discovered batch size distribution per activity. Batch size distribution describes the number of cases that are processed in one batch. In this case, for “Lapping”, the batch size distribution is 2 to 4 cases. Most commonly, in 85% of batch processing, batches include 2 cases.

Waiting time analysis. Having discovered the batch processing behavior, we focus on quantitatively analyzing the waiting times. The output for this step is a report that measures average waiting times per type and their impact on batch processing CTE (Table 3). For each activity, the average processing time (*PT*

Table 2. Batch processing analysis results.

Batch Processing Activity	Lapping	Packing																														
Batch Processing Type	Parallel, Sequential task-based, Concurrent task-based	Parallel																														
Activation Rules	[<i>hour</i> = 4.0 – 5.0]	[<i>hour</i> ≤ 5.0]																														
Activation Rules Quality	<i>Conf.</i> = 0.94, <i>Supp.</i> = 0.12	<i>Conf.</i> = 0.92, <i>Supp.</i> = 0.34																														
Batch Size Distribution	 <table><tr><th>Batch size</th><th>Nr of batches</th></tr><tr><td>2</td><td>40</td></tr><tr><td>3</td><td>6</td></tr><tr><td>4</td><td>1</td></tr></table>	Batch size	Nr of batches	2	40	3	6	4	1	 <table><tr><th>Batch size</th><th>Nr of batches</th></tr><tr><td>2</td><td>16</td></tr><tr><td>3</td><td>13</td></tr><tr><td>4</td><td>13</td></tr><tr><td>5</td><td>4</td></tr><tr><td>6</td><td>6</td></tr><tr><td>7</td><td>1</td></tr><tr><td>8</td><td>2</td></tr><tr><td>9</td><td>1</td></tr><tr><td>10</td><td>1</td></tr><tr><td>11</td><td>1</td></tr></table>	Batch size	Nr of batches	2	16	3	13	4	13	5	4	6	6	7	1	8	2	9	1	10	1	11	1
Batch size	Nr of batches																															
2	40																															
3	6																															
4	1																															
Batch size	Nr of batches																															
2	16																															
3	13																															
4	13																															
5	4																															
6	6																															
7	1																															
8	2																															
9	1																															
10	1																															
11	1																															

in Table 3) and the average waiting times per type are calculated (WT_{accum} , WT_{ready} , WT_{other} in Table 3).

The waiting times show how long, on average, cases wait for the batch to be accumulated and then processed. For instance, cases wait on an average of 4d,5h,7m while the batch is accumulated for “Lapping”. Then, once the batch is accumulated, the cases wait for 2d,14h,20m before the batch processing starts. When cases are processed sequentially or concurrently, these cases wait for an additional 34m on average while other cases are processed. The existing batch processing strategy, therefore, results in a CTE of 1.10% (CTE_b). The CTE value for this activity indicates a potential improvement opportunity. Table 3 shows the impact on the CTE if the waiting times are eliminated. For instance, for “Lapping”, if the waiting time for other cases to be processed is eliminated, a slight improvement will be achieved, but the CTE will remain at 1.1% ($CTE_{im3} = 0.0\%$). However, if analysts focus on reducing the waiting time of a ready batch, the CTE could be increased up to 1.78%, which corresponds to an improvement of $CTE_{im2} = 62\%$. The most significant CTE improvement can be achieved if the efforts are focused on reducing the waiting time for batch accumulation (WT_{accum}) that could improve CTE up to 2.78% ($CTE_{im1} = 153\%$).

Table 3. Waiting time analysis results.

Batch processing activity	PT	WT_{accum}	CTE_{im1}	WT_{ready}	CTE_{im2}	WT_{other}	CTE_{im3}	CTE_b
Lapping	0d,1h,50m	4d,4h,39m	153%	2d,15h,31m	62%	34m	0%	1.1%
Packing	0d,1h,0m	10d,8h,34m	267%	3d,17h,56m	33%	0d,0h,0m	0%	0.3%

The analysis demonstrates that attempts to reduce WT_{other} by changing the batch processing type (e.g., use only parallel batch processing in “Lapping”) will not add much value. The analysis also shows the reason, i.e., that almost all the waiting time is related to the accumulation and waiting time of a ready batch. We also note from the analysis that the greatest improvements can be achieved by addressing the waiting times for accumulation for “Lapping” and “Packing”. This is due to, as the analysis shows, cases being processed relatively fast w.r.t. the time they spend waiting in for accumulation. Based on the analysis, the analyst can consider discarding batch processing in favor of processing cases individually when they are available for these two activities.

5 Conclusion

In this paper, we propose an approach for analyzing event logs to identify improvement opportunities in processes with batch processing activities. In addressing RQ1, we define three types of waiting times associated with batch processing, namely *waiting time for batch accumulation* and *waiting time of a ready batch* in parallel, sequential, and concurrent batch processing, and *waiting time for other cases to be processed* in sequential and concurrent batch processing. We use these definitions to identify waiting times from event logs.

In our approach, we first discover different types of batch processing from an event log. For this, we extend existing research by also considering the waiting times associated with batch processing and discovering their impact on the CTE of batch processing activities (RQ2). Finally, we identify batch processing inefficiencies by measuring the impact of batch processing waiting times on the activity CTE. This enables analysts to identify where there are improvement opportunities and where to target the process changes (RQ3). We evaluated the approach with synthetic data and a real-life event log. The evaluation indicates that our approach can provide analysts with insights on potential batch processing inefficiencies. Thus, the analysts can take a data-driven approach to improve batch processing efficiency.

As a result of the experimentation, we detected potential improvements in the approach. Our approach uses all observations available per batch for the discovery of batch activation rules and reports on confidence and support. This process can be improved by establishing training/test partitions to discover and validate the rules. The approach applicability is, however, limited to the event logs that have data on the resources, start and end timestamps. If the log miss any of these data, the approach cannot be executed and the analyst is informed accordingly. In addition, the non-working periods of resources are currently part of the measured waiting times. Our approach could be extended by adding calendar information to improve the accuracy of both measures. Finally, the evaluation could be extended by including other synthetic event logs and a larger real-life event log with multitasking.

As future work, we plan to implement a what-if simulation analysis for batch processing activities to identify the impact of particular changes on the CTE.

This would allow analysts to change the parameters of batch processing activities and explore what changes would improve batch processing performance and by how much.

Acknowledgments. This research is funded by the European Research Council (PIX Project).

References

1. van der Aalst, W.M.P.: *Process Mining - Data Science in Action*. 2nd edn. Springer, Cham (2016). <https://doi.org/10.1007/978-3-662-49851-4>
2. Andrews, R., Wynn, M.: Shelf time analysis in CTP insurance claims processing. In: Kang, U., Lim, E.-P., Yu, J.X., Moon, Y.-S. (eds.) *PAKDD 2017. LNCS (LNAI)*, vol. 10526, pp. 151–162. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67274-8_14
3. Armas-Cervantes, A., Dumas, M., Rosa, M.L., Maaradji, A.: Local concurrency detection in business process event logs. *ACM Trans. Internet Tech.* **19**(1), 16:1–16:23 (2019)
4. Arora, S., Rudnisky, C.J., Damji, K.F.: Improved access and cycle time with an “in-house” patient-centered teleglaucoma program versus traditional in-person assessment. *Telemed. e-Health* **20**(5), 439–445 (2014)
5. Baba, Y.: A bulk service gi/m/1 queue with service rates depending on service batch size. *J. Operat. Res. Soc. Japan* **39**(1), 25–35 (1996)
6. Burattin, A., Sperduti, A., Veluscek, M.: Business models enhancement through discovery of roles. In: *IEEE Symposium on Computational Intelligence and Data Mining, CIDM*, pp. 103–110. IEEE (2013)
7. Cachon, G., Terwiesch, C.: *Matching Supply with Demand*. McGraw-Hill Publishing, New York (2008)
8. Camargo, M., Dumas, M., González, O.: Automated discovery of business process simulation models from event logs. *Decis. Support Syst.* **134** (2020). Article no. 113284
9. Çetinkaya, S.: Coordination of inventory and shipment consolidation decisions: a review of premises, models, and justification. *Applications of supply chain management and e-commerce research*, pp. 3–51 (2005)
10. Cigolini, R., Perona, M., Portioli, A., Zambelli, T.: A new dynamic look-ahead scheduling procedure for batching machines. *J. Schedul.* **5**(2), 185–204 (2002)
11. Claeys, D., Walraevens, J., Laevens, K., Bruneel, H.: A queueing model for general group screening policies and dynamic item arrivals. *Eur. J. Oper. Res.* **207**(2), 827–835 (2010)
12. Delgado, A., Weber, B., Ruiz, F., de Guzmán, I.G.-R., Piattini, M.: Continuous improvement of business processes realized by services based on execution measurement. In: Maciaszek, L.A., Zhang, K. (eds.) *ENASE 2011. CCIS*, vol. 275, pp. 64–81. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-32341-6_5
13. Delias, P.: A positive deviance approach to eliminate wastes in business processes: the case of a public organization. *Ind. Manag. Data Syst.* **117**(7), 1323–1339 (2017)
14. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: *Fundamentals of Business Process Management*. Springer, Cham (2013). <https://doi.org/10.1007/978-3-642-33143-5>

15. Estrada-Torres, B., et al.: Discovering business process simulation models in the presence of multitasking and availability constraints. *Data Knowl. Eng.* **134** (2021). Article no. 101897
16. Henn, S., Koch, S., Wäscher, G.: Order batching in order picking warehouses: a survey of solution approaches. In: Manzini, R. (eds.) *Warehousing in the Global Supply Chain*, pp. 105–137. Springer, Cham (2012). https://doi.org/10.1007/978-1-4471-2274-6_6
17. Higginson, J., Bookbinder, J.H.: Policy recommendations for a shipment-consolidation program. *J. Busi. Logist.* **15**(1) (1994)
18. Ignizio, J.P.: The impact of operation-to-tool dedications on factory stability. In: *Proceedings of the 2010 Winter Simulation Conference, WSC 2010*, pp. 2606–2613. IEEE (2010)
19. Klijn, E.L., Fahland, D.: Performance mining for batch processing using the performance spectrum. In: Di Francescomarino, C., Dijkman, R., Zdun, U. (eds.) *BPM 2019. LNBIP*, vol. 362, pp. 172–185. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-37453-2_15
20. Kren, L., Tyson, T.: Using cycle time to measure performance and control costs in focused factories. *J. Cost Manage.* **16**(6), 18–23 (2002). Article no. 101897
21. Levy, D.: Production analysis with process mining technology (2014). <https://doi.org/10.4121/uuid:68726926-5ac5-4fab-b873-ee76ea412399>
22. Maity, A., Gupta, U.C.: Analysis and optimal control of a queue with infinite buffer under batch-size dependent versatile bulk-service rule. *Opsearch* **52**(3), 472–489 (2015). <https://doi.org/10.1007/s12597-015-0197-6>
23. Martin, N., Pufahl, L., Mannhardt, F.: Detection of batch activities from event logs. *Inf. Syst.* **95** (2021). Article no. 101642
24. Martin, N., Solti, A., Mendling, J., Depaire, B., Caris, A.: Mining batch activation rules from event logs. *IEEE Trans. Serv. Comput.* **14**(6), 1837–1848 (2021)
25. Martin, N., Swennen, M., Depaire, B., Jans, M., Caris, A., Vanhoof, K.: Retrieving batch organisation of work insights from event logs. *Decis. Support Syst.* **100**, 119–128 (2017)
26. Nakatumba, J.: Resource-aware business process management: analysis and support. Ph.D. thesis, Mathematics and Computer Science (2013)
27. Pika, A., Ouyang, C., ter Hofstede, A.: Configurable batch-processing discovery from event logs. *ACM Trans. Manage. Inf. Syst.* **13**(3) (2021)
28. Pufahl, L.: Modeling and executing batch activities in business processes. Ph.D. thesis, Universität Potsdam (2018)
29. Pufahl, L., Bazhenova, E., Weske, M.: Evaluating the performance of a batch activity in process models. In: Fournier, F., Mendling, J. (eds.) *BPM 2014. LNBIP*, vol. 202, pp. 277–290. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15895-2_24
30. Pufahl, L., Weske, M.: Batch activities in process modeling and execution. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) *ICSOC 2013. LNCS*, vol. 8274, pp. 283–297. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45005-1_20
31. Pufahl, L., Weske, M.: Batch activity: enhancing business process modeling and enactment with batch processing. *Computing* **101**(12), 1909–1933 (2019). <https://doi.org/10.1007/s00607-019-00717-4>
32. Reijers, H.A., Mansar, S.L.: Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics. *Omega* **33**(4), 283–306 (2005)

33. Rohleder, T.R., Silver, E.A.: A tutorial on business process improvement. *J. Operat. Manage.* **15**(2), 139–154 (1997). Article no. 101897
34. Selvarajah, E., Steiner, G.: Approximation algorithms for the supplier's supply chain scheduling problem to minimize delivery and inventory holding costs. *Oper. Res.* **57**(2), 426–438 (2009)
35. Sha, D., Hsu, S.Y., Lai, X.: Design of due-date oriented look-ahead batching rule in wafer fabrication. *Int. J. Adv. Manuf. Technol.* **35**(5), 596–609 (2007)
36. Simons, J.V., Jr., Russell, G.R.: A case study of batching in a mass service operation. *J. Operat. Manage.* **20**(5), 577–592 (2002). Article no. 101897
37. Venkatraman, S., Venkatraman, R.: Process innovation and improvement using business object-oriented process modelling (BOOPM) framework. *Appl. Syst. Innovat.* **2**(3), 23 (2019). Article no. 101897
38. Verenich, I., Dumas, M., La Rosa, M., Maggi, F.M., Di Francescomarino, C.: Minimizing overprocessing waste in business processes via predictive activity ordering. In: Nurcan, S., Soffer, P., Bajec, M., Eder, J. (eds.) *CAiSE 2016. LNCS*, vol. 9694, pp. 186–202. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39696-5_12
39. Waibel, P., Novak, C., Bala, S., Revoredo, K., Mendling, J.: Analysis of business process batching using causal event models. In: Leemans, S., Leopold, H. (eds.) *ICPM 2020. LNBIP*, vol. 406, pp. 17–29. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-72693-5_2
40. Weijters, A.J.M.M., Ribeiro, J.T.S.: Flexible heuristics miner (FHM). In: *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2011*, pp. 310–317. IEEE (2011)
41. Wen, Y., Chen, Z., Liu, J., Chen, J.: Mining batch processing workflow models from event logs. *Concurr. Comput. Pract. Exp.* **25**(13), 1928–1942 (2013)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

