

Article

Reliable Process Tracking Under Incomplete Event Logs Using Timed Genetic-Inductive Process Mining

Yutika Amelia Effendi ¹ and Minsoo Kim ^{2,*}

¹ Robotics and Artificial Intelligence Engineering, Faculty of Advanced Technology and Multidiscipline, Airlangga University, Surabaya 60115, Indonesia; yutika.effendi@ftmm.unair.ac.id

² Department of Industrial and Data Engineering, College of Engineering, Pukyong National University, Busan 48513, Republic of Korea

* Correspondence: minsky@pknu.ac.kr; Tel.: +82-51-629-6487

Abstract: Process mining facilitates the discovery, conformance, and enhancement of business processes using event logs. However, incomplete event logs and the complexities of concurrent activities present significant challenges in achieving accurate process models that fulfill the completeness condition required in process mining. This paper introduces a Timed Genetic-Inductive Process Mining (TGIPM) algorithm, a novel approach that integrates the strengths of Timed Genetic Process Mining (TGPM) and Inductive Mining (IM). TGPM extends traditional Genetic Process Mining (GPM) by incorporating time-based analysis, while the IM is widely recognized for producing sound and precise process models. For the first time, these two algorithms are combined into a unified framework to address both missing activity recovery and structural correctness in process discovery. This study evaluates two scenarios: a sequential approach, in which TGPM and IM are executed independently and sequentially, and the TGIPM approach, where both algorithms are integrated into a unified framework. Experimental results using real-world event logs from a health service in Indonesia demonstrate that TGIPM achieves higher fitness, precision, and generalization compared to the sequential approach, while slightly compromising simplicity. Moreover, the TGIPM algorithm exhibits lower computational cost and more effectively captures parallelism, making it particularly suitable for large and incomplete datasets. This research underscores the potential of TGIPM to enhance process mining outcomes, offering a robust framework for accurate and efficient process discovery while driving process innovation across industries.

Keywords: incomplete event log; genetic process mining; inductive mining; process model; parallel processes; process mining; discovery; enhancement; process innovation; health service



Academic Editors: Moutaz Khouja, Tai-Woo Chang and Gyusun Hwang

Received: 26 January 2025

Revised: 27 February 2025

Accepted: 24 March 2025

Published: 27 March 2025

Citation: Effendi, Y.A.; Kim, M.

Reliable Process Tracking Under Incomplete Event Logs Using Timed Genetic-Inductive Process Mining. *Systems* **2025**, *13*, 229.

<https://doi.org/10.3390/systems13040229>

Copyright: © 2025 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license

(<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In the modern industrial landscape, where artificial intelligence (AI) plays an increasingly central role, process mining has emerged as a crucial bridge between data-driven methodologies and the operational analysis of complex systems [1]. Process mining leverages event logs—structured records of activities within processes—to uncover hidden patterns, diagnose inefficiencies, and predict system behaviors [2]. By enabling organizations to derive actionable insights, process mining fosters transparency and continuous improvement, ensuring that operations align with data-driven strategies [3]. This capability is particularly essential in industrial systems, where processes are intricate and data sources

are fragmented. The integration of AI and process mining allows organizations to move beyond static analyses, adopting dynamic process tracking and workflow optimization [4].

At the core of process mining lies the event log, a structured collection of events that captures the sequential execution of processes within a system [5]. Each event represents a specific action or activity, making event logs indispensable for accurate and reliable analysis [6]. However, event log completeness is crucial for ensuring the quality of process mining outcomes [7]. Incomplete event logs can lead to flawed process models, inaccurate predictions, and suboptimal decision-making. The primary causes of incompleteness include limited data extraction methods, which retrieve only events within specific timeframes, unfinished ongoing cases, and cases that never reach completion [8]. Addressing these issues is essential to improving the reliability and applicability of process mining techniques.

Beyond technical limitations, organizational and human factors further contribute to the exacerbation of event log incompleteness. Many businesses still rely on periodic and asynchronous systems that lack full integration, leading to unrecorded activities and missing timestamps [9]. Human errors, such as manual data entry mistakes or delayed inputs due to workload pressures, also contribute to missing data, reducing the overall accuracy of process models [10]. These gaps undermine process mining's effectiveness, making it necessary to develop advanced approaches that enhance event log completeness and mitigate temporary absences of activity data.

Despite significant advancements in process mining, existing discovery algorithms struggle with incomplete event logs and complex process dependencies. Inductive Mining (IM), while effective in producing sound and precise models, heavily relies on event log completeness and struggles with missing data, often leading to overly generalized or incorrect process structures. On the other hand, Genetic Process Mining (GPM) is more adaptable in handling missing activities but lacks a built-in structural correctness mechanism, making it prone to generating inaccurate dependencies and suboptimal process models.

Previous studies on GPM have explored various strategies for optimizing process discovery using evolutionary principles. Ref. [11] introduced one of the earliest frameworks for genetic-based process discovery, focusing on optimizing fitness scores relative to observed event traces. However, this approach lacked strong structural correctness guarantees. A significant advancement in this area is [12]'s work on genetic process discovery using process trees, which ensures soundness by construction. This approach generates process models that inherently adhere to structural correctness by encoding process trees into genetic representations. This contrasts with the traditional genetic algorithms, which operate on control-flow graphs, where soundness violations (e.g., deadlocks, livelocks) are harder to detect and correct.

To address these challenges, this research introduces the Timed Genetic-Inductive Process Mining (TGIPM) algorithm, an innovative approach that integrates Timed Genetic Process Mining (TGPM) and IM while leveraging dual timestamp information from event logs. TGPM is utilized for its ability to recover missing activities, reconstruct incomplete event logs, and handle noise and uncertainty through iterative optimization [13]. TGPM extends GPM [14] by incorporating time-based analysis into the process discovery phase, enhancing the accuracy of process model generation. Meanwhile, IM is selected for its ability to generate sound and precise process models, ensuring that the discovered workflows are free from structural anomalies such as deadlocks and deviations [15].

Although GPM and IM can be executed sequentially, we propose TGIPM, which integrates both approaches into a unified framework. This integration enables the simultaneous recovery of missing activities and the generation of structurally robust process models, resulting in higher accuracy, improved conformance, and better preservation of

process dependencies. TGIPM ensures that temporal gaps in event logs are correctly reconstructed, making it particularly effective for handling incomplete datasets in real-world industrial applications. Moreover, TGIPM optimizes genetic operations with heuristics and leverages time-aware fitness measures, thereby reducing computational complexity while maintaining high model accuracy and robustness.

To validate the effectiveness of TGIPM, its performance is evaluated in two scenarios: (1) sequential execution: TGPM is used for recovering missing activities, followed by IM for process model discovery; and (2) the integrated TGIPM approach: both algorithms are combined into a single framework, where missing activity recovery and process model generation occur simultaneously.

The performance of these scenarios is assessed based on their ability to recover missing activities, achieve temporal fitness, and generate structurally sound and accurate process models. Additionally, conformance metrics and time complexity are analyzed to evaluate the practical applicability of TGIPM in real-world industrial settings.

This research not only advances the theoretical foundations of process mining but also provides practical solutions for industries seeking to optimize operations despite incomplete and fragmented event data. By addressing the persistent issue of event log incompleteness, this study establishes a robust framework for developing accurate and reliable process models, ultimately facilitating data-driven decision-making and continuous operational improvement.

The structure of this document is as follows: Section 2 explores the foundational research background, Section 3 focuses on the proposed algorithm, and Sections 4 and 5 provide the experimental evaluation and conclusion, respectively.

2. Related Research

2.1. Process Mining

Process mining acts as a critical bridge between data-driven methodologies and the operational analysis of industrial systems. In an era where industries increasingly leverage AI to optimize workflows, extracting actionable insights from event logs has become fundamental to informed decision-making [16]. By uncovering process models, diagnosing inefficiencies, and predicting system behaviors, process mining enables organizations to align their operations with robust, data-driven strategies [17]. This capability is especially valuable in industrial systems, where processes are inherently complex, and data sources are often fragmented. Process mining enhances transparency and establishes a solid foundation for AI-driven optimization, allowing industries to move beyond static analyses toward dynamic process tracking and continuous operational improvement [18].

In the context of Industry 4.0, nearly all process activities can now be recorded, enabling detailed real-time or offline analysis. Accurate event logs, which serve as the primary data source for process mining, are essential for extracting meaningful insights [19]. These logs document the sequence of activities within organizational workflows, facilitating the discovery, monitoring, and improvement of real-world processes [20]. Through the analysis of event logs, organizations can gain a deeper understanding of their workflows, identify inefficiencies, ensure procedural compliance, and enhance performance based on empirical data.

To process event logs effectively, process mining encompasses three main objectives: discovery, conformance, and enhancement [2]. Discovery involves identifying and visualizing actual processes from event data, often resulting in the creation of process models [21]. Conformance compares the discovered process model with a predefined model to detect deviations and ensure compliance with established procedures [22]. Enhancement

refines process models by incorporating additional event log data to optimize performance, efficiency, and overall understanding [23].

2.2. Completeness of Event Log

Event log completeness is a fundamental concept in process mining, referring to how accurately and comprehensively an event log represents the underlying business processes [24]. It encompasses several dimensions, including trace completeness, event completeness, attribute completeness, frequency completeness, and temporal completeness. Ensuring completeness is crucial for the reliability and validity of process mining analyses, as it directly impacts process discovery, conformance checking, and enhancement, all of which contribute to effective business process management and decision-making [25].

In real-world applications, event logs often suffer from imperfections and data quality issues, leading to incomplete or inaccurate process models [26]. These imperfections manifest in various forms, making process discovery more challenging. Ref. [27] categorized four primary data quality issues affecting process mining: missing data, where events or attributes are not recorded; incorrect data, involving errors or inconsistencies in recorded events; imprecise data, which contains ambiguous or vague event information; and irrelevant data, referring to redundant or noisy event logs that do not contribute to meaningful process discovery. These data quality issues can significantly distort process mining results, leading to unreliable and misleading process models.

Building on these classifications, refs. [28,29] worked on how these quality issues manifest across different entities within an event log, identifying structural and semantic imperfection patterns. At the case level, incompleteness may arise from missing case identifiers, truncated traces, or cases that never reach completion. At the event level, issues include missing, duplicated, or misordered events. At the attribute level, incorrect timestamps, missing resource allocations, or ambiguous event labels can affect the quality of the log. Additionally, temporal inconsistencies may cause activities to appear out of sequence due to incorrect or delayed timestamp recording, further complicating process discovery. Understanding these imperfections is essential, as they introduce challenges in control-flow relationships, conformance checking, and event correlation.

Before conducting a process mining analysis, it is critical to assess data suitability and identify incomplete cases within the dataset. Several factors contribute to event log incompleteness.

1. Data extraction limitations. Logs often capture events within specific timeframes, which may lead to truncated cases if a process started before or extended beyond the extraction period [8].
2. Ongoing cases. Some processes remain active at the time of extraction, making them appear incomplete [8].
3. Abandoned cases. Certain processes may never reach completion due to customer inaction, system failures, or external dependencies [8].
4. Lack of system integration. Many organizations rely on periodic or asynchronous data entry systems, where activities may be logged with delays or not recorded at all due to system inefficiencies [13].
5. Human factors. Manual data entry errors, workload pressures, and inconsistent recording practices often result in missing or incorrect timestamps, further exacerbating incompleteness [13].

Addressing these data quality challenges is essential, as missing or incorrect data can distort process models and lead to inaccurate decision-making. Researchers have proposed various techniques to mitigate the impact of incomplete event logs. Trace clustering and log preprocessing methods help identify and filter unreliable traces before process

discovery [30]. Data imputation techniques, using statistical and machine learning models, estimate missing event timestamps or attributes [31]. Stochastic process discovery approaches leverage probabilistic modeling to reconstruct missing sequences [32]. Additionally, hybrid techniques combining genetic algorithms and heuristic search improve event log recovery by integrating structural corrections and missing activity inference [13]. By incorporating imperfection pattern analysis into event log preprocessing, process mining techniques can better adapt to real-world datasets, ensuring greater accuracy, reliability, and completeness.

2.3. Genetic Process Mining

GPM has emerged as a powerful approach for handling incomplete event logs by leveraging evolutionary search techniques to optimize process model discovery. Unlike deterministic algorithms, which strictly follow predefined rules to construct process models, GPM iteratively explores candidate process models, refining them based on fitness evaluations derived from event logs [8]. This approach enables GPM to infer missing activities and correct structural inaccuracies caused by incomplete data, making it particularly useful for real-world process discovery.

Inspired by evolutionary principles, GPM refines a population of candidate process models over successive iterations to identify the one that best aligns with the observed event data [2]. As presented in Figure 1, the algorithm begins by initializing a diverse set of process models, which are then evaluated using a fitness function that measures their correspondence with the event log. Through genetic operations such as selection, crossover, and mutation, new models are generated, inheriting characteristics from high-performing models while exploring alternative structures. This iterative process continues until a stopping criterion is met, such as reaching a predefined number of generations or achieving a satisfactory fitness score.

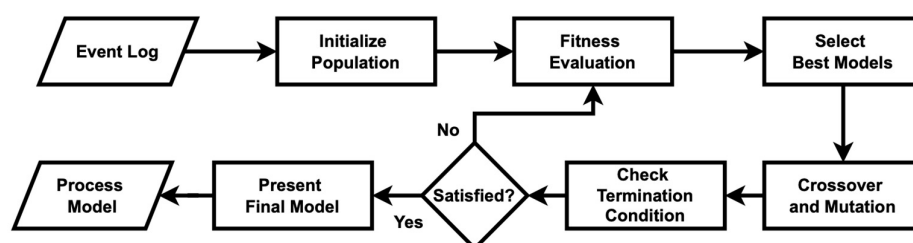


Figure 1. A flowchart of the GPM algorithm to discover a process model.

One of GPM's key strengths is its ability to handle complex and noisy event logs, where traditional process discovery algorithms often struggle [11]. It excels at discovering process models with concurrency, loops, and non-linear structures, making it well-suited for processes with high variability. Unlike deterministic algorithms that operate within a fixed rule set, GPM explores a broader search space, increasing the likelihood of identifying optimal or near-optimal process models. However, this flexibility comes at a cost: GPM is computationally intensive, as it requires evaluating numerous candidate models across multiple generations [14]. Despite this challenge, GPM remains a robust and adaptable tool for process discovery in complex datasets.

Beyond process discovery, GPM also demonstrates an inherent capability to recover missing activities in event logs, addressing a critical challenge for traditional process mining methods [13]. By leveraging its iterative search mechanism, GPM explores various process model configurations, allowing it to infer and reconstruct missing portions of processes that align with the available data, as shown in Figure 2. This feature is particularly advantageous in scenarios where event logs are incomplete or contain gaps. However,

while GPM's flexibility allows it to handle missing data, its recovery capabilities are not always optimal [13]. The reconstructed models may sometimes lack precision or fail to fully capture the true sequence of activities, especially in highly complex or noisy logs.

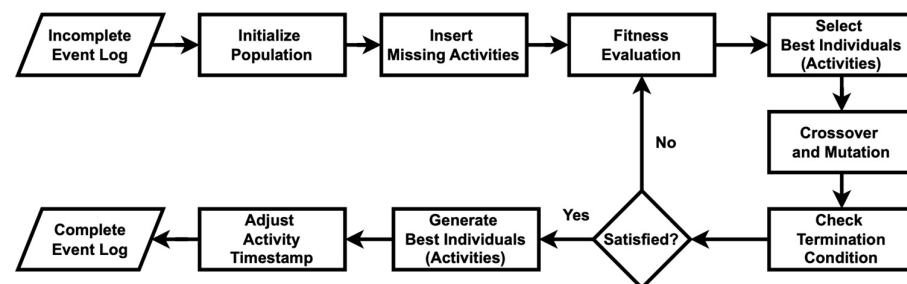


Figure 2. A flowchart of the GPM algorithm used to recover missing activities.

Several studies have investigated the application of GPM for missing data recovery. Ref. [11] introduced a genetic algorithm-based process discovery framework that reconstructs process models by optimizing fitness scores relative to observed event traces. Similarly, ref. [13] extended GPM with time-based analysis, enabling the recovery of missing activities by considering dual timestamps (start and end times). However, a common limitation of standalone GPM is its computational overhead, as evaluating multiple generations of process models requires significant processing time. Nonetheless, such approaches still face challenges in ensuring the soundness and precision of the discovered process models, as GPM alone does not explicitly enforce structural correctness. Therefore, integrating GPM with other process discovery techniques has been a focus of recent research to enhance its efficiency and accuracy while maintaining computational feasibility.

2.4. Inductive Mining

IM is one of the most widely used and reliable process discovery algorithms, known for its ability to generate sound and precise process models [2]. Unlike traditional process discovery techniques, IM employs a divide-and-conquer strategy, recursively partitioning event logs into smaller components before reconstructing a hierarchical process model [15]. This structured approach ensures that the resulting models are free of deadlocks, behavioral anomalies, and structural inconsistencies, making IM particularly useful for complex real-world processes. The flowchart of the IM algorithm to discover a process model is explained in Figure 3.

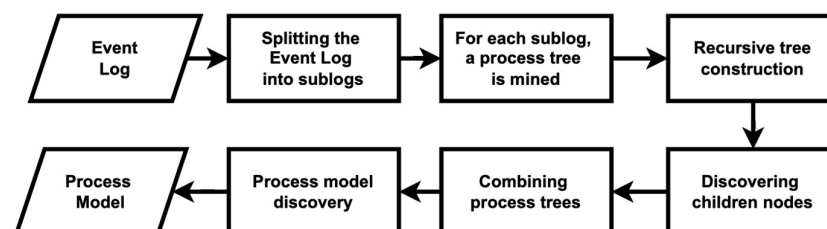


Figure 3. A flowchart of the IM algorithm to discover a process model.

A key strength of IM is its guarantee of model soundness, ensuring that the discovered models are executable and free from logical errors. This eliminates common issues such as deadlocks or unreachable states, which are often found in models generated by other algorithms. Additionally, IM excels at representing concurrency and parallel processes, making it particularly suitable for event logs that feature complex interdependencies [2]. Its computational efficiency and robustness against noise further solidify its reputation as a

leading algorithm in process mining, making it a preferred choice for applications such as conformance checking and process optimization across various industries.

Despite its advantages, IM has two fundamental limitations. First, it heavily relies on event log completeness. When applied to incomplete event logs, IM often produces inaccurate or overly generalized models due to its deterministic nature [16,33]. This issue arises because IM assumes that the event log fully represents the underlying process, making it difficult to infer missing activities without additional guidance.

Second, IM prioritizes perfect fitness without accounting for the frequency of activities, which can impact model quality [15]. Infrequent behaviors refer to valid but rare actions within the system, while deviating behaviors are actions recorded in the event log but not inherent to the process. To address this issue, IM introduces dummy activities, often represented as invisible transitions or black boxes, to capture infrequent activities that occur below a predefined threshold. While this approach helps improve process model completeness, it also highlights the IM's difficulty in balancing fitness and model quality, especially when dealing with rare or anomalous activities.

To overcome these limitations, several enhancements have been proposed to improve the IM's handling of missing data. Ref. [34] introduced an event log repair mechanism that augments IM with preprocessing techniques to infer missing cases before process discovery. While these approaches enhance the IM's performance, they do not fully address missing activity recovery, as they focus primarily on structural corrections rather than data reconstruction.

Beyond these improvements, several other research efforts have aimed to enhance the IM algorithm by addressing accuracy, model quality, privacy, and interpretability. One notable advancement is the Probabilistic Inductive Miner (PIM), which integrates frequency information from event logs into the IM framework [35]. By selecting the most probable operators and structures, PIM generates block-structured models that balance accuracy and simplicity more effectively than traditional IM. Evaluations show that PIM produces more accurate models while maintaining an acceptable level of complexity, making it more practical for real-world applications.

Another significant development is Differentially Private Inductive Miner (DPIM), designed to protect sensitive information in event logs [36]. DPIM approximates the IM algorithm while ensuring differential privacy, preventing process models from exposing personal or confidential data. Experimental results indicate that DPIM maintains a balance between privacy and model utility, producing faithful process trees with minimal loss in accuracy compared to the standard IM.

Additionally, Inductive Visual Miner (IvM) was introduced to improve the interpretability and educational value of IM [37]. IvM provides a step-by-step visualization of the IM process, allowing users to interactively explore how the algorithm processes event data. This tool is particularly beneficial for students and practitioners in process mining, enhancing their understanding of the IM's divide-and-conquer approach.

These advancements collectively contribute to refining the IM algorithm, making it more robust, user-friendly, privacy-preserving, and effective in handling incomplete event logs. Future research should continue to explore hybrid approaches that combine event log recovery, probabilistic modeling, and structural optimization, ensuring that IM remains adaptable to real-world process discovery challenges.

2.5. Timed-Based Process Discovery

Among the various techniques in process mining, process discovery is widely regarded as the most challenging [2,21]. It involves extracting a process model from event logs that record the sequence of activities in a business process. The goal is to generate a model

that accurately represents real-world behavior captured in the log. However, this task is inherently complex due to factors such as variability in process executions, data noise, and the presence of concurrent activities. To address these challenges, a range of algorithms has been developed, evolving significantly over the years.

Traditional process discovery algorithms, such as Alpha Mining [38,39], Heuristic Mining [40], Fuzzy Mining [33], GPM [11], and IM [15], typically rely on sequential approaches using single time event logs [5]. While these methods have significantly advanced the field, they primarily focus on control-flow perspectives and are less equipped to handle intricate temporal dependencies or concurrency in processes.

The recent introduction of dual time information has revolutionized process discovery, enabling time-based methodologies that integrate temporal aspects into model generation. These approaches enhance the accuracy and expressiveness of discovered models by leveraging both the start and end times of activities. Several algorithms exemplify this paradigm, including Stochastic Process Discovery by Weight Estimation [41], Timeline-Based Process Discovery [42], Determining Process Models Using Time-Based Process Mining and Control-Flow Patterns [43], and Time-Based Discovery of Parallel Business Processes [44]. These advancements address key limitations of traditional methods, offering improved insights and expanding the scope of process mining applications.

Time-based process discovery has gained significant attention for its ability to enhance model accuracy and utility by incorporating temporal data. Traditional techniques often overlook temporal patterns, leading to models that fail to capture dynamic process behaviors. To address this, time-based approaches integrate temporal information into control-flow patterns, enabling the generation of more comprehensive and realistic models [5]. For instance, ref. [41] proposed a stochastic process discovery method that uses weight estimation techniques to decouple temporal aspects from control-flow discovery, generating Generalized Stochastic Petri Nets (GSPNs) that incorporate time and probability components. Similarly, timeline-based process discovery [42] methods align process models with a time axis, emphasizing waiting times as critical elements of business processes, thereby improving model interpretability and real-world applicability.

Specific algorithms were also developed to tackle concurrency in business processes. For example, a novel time-based discovery algorithm utilizes non-linear dependence principles [43,44] to distinguish between AND-parallel and OR-conditional relations, resulting in improved accuracy in detecting concurrent processes. Moreover, ref. [13] introduces a novel TGPM algorithm to address challenges posed by incomplete event logs, as presented in Figure 4. Leveraging dual time information, TGPM enhances traditional GPM to recover missing activities and reconstruct processes in noisy or incomplete data. The algorithm uses start and end times, time-aware fitness measures (sequence, temporal, and concurrency fitness), and genetic operations (selection, crossover, and mutation) while emphasizing temporal consistency. Through iterative evolution, the algorithm selects the best model based on fitness scores. Experimental validation demonstrates its superior performance, enabling more accurate process models and reliable conformance checking.

These studies collectively underscore the importance of incorporating temporal information into process discovery. By addressing challenges such as concurrency, noise, and incomplete data, time-based methodologies significantly enhance the interpretability, reliability, and quality of process models. This foundation is instrumental in advancing the state-of-the-art in process discovery and its practical applications.

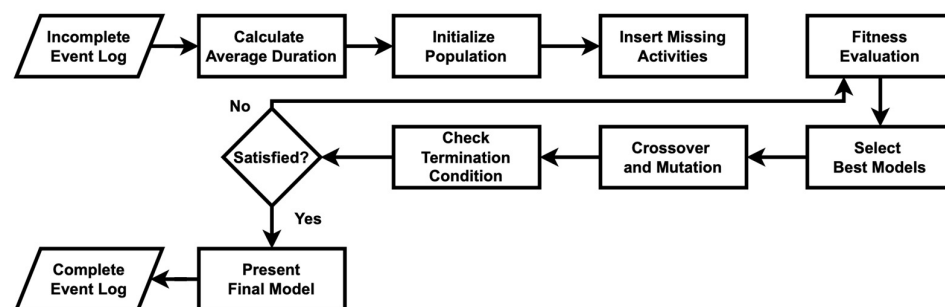


Figure 4. A flowchart of TGPM algorithm to recover and discover a process model.

3. Timed Genetic-Inductive Process Mining

The TGIPM algorithm is a novel approach to process discovery and represents the first integration of two widely recognized process mining algorithms, TGPM and IM, into a unified framework. By combining the strengths of these algorithms, TGIPM effectively recovers missing activities from incomplete event logs. This integrated approach produces a complete event log with successfully recovered activities, a refined process model with minimized dummy activities, and the ability to accurately identify and distinguish parallel gateways in a process.

The combination of the modified GPM, which utilizes dual time information (referred to as TGPM [13]), and IM is essential because TGPM alone cannot fully overcome the challenges of process discovery from incomplete event logs. While TGPM excels at recovering missing activities and reconstructing event logs, it does not guarantee that the resulting process model will be sound or precise. Its primary focus on filling data gaps often results in models with behavioral anomalies or structural inaccuracies, limiting their reliability and usefulness. Additionally, TGPM lacks the ability to optimize structural elements such as parallelism, gateways, or conditional behaviors, which are crucial for capturing the complexity of real-world processes.

TGPM's reliance on genetic algorithms also makes it computationally intensive, especially for large datasets with numerous missing activities. While effective at recovering activities, TGPM may retain irrelevant or infrequent behaviors due to its sensitivity to noise in the event log. Without additional processing, this can lead to overly complex or unreliable models.

The integration of TGPM with IM overcomes these limitations. IM ensures sound and precise process models by eliminating behavioral anomalies, such as deviations and infrequent behaviors, while refining structural elements. It is particularly effective at discovering complex process structures, including parallelism, sequential relations, and gateways, which TGPM alone cannot optimize. Additionally, IM inherently filters noise and focuses on frequent patterns in the event log, complementing TGPM's recovery process by ensuring that only relevant activities are included in the final model.

Unlike traditional process mining approaches that rely solely on a single timestamp, TGIPM leverages dual timestamps (start and end times), which are now commonly stored in modern enterprise databases. Timestamp information recorded in event logs plays a crucial role in uncovering meaningful insights into business process performance and behavior through process mining techniques [45,46]. Most contemporary systems, such as Enterprise Resource Planning (ERP), business process management (BPM), and Hospital Information Systems (HIS), generally track both start and end times for process monitoring, compliance, and optimization. The availability of dual timestamps enhances TGIPM's ability to reconstruct event sequences, ensuring accurate activity recovery and improving conformance checking.

TGIPM leverages the strengths of both TGPM and IM while utilizing dual time information to enhance both activity recovery and process model generation. This integrated approach ensures that missing activities are accurately recovered and that the final process model is reliable, comprehensive, and structurally sound. Furthermore, TGIPM reduces computational overhead by combining recovery and model discovery into a unified workflow, making it more scalable for large and complex datasets. By addressing TGPM's shortcomings and incorporating IM's structural refinement capabilities, TGIPM provides a robust and efficient solution for process mining from incomplete event logs.

3.1. The Algorithm

The proposed TGIPM algorithm integrates the strengths of TGPM and IM to enhance process model discovery from event logs. As illustrated in Figure 5, TGIPM combines TGPM's ability to recover missing activities with IM's capability to generate sound and precise process models. The flowchart in Figure 5 provides a step-by-step overview of the TGIPM workflow, spanning from preprocessing incomplete event logs to producing robust process models. The algorithm addresses key challenges such as recovering missing activities, minimizing the appearance of dummy activities, and accurately representing parallelism in a process, ensuring reliable outcomes even under challenging conditions.

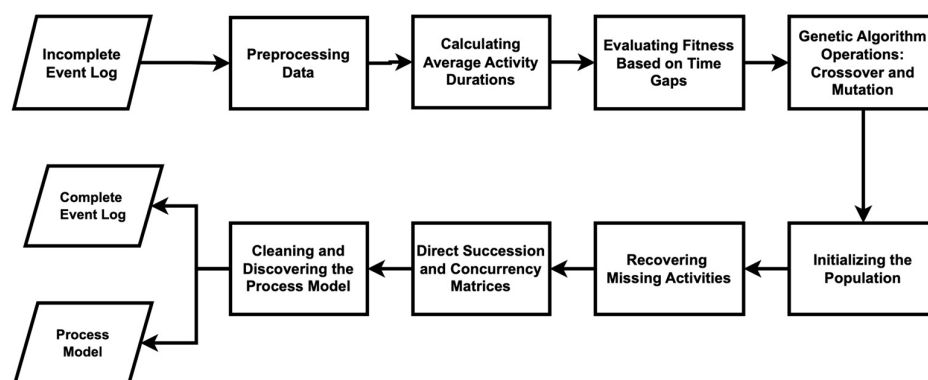


Figure 5. A flowchart of the TGIPM algorithm used to recover missing activities from the event logs.

The detailed steps of the TGIPM algorithm are outlined in Algorithm A1 (see Appendix A). The process begins with data preprocessing, where the event log is standardized by formatting timestamps, ensuring consistent column names, and validating the completeness of key attributes, including Case IDs, Activities, Start Time, and End Time. Next, the algorithm calculates average activity durations by grouping the event log by activity and computing the mean of the time differences between start and end times. These durations serve as benchmarks for evaluating the fitness of activity sequences in subsequent steps.

In the fitness evaluation phase, the algorithm compares the actual time gaps between consecutive activities with the expected gaps derived from the computed average durations. Sequences that align with these expected timings are assigned higher fitness scores, increasing their likelihood of selection for further optimization. Optimization is achieved through genetic algorithm operations, including crossover and mutation. Crossover combines segments of two parent sequences to generate new candidate solutions, while mutation introduces random variations to maintain diversity within the population.

The genetic process recovery step identifies missing activities in each trace of the event log. A population of candidate sequences is initialized to recover these activities. Over multiple generations, the sequences are refined through iterative fitness evaluations, crossover, and mutation. The best-performing sequence is selected to reconstruct the event

log, where timestamps for recovered activities are interpolated based on the temporal relationships with their neighboring activities.

Following the reconstruction of the event log, the algorithm computes direct succession and concurrency matrices to analyze the relationships between activities and identify parallelism within the process. Dummy activities, which represent low-frequency or irrelevant entries in the event log, are addressed by removing activities that fall below a specified occurrence threshold. This ensures the event log is clean and suitable for process model discovery.

The cleaned event log is then processed using the steps of IM, which generates a hierarchical process tree. This process tree is then converted into a simple process model visualization with initial and final markings, representing the final process model. By combining TGPM's capabilities in activity recovery with IM's structural precision, TGIPM delivers process models that are both accurate and robust.

The TGIPM approach not only addresses the limitations of its individual components but also reduces computational overhead by unifying recovery and model discovery into a single workflow. This integration makes TGIPM scalable and practical for large and complex datasets. The entire methodology is visually summarized in Figure 5, while the pseudocode implementation provided in Algorithm A1 (Appendix A) serves as a comprehensive guide for implementation.

3.2. Definition of Related Terms for TGIPM Algorithm

Process mining leverages event logs to reconstruct process models. The TGIPM algorithm extends existing process discovery techniques by integrating TGPM and IM. This section provides several definitions of the terms to help understand the TGIPM algorithm. All notations used in this section are borrowed from established process mining concepts and theories, as presented in [2], with necessary modifications to align with our approach.

Definition 1 (Dual Timestamps Event Log). A dual timestamps event log L is a finite set of events

$$L = \{e_1, e_2, \dots, e_n\}$$

where each event e_i ($1 \leq i \leq n$) is represented as a tuple

$$e_i = (c_i, a_i, t_i^s, t_i^e)$$

where

c_i is the case identifier (trace identifier) of the i -th event.

a_i is the activity's label executed at the i -th event.

t_i^s is the non-empty (or non-null) start time of the i -th event.

t_i^e is the non-empty (or non-null) end time of the i -th event.

Definition 2 (Process Model). A process model P is a tuple defined as follows:

$$P = (\mathcal{A}, R)$$

where

$\mathcal{A}$ is a finite set of all activity labels that are observed in the process model.

$R \subseteq \mathcal{A} \times \mathcal{A}$ is a union of the following relation sets—direct succession (R_d), concurrency (R_p), and choice (R_c). The three relation sets R_d , R_p , and R_c are mutually disjointed.

1. Direct Succession: $(a_i, a_j) \in R_d$ if activity a_i is directly followed by a_j in the same case, where $t_i^e \leq t_j^s$ and $c_i = c_j$.

2. Concurrency: $(a_i, a_j) \in R_p$ if activities a_i and a_j can be executed in parallel in the same case, where $t_i^s \leq t_j^s \leq t_i^e$ or $t_j^s \leq t_i^s \leq t_j^e$ and $c_i = c_j$.
3. Choice: $(a_i, a_j) \in R_c$ if either a_i or a_j directly occurs after the same activity a_k , but not both, where $(a_k, a_i) \in R_d \oplus (a_k, a_j) \in R_d$, $(a_i, a_j) \notin R_p$ and $c_i = c_j = c_k$. Here, $\oplus$ operator means exclusive or relationship.

Definition 3 (Fitness Function). Given a dual timestamps event log L and a discovered process model P , the fitness function $F(L, P)$ measures how well P represents L .

$$F(L, P) = \sum_{(a_i, a_j) \in R} \delta(e_i, e_j)$$

where

$\delta(x, y)$ is an indicator function, returning 1 if the condition (i.e., direct succession, concurrency, or choice) holds, and 0 otherwise.

Definition 4 (Completeness of Process Model). A process model P that is derived from a dual timestamps event log L is complete if and only if the following occurs:

$$\mathcal{A} = A_L$$

where A_L is the set of all activity labels observed in the dual timestamps event log L .

Definition 5 (Timed Fitness). Timed fitness tries to measure how well the observed execution durations align with the expected execution times in a process model P . The expected execution time of P is the shortest execution time of an ideal process instance p , when all the activity instances within p is considered to finish exactly in their average time.

Let T_{obs} be the observed time gaps between consecutive activities in the event log L and T_{exp} be the expected time gaps based on the process model P . The timed fitness function F_T is defined as:

$$F_T(L, P) = 1 - \frac{\sum_{(a_i, a_j) \in R} |T_{obs}(a_i, a_j) - T_{exp}(a_i, a_j)|}{\sum_{(a_i, a_j) \in R} T_{exp}(a_i, a_j)}$$

In addition, soundness in process models, particularly those generated by the IM algorithm, ensures that the discovered models are both correct and executable. According to [2], a process model is considered sound if it meets three key criteria: proper completion, the absence of deadlocks or livelocks, and no unreachable states or activities. Proper completion ensures that at least one possible execution sequence reaches a final state, meaning the process can always be completed correctly. The absence of deadlocks or livelocks guarantees that the model does not contain states where execution cannot proceed or infinite loops prevent completion. Additionally, all activities must be reachable from the initial state, ensuring that every part of the process model contributes to its overall execution. IM inherently produces sound models by applying a divide-and-conquer approach, which constructs well-formed, block-structured models that comply with these criteria [2,15]. This guarantees that the generated process models are executable and free from structural errors, making them reliable for process mining applications.

Similarly, the concept of soundness applies to our approach, TGIPM. After generating the complete event log using TGPM, we proceed with IM. TGIPM does not interfere with the operation of the IM algorithm. It simply adds preprocessing steps for dual timestamp event log before the execution of the IM algorithm, as explained in Algorithm A1 (Appendix A). This ensures the soundness of the process model generated by the TGIPM algorithm.

4. Experiment Evaluation

In this section, we discuss the purpose, setup, and results of the experiments conducted to verify and validate the proposed TGIPM algorithm. The experiments are designed to evaluate the algorithm's ability to recover missing activities, generate complete event logs, and produce sound process models. Furthermore, we assess its performance in handling real-world scenarios with incomplete event logs and compare it against benchmark algorithms to demonstrate its effectiveness.

Real-world event logs are often significantly more complex than those used in controlled research settings, as they involve numerous cases, activities, timestamps, and intricate dependencies. To address this complexity, the analysis in this study focuses on the core attributes of event logs: Case ID, Activity, and Timestamps. These elements are essential for reconstructing process models and analyzing process behaviors, ensuring a robust and practical evaluation of the algorithm [8].

4.1. Experiment Purpose

The purpose of this experiment is to evaluate the effectiveness of the proposed method in recovering missing activities and their associated execution times from an event log while simultaneously generating sound and robust process models. The objective is to ensure that the reconstructed event log accurately represents the original company processes until the actual data are entered into the company database.

The experimental evaluation aims to address the following two key research questions: (1) Can the proposed method accurately recover missing activities and their execution times? This involves assessing whether the method can correctly infer and reconstruct the missing parts of the event log in a way that aligns with the underlying process data. (2) Can the proposed method accurately reconstruct the original process model with a complex structure from incomplete event logs? This involves determining whether the reconstructed event log enables the discovery of a process model that reflects the true complexity and structure of the original process.

To ensure a robust evaluation, the experiment addresses real-world challenges, including incomplete event logs, dummy activities, concurrency, and varying activity durations. The effectiveness of the proposed method is assessed by comparing the reconstructed event log and process model against ground-truth data. Key conformance metrics [22,47] such as fitness (Equation (1)), precision (Equation (2)), generalization (Equation (3)), simplicity (Equation (4)), and time complexity are used to measure performance. These evaluations validate the practicality and reliability of the proposed method for real-world applications, particularly in complex domains like healthcare.

$$Q_f = \frac{\#casesCaptured}{\#casesLog} \quad (1)$$

$$Q_p = \frac{\#traceCaptured}{\#tracesModel} \quad (2)$$

$$Q_g = 1 - \frac{\sum_1^{nodeTree} \left(\sqrt{\#exactNode} \right)^{-1}}{\#nodeTree} \quad (3)$$

$$Q_s = 1 - \frac{\#dupAct + \#misAct}{\#nodeTree} \quad (4)$$

where

Q_f is the fitness value.

Q_p is the precision value.

Q_g is the generalization value.

Q_s is the simplicity value.

$\#casesCaptured$ is the number of processes depicted in the process model from the event log.

$\#casesLog$ is the number of processes in the event log.

$\#tracesCaptured$ is the number of traces recorded in the event log.

$\#tracesModel$ is the number of traces that can be formed from the process model.

$\#dupAct$ is the number of activities depicted as duplications in the tree model and the total number of duplications.

$\#misAct$ is the number of activities in the event log that are not depicted in the tree model, and the activities that are depicted in the tree model but not in the event log.

$\#exactNode$ is the number of nodes in the tree model traversed corresponds to the process in the event log.

$\#nodeTree$ is the number of nodes contained in the tree model.

4.2. Experiment Setup

The experiments in this study utilized real-world data collected from a private hospital in Surabaya, Indonesia, during March and April 2023. The event log, as shown in Figure 6, includes case identifiers (Case IDs) to differentiate individual process instances, activity names, start times, and end times. The structure of the event log highlights both sequential and parallel processes, as evidenced by the timestamps recorded for each activity. The dataset comprises 50 cases and 13 distinct activities.

COMPLETE EVENT LOG					INCOMPLETE EVENT LOG				
CASE_ID	ACTIVITY	START_TIME	END_TIME		CASE_ID	ACTIVITY	START_TIME	END_TIME	
PRO004	Patient_Check_In	2023-03-12 08:13:00	2023-03-12 08:21:00		PRO019	Patient_Check_In	2023-03-19 08:15:22	2023-03-19 08:31:22	
PRO004	Patient_Assignment	2023-03-12 08:20:21	2023-03-12 08:47:21	MISSING_ACT	PRO019	Insurance_Verification	2023-03-19 08:25:07	2023-03-19 08:53:07	
PRO004	Insurance_Verification	2023-03-12 08:26:24	2023-03-12 08:39:24		PRO019	Initial_Assessment	2023-03-19 09:09:38	2023-03-19 09:14:38	
PRO004	Initial_Assessment	2023-03-12 08:58:43	2023-03-12 09:07:43		PRO019	Diagnostic_Tests	2023-03-19 09:20:33	2023-03-19 09:40:33	
PRO004	Diagnostic_Tests	2023-03-12 09:03:49	2023-03-12 09:29:49		PRO019	Doctor_Consultation	2023-03-19 09:09:39	2023-03-19 09:33:39	
PRO004	Doctor_Consultation	2023-03-12 09:11:16	2023-03-12 09:27:16		PRO019	Treatment_Planning	2023-03-19 09:54:46	2023-03-19 10:24:46	
PRO004	Treatment_Planning	2023-03-12 09:52:08	2023-03-12 10:18:08		PRO019	Medication_Administration	2023-03-19 10:06:26	2023-03-19 10:21:26	
PRO004	Medication_Administration	2023-03-12 09:52:26	2023-03-12 10:04:26		PRO019	Therapy_Sessions	2023-03-19 10:07:09	2023-03-19 10:31:09	
PRO004	Therapy_Sessions	2023-03-12 09:48:38	2023-03-12 10:02:38		PRO019	Generate_Bill	2023-03-19 10:56:48	2023-03-19 11:22:48	
PRO004	Generate_Bill	2023-03-12 10:25:03	2023-03-12 10:49:03		PRO019	Payment_Processing	2023-03-19 10:52:01	2023-03-19 11:08:01	
PRO004	Payment_Processing	2023-03-12 10:45:13	2023-03-12 10:57:13		PRO019	Discharge_Summary	2023-03-19 10:58:27	2023-03-19 11:12:27	
PRO004	Discharge_Summary	2023-03-12 10:21:00	2023-03-12 10:30:00		PRO019	Patient_Discharged	2023-03-19 11:22:48	2023-03-19 11:31:48	
PRO004	Patient_Discharged	2023-03-12 10:57:13	2023-03-12 11:19:13		PRO033	Patient_Check_In	2023-04-02 07:50:02	2023-04-02 08:00:02	
PRO004	Patient_Check_In	2023-04-14 08:18:57	2023-04-14 08:39:57		PRO033	Insurance_Verification	2023-04-02 08:00:02	2023-04-02 08:30:02	
PRO044	Insurance_Verification	2023-04-14 08:27:31	2023-04-14 08:49:31		PRO033	Patient_Assignment	2023-04-02 08:22:30	2023-04-02 08:51:30	
PRO044	Patient_Assignment	2023-04-14 08:13:22	2023-04-14 08:35:22		PRO033	Initial_Assessment	2023-04-02 09:07:32	2023-04-02 09:18:32	
PRO044	Initial_Assessment	2023-04-14 09:16:05	2023-04-14 09:46:05		PRO033	Diagnostic_Tests	2023-04-02 08:54:15	2023-04-02 09:15:15	
PRO044	Diagnostic_Tests	2023-04-14 08:49:40	2023-04-14 09:17:40		PRO033	Doctor_Consultation	2023-04-02 09:14:09	2023-04-02 09:34:09	
PRO044	Doctor_Consultation	2023-04-14 09:11:14	2023-04-14 09:40:14		PRO033	Treatment_Planning	2023-04-02 09:44:36	2023-04-02 10:08:36	
PRO044	Treatment_Planning	2023-04-14 10:11:30	2023-04-14 10:22:30		PRO033	Medication_Administration	2023-04-02 09:57:47	2023-04-02 10:27:47	
PRO044	Medication_Administration	2023-04-14 10:03:00	2023-04-14 10:22:00						
PRO044	Therapy_Sessions	2023-04-14 10:00:42	2023-04-14 10:07:42	MISSING_ACT	PRO033	Generate_Bill	2023-04-02 10:42:10	2023-04-02 11:05:10	
PRO044	Generate_Bill	2023-04-14 10:41:04	2023-04-14 10:52:04						
PRO044	Payment_Processing	2023-04-14 10:25:45	2023-04-14 10:54:45	MISSING_ACT					
PRO044	Discharge_Summary	2023-04-14 10:46:01	2023-04-14 10:57:01						
PRO044	Patient_Discharged	2023-04-14 10:57:01	2023-04-14 11:19:01	MISSING_ACT	PRO033	Patient_Discharged	2023-04-02 11:32:35	2023-04-02 11:42:35	

Figure 6. Hospital data showing both complete and incomplete event logs.

During the data collection period, certain activities were manually recorded by hospital staff and entered into the hospital information system after working hours, typically around 6 PM. This asynchronous data entry resulted in incomplete event logs, where some executed processes were not fully captured in the system. Specifically, 10 cases from March and April 2023 were identified as incomplete. Figure 6 illustrates examples of incomplete event logs, including one missing activity, ‘Patient_Assignment’, in case PRO019, and three missing activities, ‘Therapy_Sessions’, ‘Payment_Processing’, and ‘Discharge_Summary’, in case PRO033.

The experiments were conducted on a system configured with a Windows 10 Ultimate 64-bit operating system, 8 GB of memory, and a GPM setup with a maximum generation limit of 50 and a population size of 50. This configuration ensured the computational feasibility of the GPM algorithm in effectively handling incomplete event logs.

This study also evaluates the proposed methods by comparing two scenarios designed to discover process models from incomplete event logs. The objective is to analyze each scenario's effectiveness in recovering missing activities, reconstructing event logs, and generating accurate process models.

Scenario 1: Sequential Execution of TGPM and IM. In this scenario, the TGPM algorithm is executed independently to recover missing activities from incomplete event logs. The reconstructed event log generated from this step is then used as input for IM, which processes the log to discover a process model. This sequential execution separates the recovery of missing activities from the process model discovery phase, allowing for independent analysis and validation of each stage. However, this approach may introduce potential inefficiencies, as the genetic algorithm and IM operate in isolation, without leveraging combined optimization.

Scenario 2: Execution of TGIPM. In the second scenario, the TGIPM algorithm is executed as an integrated framework (Algorithm A1). This unified approach combines the recovery of missing activities with process model discovery in a single workflow. By integrating the strengths of TGPM and IM, this method simultaneously optimizes activity recovery and process model generation. The TGIPM algorithm leverages dual times and temporal dependencies to guide genetic optimization while ensuring the soundness and accuracy of the resulting process model through IM principles. This combined approach is expected to improve efficiency and produce more precise models compared to the sequential method.

The results of these two scenarios are analyzed to evaluate their performance in terms of recovering missing activities, achieving temporal fitness, and generating structurally sound and accurate process models. Additionally, conformance metrics and time complexity are compared to determine the practicality of the proposed methods for real-world applications.

Moreover, the quality of the repaired event log in this study is evaluated using two key criteria to ensure its accuracy and reliability. First, temporal alignment is assessed by comparing the timestamps of recovered activities in the reconstructed log against real-world process data. This step verifies whether the inserted activities follow a logical time sequence and align with the expected execution patterns observed in actual process logs. Second, process conformance is measured using standard conformance checking metrics, including fitness, precision, generalization, and simplicity. These metrics determine how well the repaired event log adheres to expected process execution rules, ensuring that missing activities are recovered without introducing inconsistencies or deviations from the original process model.

4.3. Experiment Results

The two scenarios utilize the incomplete event log shown in Figure 6 as their input. In Scenario 1, the algorithm generates a complete event log, illustrated in Figure 7a, while in Scenario 2, the TGIPM algorithm produces another complete event log, depicted in Figure 7b. These reconstructed event logs are compared to the original event log from the hospital, presented in Figure 8a. For case PRO019, the complete event log from Scenario 1 aligns more closely with the original timestamps. However, in case PRO033, the output from Scenario 2 better reflects the original event log by capturing the concurrency of activities, such as 'Generate_Bill', 'Payment_Processing', and 'Discharge_Summary', through timestamps indicating parallel execution of processes. In contrast, the sequential approach in Scenario 1 primarily represents timestamps in a linear order.

CASE_ID	ACTIVITY	START_TIME	END_TIME
PRO019	Patient_Check_In	2023-03-19 08:15:22	2023-03-19 08:31:22
PRO019	Insurance_Verification	2023-03-19 08:25:07	2023-03-19 08:53:07
PRO019	Patient_Assignment	2023-03-19 08:58:07	2023-03-19 09:08:07
PRO019	Initial_Assessment	2023-03-19 09:09:38	2023-03-19 09:14:38
PRO019	Doctor_Consultation	2023-03-19 09:09:39	2023-03-19 09:33:39
PRO019	Diagnostic_Tests	2023-03-19 09:20:33	2023-03-19 09:40:33
PRO019	Treatment_Planning	2023-03-19 09:54:46	2023-03-19 10:24:46
PRO019	Medication_Administration	2023-03-19 10:06:26	2023-03-19 10:21:26
PRO019	Therapy_Sessions	2023-03-19 10:07:09	2023-03-19 10:31:09
PRO019	Payment_Processing	2023-03-19 10:52:01	2023-03-19 11:08:01
PRO019	Generate_Bill	2023-03-19 10:56:48	2023-03-19 11:22:48
PRO019	Discharge_Summary	2023-03-19 10:58:27	2023-03-19 11:12:27
PRO019	Patient_Discharged	2023-03-19 11:22:48	2023-03-19 11:31:48
PRO033	Patient_Check_In	2023-04-02 07:50:02	2023-04-02 08:00:02
PRO033	Insurance_Verification	2023-04-02 08:00:02	2023-04-02 08:30:02
PRO033	Patient_Assignment	2023-04-02 08:22:30	2023-04-02 08:51:30
PRO033	Diagnostic_Tests	2023-04-02 08:54:15	2023-04-02 09:15:15
PRO033	Initial_Assessment	2023-04-02 09:07:32	2023-04-02 09:18:32
PRO033	Doctor_Consultation	2023-04-02 09:14:09	2023-04-02 09:34:09
PRO033	Treatment_Planning	2023-04-02 09:44:36	2023-04-02 10:08:36
PRO033	Medication_Administration	2023-04-02 09:57:47	2023-04-02 10:27:47
PRO033	Therapy_Sessions	2023-04-02 10:04:47	2023-04-02 10:30:10
PRO033	Generate_Bill	2023-04-02 10:42:10	2023-04-02 11:05:10
PRO033	Discharge_Summary	2023-04-02 11:05:35	2023-04-02 11:27:35
PRO033	Payment_Processing	2023-04-02 11:17:35	2023-04-02 11:42:35
PRO033	Patient_Discharged	2023-04-02 11:32:35	2023-04-02 11:42:35

(a)

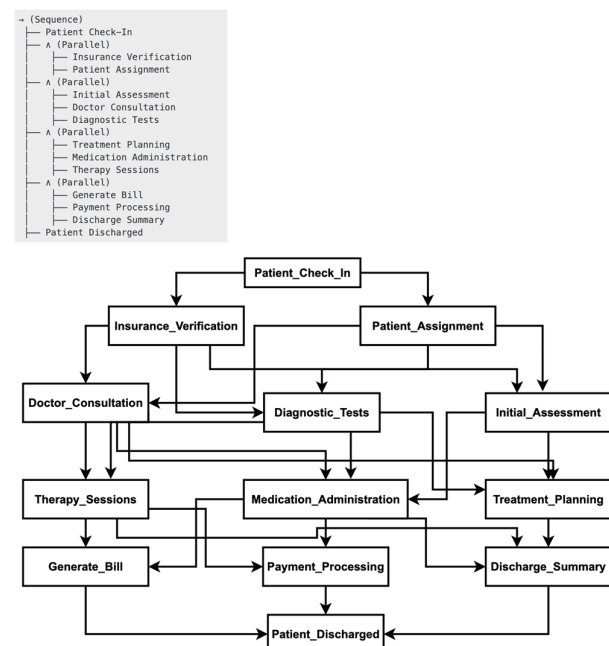
CASE_ID	ACTIVITY	START_TIME	END_TIME
PRO019	Patient_Check_In	2023-03-19 08:15:22	2023-03-19 08:31:22
PRO019	Insurance_Verification	2023-03-19 08:25:07	2023-03-19 08:53:07
PRO019	Patient_Assignment	2023-03-19 08:53:08	2023-03-19 09:15:48
PRO019	Initial_Assessment	2023-03-19 09:09:38	2023-03-19 09:14:38
PRO019	Diagnostic_Tests	2023-03-19 09:09:39	2023-03-19 09:33:39
PRO019	Doctor_Consultation	2023-03-19 09:20:33	2023-03-19 09:40:33
PRO019	Treatment_Planning	2023-03-19 09:54:46	2023-03-19 10:24:46
PRO019	Medication_Administration	2023-03-19 10:06:26	2023-03-19 10:21:26
PRO019	Therapy_Sessions	2023-03-19 10:07:09	2023-03-19 10:31:09
PRO019	Generate_Bill	2023-03-19 10:52:01	2023-03-19 11:08:01
PRO019	Payment_Processing	2023-03-19 10:56:48	2023-03-19 11:22:48
PRO019	Discharge_Summary	2023-03-19 10:58:27	2023-03-19 11:12:27
PRO019	Patient_Discharged	2023-03-19 11:22:48	2023-03-19 11:31:48
PRO033	Patient_Check_In	2023-04-02 07:50:02	2023-04-02 08:00:02
PRO033	Insurance_Verification	2023-04-02 08:00:02	2023-04-02 08:30:02
PRO033	Patient_Assignment	2023-04-02 08:22:30	2023-04-02 08:51:30
PRO033	Initial_Assessment	2023-04-02 08:54:15	2023-04-02 09:15:15
PRO033	Diagnostic_Tests	2023-04-02 09:07:32	2023-04-02 09:18:32
PRO033	Doctor_Consultation	2023-04-02 09:14:09	2023-04-02 09:34:09
PRO033	Treatment_Planning	2023-04-02 09:44:36	2023-04-02 10:08:36
PRO033	Medication_Administration	2023-04-02 09:57:47	2023-04-02 10:27:47
PRO033	Therapy_Sessions	2023-04-02 09:48:29	2023-04-02 10:06:29
PRO033	Generate_Bill	2023-04-02 10:42:10	2023-04-02 11:05:10
PRO033	Payment_Processing	2023-04-02 10:29:23	2023-04-02 10:48:23
PRO033	Discharge_Summary	2023-04-02 10:36:10	2023-04-02 10:58:10
PRO033	Patient_Discharged	2023-04-02 11:32:35	2023-04-02 11:42:35

(b)

Figure 7. Comparison of hospital obtained complete event logs: (a) generated by TGPM and IM algorithms, which executed sequentially; (b) generated by TGIPM algorithm.

CASE_ID	ACTIVITY	START_TIME	END_TIME
PRO019	Patient_Check_In	2023-03-19 08:15:22	2023-03-12 08:31:22
PRO019	Insurance_Verification	2023-03-19 08:25:07	2023-03-12 08:53:07
PRO019	Patient_Assignment	2023-03-19 08:57:21	2023-03-12 09:10:07
PRO019	Initial_Assessment	2023-03-19 09:09:38	2023-03-19 09:14:38
PRO019	Diagnostic_Tests	2023-03-19 09:09:39	2023-03-19 09:33:39
PRO019	Doctor_Consultation	2023-03-19 09:20:33	2023-03-19 09:40:33
PRO019	Treatment_Planning	2023-03-19 09:54:46	2023-03-19 10:24:46
PRO019	Medication_Administration	2023-03-19 10:06:26	2023-03-19 10:21:26
PRO019	Therapy_Sessions	2023-03-19 10:07:09	2023-03-19 10:31:09
PRO019	Generate_Bill	2023-03-19 10:52:01	2023-03-19 11:08:01
PRO019	Payment_Processing	2023-03-19 10:56:48	2023-03-19 11:22:48
PRO019	Discharge_Summary	2023-03-19 10:58:27	2023-03-19 11:12:27
PRO019	Patient_Discharged	2023-03-19 11:22:48	2023-03-19 11:31:48
PRO033	Patient_Check_In	2023-04-02 07:50:02	2023-04-02 08:00:02
PRO033	Insurance_Verification	2023-04-02 08:00:02	2023-04-02 08:30:02
PRO033	Patient_Assignment	2023-04-02 08:22:30	2023-04-02 08:51:30
PRO033	Initial_Assessment	2023-04-02 08:54:15	2023-04-02 09:15:15
PRO033	Diagnostic_Tests	2023-04-02 09:07:32	2023-04-02 09:18:32
PRO033	Doctor_Consultation	2023-04-02 09:14:09	2023-04-02 09:34:09
PRO033	Treatment_Planning	2023-04-02 09:44:36	2023-04-02 10:08:36
PRO033	Medication_Administration	2023-04-02 09:57:47	2023-04-02 10:27:47
PRO033	Therapy_Sessions	2023-04-02 09:50:42	2023-04-14 10:07:42
PRO033	Generate_Bill	2023-04-02 10:42:10	2023-04-02 11:05:10
PRO033	Payment_Processing	2023-04-02 10:27:45	2023-04-14 10:49:03
PRO033	Discharge_Summary	2023-04-02 10:38:25	2023-04-14 10:57:01
PRO033	Patient_Discharged	2023-04-02 11:32:35	2023-04-02 11:42:35

(a)



(b)

Figure 8. Hospital data: (a) real event logs after being input; (b) original process model.

The differences in results arise from the structural variations between the two algorithms. In Scenario 1, the TGPM algorithm is executed first to recover missing activities from the incomplete event log. This generates a complete event log, which is subsequently processed by IM. While this sequential execution effectively improves the process model by removing dummy activities and identifying accurate parallel gateways, it does not adjust or refine timestamps beyond the initial recovery phase.

In Scenario 2, TGIPM uses crossover and mutation to recover missing activities while simultaneously adjusting timestamps based on direct succession and concurrency matrices. The integration enables the algorithm to better capture parallelism in the reconstructed event log, leveraging dual time information to refine the discovered process model. In this

case study, although the recovered missing activities align reciprocally with the original data, the integrated approach significantly enhances the accuracy of the process model and improves conformance checking results.

The process models generated by Algorithm A1 are represented as process trees, requiring conversion into a visualized process model to better understand the flow from start to finish. The process model obtained from Scenario 1, shown in Figure 9a, exhibits a simpler structure with fewer relationships. In contrast, the model generated in Scenario 2, displayed in Figure 9b, presents a more intricate and detailed representation. Despite these structural differences, both models successfully capture the sequential and parallel relationships of the original hospital process model, as illustrated in Figure 8b, although neither is entirely perfect. The primary distinction between the two lies in temporal fitness—while both scenarios effectively recover missing activities and produce structurally sound process models, Scenario 2 demonstrates superior performance in aligning timestamps and accurately representing concurrency within the process.

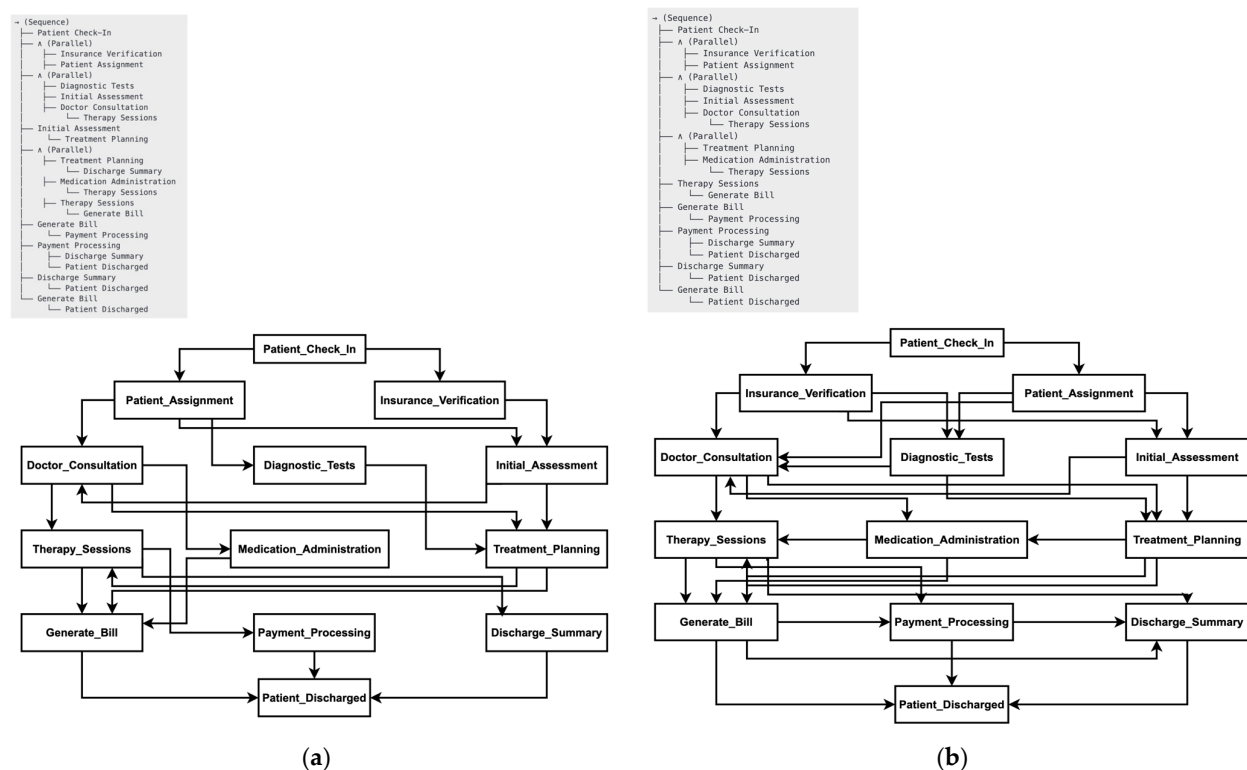


Figure 9. Comparison of hospital process models: (a) generated by TGPM and IM algorithms, which executed sequentially; (b) generated by TGIPM algorithm.

When comparing the process models from Scenario 1 and Scenario 2 to the original model, both approaches introduce improvements in handling missing activities but differ in their trade-offs. The process model in Scenario 1 provides a simplified, step-by-step approach, ensuring clarity and process separation at the cost of losing some interdependencies present in the original model. In contrast, the process model in Scenario 2 retains the real-world complexity of the original model while enhancing its ability to recover missing activities, making it a more adaptive and robust approach.

Overall, the process model in Scenario 1 is the better choice when clarity and sequential execution are the primary objectives, as it provides a structured and well-separated process discovery approach. However, the process model in Scenario 2 is more effective when handling incomplete event logs, as it integrates missing activity recovery directly into process model generation, ensuring greater accuracy and adaptability. While the process

model in Scenario 1 is easier to interpret, the process model in Scenario 2 offers a more comprehensive representation of real-world processes, making it the superior model for handling dynamic and complex workflows.

Based on the event logs and process models above, we have listed the pros and cons of each algorithm in Table 1. The sequential method separates the tasks into distinct steps, first completing the event log and then refining the process model. Its strengths include clear task separation, enhanced control and validation due to intermediate outputs, and the ability to produce a refined process model with minimal dummy activities. However, this approach is computationally intensive, may result in slightly lower log quality compared to the combined approach, and introduces potential redundancy in handling tasks separately.

Table 1. Pros and cons of each scenario for handling incomplete cases.

	Scenario 1	Scenario 2
Flow	Incomplete event log → TGPM → Complete Event Log → IM → Process Model	Incomplete event log → TGIPM → Complete Event Log and Process Model
Pros	Clear Separation of Tasks: Each algorithm performs a distinct function, allowing TGPM to focus solely on completing the event log and IM on model refinement. Control and Validation: Having a complete event log as an intermediate step allows us to validate it independently before proceeding to model generation, enhancing transparency. Quality of Process Model: Since IM operates on a complete event log, it may generate a refined and optimized process model with minimal dummy activities.	Improved Log Quality: TGIPM appears to produce a slightly better complete event log, potentially capturing nuanced dependencies and context between tasks. Efficiency: Combining both steps into one phase can reduce processing time, as it minimizes redundancy in handling the data. Integrated Task: With both algorithms working together, the completion and model discovery tasks may complement each other, enhancing coherence in the event log and process model.
Cons	Computational Cost: This approach may be computationally intensive and time-consuming, as each step is handled separately. Slightly Lower Log Quality: The complete event log generated here is slightly less accurate than that in Scenario 2. Potential Redundancy: Running two separate steps might introduce redundancy if the combined approach can achieve similar or better results.	Less Intermediate Control: Without a separate complete event log as an output, validation between steps is limited, which could make debugging and quality assessment harder. Potential Model Complexity: Depending on how well the combined algorithm handles both tasks, the process model could risk becoming more complex or less refined if the integration is not perfectly balanced.

In contrast, the TGIPM approach integrates event log completion and process model discovery into a single unified step, which improves efficiency by reducing processing time and redundancy. It also tends to produce slightly better log quality, capturing nuanced dependencies and context between tasks. Additionally, the combined approach enhances coherence in event logs and process models by integrating both algorithms. However, it lacks intermediate outputs for debugging and validation, potentially complicating quality assessment. Furthermore, improper integration could lead to overly complex or less refined process models. Each approach offers unique advantages and trade-offs depending on the goals and requirements of the application.

Table 2 provides a comprehensive comparison of their performance across key conformance metrics, calculated using Equations (1)–(4). The TGIPM algorithm outperforms the sequential approach in fitness (0.92 vs. 0.85) and precision (0.91 vs. 0.88), indicating that it aligns more closely with the event log and better captures the correct sequences of activities. Furthermore, the TGIPM algorithm demonstrates superior generalization, scoring 0.90 compared to 0.83 for the sequential approach, highlighting its ability to adapt to unseen

cases. However, the sequential execution algorithm retains an edge in simplicity, achieving 0.90 versus 0.87 for the TGIPM algorithm, suggesting that separating tasks results in a less complex process model.

Table 2. Conformance checking results.

Metric	Scenario 1	Scenario 2
Fitness	0.85	0.92
Precision	0.88	0.91
Simplicity	0.90	0.87
Generalization	0.83	0.90
Time Complexity	$O(g \cdot p \cdot n(m + \text{clog } c)) + O(n \log n + m^2 + t \cdot k + s)$	$O(g \cdot p \cdot n(m + \text{clog } c)) + O(m^2 + t \cdot k)$

where g is the number of generations in the genetic algorithm. p is the population size in the genetic algorithm. n is the number of cases in the event log. m is the total number of unique activities in the event log (global across all cases). c is the number of missing activities in the incomplete event log (global count). t is the number of tokens used in process model generation or conformance checking. k is the number of activities in a single case. s is the population size used in token replay and validation.

The combined TGIPM approach shows significant advantages in terms of computational efficiency and robustness. It has a relatively lower computational cost compared to the sequential approach, making it faster and more efficient, especially when handling large datasets or incomplete event logs. This efficiency stems from the integration of task completion and process model discovery, which minimizes redundancy and maximizes data dependency utilization. Scenario 2's ability to accurately fill gaps and adapt to data dependencies leads to a more comprehensive event log and a process model that excels in fitness and generalization. While both approaches achieve high precision, the TGIPM algorithm marginally surpasses the sequential method due to improved coherence in event log completion.

In summary, while Scenario 1 benefits from a simpler and more straightforward process model, Scenario 2 offers superior performance across most metrics, particularly in fitness, precision, and generalization. These results suggest that the combined TGIPM approach is the preferred choice for practical implementation, particularly for complex, incomplete, or large-scale event logs.

For time complexity, Scenario 2 demonstrates a lower overall time complexity compared to Scenario 1 due to its integrated nature, which effectively reduces redundant computations. In both scenarios, for large values of n (number of cases), m (number of unique activities), or c (number of missing activities), the genetic operations dominate, making $g \cdot p \cdot n \cdot \text{clog } c$ the primary term influencing computational cost. However, Scenario 2 mitigates additional overhead from the IM phase by integrating recovery and process model discovery into a unified framework.

When comparing average complexities, Scenario 2 consistently outperforms Scenario 1 as the dataset size increases, with Scenario 2 requiring fewer resources to achieve the same or better outcomes. This analysis underscores Scenario 2's efficiency advantage, particularly for large or incomplete datasets, making it the preferred choice for practical applications in process mining.

To conclude the experimental research, this study acknowledges the complexities inherent in real-world event logs compared to simplified case studies but focuses on the core attributes of event logs, Case ID, Activity, and Execution Time (Start Time and End Time), as these three elements are essential for reconstructing process models and understanding process behaviors, forming the foundation for effective process mining methodologies [8]. The study also highlights the importance of analyzing the structural and sequential ordering of activity execution to improve the recovery of missing activities in periodic and asynchronous systems. By examining the sequence and structure of activities, the research

demonstrates that even incomplete event logs can yield robust and complex process models. This is particularly significant in environments where activities are recorded periodically or asynchronously, resulting in data gaps. The study's findings propose first integrating methods in process mining to bridge these gaps by generating temporary event logs using the proposed algorithm until the actual logs become available in the system.

4.4. Comparison with Existing Genetic and Inductive Mining Approaches

Process discovery techniques have traditionally focused on either optimizing fitness or ensuring structural correctness, but very few methods effectively address both missing activity recovery and model soundness simultaneously. TGIPM overcomes this gap by integrating TGPM with IM into a unified framework, enhancing both conformance and structural integrity.

Previous GPM approaches, such as [12], constructed process trees to enforce soundness by construction. However, these methods did not actively recover missing activities, making them less effective in handling incomplete event logs. Similarly, ref. [11] optimized process model fitness using evolutionary techniques but lacked structural constraints, which may lead to inconsistent or infeasible process models. TGIPM addresses these limitations through the following:

1. Dual timestamps integration, which improves missing activity reconstruction compared to [12]'s method.
2. Genetic selection optimization, which refines process model fitness while preserving process structure, unlike [11]'s approach.
3. Simultaneous recovery and discovery, ensuring that missing activities are reconstructed while the discovered process model remains structurally sound.

Experimental results in Section 4.3 and comparison results in Table 3 demonstrate that TGIPM consistently outperforms existing techniques in handling missing activities, ensuring structural correctness, and maintaining computational efficiency. Unlike previous approaches, which either focus on fitness [11], or enforce soundness [12] and IM, TGIPM integrates both objectives, making it a practical, scalable, and effective solution for real-world process mining challenges.

Table 3. Comparison results.

Approach	Handles Missing Activities	Ensures Structural Correctness	Computational Cost	Captures Parallelism
[12]	No	Yes	High	No
[11]	Yes	No	High	Yes
The IM	No	Yes	Low	Yes
Scenario 1	Yes	Yes	Medium	Yes
Scenario 2 (Proposed Approach)	Yes	Yes	Low	Yes

5. Conclusions

This study addresses the challenges of process discovery from incomplete event logs by introducing the TGIPM algorithm, which integrates TGPM and IM. These two well-established process discovery algorithms are combined to recover missing activities while ensuring the generation of sound and precise process models. Unlike the traditional IM, which struggles with missing activities, or standalone GPM, which lacks structural correctness, TGIPM combines the strengths of both approaches to recover missing activities while generating sound and precise process models. The proposed approach successfully answers two key research questions: (1) it effectively recovers missing activities and their

execution times, and (2) it reconstructs an accurate process model that preserves complex structural dependencies, even under incomplete event log conditions.

Experimental evaluations demonstrate that the TGIPM approach outperforms the sequential execution of TGPM and IM across key conformance metrics, including fitness, precision, generalization, and computational efficiency. By integrating activity recovery with process model discovery, TGIPM reduces redundant computations while better capturing parallelism in process executions. While the sequential approach produces simpler models due to the clear separation of tasks, TGIPM offers greater accuracy in reconstructing event logs and process models, particularly for datasets containing complex and concurrent activities.

However, a notable limitation of TGIPM is its compromise in simplicity. While the integration of two algorithms into a unified framework improves accuracy, it also introduces additional model complexity. The process models generated by TGIPM tend to capture more intricate dependencies, which may lead to higher structural complexity compared to models derived from the sequential approach. This trade-off between simplicity and completeness is inherent to the methodology, as ensuring a faithful reconstruction of real-world processes often requires handling nuanced dependencies. Future research could explore optimization techniques to mitigate excessive complexity while maintaining the accuracy and robustness of the process model. Additionally, while this study prioritizes completeness, other data quality issues have not yet been fully addressed and should be explored in future work.

These findings highlight the importance of integrating recovery and discovery algorithms in process mining to achieve more robust and efficient process discovery. Future work will focus on extending the TGIPM algorithm for real-time applications and further enhancing its scalability for industrial settings, ensuring its applicability in dynamic environments with high volumes of event data.

Author Contributions: This study is the result of a collaborative intellectual effort by the entire team. Conceptualization, Y.A.E. and M.K.; methodology, Y.A.E.; software, Y.A.E.; validation, Y.A.E. and M.K.; formal analysis, Y.A.E. and M.K.; investigation, Y.A.E.; resources, Y.A.E.; data curation, Y.A.E.; writing—original draft preparation, Y.A.E.; writing—review and editing, M.K.; visualization, Y.A.E.; supervision, M.K.; project administration, M.K.; funding acquisition, M.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant, and was funded by the government of the Republic of Korea (MSIT) (RS-2023-00242528).

Data Availability Statement: The experimental data and code associated with this paper are available upon request from the corresponding author.

Acknowledgments: The authors express their appreciation to the Business Computing Lab, Pukyong National University and the Robotics, Mechatronics, & Intelligent System Research Group at the Faculty of Advanced Technology and Multidiscipline, Airlangga University, for their valuable support throughout the research process.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	artificial intelligence
GPM	Genetic Process Mining
IM	Inductive Mining
TGPM	Timed Genetic Process Mining
TGIPM	Timed Genetic-Inductive Process Mining
GSPNs	Generalized Stochastic Petri Nets
PIM	Probabilistic Inductive Miner
DPIM	Differentially Private Inductive Miner
IvM	Inductive Visual Miner
ERP	Enterprise Resource Planning
BPM	business process management
HIS	Hospital Information Systems

Appendix A

Algorithm A1. Pseudocode of TGIPM to recover missing activities and discover a process model

Input:

EventLog: Event log containing CaseID, Activity, Start Time, End Time

Generations: Number of generations for the genetic algorithm

PopulationSize: Number of individuals in the genetic population

Output:

ProcessModel: A process model visualization

Step 1: Read and Preprocess Event Log

Algorithm Read_Event_Log(file_path)

Read event log from file_path

Return event_log

Step 2: Calculate Average Activity Durations

Algorithm Calculate_Average_Durations(event_log)

Initialize average_durations as an empty dictionary

Group event_log by Activity

For each activity in grouped_event_log:

Compute average_duration = Mean(End Time – Start Time)

Store average_duration in average_durations dictionary

Return average_durations

Step 3: Evaluate Fitness of an Activity Sequence

Algorithm Evaluate_Fitness(sequence, start_times, end_times, average_durations)

Initialize fitness_score = 0

For i = 1 to length(sequence) – 1:

If i < length(start_times) – 1:

expected_gap = average_durations(sequence [i + 1])

actual_gap = start_times [i + 1] – end_times [i]

If actual_gap >= expected_gap:

fitness_score += 1

Return fitness_score

Step 4: Genetic Operations (Crossover and Mutation)

Algorithm Perform_Crossover(parent1, parent2)

If length(parent1) > 2 AND length(parent2) > 2:

Select crossover_point randomly from range(1, length(parent1) – 1)

offspring1 = parent1[0:crossover_point] + parent2[crossover_point:end]

offspring2 = parent2[0:crossover_point] + parent1[crossover_point:end]

Else:

offspring1, offspring2 = parent1, parent2

Return offspring1, offspring2

Algorithm Perform_Mutation(sequence, activity_pool)

If length(sequence) > 1:

Select mutation_index randomly from range (0, length(sequence) – 1)

Replace sequence [mutation_index] with a random activity from activity_pool

Return sequence

Step 5: Initialize Genetic Algorithm Population

Algorithm Initialize_Population(size, missing_activities)

Initialize empty population list

For i = 1 to size:

Generate a random sequence of missing_activities

Add sequence to population

Return population

Step 6: Genetic Process Mining for Recovering Missing Activities

Algorithm Genetic_Process_Recovery (event_log, generations, population_size)

Extract all unique activities from event_log

Compute average_durations using Calculate_Average_Durations (event_log)

Initialize recovered_log as an empty list

For each case in event_log grouped by CaseID:

Sort case events by Start Time

Extract case_activities, start_times, and end_times

Identify missing_activities = all_activities – case_activities

Initialize population using Initialize_Population (population_size, missing_activities)

For generation = 1 to generations:

Compute fitness_scores for each individual in population

Select individuals with maximum fitness

Initialize new_population = empty list

While length(new_population) < population_size:

Select parent1, parent2 randomly from selected_population

Generate offspring1, offspring2 using Perform_Crossover (parent1, parent2)

Apply Perform_Mutation to offspring1 and offspring2

Add mutated offspring to new_population

Update population = new_population

Select best_individual with highest fitness

Insert missing activities in chronological order based on average durations

Store updated case in recovered_log

Return recovered_log

```

# Step 7: Compute Direct Succession Matrix
Algorithm Compute_Direct_Succession (event_log, activities)
Initialize direct_succession matrix with zeros for all activity pairs (a1, a2)
For each trace in event_log grouped by CaseID:
Sort trace by Start Time
For i = 1 to length(trace) – 1:
Increment direct_succession[trace[i], trace[i + 1]]
Return direct_succession

# Step 8: Identify Concurrency Matrix
Algorithm Identify_Concurrency (direct_succession, activities)
Initialize concurrency_matrix with zeros for all activity pairs (a1, a2)
For each (a1, a2) in direct_succession:
If direct_succession[a1, a2] > 0 OR direct_succession[a2, a1] > 0:
concurrency_matrix[a1, a2] = 1
Return concurrency_matrix

# Step 9: Remove Dummy Activities Based on Frequency Threshold
Algorithm Remove_Dummy_Activities (event_log, threshold)
Compute activity_counts = Frequency of each activity in event_log
Identify dummy_activities = activities with count < threshold
Remove dummy_activities from event_log
Return cleaned_event_log

# Step 10: Discover Process Model Using Inductive Mining
Algorithm Discover_Process_Model (cleaned_event_log)
Apply Inductive Mining on cleaned_event_log to generate a process tree
Convert process tree into a process model visualization
Return process_model

```

References

1. Gao, R.X.; Krüger, J.; Merklein, M.; Möhring, H.-C.; Váncza, J. Artificial Intelligence in Manufacturing: State of the Art, Perspectives, and Future Directions. *CIRP Ann.* **2024**, *73*, 723–749. [\[CrossRef\]](#)
2. Van der Aalst, W.M.P. Process mining: Data science in action. In *Process Mining: Data Science in Action*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2016; pp. 140–178. [\[CrossRef\]](#)
3. Sjödin, D.; Parida, V.; Kohtamäki, M. Artificial intelligence enabling circular business model innovation in digital servitization: Conceptualizing dynamic capabilities, AI capacities, business models and effects. *Technol. Forecast. Soc. Chang.* **2023**, *197*, 122903. [\[CrossRef\]](#)
4. Aldoseri, A.; Al-Khalifa, K.N.; Hamouda, A.M. AI-Powered Innovation in Digital Transformation: Key Pillars and Industry Impact. *Sustainability* **2024**, *16*, 1790. [\[CrossRef\]](#)
5. Effendi, Y.A.; Minsoo, K. Refining Process Mining in Port Container Terminals Through Clarification of Activity Boundaries with Double-Point Timestamps. *ICIC Express Lett. Part B Appl.* **2024**, *15*, 61–70. [\[CrossRef\]](#)
6. Bose, R.; Mans, R.; van der Aalst, W. Wanna improve process mining results? In Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM), Singapore, 16–19 April 2013; Volume 2013, pp. 127–134.
7. Pei, J.; Wen, L.; Yang, H.; Wang, J.; Ye, X. Estimating Global Completeness of Event Logs: A Comparative Study. *IEEE Trans. Serv. Comput.* **2018**, *14*, 441–457. [\[CrossRef\]](#)
8. Process Mining Book. Available online: <https://fluxicon.com/book/read/dataext/> (accessed on 1 December 2024).
9. Laplante, P.A.; Ovaska, S.J. *Real-Time Systems Design and Analysis*; IEEE Press: Piscataway, NJ, USA, 2012; Volume 3, pp. 154–196.
10. Barchard, K.A.; Pace, L.A. Preventing human error: The impact of data entry methods on data accuracy and statistical results. *Comput. Hum. Behaviour.* **2011**, *27*, 1834–1839. [\[CrossRef\]](#)
11. De Medeiros, A.K.A.; Weijters, A.J.M.M.; van der Aalst, W.M.P. Genetic Process Mining: An Experimental Evaluation. *Data Min. Knowl. Discov.* **2007**, *14*, 245–304. [\[CrossRef\]](#)
12. Buijs, J.C.A.M. Flexible Evolutionary Algorithms for Mining Structured Process Models. Ph.D. Thesis, Technische Universiteit, Eindhoven, The Netherlands, 2014. [\[CrossRef\]](#)

13. Effendi, Y.A.; Kim, M. Timed Genetic Process Mining for Robust Tracking of Processes under Incomplete Event Log Conditions. *Electronics* **2024**, *13*, 3752. [\[CrossRef\]](#)
14. De Medeiros, A.; Weijters, A.; Van der Aalst, W.M.P. *Using Genetic Algorithms to mine Process Models: Representation, Operators and Results*; Beta Working Paper Series, WP 124; Eindhoven University of Technology: Eindhoven, The Netherlands, 2004.
15. Pohl, T.; Pegoraro, M. *An Inductive Miner Implementation for the PM4PY Framework. i9 Process and Data Science (PADS)*; RWTH Aachen University: Aachen, Germany, 2019.
16. Sarker, I.H. AI-Based Modeling: Techniques, Applications and Research Issues Towards Automation, Intelligent and Smart Systems. *SN Comput. Sci.* **2022**, *3*, 158. [\[CrossRef\]](#)
17. Yang, Y.; Wu, Z.; Chu, Y.; Chen, Z.; Xu, Z.; Wen, Q. Intelligent Cross-Organizational Process Mining: A Survey and New Perspectives. *arXiv* **2024**, arXiv:2407.11280.
18. Aldoseri, A.; Al-Khalifa, K.N.; Hamouda, A.M. Re-Thinking Data Strategy and Integration for Artificial Intelligence: Concepts, Opportunities, and Challenges. *Appl. Sci.* **2023**, *13*, 7082. [\[CrossRef\]](#)
19. Effendi, Y.A.; Retrialisca, F. Transforming Timestamp of a CSV Database to an Event Log. In Proceedings of the 8th International Conference on Computer and Communication Engineering (ICCCCE), Kuala Lumpur, Malaysia, 22–23 June 2021; pp. 367–372. [\[CrossRef\]](#)
20. Effendi, Y.A.; Kim, M. Transforming Event Log with Domain Knowledge: A Case Study in Port Container Terminals. *ICIC Express Lett.* **2025**, *19*, 291–299. [\[CrossRef\]](#)
21. Huser, V. Process Mining: Discovery, Conformance and Enhancement of Business Processes. *J. Biomed. Inform.* **2012**, *45*, 1018–1019. [\[CrossRef\]](#)
22. Jans, M.; De Weerd, J.; Depaire, B.; Dumas, M.; Janssenswillen, G. Conformance Checking in Process Mining. *Inf. Syst.* **2021**, *102*, 101851. [\[CrossRef\]](#)
23. De Leoni, M. Foundations of Process Enhancement. In *Process Mining Handbook. Lecture Notes in Business Information Processing*; Van der Aalst, W.M.P., Carmona, J., Eds.; Springer: Cham, Switzerland, 2022; Volume 448. [\[CrossRef\]](#)
24. Yang, H.; van Dongen, B.F.; ter Hofstede, A.H.M.; Wynn, M.T.; Wang, J. Estimating completeness of event logs. *BPM Rep.* **2012**, *1204*, 1–23.
25. Li, C.; Ge, J.; Wen, L.; Kong, L.; Chang, V.; Huang, L.; Luo, B. A novel completeness definition of event logs and corresponding generation algorithm. *Expert Syst.* **2020**, *37*, 12529. [\[CrossRef\]](#)
26. Wang, L.; Fang, X.; Shao, C. Discovery of Business Process Models from Incomplete Logs. *Electronics* **2022**, *11*, 3179. [\[CrossRef\]](#)
27. Bose, R.P.J.C.; van der Aalst, W.M.P. Discovering Signature Patterns from Event Logs. *Knowl. Inf. Syst.* **2013**, *39*, 491–526. [\[CrossRef\]](#)
28. Suriadi, S.; Andrews, R.; ter Hofstede, A.H.M.; Wynn, M.T.; van der Aalst, W.M.P.; Budiayanto, M.A. Event Log Imperfection Patterns for Process Mining: Towards a Systematic Approach to Cleaning Event Logs. *Inf. Syst.* **2017**, *64*, 132–150. [\[CrossRef\]](#)
29. Dakic, J.; La Rosa, M.; Polyvyanyy, A. Taxonomy and Classification of Event Log Imperfections in Process Mining. *Inf. Syst.* **2023**, *113*, 102122.
30. Song, M.; Günther, C.W.; van der Aalst, W.M.P. Trace Clustering in Process Mining. In *Business Process Management Workshops*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 109–120. [\[CrossRef\]](#)
31. Lu, Y.; Chen, Q.; Poon, S.K. A Deep Learning Approach for Repairing Missing Activity Labels in Event Logs for Process Mining. *Information* **2022**, *13*, 234. [\[CrossRef\]](#)
32. Rogge-Solti, A.; Mans, R.S.; van der Aalst, W.M.P.; Weske, M. *Repairing Event Logs Using Stochastic Process Models*; Universitätsverlag Potsdam: Potsdam, Germany, 2013; p. 78. Available online: <https://publishup.uni-potsdam.de/opus4-ubp/frontdoor/index/index/docId/6370> (accessed on 14 November 2024).
33. Siek, M. Investigating inductive miner and fuzzy miner in automated business model generation. In Proceedings of the 3rd International Conference on Computer, Science, Engineering and Technology, Changchun, China, 22–24 September 2023; p. 2510. [\[CrossRef\]](#)
34. Wang, J.; Sun, Y.; Wen, L.; Wang, J. Cleaning Structured Event Logs: A Graph Repair Approach. *Comput. Ind.* **2015**, *70*, 194–205. [\[CrossRef\]](#)
35. Brons, D.; Scheepens, R.; Fahland, D. Striking a New Balance in Accuracy and Simplicity with the Probabilistic Inductive Miner. In Proceedings of the 2021 3rd International Conference on Process Mining (ICPM), Eindhoven, The Netherlands, 31 October–4 November 2021; pp. 80–87. [\[CrossRef\]](#)
36. Schulze, M.; Zisgen, Y.; Kirschte, M.; Mohammadi, E.; Koschmider, A. Differentially Private Inductive Miner. *arXiv* **2024**, arXiv:2407.04595. [\[CrossRef\]](#)
37. Leemans, S.J.J. Inductive Visual Miner Manual. *ProM Tools*. 2017. Available online: <https://www.leemans.ch/publications/ivm.pdf> (accessed on 14 February 2025).
38. De Medeiros, A. *Process Mining: Extending the α -Algorithm to Mine Short Loops*; BETA Working Paper Series, WP 113; Eindhoven University of Technology: Eindhoven, The Netherlands, 2004.

39. Mikolajczak, B.; Chen, J.L. Workflow Mining Alpha Algorithm—A Complexity Study. In *Intelligent Information Processing and Web Mining: Proceedings of the International IIS: IIPWM'05 Conference, Gdansk, Poland, 13–16 June 2005*; Kłopotek, M.A., Wierzchoń, S.T., Trojanowski, K., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; Volume 31. [\[CrossRef\]](#)
40. Porouhan, P.; Jongsawat, N.; Premchaiswadi, W. Process and deviation exploration through Alpha-algorithm and Heuristic miner techniques. In *Proceedings of the 2014 Twelfth International Conference on ICT and Knowledge Engineering*, Bangkok, Thailand, 18–21 November 2014; pp. 83–89. [\[CrossRef\]](#)
41. Burke, A.; Leemans, S.J.J.; Wynn, M.T. Stochastic Process Discovery by Weight Estimation. *Lect. Notes Bus. Inf. Process.* **2021**, *406*, 306–318. [\[CrossRef\]](#)
42. Kaur, H.; Mendling, J.; Rubensson, C.; Kampik, T. Timeline-Based Process Discovery. *arXiv* **2024**, arXiv:2401.04114.
43. Sarno, R.; Wibowo, W.A.; Kartini, K.; Amelia, Y.; Rossa, K. Determining Process Model Using Time-Based Process Mining and Control-Flow Pattern. *TELKOMNIKA* **2016**, *14*, 349–360. [\[CrossRef\]](#)
44. Sarno, R.; Kartini, W.; Wibowo, W.A.; Solichah, A. Time-Based Discovery of Parallel Business Processes. In *Proceedings of the 2015 International Conference on Computer, Control, Informatics and its Applications (IC3INA)*, Bandung, Indonesia, 5–7 October 2015; pp. 28–33. [\[CrossRef\]](#)
45. Fischer, M.; Küpper, T.; Bergener, K.; Räckers, M.; Becker, J. The Impact of Timestamp Granularity on Process Mining Insights: A Case Study. *Inf. Syst.* **2020**, *92*, 101531.
46. Fracca, G.; De Leoni, M.; Ter Hofstede, A.H.M. Managing Missing Start Timestamps in Event Logs: A Probabilistic Approach. In *Advanced Information Systems Engineering*; Springer: Cham, Switzerland, 2022; pp. 273–288.
47. Buijs, J.C.A.M.; van Dongen, B.F.; van der Aalst, W.M.P. On the role of fitness, precision, generalization and simplicity in process discovery. In *On the Move to Meaningful Internet Systems: OTM 2012*, 2nd ed.; Meersman, R., Ed.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7565, pp. 305–322.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.