ECONSTOR
Make Your Publications Visible.

A Service of

ZBW Leibniz-Informationszentrum Wirtschaft
Leibniz Information Centre for Economics

Li, Chiao-Yun et al.

Article — Published Version

Cargo distribution analysis: a process mining approach

Process Science

Provided in Cooperation with:
Springer Nature

Suggested Citation: Li, Chiao-Yun et al. (2026) : Cargo distribution analysis: a process mining approach, Process Science, ISSN 2948-2178, Springer International Publishing, Cham, Vol. 3, Iss. 1, https://doi.org/10.1007/s44311-026-00038-8

This Version is available at:
https://hdl.handle.net/10419/336716

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

Terms of use:

Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.

https://creativecommons.org/licenses/by/4.0/

WWW.ECONSTOR.EU

Mitglied der Leibniz-Gemeinschaft

Li *et al. Process Science* (2026) 3:4
https://doi.org/10.1007/s44311-026-00038-8

Process Science

RESEARCH Open Access

Cargo distribution analysis: a process mining approach

Chiao-Yun Li[1*], Anton Antonov[1], Tsung-Hao Huang[1], Marco Pegoraro[1], Sean Shing Fung Lau[2], Morgan Xian Biao Hiew[2], Ivy Yan Nei Law[2], Benny Drescher[2,3] and Wil M. P. van der Aalst[1]

*Correspondence:
Chiao-Yun Li
chiaoyun.li@pads.rwth-aachen.de
[1]Chair of Process and Data Science, RWTH Aachen University, Ahornstraße 55, 52074 Aachen, Germany
[2]Hong Kong Industrial Artificial Intelligence and Robotics Centre Limited, Kowloon Tong, NT, Hong Kong
[3]INC Innovation Center GmbH, Munich, Germany

Abstract
Air cargo terminals are indispensable hubs in the global trade network, yet inefficiencies in their cargo handling processes can trigger significant disruptions, manifesting as costly delays, inflated operational expenses, and a decline in service quality, thereby jeopardizing intricate supply chains. To gain a comprehensive understanding of operational efficiency, we leverage process mining to analyze event data that track the transportation journey of each cargo item from event-driven systems. Following the process mining project methodology, we present an end-to-end pipeline that diagnoses bottlenecks within the cargo distribution process and predicts future distribution flows based on historical event data. Real-world event data are often characterized by inherent complexities, presenting challenges in data quality and scalability. Addressing these challenges, our standardized effort focuses on processing event data from supporting information systems to automatically detect and diagnose the root causes of inefficiencies to deliver actionable, data-driven insights. These insights are integrated into an interactive GUI that supports decision-making, enabling proactive measures and streamlining operations.

Keywords Process mining project methodology, Root cause analysis, Data processing, Next activity prediction, Logistics

Introduction

As air cargo plays a vital role in facilitating international trade, timely and reliable operations at these terminals directly influence logistics costs, transit times, and service quality (Hammervoll 2009; Murphy et al. 1989; Yu and Zou 2022). Inefficient cargo handling can lead to delays, missed connections, and higher operational costs (Hoffman 2019; Liu et al. 2019). Therefore, understanding terminal performance is critical for maintaining the competitiveness of air cargo in the global supply chain.

To elevate business process operations, industries are embracing *smart logistics*, leveraging IoT technologies for real-time tracking and monitoring (Uckelmann 2008). This reliance on continuous and responsive data flows drives the adoption of *Event-Driven Systems* (EDS), which capture real-time snapshots of *activities* (i.e., well-defined process steps) by enabling IoT devices to generate *event data* in response to specific incidents,

© The Author(s) 2026. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, 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 licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence 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. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

Springer

such as sensor readings or user interactions (Michelson 2006). While these systems provide valuable insights into particular moments to monitor the current state of organizational processes, they often lack a complete view of process instances for reliable analysis. The generated event data may include noise, such as erroneous readings; inconsistencies, such as conflicting information from different sensors; and gaps, such as missing data due to communication failures (Bertrand et al. 2022). These issues make it challenging to draw accurate conclusions and make informed decisions. In contrast, process mining techniques facilitate the analysis of workflows by examining historical event data to uncover patterns, inefficiencies, and bottlenecks (van der Aalst 2016). By leveraging process mining, organizations achieve a more comprehensive understanding of their operational processes, enabling data-driven decision-making and process enhancement.

However, applying process mining to highly automated systems presents unique challenges. In systems like air cargo terminals, where processes are largely automated and governed by a fixed physical network (e.g., conveyor belts, sorting devices), the traditional process mining focus on process discovery is often less relevant. Instead, the core challenge lies in ensuring that the recorded event data accurately reflects the behavior within the physical network. The vast scale of event data generated by IoT devices also presents significant analytical hurdles. Simply applying existing process mining algorithms as a *case-driven demonstration* is insufficient to address these specific problems.

To tackle these challenges, this paper presents a novel methodology for the analysis of automated cargo distribution processes. We adapt the conformance checking technique to specifically align sensor data with the physical network structure of the distribution system. This allows us to:

1. Enhance Data Quality: We use network-based conformance checking to identify and filter out inconsistent events that do not conform to the valid pathways of the system.
2. Enable Scalable Analysis: Our approach enables the analysis of large volumes of event data by focusing on deviations from the network model, making root cause analysis more targeted and efficient.
3. Detect and Diagnose Bottlenecks: By analyzing the patterns of non-conformance and performance deviations, we pinpoint the specific root causes of delays and operational variability within the automated distribution process.

Our methodology is presented within the structured framework of the *Process Mining Project Methodology* (van Eck et al. 2015), which serves as a guiding structure for our approach rather than a mere list of steps. We detail the unique adaptations and contributions we made at each stage: from the initial project planning where we defined specific research questions for an automated environment, to the data processing steps that formalize the case notion and event types, and most critically, to the mining and analysis phase where our novel conformance checking technique is applied. This work answers the following generalizable research questions through our case study:

- How can sensor data collected from IoT be transformed into event data for process mining in automated, network-constrained systems?
- What insights into logistics performance can be derived from aligning event data with a system's underlying network structure?

- What is the role of data-driven insights—specifically from analyzing process deviations and performance—in supporting real-time and strategic decision-making in complex logistical systems?

Ultimately, our methodology provides a repeatable and systematic approach that can be adopted as a foundational standard for applying process mining to automated logistics, thereby bridging the gap between theoretical process mining and its practical application in complex industrial settings.

The remainder of this paper is structured as follows. Section 2 provides a review of the relevant literature. Section 3 presents the preliminaries. Section 4 introduces event data and other relevant datasets describing a distribution system. Section 5 formalizes the methodology, followed by the implementation based on the case study in Sect. 6. Section 7 discusses the threats to validity of our approach and the limitations for the general applicability. Finally, Sect. 8 summarizes the work.

Related work

Logistics industries are increasingly adopting Event-Driven Systems (EDS) to improve operational processes (Buchmann et al. 2010). However, event data collected by sensors in these systems often faces data quality issues (Bertrand et al. 2022), and retrofitting these systems to address these challenges can be prohibitively expensive (Lin et al. 2019). Meanwhile, process mining is a technology to derive insights from event data. In (van Eck et al. 2015), the authors provide a structured framework for analyzing event data through process mining. This section introduces related work on the analysis of event data in logistics, organized into Sects. 2.1, 2.2 and 2.3, following the framework outlined by (van Eck et al. 2015).

Event data quality and repair

Event data extracted from EDS often suffers from data quality issues (Mansouri et al. 2023; Rocha and Oliveira Rocha 2022; Melo and Aquino 2021; Li et al. 2023). For reliable process mining insights, event logs must ensure accuracy and validity. Process mining typically relies on three key elements: a case identifier, an activity label, and a timestamp. In logistics, assuming the case identifier is provided, log repair focuses on handling missing events, correcting activity sequences, and adjusting inaccurate timestamps.

Several approaches address timestamp and ordering corrections. Fischer et al. 2020 detect various timestamp inconsistencies but do not provide repair techniques. Conforti et al. 2018 repair events that incorrectly share the same timestamp by sampling corrected timestamps from activity-specific distributions. Dixit et al. (Dixit et al. 2018) apply a semi-automated method to detect and repair ordering imperfections, using indicators like timestamp granularity and order anomalies.

Other works focus on repairing missing events. Rogge-Solti et al. 2013 derive missing events by aligning logs with stochastic Petri nets and sampling timestamps from a Bayesian network. Song et al. (Song et al. 2015) reconstruct incomplete traces via minimal trace recovery but struggle when many events are missing. Liu et al. (Liu et al. 2021) cluster complete traces using self-organizing maps to infer missing activities in incomplete ones. Lu et al. (Lu et al. 2022) use LSTMs to correct missing activities via next-activity prediction.

Process mining analysis in logistics

Following event data repair, research focuses on applying process mining techniques to logistics systems (Alnahas 2023). In the area of performance and bottleneck analysis, Bemthius et al. 2021 classify bottleneck analysis techniques, while Li et al. (Li et al. 2023) apply bottleneck root cause detection to air cargo terminals. Specific to airport operations, Skorupski et al. (Skorupski et al. 2018) evaluate throughput in baggage hold security screening (HBSS) systems by employing a microscale simulation model based on coloured timed Petri nets. Their analysis, applied to a medium-sized airport, assesses the impact of diverse interferences, operator time windows, and screening configurations on system capacity, finding that parallel deployment of automatic screening lines offers the highest potential for throughput improvement. Similarly, Khosravi et al. (Khosravi et al. 2009) utilize artificial neural network metamodels trained on simulation data to rapidly estimate key performance indexes. By approximating the relationship between design factors and performance measures, such as baggage travel time, they facilitate real-time operational decision-making without the computational cost of detailed simulations. Nahavandi et al. (Nahavandi et al. 2019) demonstrate the synergy between simulation and process mining by using causal-net mining to validate discrete-event models of baggage handling systems. The approach highlights behaviors and interdependencies under varying security policies, specifically analyzing how changes in manual screening rates impact overall system throughput. Lai et al. (Lai et al. 2023) and Liu et al. (Liu et al. 2023) predict bottlenecks using Graph Neural Networks and LSTMs, respectively.

Other research detects changes and root causes in logistics processes. Prathama et al. (Prathama et al. 2019) detect sudden concept drift using trace clustering. Li et al. (Denisov et al. 2020) restore missing timestamps in FIFO systems (e.g., baggage handling) using synchronous proclets (Fahland 2019) and optimization. Xu et al. (Wang et al. 2014) analyze activity time intervals to identify root causes in bulk port operations.

Further work applies process mining in domain-specific contexts. Kropp et al. (Kropp et al. 2024) improve intra-hospital patient transport using bottleneck analysis. Piest et al. (Piest et al. 2021) show that the Open Trip Model can standardize logistics data for process mining. Knoll et al. (Knoll et al. 2019) combine multidimensional process mining with value stream mapping to reduce waste in internal logistics. Intayoad et al. (Intayoad and Becker 2018) use Markov chains to preprocess data for higher-quality models.

Support for process improvement

Several approaches aim to move beyond analysis toward actively improving logistics processes. van der Aalst et al. (Aalst et al. 2020) describe forward-looking process mining challenges in the "Internet of Production."

Kratsch et al. (Denisov et al. 2019) extend the performance spectrum (Denisov et al. 2018) for forecasting dynamic processes using real baggage handling data. Building on predictive capabilities for operational support, Gunnarsson et al. (Gunnarsson et al. 2025) implement LSTM-based sequence-to-sequence models to predict the remaining trajectory and runtime of luggage in real-time. Their work introduces a development cycle for predictive monitoring and incorporates inter-case features, such as the load state of nearby locations, to enhance prediction accuracy in complex airport environments. Complementing this, Ahmed et al. (Ahmed et al. 2016) propose a probabilistic flow graph approach for online risk prediction. By maintaining histograms of transition

durations, they enable the real-time identification of indoor moving objects, such as baggage, that are at risk of being delayed or mishandled compared to historical norms.

Piest et al. (Piest et al. 2021) highlight that adopting standardized data exchange (e.g., Open Trip Model) improves interoperability and supports cross-system process mining initiatives.

Preliminaries

We begin by defining the fundamental mathematical sets and symbols used throughout the paper. This establishes a baseline understanding for the concepts that follow.

Basic sets and functions

Definition 1 (Powerset and Cardinality) Let X be an arbitrary set.

- The powerset of X is denoted by $\mathcal{P}(X) := \{X' \mid X' \subseteq X\}$.
- The cardinality of X is denoted by $|X|$.

The powerset contains all subsets of a given set, and the cardinality simply counts how many elements the set has. For example, if $X = \{a, b\}$, then $\mathcal{P}(X) = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$, and $|X| = 2$.

Definition 2 (Common Universes) A *universe* is any collection of possible entities within a specific context.

- $\mathbb{N}$ = natural numbers.
- $\mathbb{R}_{\geq 0}$ = non-negative real numbers.
- $\mathbb{B} = \{\text{True}, \text{False}\}$ denotes Boolean values.

These universes simply specify the kinds of values we allow in different settings—for instance, counts, continuous quantities, or truth values. For example, $\mathbb{N}$ contains values like $0, 1, 2, \ldots$, while $\mathbb{R}_{\geq 0}$ includes numbers such as 0, 2.5, or 7.1, and $\mathbb{B}$ contains only `True` and `False`.

Definition 3 (Functions) Let $\perp$ denote an *undefined* value, not part of any universe, and X and Y be two arbitrary sets with $\perp \notin Y$.

- A function $f\colon X \to Y$ has $\text{dom}(f) = X$ and $\text{codom}(f) = Y$.
- Restriction to $X' \subseteq X$ is denoted by $f_{|X'}(x) = f(x)$ for $x \in X'$.
- A partial function $g\colon X \not\to Y$ is a function where $g(x) = \perp$ for some $x \in X$.

This definition distinguishes total functions (defined everywhere), restricted views of functions, and partial functions that may return an undefined value. For example, if $X = \{1, 2, 3\}$ and $Y = \{a, b\}$, a total function could be $f = \{1 \mapsto a,\ 2 \mapsto b,\ 3 \mapsto a\}$, while a partial function may assign $g(3) = \perp$.

Attributes

Definition 4 (Attribute) An *attribute* is a characteristic associated with any element $x \in X$. To describe attributes formally, we introduce the following universes:

- The universe of attribute names is denoted by $\mathcal{U}_{attr} = \{dur, time, \ldots\}$.
- The universe of attribute values is given by $\mathcal{U}_{val} = \mathcal{U}_{dur} \cup \mathcal{U}_{time} \cup \ldots$, where:

  - $\mathcal{U}_{time} = \mathbb{R}_{\geq 0}$ is the universe of timestamps.
  - $\mathcal{U}_{dur} = \mathbb{R}_{\geq 0}$ is the universe of time durations, representing an elapsed amount of time.

- The universe of attribute–value mappings is expressed as $\mathcal{U}_{map} : \mathcal{U}_{attr} \not\to \mathcal{U}_{val}$.

Attributes capture properties of elements, such as how long something takes or when it occurs. For example, an element x might have an attribute mapping $dur \mapsto 5.0$ (a time duration) and $time \mapsto 12.3$ (a timestamp). To extract attribute values, we define an attribute projection function.

Definition 5 (Attribute Projection) An attribute projection $\pi \in X \to \mathcal{U}_{map}$ assigns each $x \in X$ a mapping from attribute names to their corresponding values. In particular, for every $attr \in \mathrm{dom}(\pi(x))$, we write

$$\pi_{attr}(x) := \pi(x)(attr)$$

to denote the value of attribute *attr* for the element x.
An attribute projection lets us easily look up the value of a specific property for each element. For example, if x has attributes $dur \mapsto 5.0$ and $time \mapsto 12.3$, then $\pi_{dur}(x) = 5.0$ and $\pi_{time}(x) = 12.3$.

Sequence

Definition 6 (Sequence) A *sequence* over X is a function $\sigma : \{1, 2, \ldots, n\} \to X$, written as $\langle x_1, x_2, \ldots, x_n \rangle$, where $\sigma(i) = x_i$.

- The length of σ is denoted by $|\sigma| = n$.
- We write $x \in \sigma$ to indicate that x is an element of σ.
- The set of all finite sequences over X is denoted by X^*.

A sequence simply orders elements from a set, keeping track of their positions. For example, if $X = \{a, b, c\}$, then $\langle a, b, a \rangle$ is a sequence of length 3, with $a \in \langle a, b, a \rangle$ and $\langle a, b, a \rangle \in X^*$.

Definition 7 (Sequence Concatenation) Given two sequences $\sigma_1, \sigma_2 \in X^*$, we *concatenate* them, denoted by $\sigma_1 \circ \sigma_2 \in X^*$, to obtain a sequence σ' such that:

- $\forall\, 1 \leq i \leq |\sigma_1| : \sigma'(i) = \sigma_1(i)$,
- $\forall\, 1 \leq j \leq |\sigma_2| : \sigma'(|\sigma_1| + j) = \sigma_2(j)$.

Concatenation joins two sequences end-to-end, preserving the order of elements in each. For example, if $\sigma_1 = \langle a, b \rangle$ and $\sigma_2 = \langle c \rangle$, then $\sigma_1 \circ \sigma_2 = \langle a, b, c \rangle$.

Definition 8 (Sequence Projection) Let $attr \in \mathrm{dom}(\pi(x))$ for any $x \in X$, where $\forall x \in X : \pi_{attr}(x) \neq \bot$. We project a sequence $\sigma \in X^*$ onto *attr*, denoted by $\sigma_{\uparrow attr} : X^* \to \mathcal{U}_{val}^*$, such that $\forall\, 1 \leq i \leq |\sigma| : \sigma_{\uparrow attr}(i) = \pi_{attr}(\sigma(i))$. This projection extends to multiple attributes $\langle attr_1, attr_2, \ldots, attr_k \rangle$, written as $\sigma_{\uparrow attr_1, attr_2, \ldots, attr_k}$, where $\forall\, 1 \leq i \leq |\sigma| : \sigma_{\uparrow attr_1, attr_2, \ldots, attr_k}(i) = (\pi_{attr_1}(\sigma(i)), \pi_{attr_2}(\sigma(i)), \ldots, \pi_{attr_k}(\sigma(i)))$.
Suppose we have a sequence of events $\sigma = \langle e_1, e_2, e_3 \rangle$, and each event has attributes 'time' and 'dur' given by:

$$\pi_{time}(e_1) = 1, \quad \pi_{time}(e_2) = 3, \quad \pi_{time}(e_3) = 4,$$
$$\pi_{dur}(e_1) = 2, \quad \pi_{dur}(e_2) = 1, \quad \pi_{dur}(e_3) = 3.$$

Then the projection onto the 'time' attribute is

$$\sigma_{\uparrow time} = \langle 1, 3, 4 \rangle,$$

and the projection onto both 'time' and 'dur' is

$$\sigma_{\uparrow time,dur} = \langle (1, 2), (3, 1), (4, 3) \rangle.$$

Graph

We define notations and concepts related to graphs below.

Definition 9 (Graph) A *graph* is a tuple $G = (V, E)$, where:

- V is a set of nodes.
- $E = \{(v_1, v_2) \in V \times V \mid v_1 \neq v_2\}$ is a set of edges.

A graph represents entities (nodes) and their relationships (edges). For example, if $V = \{A, B\}$ and $E = \{(A, B)\}$, we have a graph with two nodes and one edge from *A* to *B*.

We can annotate the nodes with attributes, resulting in an *attributed graph* $G_{attr} = (V, E, \pi)$, where $\pi \colon V \to \mathcal{U}_{map}$ is an attribute projection function on *V*. For instance, $\pi(A)$ could map $dur \mapsto 5$ and $time \mapsto 12.3$, adding properties to node *A*.

Definition 10 (Node Neighborhoods) Given a node $v \in V$:

- The set of *predecessors* of *v* is denoted as $\mathrm{PRED}v = \{v' \in V \mid (v', v) \in E\}$.
- The set of *successors* of *v* is denoted as $\mathrm{SUCC}v = \{v' \in V \mid (v, v') \in E\}$.

Predecessors and successors capture which nodes come before or after a given node in a graph. For example, if $V = \{A, B, C\}$ and $E = \{(A, B), (B, C)\}$, then $\mathrm{PRED}B = \{A\}$ and $\mathrm{SUCC}B = \{C\}$.

Definition 11 (Paths and Ordering) For nodes $v, v' \in V$, $v \neq v'$:

- The set of *paths* from *v* to *v*′ is $\mathrm{PATHS}v, v' = \{\sigma \in V^* \mid \sigma(1) = v \wedge \sigma(|\sigma|) = v' \wedge \forall 1 \leq i < |\sigma| \colon (\sigma(i), \sigma(i+1)) \in E\}$.
- We write $v \prec v'$ if and only if there exists a path from *v* to *v*′ and no path from *v*′ to *v*.
- For a sequence $\sigma \in V^*$, we write $\sigma \in G$ if $\sigma \in \mathrm{PATHS}\sigma(1), \sigma(|\sigma|)$.

Paths capture sequences of nodes connected by edges, and the relation $\prec$ defines a partial ordering based on reachability. For example, if $V = \{A, B, C\}$ and $E = \{(A, B), (B, C)\}$, then $\mathrm{PATHS}A, C = \{\langle A, B, C \rangle\}$, and $A \prec C$ because there is a path from *A* to *C* but not from *C* to *A*.

Definition 12 (Induced Subgraph) For a subset of nodes $V' \subseteq V$, the *projection* of *G* onto V' is denoted by $G_{|V'} = (V', E')$, where $E' = \{(v_1, v_2) \in E \mid v_1 \in V' \wedge v_2 \in V'\}$. We refer to G' as an *induced subgraph* of *G*.

An induced subgraph contains a selection of nodes from the original graph and all edges between them. For example, if $V = \{A, B, C\}$ and $E = \{(A, B), (B, C)\}$, then for $V' = \{A, B\}$, the induced subgraph is $G_{|V'} = (\{A, B\}, \{(A, B)\})$.

Alignment

In process mining, alignments are typically applied to a process model—commonly represented as a *Petri net*—and an event log. In the context of air cargo distribution, we assume that a cargo item remains a unique physical entity throughout the system; thus, the Petri net only needs to describe sequential behavior. Consequently, we define alignment on a graph, which serves as the data structure for modeling the relationships between resources involved in distributing cargo items and can be implicitly converted into a Petri net.

Let X and Y be two sets, where $\gg \notin X \cup Y$. Let $\pi^X : X \to \mathcal{U}_{map}$ and $\pi^Y : Y \to \mathcal{U}_{map}$. Given a set of attributes $attrs \subseteq \mathcal{U}_{attr}$ such that $\forall attr \in attrs : attr \in \mathrm{dom}(\pi^X(x)) \wedge \mathrm{dom}(\pi^Y(y))$ for any $x \in X$ and $y \in Y$.

Definition 13 (Move and Legal Move) Given a sequence $\sigma \in X^*$, a *move* is a pair $((x, i), y)$ where $x \in \sigma \wedge \sigma(i) = x$ or $(x, i) = \gg$, and $y \in Y \cup \{\gg\}$. A move $((x, i), y)$ is *legal* if one of the following conditions holds:

- If $y \neq \gg$ and $\forall attr \in attrs : \pi^X_{attr}(x) = \pi^Y_{attr}(y)$, the move is a *synchronized move*.
- If $(x, i) = \gg$ and $y \in Y$, the move is a *model move*.
- If $x \in \sigma$ and $y = \gg$, the move is a *log move*.

If $attrs = \varnothing$, a synchronized move is defined simply as $y \neq \gg$ and $x = y$.
Moves represent steps in comparing a sequence (log) with a model, showing whether elements match, are missing in the model, or are missing in the log.

For example, consider a log sequence $\sigma = \langle load, scan, unload \rangle$ and a model sequence $Y = \{load, unload, scan\}$ with $attrs = \varnothing$:

- $((load, 1), load)$ is a synchronized move (matches exactly).
- $(\gg, unload)$ is a model move (model has an action not yet in the log at this step).
- $((scan, 2), \gg)$ is a log move (log has an action not matched by the model at this step).

Definition 14 (Cost Function) We define a cost function $\delta : ((X \times \mathbb{N}) \cup \{\gg\}) \times (Y \cup \{\gg\}) \to \mathbb{R}_{\geq 0}$, which assigns a non-negative cost to each legal move.
The cost function quantifies how "expensive" it is to perform each move, e.g., penalties for mismatches. For the running example, we might set $\delta(((scan, 2), \gg)) = 1$ for the log move, $\delta((\gg, unload)) = 1$ for the model move, and $\delta(((load, 1), load)) = 0$ for the synchronized move.

Definition 15 (Alignment) Given sequences $\sigma_X \in X^*$ and $\sigma_Y \in Y^*$, an *alignment* between them is a sequence of legal moves such that:

- Excluding the model moves, the resulting sequence of x_i matches σ_X.
- Excluding the log moves, the resulting sequence of y_i matches σ_Y.

The total cost of an alignment is $\sum_{k=1}^{n} \delta((x_k, i_k), y_k)$.
An alignment shows how to match a log to a model step by step, highlighting agreements and deviations, while computing a total cost. For our running example, one possible alignment could be:

$$((load, 1), load), ((scan, 2), \gg), (\gg, unload), ((unload, 3), scan)$$

with a total cost of $1 + 1 + 1 = 3$.

Given a graph $G = (Y, E)$, alignment algorithms aim to find the optimal alignment using any $\sigma \in Y^*$ such that $\sigma \in G$, with the goal of minimizing the total cost. The process of extracting optimal alignments is denoted by $\mathrm{ALIGN}\sigma_X, G, \delta, attrs$, where π^X and π^Y are applied implicitly.

For example, consider a graph with nodes $Y = \{load, scan, unload\}$ and edges $E = \{(load, scan), (scan, unload)\}$, representing the allowed order of operations. Given a log sequence $\sigma_X = \langle load, unload \rangle$, the alignment algorithm will find a sequence $\sigma \in Y^*$ such as $\langle load, scan, unload \rangle$ that follows the graph and aligns with the log, possibly introducing a model move for the missing 'scan' step. The total cost is computed using δ, e.g., assigning 1 for each log or model move and 0 for synchronized moves.

Datasets

The distribution of cargo items is described by event data and the logistics system responsible for their delivery. In this section, we formalize both the event data and the logistics system. The intuitive descriptions and practical context for how these concepts are applied within our methodology are provided in the subsequent section. First, we provide an overview of the logistics system, detailing the resources involved in distributing cargo items. Then, we define the types of event data generated during the distribution process.

Logistic system

A logistics system defines how resources distribute cargo items throughout the network. Let $\mathcal{U}_{res}$ represent the universe of resource identifiers. Grounded in the definition of resources, the logistics system is defined by resources and a distribution network.

Definition 16 (Resource) A resource $r \in \mathcal{U}_{res}$ is characterized by a set of attributes. Let $\# \in \mathcal{U}_{res} \to \mathcal{U}_{map}$ be the attribute projection function for $\mathcal{U}_{res}$. The following projection functions are mandatory, and are therefore total:

- $\#_{exd} : \mathcal{U}_{res} \to \mathcal{U}_{dur}$ denotes the expected duration an item stays on the resource.
- $\#_{cap} : \mathcal{U}_{res} \to \mathbb{N}$ represents the capacity of the resource.

For example, let *r* represent a resource, such as a conveyor. Suppose it holds a cargo item for an expected duration of 30 seconds and has a capacity of 200 items. We then write $\#_{exd}(r) = 30$ and $\#_{cap}(r) = 200$.

To formalize the structure of resources within a logistics system, we introduce the concept of a *resource hierarchy*, which organizes resources into different levels of abstraction based on certain attributes.

Definition 17 (Resource Hierarchy) Let $\sigma_{attr} \in \mathrm{dom}(\#(r))^*$ for any $r \in \mathcal{U}_{res}$, where $\forall 1 \leq i < |\sigma_{attr}| \colon \#_{\sigma_{attr}(i)}(r_1) = \#_{\sigma_{attr}(i)}(r_2) \neq \bot \implies \#_{\sigma_{attr}(i+1)}(r_1) = \#_{\sigma_{attr}(i+1)}(r_2)$ for $r_1, r_2 \in \mathcal{U}_{res}$. A *resource hierarchy*, denoted as $RH = \langle R_1, R_2, \ldots, R_n \rangle \in \mathcal{P}(\mathcal{U}_{res})^*$, is a sequence of resource collections constructed using σ_{attr} such that $\forall 1 \leq i \leq |RH| \colon RH(i) = \{\{r' \in \mathcal{U}_{res} \mid \#_{\sigma_{attr}(i)}(r') = val\} \mid val \in \mathrm{codom}(\#_{\sigma_{attr}(i)}(r)) \wedge r \in \mathcal{U}_{res}\}$. We define a labeling function, $\lambda \colon \mathcal{P}(\mathcal{U}_{res}) \not\to \mathcal{U}_{val}$, to refer to the collection in *RH*, where $\forall 1 \leq i \leq |RH|, \forall R \in RH(i) \colon \lambda(R) = \#_{\sigma_{attr}(i)}(r)$ for any $r \in R$.

Intuitively, a resource hierarchy organizes resources into a nested sequence of groups according to specified attributes. Consider, for example, the attributes *area*, *floor*, and *terminal*, and a set of resources $R = r_1, r_2, \ldots, r_8$, where:

- $r_1 \colon \#_{area}(r) = A, \#_{floor}(r) = T1{-}3, \#_{terminal}(r) = T1$,
- $r_2 \colon \#_{area}(r) = B, \#_{floor}(r) = T1{-}3, \#_{terminal}(r) = T1$,
- $r_3 \colon \#_{area}(r) = C, \#_{floor}(r) = T1{-}4, \#_{terminal}(r) = T1$,
- $r_4 \colon \#_{area}(r) = C, \#_{floor}(r) = T1{-}4, \#_{terminal}(r) = T1$,
- $r_5 \colon \#_{area}(r) = D, \#_{floor}(r) = T1{-}5, \#_{terminal}(r) = T1$,
- $r_6 \colon \#_{area}(r) = E, \#_{floor}(r) = T2{-}2, \#_{terminal}(r) = T2$,
- $r_7 \colon \#_{area}(r) = E, \#_{floor}(r) = T2{-}2, \#_{terminal}(r) = T2$,
- $r_8 \colon \#_{area}(r) = F, \#_{floor}(r) = T2{-}3, \#_{terminal}(r) = T2$.

Each resource is physically located in an area on a floor within a terminal, forming a natural hierarchical relationship. In this example, floor labels are used to distinguish distinct physical locations. Alternatively, identical floor labels could be used across terminals, relying on the terminal attribute to differentiate locations. The same logic applies to areas. Formally, the resource hierarchy $RH = \langle R_1, R_2, R_3 \rangle$, extracted for area, floor, and terminal, is defined as: $R_1 = \{\{r_1\}, \{r_2\}, \{r_3, r_4\}, \{r_5\}, \{r_6, r_7\}, \{r_8\}\}$, $R_2 = \{\{r_1, r_2\}, \{r_3, r_4\}, \{r_5\}, \{r_6, r_7\}, \{r_8\}\}$, and $R_3 = \{\{r_1, r_2, r_3, r_4, r_5\}, \{r_6, r_7, r_8\}\}$. Due to the physical structure, if two resources share the same area, they must also share the same floor; similarly, if they share a floor, they must belong to the same terminal. This establishes a hierarchical relationship among the resources. The hierarchy can be represented as a nested structure, but for computational purposes, we flatten it to directly leverage attribute projections for determining which resources belong to a given abstraction level, enabling efficient queries and manipulations.

A distribution network, or simply a network, specifies the interconnections between resources and the possible flow of items between them.

Definition 18 (Distribution Network) Let $R \subseteq \mathcal{U}_{res}$ be a set of resources. A distribution network $DN = (R, E, \mathrm{tt})$ is a graph where $E \subseteq R \times R$ represents the set of directed connections between pairs of resources, and $\mathrm{tt} \colon E \to \mathcal{U}_{dur}$ defines an expected *transition time*, representing the time it takes to transport an item from r_1 to r_2, where $(r_1, r_2) \in E$, under ideal conditions, which is denoted as $\mathrm{tt}_{(r_1, r_2)} \in \mathcal{U}_{dur}$.

Consider a simplified network consisting of only two resources, r_1 and r_2, where the time between the departure of a cargo item from r_1 and its arrival at r_2 is 3 seconds. The distribution network can then be represented as $DN = (\{r_1, r_2\}\{(r_1, r_2)\}, \mathrm{tt})$, with the transition time defined as $\mathrm{tt}_{(r_1, r_2)} = 3$.

Event data

Event data track the operation of logistic systems. We categorize them into the *operational log* and the *resource status log*. Each type captures different aspects of the system's operations. The two logs reflect abstractions that extend beyond this specific domain. The operational log corresponds to the notion of an event log, a widely adopted construct in process mining for representing sequences of activities in diverse business processes. The resource status log captures the availability and utilization of resources, a fundamental concept in virtually any operational setting. Thus, although illustrated here in the context of logistics, this categorization provides a generalizable structure for analyzing processes across a wide range of systems. In the following, we formalize the event data.

Let $\mathcal{U}_{item}$ denote the universe of items to distribute within a system. A *message* represents a piece of information indicating the status of a resource with regards to a distribution. Let $\mathcal{U}_{msg}$ denote the universe of messages. An *operational log* is structured event data extracted from information systems, serving as the primary source for tracking the distribution of items.

Definition 19 (Operational Log) An operational event is defined by a tuple of attribute values $oe = (itm, r, m, t) \in \mathcal{U}_{item} \times \mathcal{U}_{res} \times \mathcal{U}_{msg} \times \mathcal{U}_{time}$. Let $\mathcal{E}_o = \mathcal{U}_{item} \times \mathcal{U}_{res} \times \mathcal{U}_{msg} \times \mathcal{U}_{time}$ denote the universe of operational events. We define the attribute projection function $\pi^o \in \mathcal{E}_o \to \mathcal{U}_{map}$, where, for any $oe = (itm, r, m, t) \in \mathcal{E}_o$, $\mathrm{dom}(\pi^o(oe)) = \{item, res, msg, time\}$ such that $\pi_{item}(oe) = itm$, $\pi_{res}(oe) = r$, $\pi_{msg}(oe) = m$, and $\pi_{time}(oe) = t$. An *operational trace* of $itm \in \mathcal{U}_{item}$, $\sigma_{itm} = \langle oe_1, oe_2, \ldots, oe_n \rangle \in \mathcal{E}_o^*$, is a sequence of operational events where $\forall oe \in \sigma_{itm} : \pi^o_{item}(oe) = itm$ and $\forall 1 \leq i < |\sigma_{itm}| : \pi^o_{time}(\sigma_{itm}(i)) \leq \pi^o_{time}(\sigma_{itm}(i+1))$. An operational log $OL \subseteq \mathcal{E}_o^*$ is a collection of operational traces, where $\forall \sigma_1, \sigma_2 \in OL : \forall 1 \leq i \leq |\sigma_1|, \forall 1 \leq j \leq |\sigma_2| : \pi^o_{item}(\sigma_1(i)) = \pi^o_{item}(\sigma_2(j)) \iff \sigma_1 = \sigma_2$.

For instance, consider a cargo item itm_1 that *arrives* at a conveyor $r_1 \in \mathcal{U}_{res}$ at 9:24:36 on 24 November 2025. We represent this occurrence as an operational event $oe_1 = (itm_1, r_1, arrive, 9{:}24{:}36,\ 24\ \text{November}\ 2025)$. Suppose the same item then departed by the conveyor at 9:25:06 and *arrives* at $r_2 \in \mathcal{U}_{res}$ 9:25:36 and represented as oe_2 and oe_3 similarly. The operational trace of itm_1 is a sequence of these events $\sigma_{itm} = \langle oe_1, oe_2, oe_3 \rangle$ ordered temporally. An operational log OL may contain multiple such traces for different items, e.g., $OL = \sigma_{itm_1}, \sigma_{itm_2}, \ldots$, providing a complete record of the movement and status of items through the system.

A *resource status log*, or simply a status log, is event data that tracks the periods of availability for resources within the system. Table 1 exemplifies such a log, which is generally defined as:

Definition 20 (Resource Status Log) A status event tracks the unavailability of a resource $r \in \mathcal{U}_{res}$ with a tuple $se = (r, t_{\text{off}}, t_{\text{on}}) \in \mathcal{U}_{res} \times \mathcal{U}_{time} \times \mathcal{U}_{time}$, where t_{off} and t_{on} capture

Table 1 An example of resource status log

Resource	Downtime	Uptime	Cause
B	13:31:10	13:33:49	Motor Overload
C	13:34:05	13:41:12	Transient Fault
A	13:55:13	14:00:01	Transient Fault

the time of *r* going offline and coming back online, respectively, with the constraint that $t_{\text{off}} < t_{\text{on}}$. Let $\mathcal{E}_s = \mathcal{U}_{res} \times \mathcal{U}_{time} \times \mathcal{U}_{time}$ denote the universe of status events. We define the attribute projection function $\pi^s \in \mathcal{E}_s \to \mathcal{U}_{map}$, where, for any $se = (r, t_{\text{on}}, t_{\text{off}}) \in \mathcal{E}_s$, $\text{dom}(\pi^s(se)) = \{res, on, \text{off}\}$ such that $\pi^s_{res}(se) = r$, $\pi^s_{\text{on}}(se) = t_{\text{on}}$, and $\pi^s_{\text{off}}(se) = t_{\text{off}}$. A *status trace* of $r \in \mathcal{U}_{res}$, $\sigma_r = \langle se_1, se_2, \ldots, se_n \rangle \in \mathcal{E}_s^*$, is a sequence of status events where $\forall se \in \sigma_r \colon \pi^s_{res}(se) = r$ and $\forall 1 \leq i < |\sigma_r| \colon \pi^s_{\text{on}}(\sigma_r(i)) \leq \pi^s_{\text{off}}(\sigma_r(i+1))$. A status log $SL \subseteq \mathcal{E}_s^*$ is a collection of status traces, where $\forall \sigma_1, \sigma_2 \in SL \colon \forall 1 \leq i \leq |\sigma_1|$, $\forall 1 \leq j \leq |\sigma_2| \colon \pi^s_{res}(\sigma_1(i)) = \pi^s_{res}(\sigma_2(j)) \iff \sigma_1 = \sigma_2$. Thereby, we write $\text{TRACE}SL, r = \iota\sigma \in SL \colon (\exists se \in \sigma \text{s.t. } \pi^s_{res}(se) = r)$ to refer to the unique status trace of *r* in *SL*.

For simplicity, we assume that resources are available by default and only record their periods of unavailability. The availability of a resource is therefore inferred as the complement of its recorded periods of unavailability. We identify the status of a resource $r \in \mathcal{U}_{res}$ at timestamp $t \in \mathcal{U}_{time}$ using a function $\text{STATUS}^S \colon \mathcal{U}_{res} \times \mathcal{U}_{time} \times SL \to \mathcal{E}_s$. Let $S = \{se \mid \forall \sigma \in SL \forall se \in \sigma \colon \pi^s_{res}(se) = r \wedge \pi^s_{\text{off}}(se) \leq t < \pi^s_{\text{on}}(se)\}$. The function is defined as:

$$\text{STATUS}^S(r, t, SL) = \begin{cases} se, & \text{if } |S| \geq 1 \text{for the } se \in S, \\ \perp, & \text{otherwise.} \end{cases}$$

Note that, due to the binary nature of the resource being online or offline, $|S| \leq 1$.

Suppose a resource $r \in \mathcal{U}_{res}$ goes offline at 14:00 on 24 November 2025 and comes back online at 14:30 on the same day, and then again goes offline at 15:15 and comes back online at 15:45. We record these periods of unavailability as status events $se_1 = (r, 14{:}30, 14{:}00)$ and $se_2 = (r, 15{:}45, 15{:}15)$. The status trace of *r* in the status log *SL* is then $\sigma_r = \langle se_1, se_2 \rangle$. To determine the status of *r* at a specific timestamp, e.g., 14:15, we use the function STATUS^S, giving $\text{STATUS}^S(r, 14{:}15, SL) = se_1$, indicating the resource is offline at that time. For a timestamp outside the recorded periods, e.g., 14:45, we have $\text{STATUS}^S(r, 14{:}45, SL) = \perp$, indicating that the resource is available. This approach assumes that resources are available by default and only records periods of unavailability. Therefore, the availability of a resource at any given time can be inferred as the complement of the intervals captured in its status trace.

Approach: analysis of air cargo distribution

This section presents an analysis of cargo distribution, starting with an overview and illustrative example in Sect. 5.1., Sects. 5.2 and 5.3 then outline the two stages of data processing, followed by the performance analysis in Sect. 5.4.

Overview

Figure 1 presents an overview of the analysis of the cargo distribution process. We begin with the operational log, where we detect events that signify the physical movement of cargo items. This log, directly extracted from information systems, may contain missing, inaccurate, or misleading information due to system malfunctions, human errors, or technical issues, leading to discrepancies between recorded logs and actual behavior. For instance, Table 2 illustrates an operational log, with each row representing an operational event. The first row indicates that resource A is *initiated* to send out cargo item 711123 at 13:30. However, since resource B is unavailable at the time, the cargo

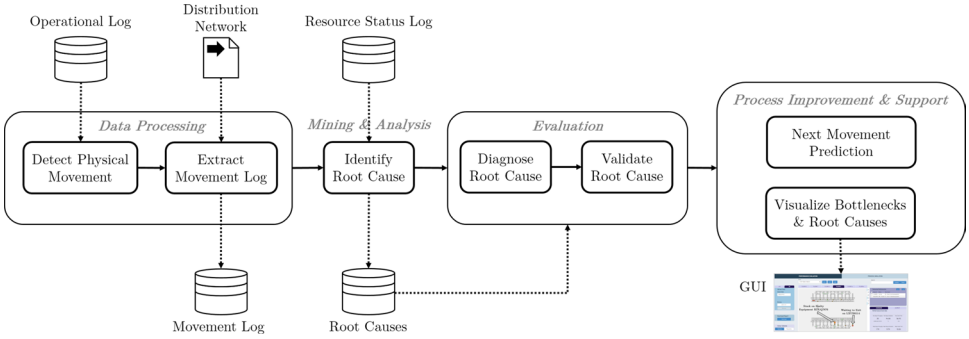


Fig. 1 Overview of the analysis of cargo distribution

Table 2 Motivating example of operational log. The last column (PM) indicates whether the operational event involves the physical movement of a cargo item

Cargo Item	Resource	Time	Message	PM
711123	A	13:30:15	Initiate	Yes
711123	A	13:32:22	Ready	No
711123	A	13:34:13	Ready	Yes
711123	B	13:35:36	Arrived	Yes
711123	D	13:43:18	Arrived	Yes
711123	D	13:47:21	Depart	Yes
711124	A	13:37:01	Initiate	Yes
711124	A	13:45:27	Depart	Yes
711124	B	13:46:41	Decide	Yes
711124	C	13:49:52	Depart	Yes
711124	D	13:52:10	Arrived	Yes
711124	D	13:54:15	Depart	Yes

must wait on resource A until 13:34. During this waiting period, resource A continues to send readiness messages. Consequently, the cargo's duration on resource A is defined as the time from the initial timestamp to the timestamp of the last readiness message, as determined by the system expert. Other events during this period are considered communication events between resources and are not relevant to the physical movement of cargo items.

Following the identification of physical movement, we construct a *movement log* that reflects the actual trajectory of cargo items. This involves aligning the operational log with the distribution network to ensure that the recorded trajectory corresponds to reality. By aligning with the network structure, we achieve a more consistent and reliable representation of the system's operations. Consider a distribution network in Fig. 2. Table 3 is the corresponding movement log extracted from Table 2. Take the case of cargo item 711124, where the trace includes operational events related to resources A, B, C, and D. However, as shown in Fig. 2, it is physically impossible for a cargo item to pass through both resources B and C simultaneously. Upon investigation, we find that the operational event from resource B is recorded as a decision during the communication between resources, rather than representing an actual physical movement of the cargo item. This event is marked as a decision without accompanying events that would indicate the cargo item's movement, as indicated by an additional *PM* column in Table 2. As a result, we correct the trace to reflect the most plausible sequence of events, where cargo item 711124 moves through resources A, C, and D—this path is consistent with

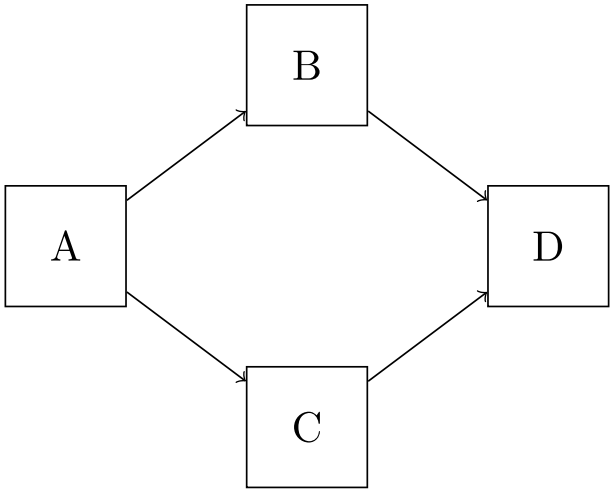


Fig. 2 Motivating example of distribution network

Table 3 An example of movement log

Cargo Item	Resource	Start Time	End Time
711123	A	13:30:15	13:34:13
711123	B	13:35:36	13:43:18
711123	D	13:43:18	13:47:21
711124	A	13:37:01	13:45:27
711124	C	13:45:27	13:49:52
711124	D	13:52:10	13:54:15

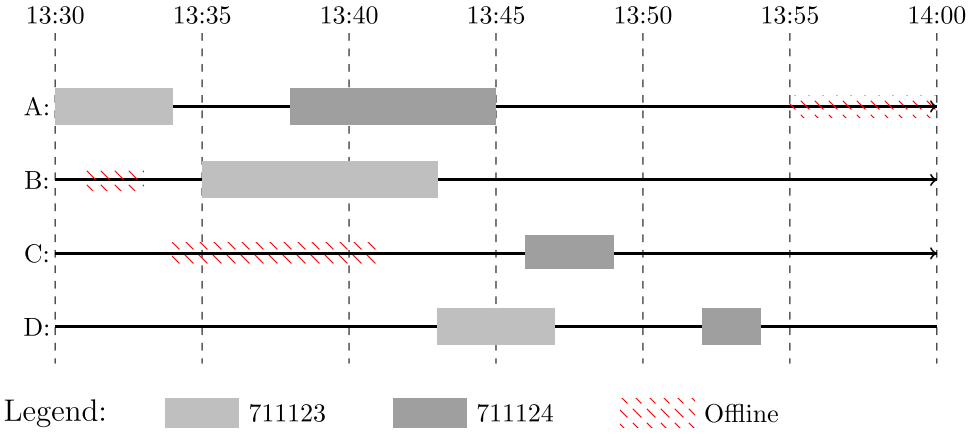


Fig. 3 Event data visualized on a timeline for each resource, showing the progression of events over time

the physical constraints of the distribution network. This process ensures that the resulting movement log reflects the actual distribution of events, reconciling discrepancies for a more truthful representation of the system.

With the movement log in place, the analysis focuses on operational efficiency. We identify the root causes of given bottlenecks—instances where a cargo item remains on a resource longer than expected. Root cause detection pinpoints the underlying causes of these bottlenecks, which may arise from other distribution activities or resource availability issues. Table 1 provides an example of a resource status log, while Fig. 3 illustrates the system based on Table 3. This visualization projects both movement and resource

status events onto timelines for each resource, providing insights into the causes of bottlenecks. For instance, in the case of cargo item 711123, the delay occurs on resource A due to the unavailability of resource B at the time when resource A is scheduled to send the cargo item to resource B. We categorize the root causes into diagnostics of inefficiencies and automate the analysis, with process owners validating the findings. To support process operations, we enhance the analysis with an interactive GUI and predictive analytics to forecast delays, enabling proactive decision-making, optimizing resource management, and improving overall distribution efficiency.

Physical movement detection

Some operational events, such as those related to communication between resources for planning or coordinating the distribution of an item, are unrelated to the physical movement of the item. Hence, we consider them as noise and isolate them from the events that capture the movement of the item, resulting in an operational log that contains only the events tracking the arrival or departure of an item relative to a resource. To detect the physical movement of items, we developed two approaches: an ad-hoc approach and a matching-based method.

Ad-hoc approach

We begin by identifying unreasonable behavior in the data, followed by iterative consultations with domain experts to refine the operational log. Since analysts often recognize challenges on process instances, we sample the cases with the identified behavior and discuss the findings with the experts. Subsequently, they analyze the underlying causes, and we collaboratively develop automated solutions that can be applied to all identified behaviors. This iterative process continues until no further issues are observed, as illustrated in Fig. 4.

This interview-based approach offers valuable insights into the context of the process by clarifying the reasons behind the issues. However, these insights are often confined to specific challenges observed and carry the risk that the solutions may not effectively address the problems, as the same unreasonable behavior can arise from different factors, each requiring distinct solutions. For instance, Fig. 5 illustrates how the same order of resource in an operational trace can result from different factors. Sensors mounted on each resource detect the movement of cargo items and transmit the data to a centralized repository (e.g., data lake). The extracted data forms an operational trace. In Fig. 5a, the misordering is due to sensor installation and sensitivity: when a cargo item is transferred from the second to the third resource, the sensor on the resource device registers the

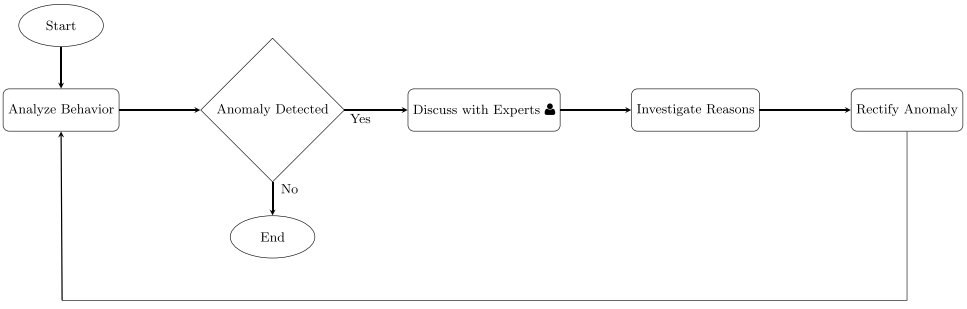
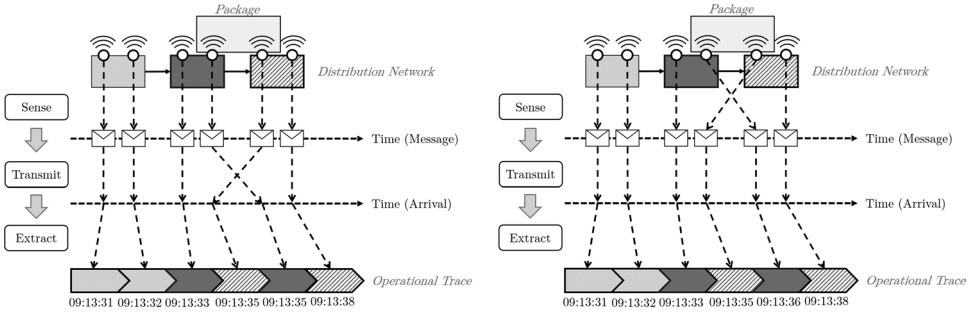


Fig. 4 The ad-hoc approach for physical movement detection



(a) Sensitive sensor (b) Transmission delay

Fig. 5 Examples demonstrating unreasonable behavior. (**a**) Sensitive sensor. (**b**) Transmission delay

Algorithm 2 Root Cause Identification

Require: ML, fault log SL, $bn = (i, \sigma)$
Ensure: a root cause $(j, \sigma') \in \mathbb{N} \cup {\mathcal{E}_m}^* \cup {\mathcal{E}_s}^*$

if $\exists \text{TRACE}(SL, \pi^m_{res}(\sigma(i))) \in SL$ **then**
  $N \leftarrow \{1 \le k \le |\sigma'| \mid \pi^m_{arr}(\sigma(i)) \le \pi^s_{off}(\sigma'(k)) \wedge \pi^m_{dep}(\sigma(i)) \ge \pi^s_{on}(\sigma'(k))\}$
  if $|N| > 1$ **then**
    return $(\min(N), \text{TRACE}(SL, \pi^m_{res}(\sigma(i))))$
  end if
end if
while $i \le |\sigma|$ **do**
  if $i = |\sigma|$ **then return** (i, σ)
  end if
  $r_{nxt} \leftarrow \pi^m_{res}(\sigma(i+1))$
  $t_{nxt} \leftarrow \text{REACH}(\sigma(i), r_{nxt})$
  if $\text{STATUS}^S(r_{nxt}, t_{nxt}, SL) \neq \perp$ **then**
    $\sigma' \leftarrow \text{TRACE}(SL, r_{nxt})$
    $N \leftarrow \{1 \le k \le |\sigma'| \mid \sigma'(k) = \text{STATUS}^S(r_{nxt}, t_{nxt}, SL)\}$
    return $(\min(N), \sigma')$
  end if
  if $|\text{STATUS}^M(r_{nxt}, t_{nxt}, ML)| < \#_{cap}(r_{nxt})$ **then**
    return (i, σ)
  else ▷ The next resource reaches the maximal capacity at the planned time
    $ME \leftarrow \{me | \forall \sigma_m \in ML \forall me \in \sigma_m : \pi^m_{res}(me) = r_{nxt} \wedge \pi^m_{arr}(me) \le t_{nxt} < \pi^m_{dep}(me)\}$
    $me \leftarrow \arg\min_{me' \in ME} \pi^m_{dep}(me')$
    if $\pi^m_{dep}(me) - \pi^m_{arr}(me) > \#_{exd}(r_{nxt})$ **then**
      $\sigma \leftarrow \text{TRACE}(ML, \pi^m_{item}(me))$
      $i \leftarrow \min(\{1 \le k \le |\sigma| \mid \sigma(k) = me\})$
    else
      return (i, σ)
    end if
  end if
end while

item's arrival before it has completely departed from the previous resource. This causes the item to appear at the next device before leaving the previous one. In Fig. 5b, a similar misordering occurs owing to a transmission delay, where the arrival and departure timestamps are recorded simultaneously. This issue cannot be resolved simply by sorting timestamps during extraction. To correct the trace in Fig. 5a, we adjust the timestamp of the arrival event to match the departure time of the previous resource and reorder the events. For Fig. 5b, we swap the order of the events without modifying the timestamps.

This example demonstrates that identical behaviors can arise from different causes, requiring tailored solutions. Moreover, this approach is labor-intensive and becomes less efficient as the scale of the operational log increases.

Matching physical movement

Owing to the limitations of the ad-hoc approach, we develop a systematic method to *match* the operational events to represent the physical movement of an item to a resource. To avoid assumption on the timestamp of uncaptured or misinterpreted message of events, e.g., the readiness event of operational trace of cargo item 711123 in Table 2 is considered as the departure of item on resource A, we approximate the timestamp of an inserted event based on the timestamp of the closest event based on the lifecycle model of messages of a resource. To avoid assumptions about the timestamps of uncaptured or misinterpreted events—such as the readiness event of resource A in the operational trace of 711,123 in Table 2, which is determined as the item's departure from resource A—we approximate the timestamp of any inserted event using the closest event's timestamp, based on the resource's message lifecycle model. We adapt the formalization in (Li et al. 2019) to define the objectives and constraints of the matching.

Given an operational log OL and a trace $\sigma \in OL$, let $G = (V_1, V_2, E, \lambda, w)$, where $V_1 = \{v \mid 1 \leq i \leq |\sigma|\}$ and V_2 is constructed in the same manner, λ is a labeling function that annotates the position of an event in the trace, i.e., $\exists! v \in V_1 : \lambda(v) = i \wedge \exists! v \in V_2 : \lambda(v) = i$. Let $ws \in \mathbb{N}$ be the *window size*, we define the edges as $E = \{(v_1, v_2) \in V_1 \times V_2 \mid 1 \leq \lambda(v_2) - \lambda(v_1) \leq ws \wedge \pi^o_{res}(\sigma(\lambda(v_1))) = \pi^o_{res}(\sigma(\lambda(v_2)))\}$. Our goal is to maximize the pairing of events by optimizing the likelihood that a message serves as the start (quantified by $rnk_{arr} \in \mathbb{N}$) and the message serves as the departure (quantified by rnk_{dep}), based on their proximity in the lifecycle model per corresponding resource. The weight of $(v_1, v_2) \in E$, denoted by $w_{(v_1,v_2)}$, is defined as

$$w_{(v_1,v_2)} = \frac{rnk_{arr}(v_1) \cdot rnk_{dep}(v_2)}{\pi^o_{time}(\sigma(\lambda(v_2))) - \pi^o_{time}(\sigma(\lambda(v_1))) + 1}$$

Let $x_{(v_1,v_2)} \in \{0, 1\}$ be a variable assigned to each $(v_1, v_2) \in E$. We formulate the pairing as an optimization problem using Integer Linear Programming (ILP) (Schrijver 1999):

$$\text{Maximize } Z = \sum_{(v_1,v_2) \in E} w_{(v_1,v_2)} \cdot x_{(v_1,v_2)}$$

$$\text{Subject to: } \begin{cases} \sum_{v' \in V_2} x_{(v,v')} \leq 1, & v \in V_1 \\ \sum_{v' \in V_1} x_{(v',v)} \leq 1, & v \in V_2 \end{cases}$$

The matching process aims to pair the arrival and departure messages for every cargo item on a resource. The ILP-based approach is chosen because it guarantees a globally optimal solution for this task, a benefit that has been previously validated in our related work, where we compared it against heuristic-based approaches (Li et al. 2024).

Movement log extraction

The identification of operational events indicating physical movement does not guarantee consistency between the log and the network. Therefore, we extract a *movement log*

that reflects the physical movement of items within the network. A movement log consists of a collection of *movement events*.

Definition 21 (Movement Log) A movement event is defined by a tuple of attribute values $me = (itm, r, t_{arr}, t_{dep}) \in \mathcal{U}_{item} \times \mathcal{U}_{res} \times \mathcal{U}_{time} \times \mathcal{U}_{time}$, where t_{arr} and t_{dep} capture the time of *itm* arriving and departing *r*, respectively, with the constraint that $t_{arr} \leq t_{dep}$. The duration $t_{dep} - t_{arr} \in \mathcal{U}_{dur}$ is the *dwell time* of *itm* on *r*. Let $\mathcal{E}_m = \mathcal{U}_{item} \times \mathcal{U}_{res} \times \mathcal{U}_{time} \times \mathcal{U}_{time}$ denote the universe of movement events. We define the attribute projection function $\pi^m \in \mathcal{E}_m \to \mathcal{U}_{map}$, where, for any $me = (itm, r, t_{arr}, t_{dep}) \in \mathcal{E}_m$, $\mathrm{dom}(\pi^m(me)) = \{item, res, arr, dep\}$ such that $\pi^m_{item}(me) = itm$, $\pi^m_{res}(me) = r$, $\pi^m_{arr}(me) = t_{arr}$, and $\pi^m_{dep}(me) = t_{dep}$. Let $DN = (R, E, \mathrm{tt})$ be a distribution network. A *movement trace* of $itm \in \mathcal{U}_{item}$, $\sigma_{itm} = \langle me_1, me_2, \ldots, me_n \rangle \in \mathcal{E}^*_m$, is a sequence of movement events where $\forall me \in \sigma_{itm} \colon \pi^m_{item}(me) = itm$ and $\forall 1 \leq i < |\sigma_{itm}| \colon$

- $\pi^m_{dep}(\sigma_{itm}(i)) \leq \pi^m_{arr}(\sigma_{itm}(i+1))$, assuring chronologically ordered events.
- $\pi^m_{res}(\sigma_{itm}(i)) \neq \pi^m_{res}(\sigma_{itm}(i+1))$, indicating that the trace captures the item's movement between different resources.
- $(\pi^m_{res}(\sigma_{itm}(i)), \pi^m_{res}(\sigma_{itm}(i+1))) \in E$, ensuring that the trace aligns with a valid path in the distribution network.

A movement log $ML \subseteq \mathcal{E}^*_m$ is a collection of movement traces, where $\forall \sigma_1, \sigma_2 \in ML \colon \forall 1 \leq i \leq |\sigma_1|, \forall 1 \leq j \leq |\sigma_2| \colon \pi^m_{item}(\sigma_1(i)) = \pi^m_{item}(\sigma_2(j)) \iff \sigma_1 = \sigma_2$ Given a timestamp $t \in \mathcal{U}_{time}$, the status of a resource $r \in \mathcal{U}_{res}$ with respect to package distributions can be identified by a function $\mathrm{STATUS}^M \colon \mathcal{U}_{res} \times \mathcal{U}_{time} \times ML \to \mathcal{P}(\mathcal{E}_m)$, where $\mathrm{STATUS}^M(r, t, ML) = \{me \mid \forall \sigma \in ML \forall me \in \sigma \colon \pi^m_{res}(me) = r \wedge \pi^m_{arr}(me) \leq t < \pi^m_{dep}(me)\}$. Thereby, we overload the notation and write $\mathrm{TRACE} ML, itm = \iota \sigma \in ML \colon (\exists me \in \sigma \mathrm{s.t.}\ \pi^m_{item} = itm)$ to refer to the movement trace of *itm* in *ML*.

The extraction of a movement log involves aligning an operational log with the distribution network and correcting the timestamps of the events. In the following, we detail the extraction of a movement log from an operational log.

Alignment

Alignments serve as a reference for manipulating a trace to align with the identified path in a network. The process can be broken down into two main phases: calculating moves and applying repairs. First, the algorithm compares a given operational trace to a reference process graph to identify the "moves" needed to align the two. These moves fall into three categories: model moves, which represent events that are missing in the log but present in the model; log moves, which represent extra events in the log not found in the model; and synchronous moves, which represent events found in both. Next, the algorithm applies the identified moves to the original trace. It systematically inserts the missing events (model moves) and deletes the extra events (log moves) to create a new, aligned trace. The result is a clean, semantically correct sequence of events that accurately represents a single, complete physical movement for a cargo item. This new trace can then be used for subsequent process mining analysis. We generalize the manipulation to arbitrary sequences.

Definition 22 (Insertion into a sequence) Let X and Y be two arbitrary sets. Consider:

- a sequence σ over X, and
- a sequence of tuples $\sigma' \in (\mathbb{N} \times Y)^*$ satisfying:
  - for every $(i, y) \in \sigma'$: $0 \leq i \leq |\sigma|$;
  - for every $1 \leq m < n \leq |\sigma'|$ with $\sigma'(m) = (i_m, y_m)$ and $\sigma'(n) = (i_n, y_n)$, we have $i_m < i_n$.

Each tuple $(i, y) \in \sigma'$ specifies that y is inserted after position i of σ. We denote the resulting sequence by $\mathrm{INSERT}(\sigma, \sigma')$.
For example, given $\sigma = \langle x_1, x_2, x_3 \rangle$, $\mathrm{INSERT}(\sigma, \langle (0, y_1), (2, y_2), (3, y_3), (3, y_4) \rangle)$ returns $\langle y_1, x_1, x_2, y_2, x_3, y_3, y_4 \rangle$. The position of 0 indicates that the elements should be inserted before the first element of σ.

Definition 23 (Deletion from a sequence) Let N be a set of positions such that $1 \leq i \leq |\sigma|$ for every $i \in N$. We denote by $\mathrm{DELETE}(\sigma, N)$ the sequence obtained by removing the elements at these positions and shifting the rest accordingly.
For instance, given $\sigma = \langle x_1, x_2, x_3, x_4 \rangle$, $\mathrm{DELETE}(\sigma, \{1, 3\}) = \langle x_2, x_4 \rangle$.

Definition 24 (Attributes and graph structure) Let X and Y be two arbitrary sets.

- $\pi^X : X \to \mathcal{U}_{map}$ is an attribute projection function on X.
- $G = (Y, E, \pi^Y)$ is a graph with nodes Y, edges $E \subseteq Y \times Y$, and attribute projection $\pi^Y : Y \to \mathcal{U}_{map}$.
- Let $attrs \in \mathcal{U}_{attr}$ such that, for every $attr \in attrs$:

  $$attr \in \mathrm{dom}(\pi^X) \quad \text{and} \quad attr \in \mathrm{dom}(\pi^Y).$$

Finally, we use δ to denote the cost function.
Algorithm 1 describes how an alignment result is transformed into a repaired sequence that is consistent with the distribution network. In this paper, an alignment is a structured description of deviations between an observed sequence and a valid network path, expressed through three types of moves: synchronized moves, model moves, and log moves. Synchronized moves indicate agreement between the log and the network, model moves indicate behavior required by the network but missing in the log, and log moves indicate events that are not compatible with the network structure. The algorithm operationalizes this alignment by translating the identified deviations into concrete modifications of the original sequence. Rather than constructing a new trace from scratch, the algorithm performs minimal edits to the observed sequence, inserting missing elements where required and removing elements that violate the network constraints. This design preserves as much of the recorded behavior as possible while enforcing consistency with the reference network. To achieve this, the algorithm derives two sets of repair actions from the alignment. Model moves are collected as insertion instructions, specifying where elements implied by the network should be added to the sequence. Log moves are collected as deletion instructions, identifying events that must be removed to eliminate infeasible behavior. These repair actions are derived by iterating over the alignment and interpreting each move in its execution context, ensuring that insertions and deletions are applied relative to the observed ordering of events. The

resulting sequence forms a clean and semantically consistent basis for subsequent steps in the analysis.

Given an operational log OL and a distribution network $DN = (R, E, \mathrm{tt})$, we apply an alignment algorithm to detect deviations between every operational trace $\sigma \in OL$ and the network. To facilitate alignment, we *split* every node in *DN*, resulting in an *expanded network*. This is done by splitting each resource node into two distinct nodes: a start node and a complete node, where the start node represents a cargo item's arrival at a resource, and the complete node represents the item's departure from that resource. The original edges in the network are then re-routed to connect these new start and complete nodes. Specifically, edges entering an original resource node now link to its corresponding start node, while edges exiting the original node now originate from its corresponding complete node. A new edge is also added to connect the start and complete nodes for each resource. This transformation enables our alignment algorithm to precisely track the beginning and end of each movement event. Formally, the expanded network $DN' = (V, E', \pi)$ constructed from *DN* such that for every $r \in R$, we create two nodes v_1 and v_2, where $\pi_{res}(v_1) = r \wedge \pi_{res}(v_1) = m_{arr}$ and $\pi_{res}(v_2) = r \wedge \pi_{res}(v_2) = m_{dep}$, and $E' = \{(v_1, v_2)) \mid \pi_{res}(v_1) = \pi_{res}(v_2) \wedge \pi_{msg}(v_1) = m_{arr} \wedge \pi_{msg}(v_2) = m_{dep}$ or $(\pi_{res}(v_1), \pi_{res}(v_2)) \in E \wedge \pi_{msg}(v_1) = m_{dep} \wedge \pi_{msg}(v_2) = m_{arr}\}$. Given an operational trace $\sigma \in OL$, we denote the application of the algorithm as $\mathrm{ALIGN\_REPAIR}\sigma, DN', \delta, \{res, msg\}$. The resulting operational trace $\sigma' \in \mathcal{E}_o^*$ consists of pairs of consecutive events, each representing the arrival and departure of a cargo item on the corresponding resource. For each pair, we create a movement event as follows: for every $1 \leq i \leq |\sigma'|$, if $i\%2 = 1$, we create a movement event $me = (\pi^o_{item}(\sigma(i)), \pi^o_{res}(\sigma(i)), \pi^o_{time}(\sigma(i)), \pi^o_{time}(\sigma(i+1)))$.

However, alignment incurs high computational costs (van Zelst et al. 2020; Sani et al. 2020). We leverage the resource hierarchy (cf. Definition 17) to modify the log more efficiently. By abstracting both the log and the network to a higher level using the hierarchy, we facilitate alignment on a smaller scale. This approach reduces computational complexity and enhances overall efficiency (van der Aalst 2013; Leoni et al. 2014). Figure 6 illustrates the concept. For the distribution network, we aggregate it according to the resource hierarchy and project it onto a lower level of abstraction. Similarly, we abstract the operational log, resulting in an abstracted log and multiple sublogs at the lower level. First, we align the log at the higher level, which serves as guidance for the trajectory at the corresponding level. Next, we identify the events at the lower level for each abstracted event and its corresponding subnetwork, then perform the alignment

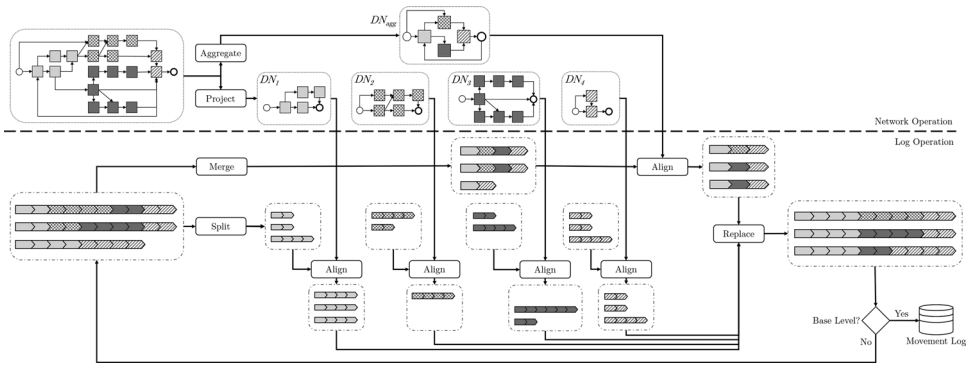


Fig. 6 The approach based on hierarchical alignments

at the lower level. This process is iteratively applied until we reach the base level of the hierarchy. The concatenation of the aligned traces at each level forms the final movement trace.

The abstraction of both the operational log and the distribution network is guided by specific attributes. To formalize this, we define a function for each abstraction.

Definition 25 (Trace abstraction by attribute) Let OL be an operational log. For every trace $\sigma \in OL$, the abstraction of σ is achieved by *merging* consecutive operational events that share the same value of any $attr \in \sigma_{attr}$, where $\forall oe \in \mathcal{E}_o \colon \pi^o_{oe} \neq \perp$. This process is formalized by two functions. The function

$$\mathrm{SPLIT} \colon \mathcal{E}_o^* \times \mathcal{U}_{attr} \to {\mathcal{E}_o}^{**}$$

splits a trace σ according to an attribute *attr*:

$$\mathrm{SPLIT}\sigma, attr = \sigma' = \langle \sigma_1, \sigma_2, \ldots, \sigma_n \rangle$$

such that:

- $\sigma_1 \circ \sigma_2 \circ \ldots \circ \sigma_n = \sigma$,
- $\forall 1 \leq i < n : \pi^o_{attr}(\sigma_i(1)) \neq \pi^o_{attr}(\sigma_{i+1}(1))$.

The function

$$\mathrm{MERGE} \colon {\mathcal{E}_o}^{**} \to \mathcal{U}^*_{attr}$$

consolidates consecutive sequences produced by SPLIT:

$$\mathrm{MERGE}\sigma, attr = \sigma_{attr} \in \mathcal{U}^*_{attr},$$

where

$$\forall 1 \leq i \leq |\sigma| \colon \sigma_{attr}(i) = \pi^o_{attr}(\mathrm{SPLIT}\sigma, attr(i)(1)).$$

Let $DN = (R, E, \mathrm{tt})$ be the network and $attr_{res} \in \mathrm{dom}(\#(r))$, for any $r \in \mathcal{U}_{res}$, where $\forall r \in \mathcal{U}_{res} \colon \#_{attr_{res}}(r) \neq \perp$. The aggregation of *DN* based on $attr_{res}$, denoted by $\mathrm{AGGRE}DN, attr_{res}$, returns a graph $\mathrm{AGGRE}DN, attr_{res} = (V, E')$, where $V = \{\#_{attr}(r) \mid r \in R\}$ and $E' = \{(R_1, R_2) \in V \times V \mid \exists (r_1, r_2) \in E \colon \#_{attr}(r_1) = R_1 \wedge \#_{attr}(r_2) = R_2 \wedge R_1 \neq R_2\}$.

Definition 26 (Aggregation of a distribution network) Let $DN = (R, E, \mathrm{tt})$ be a distribution network, and let $attr_{res} \in \mathrm{dom}(\#(r))$ for any $r \in \mathcal{U}_{res}$, where $\forall r \in \mathcal{U}_{res} \colon \#_{attr_{res}}(r) \neq \perp$. The aggregation of *DN* based on $attr_{res}$, denoted by $\mathrm{AGGRE}DN, attr_{res}$, returns a graph

$$\mathrm{AGGRE}DN, attr_{res} = (V, E')$$

where:

- $V = \{\#_{attr}(r) \mid r \in R\}$, and
- $E' = \{(R_1, R_2) \in V \times V \mid \exists (r_1, r_2) \in E \colon \#_{attr}(r_1) = R_1 \wedge \#_{attr}(r_2) = R_2 \wedge R_1 \neq R_2\}$.

The resource hierarchy serves as a reference for abstracting an operational trace and the network to a higher level. Utilizing this hierarchy, we correct the order of merged events

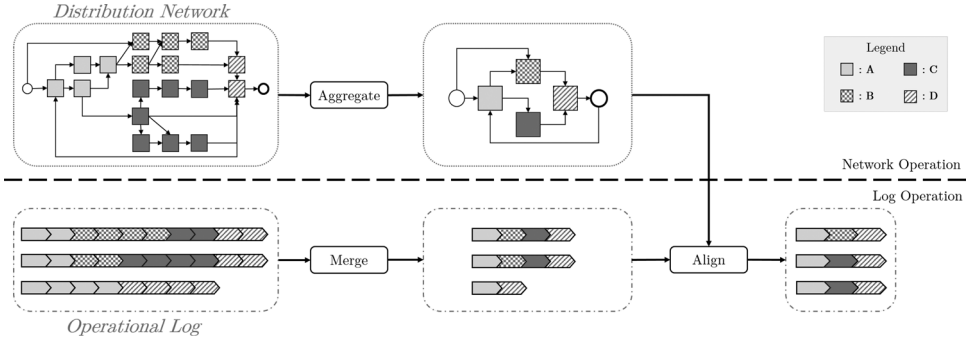


Fig. 7 Example of trace repair based on the custom cost definition

based on the aggregated network. We then expand both the merged event and the aggregated node down to the lower levels. This process is recursively repeated until we reach the base level of the hierarchy. The algorithm works in three key steps:

1. **Top-Down Alignment:** The process begins by merging events in the raw trace to match the most abstract level of the resource hierarchy. This creates a high-level, aggregated trace. This aggregated trace is then aligned with the simplified network, correcting the order of events and filling in missing steps.
2. **Recursive Refinement:** For each event in the aligned, high-level trace, the algorithm expands that event back down to the next lower level of the hierarchy. It then performs the same alignment and repair process on the corresponding lower-level events. This step is recursively repeated until the algorithm reaches the base level.
3. **Final Output:** The final output is a fully aligned event log at the lowest level of detail, with all events correctly ordered and linked to their specific resources for subsequent process mining analysis.

We repeatedly move down the resource hierarchy from the highest level. At each level, we merge events to match the current abstraction, align the merged sequence with the correspondingly aggregated network, and then expand each aligned event back into its underlying subsequence for further refinement. Inserted events are handled by aligning an empty subsequence, while existing events trigger a recursive call on the actual subsequence from which they were merged. This continues until the algorithm reaches the base level, where the fully detailed alignment is performed, and all refined subsequences are concatenated to produce the final, lowest-level repaired trace.

The alignment algorithm aims to identify the optimal path based on costs. Typically, the cost function δ assigns a value of 1 for both log and model moves and 0 for synchronized moves. However, applying this cost function to the abstracted log and network may obscure the actual distribution of events. For instance, as illustrated in Fig. 7, the first two traces are merged into an identical flow of resource paths, A, B, C, and D. However, based on the underlying events, the desired alignment is for the first trace to follow the path A, B, D, while the second trace should align with the path A, C, D. Assigning a cost of 1 to misalignments fails to accurately represent the underlying resource utilization, overlooking the actual occurrence frequency of events. Hence, we define two cost functions for alignments based on the abstracted log and network, incorporating utilization rates for each abstracted entity. This ensures that the functions reflect the underlying resource utilization within each abstracted entity.

Algorithm 1 Align_Repair

Require: $\sigma \in X^*$, $G = (Y, E, \pi^Y)$, Cost function δ, $attrs \subseteq \mathcal{U}_{attr}$
Ensure: $\sigma_{aligned} \in \{val \in \mathcal{U}_{attr} \mid attr \in attrs\}^*$

```
MM ← ⟨ ⟩                                   ▷ Model moves with the position to insert
LM ← { }                                   ▷ Position to remove based on log moves
σ_moves ← ALIGN(σ, G, δ, attrs)
for 1 ≤ k ≤ |σ_moves| do
    ((x, i_k), v) ← σ_moves(k)
    if ((x, i_k), y) is a model move then
        m ← max({i_j | 1 ≤ j < k : σ_moves(j) is not a model move})
        if m = ⊥ then
            m ← 0
        end if
        MM ← MM∘⟨(m, {π^Y_attr(y) | attr ∈ attrs})⟩
    end if
    if ((x, i_k), y) is a log move then
        n ← |{j | 1 ≤ j < k : σ_moves(j) is a model move}|
        LM ← LM ∪ {i_k + n}
    end if
end for
return DELETE(INSERT(σ, MM), LM)
```

Let OL be an operational log, DN be the distribution network, and $RH \in \mathcal{P}(\mathcal{U}_{res})^*$ be a resource hierarchy constructed from a sequence of attributes $\sigma_{attr} \in \mathcal{U}^*_{attr}$. Given $\sigma \in OL$, for any $1 < i \leq |\sigma_{attr}|$, let $\sigma_{merge} = \text{MERGE}\sigma, \sigma_{attr}(i)$ and $G = \text{AGGRE}DN, \sigma_{attr}(i)$. Let $\sigma' = \langle \#_{\sigma_{attr}(i)}(\pi^o_{res}(\sigma_1(1))), \#_{\sigma_{attr}(i)}(\pi^o_{res}(\sigma_2(1))), \ldots, \#_{\sigma_{attr}(i)}(\pi^o_{res}(\sigma_{|\sigma_{merge}|}(1)))\rangle$ be the sequence of values of $\sigma_{attr}(i)$; we define δ for every $((R_L, k), R_M) \in \text{ALIGN}(\sigma', G, \delta)$ through two implementations:

- **Hierarchical cost**, which measures the percentage of non-utilized resources at the lower level:

$$\delta((R_L, k), R_M) = 1 - \frac{|\{\#_{\sigma_{attr}(i-1)}(\pi^o_{res}(oe)) \mid \sigma'' \in \sigma_{merge}(k) \colon oe \in \sigma''\}|}{|\{\#_{\sigma_{attr}(i-1)}(r) \mid R \in RH(i) \colon \lambda(R) = R_M \colon r \in R\}|}$$

- **Flatten cost**, which computes the percentage of non-utilized resources within the abstraction entity:

$$\delta((R_L, k), R_M) = 1 - \frac{|\{\pi^o_{res}(oe) \mid \sigma'' \in \sigma_{merge}(k) \colon oe \in \sigma''\}|}{|\{r \in R \mid R \in RH(i) \colon \lambda(R) = R_M\}|}$$

The second cost, the flatten cost, provides a more granular perspective of resource utilization during the alignment process.

Timestamp repair

Let $DN = (R, E, \text{tt})$ be the distribution network, M_r be the probabilistic dwell time model for *r*, and $M_{(r_1, r_2)}$ be the probabilistic transition time model from r_1 to r_2, where $(r_1, r_2) \in E$. Let $\mathcal{M}$ denote a set of duration distribution models. Sampling from a model $M \in \mathcal{M}$ is defined by the function

$$\text{SAMPLE}\colon \mathcal{M} \to \mathcal{U}_{dur} \times \mathbb{R}_{\geq 0},$$

where $\text{SAMPLE}M = (dur, prob)$ returns a tuple whose first element is a sampled duration and whose second element is the probability of *dur* in *M*.

Let $\sigma \in (\mathcal{U}_{item}, \mathcal{U}_{res}, \mathcal{U}_{msg} \times \mathcal{U}_{time} \cup \{\perp\})^*$ be an aligned trace. Let $\{i, i+1, \ldots, i+n\}$, with $i+n \leq |\sigma|$, denote the positions of consecutive inserted events in σ, such that

$$\forall i \leq k \leq i+n\colon \pi^o_{ts}(\sigma(k)) = \perp,$$

and if $i > 1$ then $\sigma(i-1) \neq \perp$, and if $i+n < |\sigma|$ then $\sigma(i+n+1) \neq \perp$. Let

$$\sigma' = \langle \sigma(i), \sigma(i+1), \ldots, \sigma(i+n) \rangle$$

be the sequence of inserted events extracted from σ, and let

$$\sigma_s = \langle (dur_1, prob_1), (dur_2, prob_2), \ldots, (dur_{i+1}, prob_{i+1}) \rangle$$

be a sequence of samples for σ'.

To ensure that the operational events are ordered chronologically according to the corrected sequence, we optimize the sample probabilities over the inserted events:

$$\Pi_{j=1}^{|\sigma_s|} prob_j,$$

while respecting the *available duration* constraints: if $i > 1$ and $i+n < |\sigma|$,

$$\Sigma_{j=1}^{|\sigma_s|} dur_j = \pi^o_{ts}(\sigma(i+n+1)) - \pi^o_{ts}(\sigma(i-1)),$$

otherwise

$$\Sigma_{j=1}^{|\sigma_s|} dur_j \geq 0.$$

We then inject the timestamps for every inserted event in the aligned trace. Let $\sigma \in \mathcal{E}_o^*$ be the resulting operational trace. For each $1 \leq i \leq |\sigma|$, if $i\%2 = 1$, we instantiate a movement event

$$(\pi^o_{itm}(\sigma(i)), \pi^o_{res}(\sigma(i)), \pi^o_{ts}(\sigma(i)), \pi^o_{ts}(\sigma(i+1))) \in \mathcal{E}_m.$$

Consequently, a movement log is created from the collection of these movement events.

Performance analysis

We evaluate the performance of the distribution process by identifying and diagnosing the root causes of inefficiencies, and using these insights to predict future process execution. The root cause analysis involves identifying and diagnosing underlying issues that impact process efficiency. Additionally, predictive analysis supports process operations by forecasting potential delays and optimizing resource allocation.

Root cause analysis

We define bottlenecks using clear and quantifiable criteria, adaptable to various processes as a flexible yet standardized data-driven criterion to define inefficiencies. Building on this, we detect the root causes for delays in logistics. Specifically, a bottleneck is

described by a movement event where the dwell time exceeds the expected duration of the corresponding resource.

Definition 27 (Bottleneck) Let $ML \subseteq \mathcal{E}_m^*$ be a movement log. For any movement trace $\sigma \in ML$ and any $1 \leq i \leq |\sigma|$, $(i, \sigma) \in \mathbb{N} \times \mathcal{E}_m^*$ is a bottleneck if $\pi_{dep}^m(\sigma(i)) - \pi_{arr}^m(\sigma(i)) > \#_{exd}(\pi_{res}^m(\sigma(i)))$.

Once a bottleneck is identified, it is essential to investigate the underlying causes of the delay. The root cause detection determines whether the bottleneck arises from a movement event or a status event.

Definition 28 (Root Cause) Let $ML \subseteq \mathcal{E}_m^*$ be a movement log and $SL \subseteq \mathcal{E}_s^*$ be a status log. Given a bottleneck $bn = (i, \sigma(i))$, where $\sigma \in ML$ and $1 \leq i \leq |\sigma|$, the root cause of *bn* is a tuple $(j, \sigma') \in \mathbb{N} \times \mathcal{E}_m^* \cup \mathcal{E}_s^*$, where $\sigma' \in ML \cup SL$ and $\sigma'(j)$ is an incident that causes *bn*.

An item may remain idle on a resource if upstream resources are unavailable or no further resources exist on its route. A resource is unavailable if it is offline or at maximum capacity, and available otherwise. The system's status is time-dependent, and event data provides snapshots of this status. Let *ML* be a movement log and $DN = (R, E, \mathrm{tt})$ be a network. For any movement trace $\sigma \in ML$, let $me \in \sigma$ be a movement event. Given a resource $r \in R$, where $(\pi_{res}^m(me), r) \in E$, we define the function $\mathrm{REACH}\colon \mathcal{E}_m \times \mathcal{U}_{res} \to \mathcal{U}_{time}$, where $\mathrm{REACH}me, r = \pi_{arr}^m(me) + \#_{exd}(\pi_{res}^m(me)) + \mathrm{tt}_{(\pi_{res}^m(me), r)}$ is the time at which $\pi_{item}^m(me)$***should*** arrive on *r*. By checking the status of the resource to which an item is scheduled to be distributed at the planned time, we trace root causes through the interactions between distributions.

Algorithm 2 describes the identification of the root cause of a bottleneck, defined as a movement event whose dwell time exceeds the expected duration of its resource. The algorithm aims to determine the most plausible incident responsible for the delay by analyzing both the movement log, which captures resource occupation by cargo items, and the status log, which records periods of resource unavailability. The diagnostic procedure follows an operational reasoning pattern. It first evaluates whether the resource on which the bottleneck occurs is offline during the bottleneck interval. If an overlapping offline period is found in the status log, this status event is immediately identified as the root cause, as resource unavailability provides a direct explanation for the observed delay.

If the current resource is available, the algorithm examines downstream conditions along the item's planned route. For each step, it computes the planned arrival time at the next resource. If the next resource is offline, the corresponding status event is returned as the root cause. Otherwise, the algorithm assesses capacity constraints. When capacity is available, neither resource failure nor congestion explains the delay, and the bottleneck is attributed to factors not captured in the logs, such as manual interventions or implicit operational rules. If the resource is at maximum capacity, the algorithm identifies the occupying movement event with the earliest departure time. If that event itself exceeds its expected duration, the delay is classified as a cascading bottleneck and the procedure is recursively applied to the upstream item; otherwise, the congestion is treated as normal utilization. Overall, Algorithm 2 returns a single, concrete root-cause incident,

either a status event or a blocking movement event, and provides a systematic basis for the diagnostic categories introduced subsequently.

To support interpretation and operational decision-making, the root cause identified by Algorithm 2 is mapped to a set of diagnostic categories that characterize the nature of the delay.

- **Stalled on Offline Resource:** An item is halted on an offline resource, with the root cause identified in line 4 of Algorithm 2.
- **Awaiting Online Activation:** A distribution is delayed while waiting for resources on its route to be online, with the root cause identified under the conditions in line 9 to 12 in Algorithm 2.
- **Pending Exit:** A distribution is awaiting exit from the system, with the root cause identified in line 6 of Algorithm 2.
- **Cascading Bottleneck:** The distribution is delayed because the next resource on its planned route is unavailable because it is occupied by another item, which is itself delayed. This recursive process is shown in lines 22–26 of Algorithm 2.
- **Extraneous Factor:** The delay is caused by reasons not captured in the logs, such as a business decision to cancel a package during its distribution. If none of the above conditions apply, the root cause is diagnosed as an extraneous factor.

Note that a root cause may be associated with multiple contributing factors.

Predictive analysis

To enhance operational efficiency and enable proactive adjustments, we developed a predictive model that forecasts two key outcomes for each cargo item: its next resource and the time it will take to arrive. This capability is used for strategic resource management and to prevent potential issues such as bottlenecks.

Two models are evaluated to address this prediction task: a higher-order Markov chain and a Long Short-Term Memory (LSTM) network. The Markov chain serves as an interpretable baseline that captures sequential movement patterns, while the LSTM model leverages a richer set of features to capture more complex, non-linear dependencies.

Feature Engineering We categorize the features used to train our LSTM model into two main types: static and dynamic. Static features represent the inherent nature of objects that remain unchanged throughout the distribution process. In our case study, these static features include the location of resources, the types of resources involved, and the types of cargo items being transported. For example, static features might identify whether the cargo is perishable, hazardous, or fragile—factors that influence handling and storage requirements but remain constant.

Dynamic features, by contrast, are derived from the evolving status of the system during distribution. These features capture real-time changes in the system, which are essential for predicting the next movement in the sequence. We extract dynamic information from root cause analysis, using past instances of system behavior to identify patterns that could inform future states. This approach is possible because root causes are based on events and conditions that have historically impacted the distribution process, providing a basis for understanding dynamic conditions. For each movement event me, we encode the following dynamic features to predict the distribution's next state:

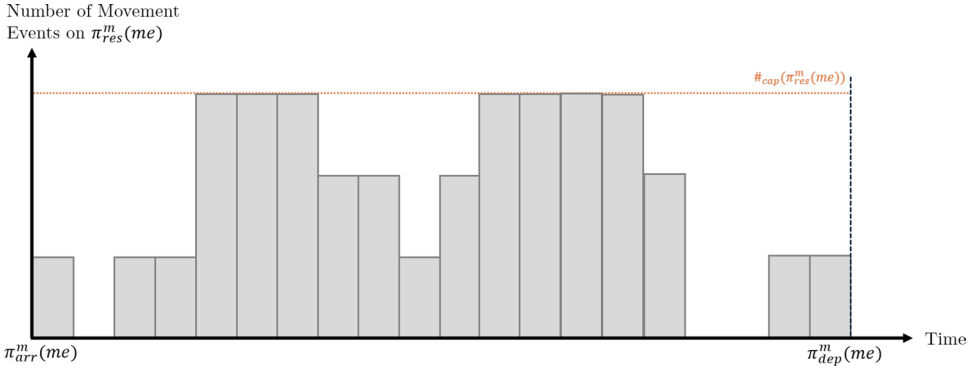


Fig. 8 Example of utilization rate

- *Capacity* ($\in \mathbb{N}$): The maximum number of cargo items that the resource associated with the movement event can accommodate, i.e., $\#_{cap}(\pi^m_{res}(me))$.
- *Bottleneck Flag* ($\in \mathbb{B}$): A binary flag, where *True* if $\pi^m_{dep}(me) - \pi^m_{arr}(me) > \#_{exd}(\pi^m_{res}(me))$.
- *Root Cause Resource*: The resource responsible for the root cause when *me* is identified as a bottleneck.
- *Root Cause Begin Time* ($\in \mathcal{U}_{time}$): The timestamp at which the root cause of a bottleneck begins.
- *Root Cause End Time* ($\in \mathcal{U}_{time}$): The timestamp at which the root cause of a bottleneck ends.
- *Root Cause Diagnosis*: A classification of the root cause.
- *Utilization Rate*: The utilization rate quantifies the proportion of time a resource is fully occupied during the movement event. It is calculated as the ratio of the time the resource is at its maximum capacity to the total duration of the event. We focus on this specific metric because a resource only truly becomes a bottleneck when it is unable to handle additional demand due to capacity constraints, which directly impacts the flow of a cargo item. Figure 8 provides an illustrative example: a movement event spans 20 time units, during which the resource is at full capacity for 7 time units. This results in a utilization rate of $\frac{7}{20} = 0.35$. This measure captures the direct impact of capacity limitations on a specific cargo item's distribution.

Prediction Models A higher-order Markov chain serves as our primary baseline model. We chose this model for its computational efficiency and high degree of interpretability, which is crucial for operational deployment. Unlike more complex models, a Markov chain provides clear, probability-based insights into common movement patterns. This transparency allows terminal managers to easily understand and trust the model's output.

The input to the Markov chain model is a sequence of resources, which is extracted from a movement trace σ_{itm} by projecting the res attribute of each movement event. We denote a resource sequence as $\sigma^{res}_{itm} = \langle \pi^m_{res}(me_1), \pi^m_{res}(me_2), \ldots, \pi^m_{res}(me_n) \rangle$. Figure 9 illustrates conceptually how to transform the movement log into a Markov chain model.

To capture dependencies beyond a single transition, we employ a higher-order Markov chain, where transition probabilities consider sequences of the previous states. This allows for modeling more complex movement patterns and improves predictive

Cargo Item	Resource	Time	Message	PM
711123	A	13:30:15	Initiate	Yes
711123	A	13:32:22	Ready	No
711123	A	13:34:13	Ready	Yes
711123	B	13:35:36	Arrived	Yes
711123	D	13:43:18	Arrived	Yes
711123	D	13:47:21	Depart	Yes
711124	A	13:37:01	Initiate	Yes
711124	A	13:45:27	Depart	Yes
711124	B	13:46:41	Decide	Yes
711124	C	13:49:52	Depart	Yes
711124	D	13:52:10	Arrived	Yes
711124	D	13:54:15	Depart	Yes

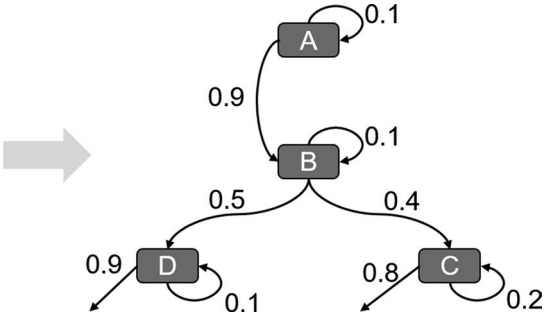


Fig. 9 An example of transforming the operational log to a Markov chain

reliability. Formally, a Discrete-Time Markov Chain (DTMC) is defined by a set of states S and a transition probability matrix P. In our context:

- The set of states S corresponds to all unique resources in the distribution network.
- For a higher-order Markov chain, a state is defined by a sequence of the last k resources visited. The transition probability for a cargo item to move from one state to the next is calculated directly from the historical movement log ML:

$$P(\pi^m_{res}(me_{i+1})|\pi^m_{res}(me_i),\ldots,\pi^m_{res}(me_{i-k+1})) = \frac{\text{count}(\pi^m_{res}(me_{i-k+1}) \to \ldots \to \pi^m_{res}(me_i) \to \pi^m_{res}(me_{i+1}))}{\text{count}(\pi^m_{res}(me_{i-k+1}) \to \ldots \to \pi^m_{res}(me_i))}$$

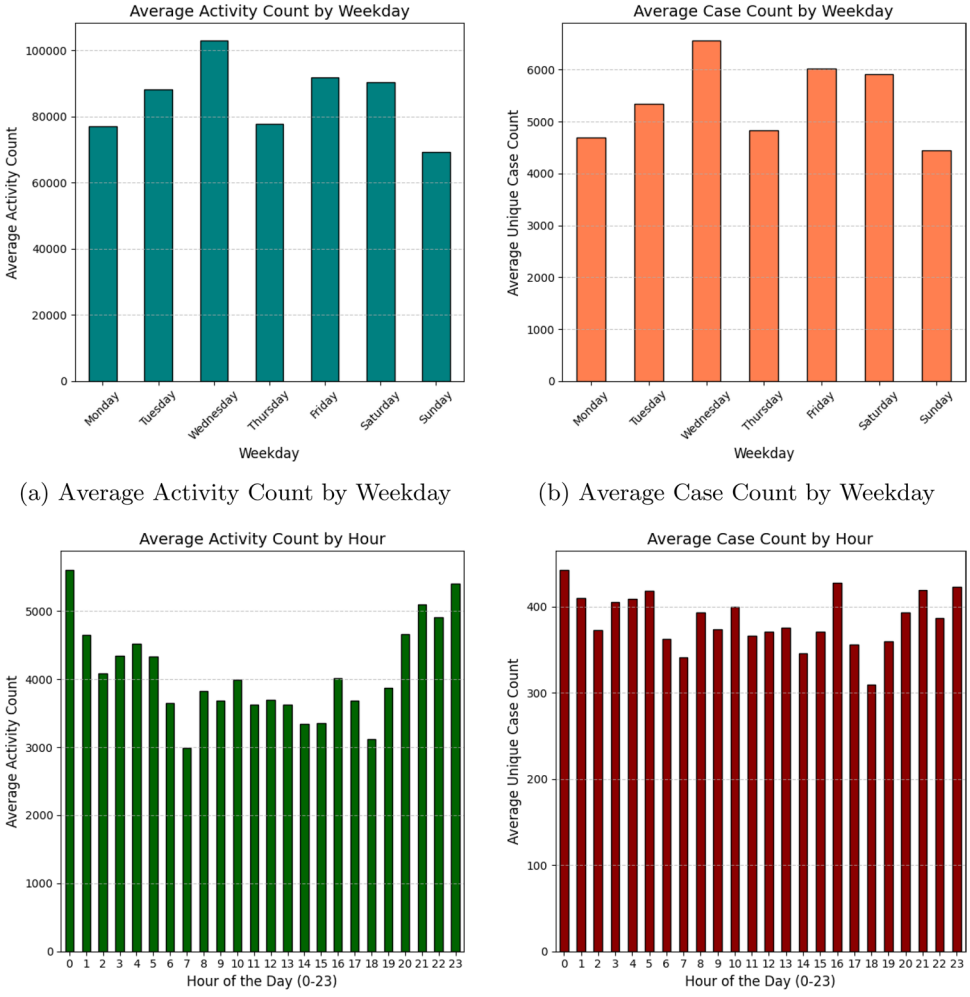
A key limitation of the Markov chain model is that its performance degrades significantly when incorporating the numerous static and dynamic features we identified, as this would lead to an extremely sparse state-transition matrix. Therefore, the Markov chain model only uses the historical resource sequence as its input.

To complement the Markov chain and leverage the rich set of static and dynamic features, we implemented an LSTM network. As a type of recurrent neural network, LSTMs are well-suited for sequence prediction tasks where capturing long-term dependencies is important. They can also effectively integrate additional feature data at each step of the sequence.

The LSTM model takes a sequence of historical movement events from a movement trace σ_{itm} as input. At each step, the model receives a feature vector with static and dynamic features discussed above. The hidden states of the LSTM learn to capture complex, non-linear relationships among these features and the sequence itself, potentially allowing for a more nuanced prediction than the Markov chain.

The predictive model's output may serve as the foundation for proactive operational management. A terminal can forecast the exact flow of items, enabling the optimization of resource allocation. For example,

- Prevent Bottlenecks: The models can identify potential chokepoints. For example, if a specific conveyor belt is predicted to have an increased dwell time, the system can automatically reroute packages to an alternative path to maintain a smooth flow.
- Optimize Sorting and Queuing: Insights from the models can enable dynamic queuing. High-priority packages can be prioritized for immediate processing, while others can be strategically delayed to optimize overall throughput.



(a) Average Activity Count by Weekday

(b) Average Case Count by Weekday

(c) Average Activity Count by Hour

(d) Average Case Count by Hour

Fig. 10 Average activity and case counts grouped by Weekday and hour. (**a**) Average activity count by Weekday. (**b**) Average case count by Weekday. (**c**) Average activity count by hour. (**d**) Average case count by hour

Case study

We analyze a real-world logistics dataset from a major cargo distribution terminal in Asia, comprising 21,341 operational traces collected from 4,839 devices over 15 consecutive days of continuous operation. While this horizon does not capture long-term seasonal effects, it spans multiple weekdays and weekends, thereby including typical variations in operational load and device usage patterns.

To give the reader a brief overview, Figure 10 provides the data volume in terms of average activity (resource) and case counts grouped by weekday and hour. The average daily volume of activities, shown in Fig. 10a, indicates that the busiest day is Wednesday, with an average of 103,112 events. Activity volume is lowest on Sunday (69,265 events). Notably, Saturday maintains a relatively high volume (90,432 events), suggesting significant weekend operations. The average number of unique cases processed, shown in Fig. 10b, follows a highly similar pattern to the activity count. Wednesday also registers the highest average number of unique cases (6,570), confirming it as the peak day for both resources involved and the number of distinct processes being handled.

When grouped by the hour of the day, the average total activity count (Fig. 10c) is dominated by nightly automation. Activity peaks around the midnight hour (Hour 0) with an average of 5,605 events, and remains high late at night (Hour 23). The lowest activity occurs around 7 AM (2,989 events), marking the end of the major overnight batch processing window. The average distinct case count by hour (Fig. 10d) shows a much more stable profile throughout the day. While the peak still aligns with the automated peak at midnight (443 cases), the counts during standard working hours are relatively constant.

This case study evaluates (1) the effectiveness of our data processing and (2) the analysis of the performance.

Data processing

Our data processing pipeline consists of two stages: (1) physical movement detection from raw event data, and (2) log alignment with the network. This section evaluates both components sequentially—first establishing the accuracy of movement detection as the foundation for subsequent alignment quality assessment.

Physical movement detection

Since direct validation of movement event identification is impossible without ground truth tracking, we establish a duration-based validation approach using the most reliable event sequences available—specifically cases where a device's arrival is immediately followed by its departure. These consecutive pairs yield accurate duration measurements that domain experts have verified as representing clean operational patterns free from noise. We construct our operational ground truth from 21,400 validated pairs across device types, forming duration distributions that serve as reference benchmarks.

The baseline for each device type is defined based on its characteristic behavior, as informed by domain experts—for example, elevators operate in a specific manner, whereas rotational devices behave differently.

To determine suitable duration distributions for each device type, we first applied exploratory statistical tools to identify candidate parametric distributions (e.g., Gamma, Weibull). Empirical analysis, however, revealed multi-modal operational patterns that cannot be adequately captured by a single standard distribution. We therefore use a Gaussian Mixture Model (GMM), which provides the flexibility to model multiple modes for each device type, closely matches the observed data, and enables realistic stochastic generation of durations for timestamp reconstruction. Sampling from the fitted GMMs follows the standard procedure in scikit-learn: the model first selects a Gaussian component according to its learned mixture weight, and then draws a duration from the corresponding Gaussian distribution.

Earth Mover's Distance (EMD) quantifies the alignment between the time of the extracted physical movement events and reference distributions, measuring the minimal temporal displacement required to match their characteristics. This metric accommodates varying sample sizes while preserving the temporal relationships—lower values indicate better preservation of the duration patterns observed in the validated event sequences. Due to sparse event counts per device, we aggregate at the device-type level, justified by operational homogeneity confirmed through statistical analysis and domain expertise.

To further isolate the contribution of our matching solver, we also implement a simple greedy method using the same candidate windows and scoring as the ILP-based matching. For each start event, the highest-scoring corresponding end event within the defined window is selected, ensuring that events are not matched more than once. This greedy ablation allows us to directly compare the ILP solver with a simpler heuristic under identical conditions.

Table 4 reveals distinct patterns in physical movement detection accuracy across device types, with the matching method demonstrating superior performance for most categories. The most significant improvement occurs for stacker cranes (SC), where EMD reduces by 71.8% (from 0.618 to 0.174), indicating exceptional handling of their variable movement patterns during package stacking operations. Similarly strong results emerge for pallet routers (PR) and rotation angles (RA), both achieving near-perfect alignment (EMD < 0.02), confirming their highly regular movement patterns in operational environments. The cargo hoists (CH) present a challenging case with persistently high EMD values (1.000 ad-hoc vs 0.614 matching), as these vertical transport buffers frequently experience irregular dwell times during path conflicts, queuing delays, and priority handling scenarios—though the matching method's 38.6% improvement demonstrates better accommodation of these operational variabilities.

Comparing the two approaches with the greedy method shows that greedy offers clear improvements over the ad-hoc baseline, particularly for SC and TT, where EMD decreases from 0.618 to 0.037 and from 0.147 to 0.069, respectively. However, greedy still falls short of the matching method for most device types. This is expected, as greedy decisions optimize locally rather than globally, leading to less accurate alignment in categories with more complex or irregular movement patterns. Overall, greedy provides a reasonable middle ground—substantially better than ad-hoc estimation but consistently outperformed by the matching approach.

This duration-based validation framework offers a robust solution when direct validation is impractical, enabling quantitative evaluation of how effectively different methods capture genuine operational patterns. The results reveal that while the ad-hoc approach can help identify and address localized data quality issues through manual inspection and correction, its reliability diminishes with dataset scale—systematically detecting and correcting problems becomes infeasible in large datasets. In contrast, the matching method demonstrates consistent superiority, providing more reliable results while being fundamentally scalable.

Movement log extraction

Building on the superior performance of the matching method from our previous experiment, we use its output logs for this evaluation. Due to the prohibitive computational cost of standard alignment, we restrict our analysis to cases with trace lengths less than or equal to 26 events, covering 3,044 cases representing 2,292 unique variants—demonstrating the inherent complexity of the logistic log. We evaluate two hierarchical alignment cost definitions: (1) hierarchical cost and (2) flatten cost. Both approaches achieve comparable similarity to standard alignment (37.2%), with the flatten cost showing a marginal 0.17% advantage. Figure 11 reveals the dramatic runtime differences, highlighting the scalability and efficiency of our hierarchical approach. While a standard alignment, non-hierarchical alignment (naive) frequently exceeds 100 seconds per trace, both

Table 4 EMD and Sample size by device type

	CH		PR		RA		SC		TP		TT		TV	
	EMD	**N**	**EMD**	**N**	**EMD**	**N**	**EMD**	**N**	**EMD**	**N**	**EMD**	**N**	**EMD**	**N**
Ad-Hoc	1.000	8,166	0.029	177,233	0.029	49,968	0.618	32,786	0.024	2,879	0.147	9,464	0.012	11,670
Matching	0.614	7,574	0.006	30,035	0.019	6,445	0.174	17,415	0.043	4,012	0.024	1,277	0.017	13,684
Greedy	0.199	8,406	0.029	71,020	0.028	25,529	0.037	27,434	0.088	2,321	0.069	5,095	0.047	15,374

hierarchical methods complete in under 1 second most of the time—demonstrating their practical viability for operational deployment. This fundamental difference in scalability highlights the timeliness and resource efficiency with hierarchical approaches under operational constraints.

Performance analysis

This section demonstrates the analysis using our framework. First, we analyze the root causes of the bottlenecks; then, we present the results of the predictive analysis.

Root cause analysis

Figure 12 shows the bottleneck frequency by diagnosis. Extraneous factors cause 60% of all bottlenecks. To provide a more granular diagnosis, we further distinguish between two types of *pending exit* root causes: those where the item waiting on the exiting resource is itself a bottleneck, and those where it is not. Our preliminary analysis identifies two likely reasons: unexpected manual workflow interruptions and mismatches between configured dwell times and actual processing requirements.

We distinguish between two categories of pending exits based on their operational context. The first type, standard pending exits, occurs during normal processing operations prior to reaching the system boundary. These typically represent temporary workflow delays that can be addressed through operational improvements such as dynamic route optimization or enhanced load sequencing protocols. The second category, which we classify as terminal pending exits (bottlenecks), occurs specifically at the system's exit boundary. These cases represent cargo that has completed all processing requirements but encounters finalization barriers. This more severe category indicates fundamental synchronization issues in our distribution batch management, requiring comprehensive recalibration of the entire batch entry/exit scheduling system. Figure 12 reveals that bottlenecks as pending exit have the greatest operational impact compared to other identified root causes. The observation suggests a fundamental redesign of the batch scheduling system to resolve the misalignment between inbound arrivals and outbound departures.

Meanwhile, cascading bottlenecks further underscore how poor upstream decisions may propagate delays downstream. Addressing these through predictive scheduling or dynamic resource allocation could yield significant efficiency gains.

We further analyze the number of cases involved in Fig. 13. The close alignment between the distribution of bottleneck events per root cause and the distribution of affected cases per root cause reveals a fundamentally consistent pattern of operational disruptions. The uniformity of impact across these metrics implies that solutions should focus on comprehensive process improvements addressing entire workflows, rather than targeted local fixes.

Finally, Fig. 14 presents the bottleneck duration distribution per diagnosis, quantifying the impact on efficiency in the system. The analysis reveals that pending exit (bottleneck) is the most severe contributor, with prolonged delays occurring when cargo awaits system exit. This again underscores how synchronization failures in batch scheduling disproportionately degrade overall system performance.

While less frequent, as shown in Fig. 12, device status-related bottlenecks (online activation delays and stalled on offline resource) are particularly disruptive due to their

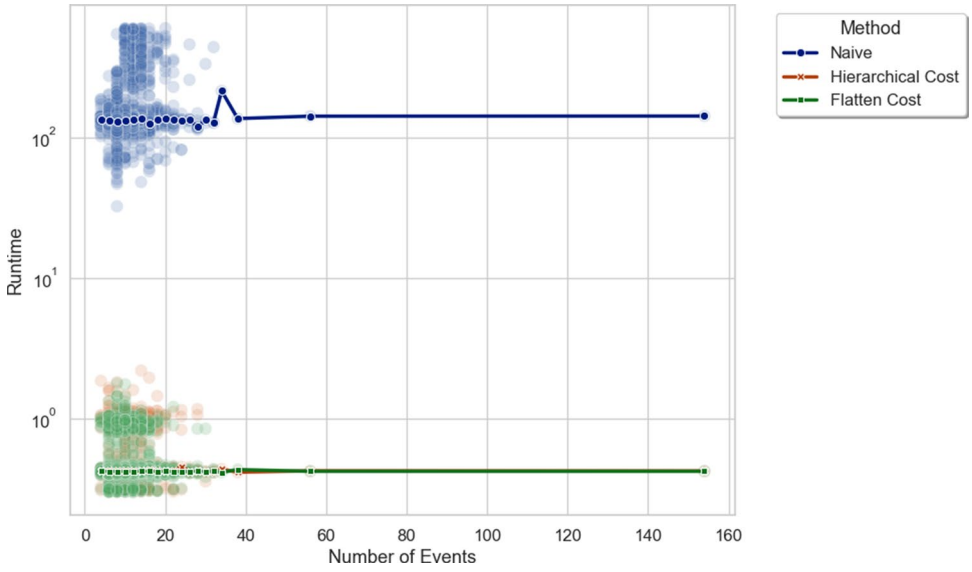


Fig. 11 Comparison of alignment method runtimes showing inefficiency of standard alignment (naive) versus near-constant time hierarchical approaches (logarithmic time scale in seconds). The lines are the median runtime for the alignment per number of events in a case

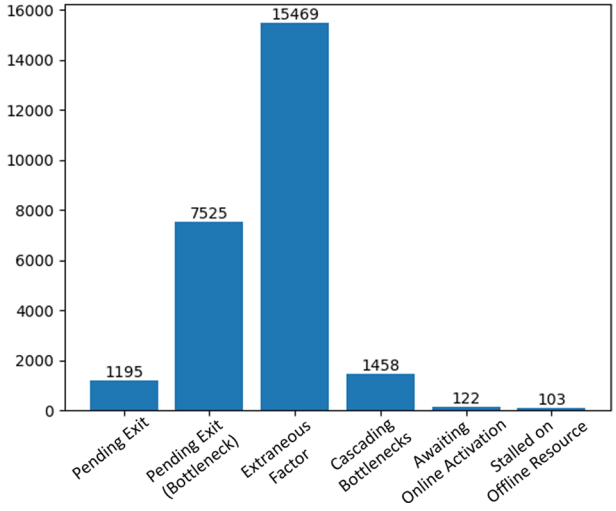


Fig. 12 Bottleneck frequency by diagnosis

dependence on manual resolution. These incidents, especially stalled offline resource, exert significant temporal performance penalties, emphasizing the need for automated monitoring and failover mechanisms to mitigate downtime.

Predictive analysis

In this section, we present the results of our two predictive models: a higher-order Markov chain and an LSTM network. We compare their performance on flow prediction (next resource) and duration estimation (time to the next resource) to highlight the trade-offs between model complexity and predictive accuracy.

Markov chain results The higher-order Markov chain was trained to forecast the next resource and the duration based solely on the sequence of past resources without the

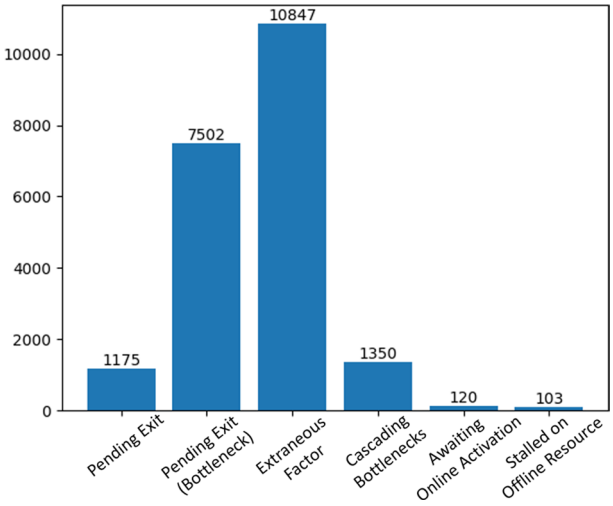


Fig. 13 Case count across different diagnoses

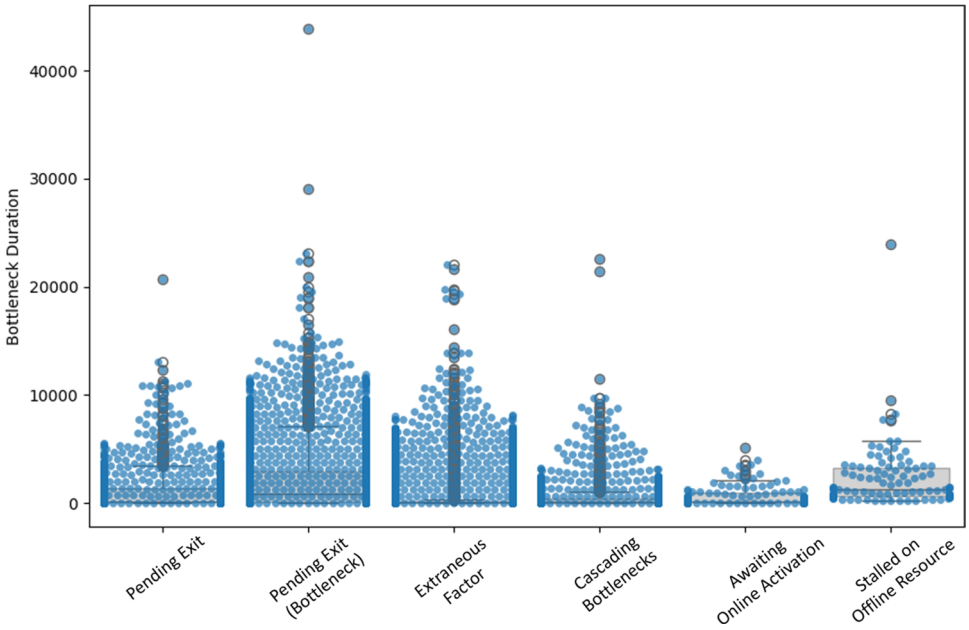


Fig. 14 Bottleneck duration (in seconds) by diagnosis

dynamic factors. We systematically evaluated different orders and selected the optimal configuration based on a trade-off between prediction accuracy and computational efficiency. We estimate the time until the next transition by analyzing historical duration data. In particular, the Markov chain model estimates the duration based on the historical time packages taken to travel along the same route as the to-be-predicted case, with the length of the historical sequence determined by the model's order. Eventually, a 5th-order Markov chain, which considers the last five resources in a sequence, was identified as the optimal configuration for this task. The Markov chain model demonstrated strong performance in flow prediction, surprisingly outperforming the more complex LSTM baseline.

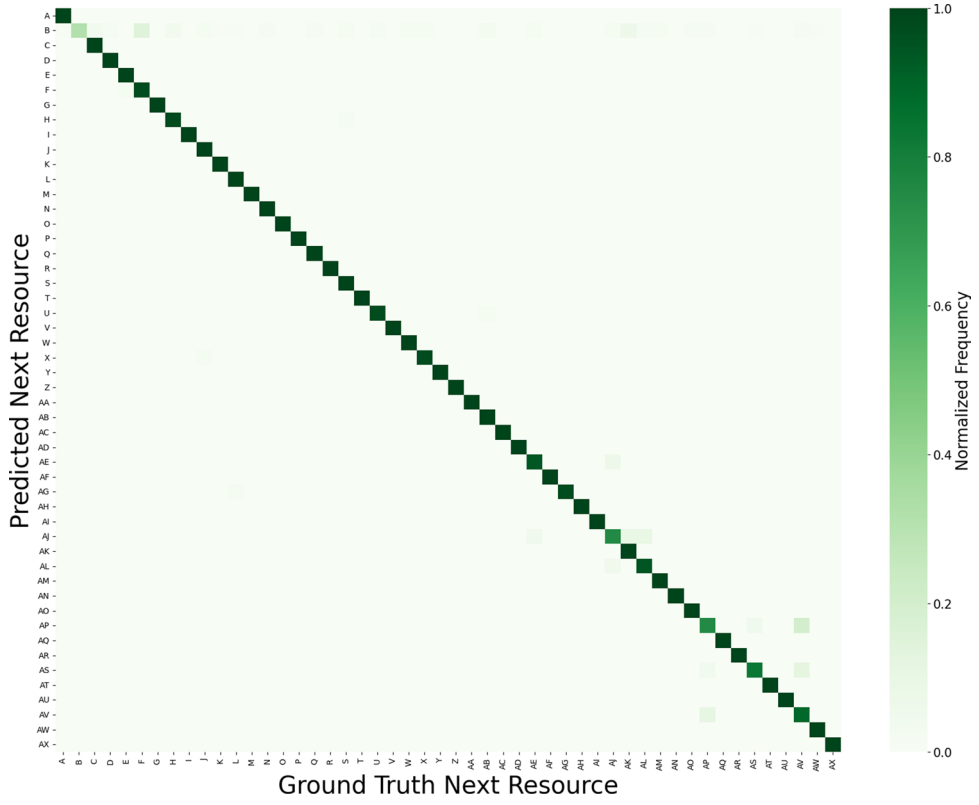


Fig. 15 The normalized confusion matrix of the top 50 most frequent resources

Dynamic factors, such as bottleneck indicators and utilization rates, adjust the transition likelihoods based on real-time system conditions. Note that the Markov chain model can be continuously updated as new data is collected. This enables real-time forecasting, facilitating proactive resource management and optimization of cargo movement within the distribution network.

As a result, the Markov chain model demonstrates strong performance in flow prediction tasks. Using the test data, we observed the following:

- Flow prediction: The Markov chain model achieves a prediction accuracy of 77.51%.
- Duration prediction: The Markov chain model yields a Mean Absolute Error (MAE) of 142.73 seconds.

Figure 15 shows the confusion matrix of the system's top 50 most frequent resources of the Markov chain model. These results indicate that the Markov chain effectively captures the essential sequential dependencies and transition patterns, providing sufficient accuracy for a classification task involving approximately 5,000 labels (resources). Nevertheless, the duration prediction shows room for improvement.

LSTM results

We tested an LSTM model as a more sophisticated baseline to see if it could leverage additional static and dynamic features to improve performance. While the LSTM excelled at duration prediction, it surprisingly fell short of the Markov chain in forecasting the next resource.

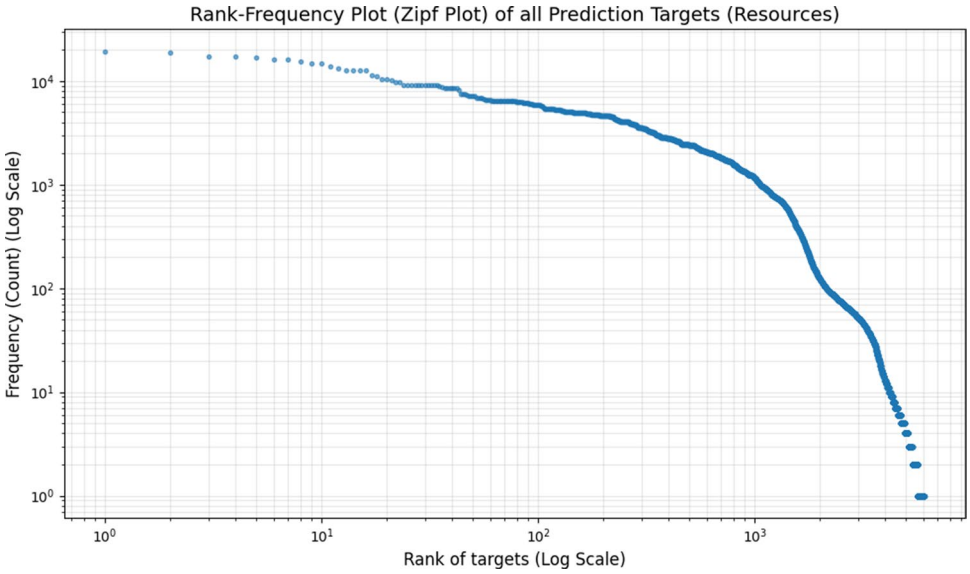


Fig. 16 Rank-frequency Plot (Zipf Plot) of all prediction targets (resources). The log-log scale confirms a heavy-tailed distribution, where the majority of targets are extremely rare (occur only once or twice)

- Flow Prediction: The LSTM model achieved a prediction accuracy of 62.57%, significantly lower than the Markov chain's performance.
- Duration Prediction: The LSTM model yielded an MAE of 26.63 seconds, a remarkable 81% reduction in error compared to the Markov chain.

Discussion and practical implications Our findings present an interesting insight: A simpler, sequential-based model appears to outperform a feature-rich deep learning model on this complex classification task.

The primary rationale seems to lie in the inherent data characteristics: the flow prediction task involves high target cardinality ($\approx \mathbf{6,000}$ unique labels) and extreme class imbalance. As visually confirmed by the Rank-Frequency Plot (Zipf Plot) in Fig. 16, the class distribution is extremely skewed, with the most frequent target occurring over 10^4 times, while thousands of targets occur only once.

The strength of the Markov chain may reside in its direct and deterministic capture of short-term dependencies, and, critically, its efficiency in exploiting this imbalance. The high overall accuracy of the Markov Chain is likely attributable to its ability to correctly predict the small number of dominant, high-frequency targets, which constitute the majority of the data. This tendency to prioritize the most stable, frequent transitions suggests a reason why the simpler, frequency-based Markov chain can outperform the more complex LSTM in this specific classification task. This complex label space may have challenged the LSTM, whose complexity may have inhibited its ability to generalize effectively across the entire, massive, and highly imbalanced label space, leading to lower performance on the flow prediction task. In contrast, the more accurate result of the LSTM in predicting duration is notable. Its ability to process detailed temporal patterns and contextual features, such as resource utilization and capacity, suggests it is better suited to model the subtle factors that influence an item's time in a resource.

This suggests a hybrid approach could yield the best of both worlds. The Markov chain could be used for highly accurate next-resource predictions, while the LSTM could

simultaneously be used to provide precise duration estimates. This combined model would take advantage of the strengths of each approach, translating directly into more effective resource allocation and better proactive management of cargo flow within a distribution network.

Among the possible application scenarios discussed in subsection 5.4.2, the development of a routing recommendation system is being investigated. As discussed, when predictive models forecast a significant increase in arrival volumes in a specific sorting area, a dashboard automatically flags the resource, prompting the floor manager to redistribute staff or activate an additional sorting line. To be even more proactive, alternative routes could be suggested based on the models' predictions of reduced dwell times, leveraging the strengths of both the Markov chain and the LSTM.

Limitations & threats to validity

This paper presents a novel methodology for applying process mining to automated, network-constrained systems, with a focus on air cargo distribution. We bridge the gap between theoretical process mining techniques and their practical application in a data-rich, yet often computationally challenging, real-world setting. In this section, we discuss the implications and potential limitations of our approach, particularly regarding the scale of the data, physical movement detection, root cause diagnosis, and performance prediction, while also addressing the generalizability of our framework.

Dataset scope and generalizability

Our study relies on a dataset of 21,341 operational traces from a single terminal collected over a 15-day period. While this dataset provides a detailed snapshot of operations, the limited horizon poses several important constraints. First, the 15-day window is too short to capture long-term trends, seasonal effects, or atypical operational conditions that may occur outside this period. Second, although it includes multiple weekdays and weekends, it may not fully represent variability in peak loads, device usage patterns, or rare events that could significantly influence predictive performance. Third, the short observation period limits our ability to assess how the models would generalize to periods with unusual operational cycles, maintenance schedules, or extraordinary traffic volumes. These limitations mean that the observed superiority of the Markov chain for next-resource prediction and the LSTM for duration may partly reflect patterns specific to this limited timeframe, rather than general operational behavior. Overall, while the dataset allows for an initial evaluation of predictive methods, caution is warranted when extrapolating these findings to longer horizons, different terminals, or broader operational contexts. Future research should consider longer and multi-terminal datasets to more robustly evaluate model performance and external validity.

Transforming sensor data to event data

One of the key challenges we addressed was the need to transform imperfect real-world sensor data into event data for process mining. Our methodology relies on aligning an incomplete log with a known network model to reconstruct the true process flow. In our case study, this process was simplified by the fact that cargo items consistently followed the shortest route, allowing us to use a cost function based on edit distance. However, this assumption may not hold true in other systems where items might traverse a path

multiple times or follow non-optimal routes. To address this, we introduced a customized cost function that incorporates the utilization rate to guide the alignment towards the most probable path.

Due to the length and complexity of some traces, alignment was validated by randomly sampling traces and manually verifying their consistency with the reference model in the case study. A more systematic validation approach, which is capable of rigorously assessing correctness and addressing the scalability challenges of large-scale alignment, is left for future work. Meanwhile, in the absence of ground-truth timestamps, timestamps can either be assigned from the closest observed events or generated via temporal simulation. We adopt the latter, producing synthetic timestamps that reflect device-specific operational characteristics, ensuring that the repaired traces are temporally plausible while preserving the integrity of the event sequence. Although absolute correctness cannot be fully verified without ground truth, the combination of model-consistent alignment and distribution-based temporal reconstruction ensures that the resulting traces are both semantically valid and operationally plausible for downstream analysis.

Threats to validity of the cost function

A key threat to validity lies in the generalization of this approach. While our cost function is effective for our system, which prevents items from using the same path more than once, a more dynamic cost assignment model would be required for systems with complex, repetitive pathways. It is possible that our repair methods, while pragmatic, could introduce some level of distortion. However, we contend that this is a necessary trade-off. Our primary goal is to transform incomplete event logs into a state that allows for meaningful, data-driven analysis. By providing a clear and documented method for this data repair process, we offer a valuable blueprint for researchers and practitioners facing similar data quality issues.

In addition, the choice of utilization-weighted cost can directly influence alignment decisions. Consider a simplified hierarchy in which each floor contains ten zones and each zone contains ten devices, with device identifiers encoded as `DXYZ` (floor X, zone Y, device Z). Suppose a synthetic movement trace σ, projected only to the device identifiers, as shown below, in which the second floor uses all twenty devices across two zones while the third floor uses only five devices spread across five distinct zones. After abstraction to the floor level, we obtain the sequence $\sigma = \langle F1, F3, F2, F4 \rangle$. Figures 17 and 18 illustrate the corresponding cost for floor 2 and floor 3, where the exclusive decision needs to be taken during alignment. Figure 18 illustrates the corresponding hierarchical and flatten costs for each floor.

$$\sigma = \langle \begin{array}{l} \mathrm{D100}, \mathrm{D110}, \ldots, \mathrm{D190}, \mathrm{D200}, \mathrm{D201}, \mathrm{D202}, \ldots, \mathrm{D209}, \mathrm{D210}, \mathrm{D211}, \ldots, \mathrm{D219}, \\ \mathrm{D300}, \mathrm{D310}, \mathrm{D320}, \mathrm{D330}, \mathrm{D340}, \mathrm{D400}, \mathrm{D410}, \ldots, \mathrm{D490} \rangle \end{array}$$

The two cost definitions lead to different alignments: hierarchical cost, emphasizing zone coverage, selects the third floor for its broader spatial engagement, while flatten cost, emphasizing device coverage, selects the second floor for its higher number of active devices. This divergence directly affects the operational interpretation of the alignment. Under the hierarchical cost, the same trace is attributed to the third floor, which can be interpreted as broader spatial engagement reflecting the actual trajectory of the cargo item across multiple zones. In contrast, under the flatten cost, the trace is

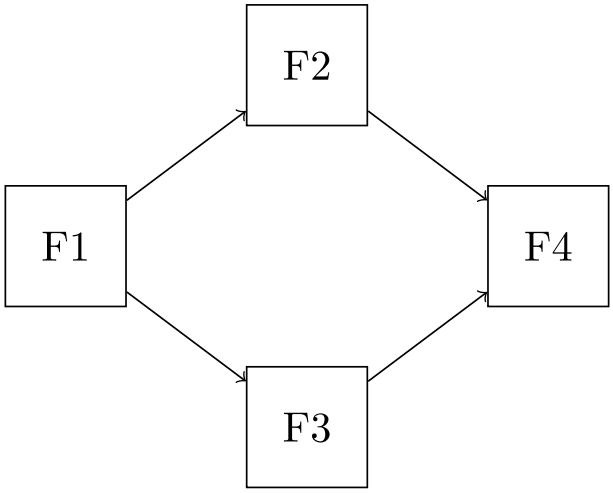


Fig. 17 Aggregated network

	F1	F3	F2	F4
HC	0	0.5	0.8	0
FC	0.9	0.95	0.8	0.9

Fig. 18 Hierarchical cost (HC) and flatten cost (FC) for motivating example

attributed to the second floor, which can be interpreted as the system initially attempting to send the cargo item to the third floor but, after several rejections, ultimately routing it to the alternative second floor. These two perspectives yield different narratives: the flatten cost emphasizes concentrated device usage, while the hierarchical cost emphasizes spatial dispersion. This example demonstrates that the choice of cost function shapes the inferred operational story, influencing subsequent artificial event insertion and performance analysis.

Root cause diagnosis and bottleneck criteria

Our approach to root cause diagnosis empowers users to define bottlenecks based on domain-specific criteria. This design choice provides essential flexibility, enabling the method to be adapted to different systems and business objectives. For instance, our case study defines bottlenecks per resource type, which was a practical decision given the sheer number of resources and the available domain knowledge. However, this flexibility introduces a significant dependence on the chosen criteria. If the defined thresholds do not capture all relevant incidents, some *unknown* root causes may appear, which are simply events that fall outside the defined bottleneck criteria. While this could be seen as a limitation, it also serves as a diagnostic tool, highlighting a mismatch between the chosen criteria and the actual process behavior. To improve the generalizability of this aspect, future work could explore more adaptive and automated methods for defining bottlenecks, or even for setting dynamic, per-resource thresholds. This would further

reduce the reliance on manual intervention and deep domain knowledge, making the methodology even more robust.

Sequential vs. Concurrent processes

Another assumption of our root cause diagnosis is that the physical movement of a cargo item is a sequential process. This is a valid assumption within our case study, as a physical item cannot occupy two locations simultaneously. It allows us to trace a trajectory of events for identifying the precise source of delays. However, we acknowledge that in broader operational environments, concurrent activities—such as manual inspections, asynchronous administrative tasks, or parallel sub-processes—could occur. While these activities might influence the process, our current methodology focuses on the core, sequential movement of the item to ensure a targeted and effective diagnosis of physical bottlenecks. Future work could extend our framework to model and analyze these concurrent activities by developing more sophisticated alignment algorithms capable of handling multiple, parallel event streams.

Predictive analysis

A threat to the internal validity of the predictive analysis is that the better performance of the Markov chain could be a result of insufficient hyperparameter tuning or poor feature engineering for the more complex LSTM model. Moreover, the comparison might be misleading if the dataset was not large enough for the LSTM to learn complex patterns effectively, as a larger model requires more data to avoid overfitting. However, these findings present an opportunity to develop a hybrid model that takes advantage of the strengths of both approaches: using the Markov chain for robust next-resource predictions and the LSTM for its more accurate duration estimations.

Conclusion

In this paper, we address inefficiencies in air cargo terminal operations by applying process mining techniques to analyze cargo distribution. We leverage event-driven systems and real-time activity data to develop a methodology aimed at optimizing efficiency. Our approach automates the detection and diagnosis of bottlenecks and provides actionable, data-driven insights. The effectiveness of our methodology is demonstrated through a case study, with results validated by process owners.

Our research highlights the importance of addressing data quality issues in event logs, a common challenge in logistics systems utilizing IoT technologies. We outline the stages of a process mining project and detail the activities carried out for the analysis of cargo distribution, including planning, extraction, data processing, mining and analysis, evaluation, and process improvement. Key contributions of our approach include conformance checking, a systematic method to match operational events, the identification of events indicating physical movement, and timestamp repair mechanisms, all of which enhance robustness. Additionally, integrating root cause analysis with predictive modeling, using a Markov chain, enables proactive adjustments, resource optimization, and efficient cargo flow management.

The findings of the case study demonstrate the potential of process mining to provide a comprehensive understanding of cargo distribution, enabling data-driven decision-making and process optimization. While this research establishes a strong foundation

for analyzing cargo distribution using process mining, future work could explore integrating additional data sources, such as asset readiness messages, environmental factors, or detailed cargo information, to refine the analysis further. Investigating advanced techniques, such as declarative process mining or social network analysis, could also provide deeper insights into the complex interactions within air cargo terminal operations.

Author contributions
All authors contributed to the work's conception. C.L. wrote and revised the manuscript throughout all its versions and implemented the experiments. A.A. co-implemented the experiments and wrote the first version of the related work. T. H. took charge of the predictive analysis and revised the manuscript throughout all its revisions. M.P. reviewed the draft and provided feedback. S. L. and M. H. provided feedback throughout the case study. I.L. and B.D. reviewed and approved the manuscript. W.V. reviewed the draft and provided feedback.

Funding
Open Access funding enabled and organized by Projekt DEAL. Our research is funded by AIR@InnoHK research cluster of the Innovation and Technology Commission (ITC) of the HKSAR Government. The results presented in this paper within the scope of the research project 'AI-based Monitoring of Compliance and Safety Breaches in Production Environments' has come to a successful completion because of the support from ITC. We would like to express our sincere gratitude to them.

Data availability
No datasets were generated or analysed during the current study.

Declarations

Ethical approval
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 25 April 2025 / Accepted: 19 January 2026
Published online: 09 February 2026

References

Aalst WMP, Brockhoff T, Ghahfarokhi AF, Pourbafrani M, Uysal MS, Zelst SJ (2020) Removing operational friction using process mining: challenges provided by the internet of production (iop). In: Hammoudi S, Quix C, Bernardino J (eds) Data management technologies and applications - 9th international conference, DATA 2020, virtual event, July 7-9, 2020, revised selected papers. Communications in computer and information science, vol 1446. Springer, Cham, Switzerland, pp 1–31. https://doi.org/10.1007/978-3-030-83014-4_1

Ahmed T, Pedersen TB, Calders T, Lu H (2016 June) Online risk prediction for indoor moving objects. In: Chow C, Jayaraman PP, Wu W (eds) IEEE 17th International Conference on Mobile Data Management, MDM 2016, IEEE Computer Society, ???, Porto, Portugal, 102–111). https://doi.org/10.1109/MDM.2016.27. 13–16, 2016

Alnahas J (2023) Application of process mining in logistic processes of manufacturing organizations: a systematic review. Sustainability 15(15)

Bemthuis RH, van Slooten N, Arachchige JJ, Piest JPS, Bukhsh FA (2021 July) A classification of process mining bottleneck analysis techniques for operational support. In: Wijnhoven F, van Sinderen M (eds) Proceedings of the 18th International Conference on e-Business, ICE-B 2021, Online Streaming, SCITEPRESS, Setúbal, Portugal, 127–135. https://doi.org/10.5220/0010578601270135. 7–9, 2021

Bertrand Y, Van Belle R, Weerdt JD, Serral E (2022) Defining data quality issues in process mining with IoT data. In: Montali M, Senderovich A, Weidlich M (eds) Process mining workshops - ICPM 2022 international workshops, Bozen-Bolzano, Italy, October 23-28, 2022, revised selected papers. Lecture notes in business information processing, vol 468. Springer, Cham, Switzerland, pp 422–434. https://doi.org/10.1007/978-3-031-27815-0_31

Buchmann AP, Pfohl H, Appel S, Freudenreich T, Frischbier S, Petrov I, Zuber C (2010 December) Event-driven services: integrating production, logistics and transportation. In: Maximilien, EM, Rossi G, Yuan S, Ludwig H, Fantinato M, Paasc, WESOA, See, SOC-LOG (eds) Service-oriented computing - ICSOC 2010 international workshops, vol 6568. San Francisco, CA, USA, pp 237–241. https://doi.org/10.1007/978-3-642-19394-1_26. 7–10, 2010, Revised Selected Papers. Lecture Notes in Computer Science

Conforti R, La Rosa M, Ter Hofstede A (2018) Timestamp repair for business process event logs. University of Melbourne, Melbourne, Australia

Denisov V, Belkina E, Fahland D, van der Aalst WMP (2018) The performance spectrum miner: visual analytics for fine-grained performance analysis of processes. In: van der Aalst WMP, Casati F, Conforti R, Leoni M, Dumas M, Kumar A, Mendling J, Nepal S, Pentland BT, Weber B (eds) Proceedings of the Dissertation Award, Demonstration, and Industrial Track at BPM 2018 Co-located with 16th International Conference on Business Process Management (BPM 2018), Sydney, Australia, September 9-14, 2018. CEUR Workshop Proceedings, vol. 2196. CEUR-WS.org, Aachen, Germany, 96–100. https://ceur-ws.org/Vol-2196/BPM_2018_paper_20.pdf

Denisov V, Fahland D, van der Aalst WMP (2019 June) Predictive performance monitoring of material handling systems using the performance spectrum. In International Conference on Process Mining, ICPM 2019, IEEE, Aachen, Germany,, New York, USA, 137–144. https://doi.org/10.1109/ICPM.2019.00029. 24–26, 2019

Denisov V, Fahland D, van der Aalst WMP (2020) Repairing event logs with missing events to support performance analysis of systems with shared resources. In: Janicki R, Sidorova N, Chatain T (eds) Application and Theory of Petri Nets and Concurrency - 41st International Conference, PETRI NETS 2020, Paris, France, June 24-25, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12152. Springer, Cham, Switzerland, 239–259. https://doi.org/10.1007/978-3-030-51831-8_12

Dixit PM, Suriadi S, Andrews R, Wynn MT, Hofstede AHM, Buijs JCAM, van der Aalst WMP (2018) Detection and interactive repair of event ordering imperfection in process logs. In: Krogstie J, Reijers HA (eds) Advanced Information Systems Engineering - 30th International Conference, CAiSE 2018, Tallinn, Estonia, June 11-15, 2018, Proceedings. Lecture Notes in Computer Science, vol 10816. Springer, Cham, Switzerland, 274–290. https://doi.org/10.1007/978-3-319-91563-0_17

Fahland D (2019) Describing behavior of processes with many-to-many interactions. In: Donatelli S, Haar S (eds) Application and Theory of Petri Nets and Concurrency - 40th International Conference, PETRI NETS 2019, Aachen, Germany, June 23-28, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11522. Springer, Cham, Switzerland, 3–24. https://doi.org/10.1007/978-3-030-21571-2_1

Fischer DA, Goel K, Andrews R, van Dun CGJ, Wynn MT, Röglinger M (2020) Enhancing event log quality: detecting and quantifying timestamp imperfections. In: Fahland D, Ghidini C, Becker J, Dumas M (eds) Business Process Management - 18th International Conference, BPM 2020, Seville, Spain, September 13-18, 2020, Proceedings. Lecture Notes in Computer Science, vol 12168. Springer, Cham, Switzerland, 309–326. https://doi.org/10.1007/978-3-030-58666-9_18

Gunnarsson BR, Broucke S, Verhoeven T, De Weerdt J (2025) Predictive process monitoring for airport operational support. KI-Künstliche Intelligenz 1–20

Hammervoll T (2009) Value-creation logic in supply chain relationships. J Business-To-Business Marketing 16(3):220–241

Hoffman AJ (2019) Quantifying the impact of freight transport performance on the total economic cost of cargo importers. In 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp 649–654. IEEE

Intayoad W, Becker T (2018) Applying process mining in manufacturing and logistic for large transaction data. In: Freitag M, Kotzab H, Pannek J (eds) Dynamics in Logistics, Proceedings of the 6th International Conference LDIC 2018, Bremen, Germany, February 20-22, 2018. Lecture Notes in Logistics. Springer, Cham, Switzerland, 378–388. https://doi.org/10.1007/978-3-319-74225-0_51

Khosravi A, Nahavandi S, Creighton D (2009) Estimating performance indexes of a baggage handling system using metamodels. In 2009 IEEE International Conference on Industrial Technology, pp 1–6. https://doi.org/10.1109/ICIT.2009.4939626

Knoll D, Reinhart G, Prueglmeier M (2019) Enabling value stream mapping for internal logistics using multidimensional process mining. Expert Syst Appl 124:130–142

Kropp T, Faeghi S, Lennerts K (2024) Process mining for capacity planning and reconfiguration of a logistics system to enhance the intra-hospital patient transport. case study. In: Finkelstein J, Moskovitch R, Parimbelli E (eds) Artificial Intelligence in Medicine - 22nd International Conference, AIME 2024, Salt Lake City, UT, USA, July 9-12, 2024, Proceedings, Part I. Lecture Notes in Computer Science, vol. 14844. Springer, Cham, Switzerland, 138–150. https://doi.org/10.1007/978-3-031-66538-7_15

Lai X, Qiu T, Shui H, Ding D, Ni J (2023) Predicting future production system bottlenecks with a graph neural network approach. J Manuf Sys 67:201–212

Leoni M, Munoz-Gama J, Carmona J, van der Aalst WMP (2014) Decomposing alignment-based conformance checking of data-aware process models. In: Meersman R, Panetto H, Dillon TS, Missikoff M, Liu L, Pastor O, Cuzzocrea A, Sellis TK (eds) On the Move to Meaningful Internet Systems: OTM 2014 Conferences - Confederated International Conferences: CoopIS, and ODBASE 2014, Amantea, Italy, October 27-31, 2014, Proceedings. Lecture Notes in Computer Science, vol. 8841. Springer, Cham, Switzerland, 3–20. https://doi.org/10.1007/978-3-662-45563-0_1

Li C-Y, van Zelst SJ, Aalst (2019) W.M.P.: a generic approach for process performance analysis using bipartite graph matching. In: Business process management workshops: bPM 2019 international workshops, Vienna, Austria, September 1–6, 2019, revised selected papers, vol 17. Springer, Cham, Switzerland, pp 199–211

Li C, Antonov A, Aalst WM (2024) Activity instance identification using bipartite graph matching. In International Conference on Service-Oriented Computing, Springer, pp. 82–95

Li C, Joshi A, Tam NTL, Lau SSF, Huang J, Shinde T, van der Aalst WMP (2023) Rectify sensor data in IoT: a case study on enabling process mining for logistic process in an air cargo terminal. In: Sellami, M, Vidal, M, van Dongen, BF, Gaaloul, W, Panetto, H (eds) Cooperative information Systems - 29th international conference, CoopIS 2023, Groningen, the Netherlands, October 30 - November 3, 2023, proceedings. Lecture notes in computer science, vol 14353. Springer, Cham, Switzerland, pp 293–310. https://doi.org/10.1007/978-3-031-46846-9_16

Li C, Shinde T, He W, Lau SSF, Hiew MXB, Tam NTL, Joshi A, van der Aalst WMP (2023) Unveiling bottlenecks in logistics: a case study on process mining for root cause identification and diagnostics in an air cargo terminal. In: Monti, F, Rinderle-Ma, S, Cortés, AR, Zheng, Z, Mecella, M (eds) Service-Oriented Computing - 21st International Conference, ICSOC 2023, Rome, Italy, November 28 - December 1, 2023, Proceedings, Part II. Lecture Notes in Computer Science, vol. 14420. Springer, Cham, Switzerland, 291–307. https://doi.org/10.1007/978-3-031-48424-7_21

Lin W, Bakir F, Krintz C, Wolski R, Mock M (2019) Data repair for distributed, event-based iot applications. In Proceedings of the 13th ACM International Conference on Distributed and Event-based Systems, DEBS 2019, ACM, Darmstadt, Germany,, New York, USA, 139–150. https://doi.org/10.1145/3328905.3329511. June 24–28, 2019

Liu D, Guo Y, Huang S, Wang S, Wu T (2023) Dynamic production bottleneck prediction using a data-driven method in discrete manufacturing system. Adv Eng Informatics 58:102162

Liu J, Xu J, Zhang R, Reiff-Marganiec S (2021) A repairing missing activities approach with succession relation for event logs. Knowl Inf Syst 63(2):477–495

Liu Y, Yin M, Hansen M (2019) Economic costs of air cargo flight delays related to late package deliveries. Transp Res Part E: Log Transp Rev 125:388–401

Lu Y, Chen Q, Poon (2022) S.K.: a deep learning approach for repairing missing activity labels in event logs for process mining. Inf. 13(5):234

Mansouri T, Moghadam MRS, Monshizadeh F, Zareravasan A (2023) Iot data quality issues and potential solutions: A literature review. Comput J 66(3):615–625

Melo M, Aquino G (2021) The pathology of failures in IoT systems. In: Gervasi O, Murgante B, Misra S, Garau C, Blecic I, Taniar D, Apduhan, BO, Rocha, AMAC, Tarantino, E, Torre, CM (eds) Computational science and its applications - ICCSA 2021 - 21st international conference, Cagliari, Italy, September 13-16, 2021, proceedings, part IX. Lecture notes in computer science, vol 12957. Springer, Cham, Switzerland pp 437–452. https://doi.org/10.1007/978-3-030-87013-3_33

Michelson BM (2006) Event-driven architecture overview. Patricia Seybold Group 2(12):10–1571

Murphy P, Dalenberg D, Daley J (1989) Improving international trade efficiency: airport and air cargo concerns. Transp J 27–35

Nahavandi S, Gunn B, Johnstone M, Creighton D (2019) Modelling and simulation of large and complex systems for airport baggage handling security. In: Arai, K., Kapoor, S., Bhatia, R. (eds) Intelligent computing. Springer, Cham, pp 1055–1067

Piest JPS, Cutinha JA, Bemthuis RH, Bukhsh FA (2021) Evaluating the use of the open trip model for process mining: an informal conceptual mapping study in logistics. In: Filipe J, Smialek M, Brodsky A, Hammoudi S (eds) Proceedings of the 23rd International Conference on Enterprise Information Systems, ICEIS 2021, Online Streaming, vol 1. SCITEPRESS, Cham, Switzerland, 290–296. April 26–28, 2021

Prathama F, Yahya BN, Harjono DD, Er M (2019 July). The Fifth Information Systems International Conference, Trace clustering exploration for detecting sudden drift: a case study in logistic process. Proc Comp Sci 161:1122–1130. 23–24 2019, Surabaya, Indonesia

Oliveira Rocha HF, Oliveira Rocha HF (2022) Achieving resilience and event processing reliability in event-driven microservices. Pract Event-Driven Microservices Archit: Build Sustain And Highly Scalable Event-Driven Microservices 275–321

Rogge-Solti A, Mans RS, van der Aalst WMP, Weske M (2013) Repairing event logs using stochastic process models, vol 78. Universitätsverlag Potsdam, Postdam, Germany

Sani MF, Gonzalez JJG, van Zelst SJ, van der Aalst WMP (2020 October) Conformance checking approximation using simulation. In: van Dongen BF, Montali M, Wynn MT (eds) 2nd International Conference on Process Mining, ICPM 2020, IEEE, Padua, Italy,, New York, USA, 105–112. https://doi.org/10.1109/ICPM49681.2020.00025. 4–9, 2020

Schrijver A (1999) Theory of linear and integer programming. Wiley-interscience series in discrete mathematics and optimization. Wiley, Hoboken, NJ, USA

Skorupski J, Uchroński P, Łach A (2018) A method of hold baggage security screening system throughput analysis with an application for a medium-sized airport. Transp Res Part C: Emerging Tech 88:52–73

Song W, Xia X, Jacobsen H, Zhang P, Hu H (2015) Heuristic recovery of missing events in process logs. In: Miller JA, Zhu H (eds) 2015 IEEE International Conference on Web Services, ICWS 2015, New York, NY, USA, June 27 - July 2, 2015. IEEE Computer Society, Los Alamitos, CA, USA, 105–112. https://doi.org/10.1109/ICWS.2015.24

Uckelmann D (2008) A definition approach to smart logistics. In: Balandin, S., Moltchanov, D., Koucheryavy, Y. (eds) Next generation teletraffic and Wired/Wireless advanced networking. Springer, Berlin, Heidelberg, pp 273–284

van der Aalst WMP (2013) Decomposing petri nets for process mining: a generic approach. Distrib Parallel Databases 31(4):471–507

van der Aalst WMP (2016) Process mining - data Science in action, Second edn. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-662-49851-4

van Eck ML, Lu X, Leemans SJJ, van der Aalst WMP (2015) PM2: a process mining project methodology. In: Zdravkovic J, Kirikova M, Johannesson P (eds) Advanced information Systems engineering - 27th international conference, CAiSE 2015, Stockholm, Sweden, June 8-12, 2015, proceedings. Lecture notes in computer science, vol 9097. Springer, Cham, Switzerland pp 297–313. https://doi.org/10.1007/978-3-319-19069-3_19

van Zelst SJ, Buijs JCAM, Vázquez-Barreiros B, Lama M, Mucientes M (2020) Repairing alignments of process models. Bus Inf Syst Eng 62(4):289–304

Wang Y, Caron F, Vanthienen J, Huang L, Guo Y (2014) Acquiring logistics process intelligence: methodology and an application for a Chinese bulk port. Expert Syst Appl 41(1):195–209

Yu C, Zou L (2022) Air trade, air cargo demand, and network analysis: case of the United States. In: The International Air cargo industry: a modal analysis. Emerald Publishing Limited, Leeds, England, pp 207–239

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.